НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

«На правах рукопису»
УДК __004.05_____

«До захисту допущено»
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
“___”_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: Автоматизована система для тестування доступності мобільних додатків

Виконала: студентка другого курсу, групи ІТ-84мп_
(шифр групи)

_Кострикіна Діана Анатоліївна_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

І.Р. Пархомей

(підпис)

«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Кострикіній Діані Анатоліївні

(прізвище, ім'я, по батькові)

1. Тема дисертації «Автоматизована система для тестування доступності мобільних додатків»,
науковий керівник дисертації ст.викладач, к.т.н. Сирота О.П.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 28 » жовтня 2019 р. № 3770-с
2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – тестування рівня доступності мобільних додатків

4. Предмет дослідження – система для тестування динамічної та статичної доступності на мобільних пристроях, на основі змін в контексті базуючись на інформації про користувальницький інтерфейс

5. Перелік завдань, які потрібно розробити – аналіз предметної області та існуючих рішень; аналіз і розробка алгоритму автоматичної генерації і скорочення простору станів призначеного для користувача інтерфейсу; розробка архітектури системи;

6. Орієнтовний перелік ілюстративного матеріалу – шість плакатів

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області	13 вересня 2019 р.	
2	Постановка задачі	16 вересня 2019 р.	
3	Аналіз інформаційного забезпечення	20 вересня 2019 р.	
5	Аналіз алгоритмічного забезпечення	25 вересня 2019 р.	
6	Розробка алгоритмічного забезпечення	15 жовтня 2019 р.	
7	Розробка програмного забезпечення	1 листопада 2019 р.	
8	Маркетинговий аналіз стартап-проекту	14 листопада 2019 р.	
9	Висновки	15 листопада 2019 р.	

Студент _____ Д.А. Кострикіна
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ О.П. Сирота
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У роботі розглянуто проблеми адаптації мобільних додатків для людей з обмеженими можливостям та тестування доступності інтерфейсу мобільних додатків.

У розділі опису предметної області розглянуто існуючі підходи для тестування мобільних додатків та виявлення питань динамічної й статичної доступності елементів користувацького інтерфейсу. В процесі було доведено доцільність введення нових окремих принципів опису доступності саме мобільних додатків, спираючись на існуючі стандарти.

Розділ аналізу проблем при реалізації автоматизованої системи тестування доступності присвячений опису загальних відомостей про прототип та розглянуто доцільність використання парадигм Model-View-Controller, а також опису існуючих інструментів для реалізації відповідного алгоритму.

У розділі розробки та реалізації автоматизованої системи тестування доступності мобільних додатків розглянуто архітектуру проектовано системи та деталізовано її налаштування, прикладного рівня та інфраструктурного рівня системи. Також визначено архітектуру інтерфейсу додатку та розроблені вимоги до системи, й обрано сервери для роботи.

У розділі маркетингового аналізу стартап-проекту проаналізовано поточну ситуацію на ринку, розроблено стратегії та маркетингові плани для впровадження даного рішення.

Ключові слова: мобільний додаток, доступність, тестування, алгоритм, користувацький інтерфейс.

Розмір пояснювальної записки – 125 аркушів, містить 35 ілюстрацій, 31 таблиці, 2 додатки (один з яких це графічний матеріал, що містить 4 плакати).

ABSTRACT

This paper deals with the problems of adapting mobile applications for people with disabilities and testing the accessibility of the mobile application interface.

The subject section describes existing approaches for testing mobile applications and identifying dynamic and static accessibility of UI elements. In the process, it was proved appropriate to introduce new separate principles for describing the availability of mobile applications, based on existing standards.

The section on analysis of problems with the implementation of an automated accessibility testing system is devoted to the description of the general information about the prototype and the expediency of using Model-View-Controller paradigms, as well as the description of existing tools for the implementation of the corresponding algorithm.

The section on development and implementation of automated system for testing the availability of mobile applications examines the architecture of the designed system and details its configuration, application level and infrastructure level of the system. It also defines the application interface architecture and system requirements, and selected servers to work with.

The Marketing Analysis section of the startup project analyzes the current market situation, develops strategies and marketing plans to implement this solution.

Keywords: mobile application, accessibility, testing, algorithm, user interface.

The size of the explanatory note is 125 sheets, contains 35 illustrations, 31 tables, 2 appendices (one of which is graphic material containing 4 posters).

Пояснювальна записка
до магістерської дисертації

на тему: *Автоматизована система для тестування доступності мобільних додатків*

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

UI (user interface) – Інтерфейс користувача;

WCAG (Web Content Accessibility Guidelines) – Правила доступу до веб-вмісту;

iOS (iPhone Operation System) – операційна система iPhone;

W3C (World Wide Web Consortium) – Всесвітній веб-консорціум;

IDE (Integrated Development Environment) – інтегроване середовище розробки;

MVC (Model-View-Controller) – Модель-вигляд-контролер або Модель-представлення-контролер;

JSON (JavaScript Object Notation) – позначення об’єкта JavaScript;

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертекст;

XML (eXtensible Markup Language) – розшиирювана мова розмітки;

UDL (Universal data link) – універсальне посилання даних;

SDK (software development kit) – комплект розробки програмного забезпечення.

ВСТУП

Мобільні пристрої стали основним методом спілкування та пошуку інформації для людей у сучасному світі. Великий відсоток людей, які живуть з обмеженими можливостями, важливо, щоб ці пристрої та програмне забезпечення, були доступними у використані для всіх. Хоча існують стандарти для забезпечення доступності в Інтернеті, для вітчизняних мобільних додатків досить рідко застосовуються будь-які з них. На сьогодні розробники поступаються доступністю задля збільшення функціональності додатків, які у своїй більшості орієнтовані на користувачів, що не мають фізичних обмежень. Такий компроміс не повинен існувати, але в частіше за все він викликаний нестачею ресурсів та програмних інструментів, які доступні розробникам. Для людей з обмеженими можливостями, такими як проблеми зі слухом, зором, руховими навичками і когнітивними процесами, використання смартфонів може бути складним завданням. Слідуючи рекомендаціям щодо забезпечення доступності і розробці додатків з урахуванням цих груп користувачів, додатки можна модифікувати, щоб зробити їх більш доступними. Проте, хоча мобільні додатки в основному охоплюються тими ж стандартами доступності, що і веб-додатки, форм-фактор і методи взаємодії, використовувані мобільними пристроями, створюють нові проблеми, які розробники повинні вирішувати під час розробки.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Об’єкт та предмет дослідження

Тестування доступності – це частина тестування зручності використання додатків або інших веб-ресурсів, в яких розглядаються користувачі які мають обмеження, що впливають на те, як вони використовують мобільний додаток. Кінцева мета, як в зручності використання, так і в доступності, полягає в тому, щоб з’ясувати, наскільки легко люди можуть використовувати додатки для смартфонів і передавати цю інформацію назад для поліпшення майбутніх конструкцій і реалізацій.

Оцінка доступності в цілому більш формалізована, ніж перевірка зручності використання. Закони та громадська думка засуджують дискримінацію людей з обмеженими можливостями. Щоб бути справедливим по відношенню до всіх, уряд та інші організації намагаються дотримуватися певних стандартів доступності веб-ресурсів, таких як Рекомендації W3C щодо забезпечення доступності веб-контенту (WCAG).

Тим не менш, важливо розрізняти відповідність стандарту і максимальну доступність мобільного додатка. В ідеалі, обидва будуть однаковими, але будь-який з існуючих стандартів може не відповідати одній із зазначених вимог:

- задовольняти потреби людей з обмеженими можливостями;
- збалансувати потреби людей з різними порушеннями;
- зіставити ці потреби з оптимальними методами;
- використати зрозумілу мову, щоб висловити потреби або методи.

Інвалідність створює особливі проблеми при визначенні того, наскільки просто використовувати продукт, тому що вони можуть створити додаткові прогалини в досвіді між користувачами і оцінювачами. Оцінка доступності повинна враховувати те, як себе поводить користувач з різними фізичними обмеженнями, а також різні незвичайні параметри конфігурації і спеціальне програмне забезпечення.

Рекомендації щодо забезпечення доступності і інструменти допомагають подолати ці прогалини в досвіді. Однак вони є доповненням, а не заміною емпатичних уяв, технічної винахідливості і спілкування з користувачами.

Приступаючи до тестування в кінці процесу розробки має два ризики:

1. Проекти мають тенденцію працювати понаднормово і понад бюджету. Через це тестування часто прискорюють, опускають або ігнорують.

2. Це великі за обсягом роботи, щоб виправити проблеми, виявлені на пізніх етапах процесу, ніж робити все з самого початку.

Таким чином, для забезпечення якості та економії часу і грошей, оцінка доступність повинна починатися на самому початку проектування продукту і включатися в наступні ітерації розробки аж до остаточної поставки.

Перш ніж приступити до оцінки проекту на предмет доступності, необхідно визначити, які ключові вимоги пред’являються до цього проекту, з огляду на його середу, передбачувану аудиторію і ресурси. Деякі вимоги будуть встановлені третіми сторонами, такими як уряду і клієнти.

1. Зовнішні вимоги.

Часто вимоги надходять із зовнішніх джерел, таких як:

− уряд. Зазвичай приймає форму загального законодавства проти дискримінації людей з обмеженими можливостями, замість того, щоб наказувати певний стандарт або перераховувати точні вимоги відповідності. Важливим винятком є випадок, коли законодавство встановлює певний стандарт для державного сектора. На сторінці політики WAI, що стосуються веб-доступності [1], подано неповний список аналогічних законів.

− політика клієнта. Наприклад, в даний час Shell намагається забезпечити відповідність своїх веб-сайтів рівню відповідності Double-A WCAG 1.0 [2].

2. Деталі відповідності.

Важливо отримати якомога більше ясності щодо зовнішніх вимог. Деякі стандарти доступності мають більше одного можливого рівня або типу відповідності, тому особливо важливо зафіксувати, що потрібно. Наприклад, WCAG 1.0 має три рівні відповідності:

− люди з обмеженими можливостями «вважатимуть неможливим доступ до інформації» в документі, який не проходить рівень «A»;

− людям з обмеженими можливостями «буде важко отримати доступ до інформації» в документі, який не проходить рівень «Double-A»;

− людям з обмеженими можливостями «буде складно отримати доступ до інформації» в документі, який не проходить рівень «Triple-A».

Проект WCAG 2.0 також має три рівні, але можливості відповідності більш складні. Якщо ресурс є частиною серії ресурсів, що представляють процес (наприклад, виявлення продукту, вибір, перевірка і підтвердження покупки для інтернет-магазину), рівень відповідності для всіх ресурсів в серії – це рівень ресурсу з найнижчим рівнем.

Заяви про відповідність повинні ґрунтуватися на технології контенту, підтримуваної доступністю. Щоб бути контент-технологією з підтримкою доступності, технологія повинна:

− були продемонстровані можливості роботи з допоміжними технологіями користувачів;

− мати користувацьких агентів (браузери, плагіни і т.д.), які працюють з допоміжними технологіями користувачів і доступні для користувачів з обмеженими можливостями безкоштовно, ніж для користувачів без обмежень.

3. Перевершення очікувань.

Визначення зовнішніх вимог повинно бути тільки початком процесу; до них слід ставитися як до мінімального набору вимог, до яких слід додати додаткові цілі для забезпечення максимальної доступності. Ціль людини, що

оцінює доступність – підняти додаткові проблеми доступності, оскільки вона є експертом в цій галузі.

4. Важливість користувацького інтерфейсу.

Необхідно звернути на особливу важливість забезпечення доступності для користувача інтерфейсу додатку. Навіть якщо контент недоступний у відповідній формі, доступний для користувача інтерфейс може допомогти користувачам визначити необхідний контент і звернутися за зовнішньою допомогою в перетворенні його в форму, яку вони можуть використовувати. Наприклад, слабочуюча людина може бути спрямована на відео-інструкцію.

− визначення користувачів з обмеженими можливостями;

Ідеальний підхід полягає в тому, щоб вбудувати ключові обмеження для проекту безпосередньо для інших користувачів: згенерованих користувачів, які діють як архетипи того, як певні типи користувачів будуть використовувати розроблюваний додаток.

− вибір стандарту доступності. Зробити вибір між існуючими міжнародними стандартами та стандартами чи законам, як діють в країні розробника впровадження додатку.

Отже, говорячи про тестування доступності, то слід зазначити, що існує дві групи, які проводять тестування – це експерти і користувачі.

Експертне тестування важливо, тому що експерти розуміють, як взаємодіють базові технології розробки мобільних додатків, та вони можуть виступати в якості центру обміну інформацією про різні групи користувачів і мають схильність вивчати спеціальні інструменти тестування.

Користувацьке тестування має вирішальне значення, тому що користувачі є справжніми експертами в своїх власних здібностях і власних допоміжних технологіях. Користувацьке тестування також може виявити прогалини в зручності використання між більш менш технічними користувачами, а також між людьми, які знайомі з даними додатком (самі експерти-тестувальники і нові користувачі).

Знання, отримані в ході призначеного для користувача тестування, повертаються в процес експертного тестування при наступному тестуванні (або в інший ітерації тестування в тому ж проекті, або в іншому проекті повністю). Гуманізація доступності та об’єднання розробників з кінцевими користувачами може підвищити мотивацію для створення доступних веб-сайтів.

Експертне тестування складається з чотирьох компонентів:

− оцінка під керівництвом інструменту: де інструмент шукає проблеми доступності і представляє їх оцінювачу;

− скринінг: де експерт імітує взаємодію з кінцевим користувачем додатку. часто не потрібно заглядати далеко, щоб знайти проблеми з доступністю;

− перевірка на основі інструменту: де оцінювач використовує інструмент для перевірки того, як різні частини мобільного додатку працюють разом;

− огляд коду: коли оцінювач дивиться прямо на код і ресурси веб-розробки, щоб знайти проблеми.

1.2 Стан вирішуваної проблеми в Україні

Насамперед люди з обмеженими можливостями обмежені у свої взаємодії з комп’ютерами та мобільними пристроями в цілому. Для того, щоб забезпечити безперешкодне використання пристрої та програмного забезпечення людям з різними вадами, усі додатки мають бути «доступними». Доступність програмного забезпечення передбачає реалізацію конкретних функцій, які допомагають людям у використанні технологій, що часто призводить до поліпшення зручності використання для всіх користувачів, у тому числі людей з обмеженими можливостями. Відповідно до статистичними даними Благодійного фонду допомого інвалідам України за 2013 рік, 2,8 мільйона українців (6,1% населення України) мають певний тип інвалідності, за оцінками, у 1,3 мільйона (%) діагностували важку

інвалідність [3]. Google, використовуючи дані Національного банку України та CDC Ukraine [4], виявив, що Україні є країною з найменшою кількістю людей з обмеженими можливостями в Європі (3,3% мають проблеми із зором та 0,7% людей з проблемами зі слухом), але не зважаючи на це Україна має одні з найменших показників «без бар'єрного середовища», забезпеченість мегаполісів мало забезпечені спеціальними світлофорами для людей з порушеннями зору чи/та слуху, не всі підземні переходи мають ескалатори для інвалідів, що вже казати про мобільні пристрої, комп'ютери та іншу техніку. З огляду на поширеність мобільних технологій та смартфонів у сучасному суспільстві, можна зробити висновок, що доступність цих пристроїв має першорядне значення.

В даний час розробники дотримуються Рекомендації щодо забезпечення доступності веб-контенту (WCAG) 2.0 від Консорціуму Всесвітнього павутиння (W3C), щоб забезпечити широкий спектр доступних функцій в межах свого програмного забезпечення [5]. Цей стандарт робить реалізацію доступності в додатках чіткішою й більш цілеспрямованою, присвоюючи рівні доступності та критерії успіху кожній рекомендації, яку він надає. Рекомендації WCAG 2.0 були прийняті урядами та організаціями по всьому світу, щоб допомогти створити доступні веб-сайти. Однак, оскільки ці вказівки були створені для Інтернету, вони недостатньо охоплюють усі аспекти мобільних додатків та інших веб-ресурсів. Робочі групи з W3C склали карту WCAG 2.0 до мобільної екосистеми та запропонували нові рекомендації щодо сенсорних екранів, але не створили згуртованих настанов, таких як WCAG 2.0 для мобільних пристроїв. Таким чином, не існує узгодженого стандарту для створення власних мобільних додатків, доступних для Інтернету.

Хоча мобільна доступність надзвичайно важлива, інструментів і середовищ тестування для розробки доступних додатків по суті не вистачає. Підручники і керівництва фрагментовані, бібліотеки для доступною розробки складні у використанні, а тестування доступності в реальних умовах

практично відсутня. Оскільки бар’єр для входу занадто високий, розробники часто уникають завдання щодо забезпечення доступності додатків. Час і знання, необхідні для створення доступного додатки, є стримуючим фактором для більшості розробників додатків. Більш того, багато інфраструктури тестування спеціальних можливостей пов’язані з проблемами статичної доступності (проблемами зі статичними знімками екранів додатків, такими як колірні контрасти і мітки програм читання з екрану), а також з проблемами динамічної доступності (проблемами з динамічним поведінкою додатки, такими як навігація) . і відповідь користувача) звертаються рідше.

Було проведено мало досліджень по розробці інструментів тестування доступності, за винятком формалізації різних вимог до дизайну мобільного додатка, які повинні бути доступні для користувачів з різними обмеженнями [6, с. 384-389]. Що стосується бібліотек і структур, які розробники використовують для тестування доступних додатків, то функціональність низькорівневого і простого тестування забезпечується операційною системою Android [7], а деякі інструменти, такі як Google Accessicистема Scanner, забезпечують тестування проблем статичної доступності. Однак ці інструменти складні у використанні, часто не універсальні для обробки певних типів додатків і не виявляють багато проблем доступності, які існують в додатках.

1.3 Огляд варіантів існуючих реалізацій для вирішення проблеми

Було проведено мало досліджень по розробці інструментів тестування доступності, за винятком формалізації різних вимог до дизайну мобільного додатка, які повинні бути доступні для користувачів з різними обмеженнями [8]. Що стосується бібліотек і структур, які розробники використовують для тестування доступних додатків, то функціональність низькорівневого й простого тестування забезпечується операційною системою Android [9], в той час як деякі інструменти, такі як Google Accessibility Scanner, забезпечують

тестування проблем статичної доступності. Однак ці інструменти складні у використанні, часто не універсальні для роботи з певними типами додатків і не виявляють багатьох проблем з доступністю, які існують в додатках.

1.3.1 Accessibility Linting Tools

При створенні додатків Android за допомогою Android Studio в середовищі IDE передбачені інструменти для виявлення деяких порушень спеціальних можливостей під час розробки, що дає безпосередній зворотній зв'язок з користувачем. Проте, базові інструменти для малювання в Android забезпечують перевірку тільки описів вмісту в уявленнях на основі зображень [10].

Наприклад, якщо зображення або значок не мають встановленого поля contentDescription, середа IDE видасть попередження (див. рис. 1.1). Як тільки поле contentDescription для зображення встановлено, програми читання з екрану, такі як TalkBack, можуть потім описати зміст цього зображення користувачу. Ці попередження служать для швидкої перевірки в режимі реального часу проблем доступності, коли розробник створює додаток.

Рисунок 1.1. Попередження про затримку в Android Studio щодо опису відсутнього вмісту в ImageViews

1.3.2 Android's AccessicистемаNodeInfo Object

Перевірка доступності може бути автоматизована шляхом написання тестів призначених для користувацького інтерфейсу додатків Android. Популярні бібліотеки, такі як Espresso, а також базове інструментальне

середовище тестування Android, дозволяють проводити тестування доступності за допомогою об’єкта AccessicистемаNodeInfo [11]. Цей клас надає функціональні можливості для доступу і маніпулювання інформацією інтерфейсу користувача, як текстові вводи, визначення, фокусування елемента, і перевірки, чи є елемент клікабельним.

Espresso – це автоматизована середа тестування користувальницького інтерфейсу для Android. Дозволяє перевірити, що усі contentDescriptions мають сенс, а перевіряє ситуації, коли contentDescription повинен динамічно змінюватися. Приклад динамічного contentDescription показаний нижче, з кнопкою Play/Pause. В ImageButton нет слів, тому програма читання з екрану не зможе передати її значення без contentDescription. Залежно від стану медіаплеєра кнопка буде або символом відтворення, або символом паузи. Залежно від зміни символу кнопки, повинен змінюватися й contentDescription.

Інформація, яка надається класом AccessicистемаNodeInfo, відмінно підходить для перевірки низькорівневих властивостей користувацького інтерфейсу, але розробник все одно повинен надати логіку для навігації по інтерфейсу, взаємодії з компонентами і використання інформації про ці компонентах для визначення доступності. Деяка робота була пророблена з тестуванням доступності Android за допомогою платформ тестування Espresso [12] і Robolectric [13], які використовують цей клас AccessicистемаNodeInfo. Проте, інформація про те, як ефективно використовувати ці бібліотеки, відсутня. Фактично, основний репозиторій для прикладів тестування доступності, званий «без очей», не оновлювався з 2015 року [14]. Правда полягає в тому, що ці інструменти рідко використовуються при тестуванні доступності, й інформація, яку вони надають для пошуку проблем, не так корисна, як могла б бути.

1.3.3 Google accessibility Scanner

Наприкінці 2018 року на Міжнародній конференції CSUN з технологій й осіб з обмеженими можливостями, група фахівців з доступності Google оголосила про випуск нового інструменту під назвою Google Accessibility Scanner. Цю програму можна безкоштовно завантажити, і це найпростіший спосіб швидко знайти проблеми з доступністю програми.

Завантаживши програму на тестовий пристрій, необхідно увімкнути сканер спеціальних можливостей в меню «Налаштування» → «Спеціальні можливості» → «Сканер спеціальних можливостей». Відкрити програму, яку необхідно протестувати, потім натиснути на плаваючу синю галочку.

Google Accessicистема Scanner – одне з найбільш багатофункціональних рішень для тестування доступності на Android. Замість оцінки доступності за допомогою статичних перевірок коду або запуску автоматичних тестів під час розробки, Google Accessibility Scanner тестує скомпільовані додатки, забезпечуючи оцінку доступності безпосередньо на пристрої [9].

Google Accessibility Scanner забезпечує відмінну функціональність тестування двома способами. По-перше, додаток надзвичайно простий у використанні як для розробників, так і для не-розробників. Якщо клацнути значок «Перевірка», щоб почати тестування відкритої програми, користувачеві надається інтерфейс, який висвічує проблеми доступності безпосередньо в інтерфейсі. Потім можна клацнути ці питання, щоб отримати більш детальну інформацію та пропозиції, і додаток підтримує відправку результатів тесту по електронній пошті і іншим службам загального доступу. Приклад такого використання в додатку Fitbit можна побачити на рис. 1.2.

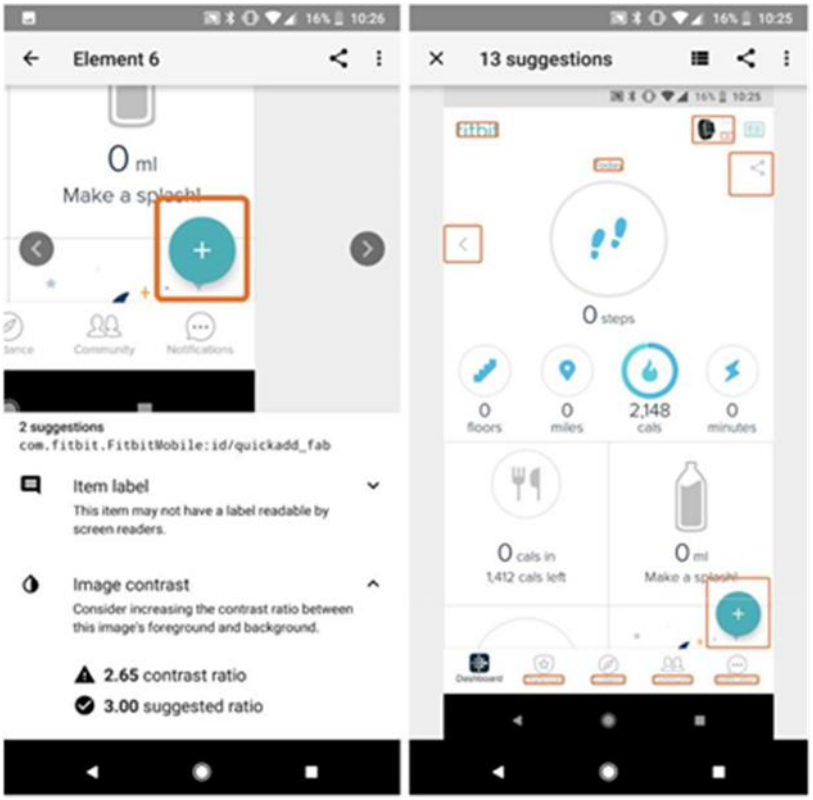


Рисунок 1.2. Приклад такого використання в додатку Fitbit

По-друге, додаток виявляє більше проблем, ніж стандартні інструменти для тестування і налаштування без зміни. Більш детально обговорюється в главі про оцінку. Сканер спеціальних можливостей Google здатний виявляти проблеми з контрастом, відсутні опису контенту, невеликі цілі торкання і інші проблеми статичного призначеного для користувача інтерфейсу. У той час як інші інструменти забезпечують поліпшену функціональність автоматизації, Google Accessibility Scanner пропонує найбільш повний набір функцій при мінімальних налаштуваннях.

Для ручного тестування необхідно увімкнути TalkBack, програму читання з екрану Android. Це допоможе орієнтуватися в додатку так само, як це роблять користувачі з вадами зору. Це також допоможе зрозуміти, як користувачі з обмеженою мобільністю орієнтуються в додатку; будь-які елементи, виділені зеленим полем в TalkBack, також доступні за допомогою d-pad або перемикача.

У той час як ручне тестування – кращий спосіб виконати тестування доступності, є деякі частини, які також можна автоматизувати. Одним з простих способів є модифікація Лінтера для збою збірки, якщо є будь-які ImageView або ImageButton елементи або відсутні елементи contentDescription.

1.3.4 VoiceOver

VoiceOver – це програма для читання з екрану на основі жестів, яка дозволяє використовувати iPhone, навіть якщо користувачь не бачить екран. Додаток дозволяє слухати опис всього, що відбувається на екрані, від рівня заряду батареї до того, хто дзвонить. Також можна налаштувати швидкість мовлення і висоту звуку відповідно до уподобань.

Оскільки VoiceOver інтегрований в iOS, він працює з усіма вбудованими додатками для iPhone. Можна створювати власні мітки для кнопок в будь-якому додатку, включаючи сторонні. Apple працює з спільнотою розробників iOS, щоб зробити ще більше додатків сумісними з VoiceOver.

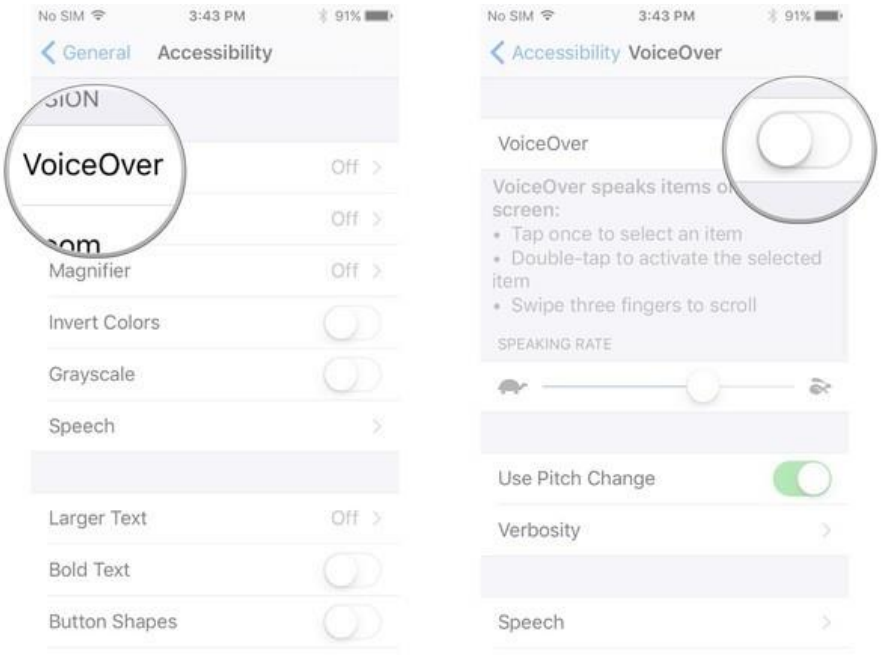


Рисунок 1.3. Робота додатку VoiceOver

Але цей додаток направлено на надання доступності для додатків, а не тестування вже існуючі додатки. Головним недоліком є мала кількість додатків, які можна адаптувати, а також цей додаток працює лише під операційною системою IOS.

1.4 Постановка задачі

Проблема з існуючими рішеннями для тестування доступності є двоякою; Налаштування ефективного тесту доступності вимагає часу, і існуючі рішення охоплюють тільки підмножину проблем доступності з мінімальною конфігурацією. Фактично, дослідження показують, що навіть стандартне автоматизоване тестування функціональності додатків і бізнес-логіки відсутня в 86% розробляються [15], і, в свою чергу, автоматичне тестування доступності також найчастіше відсутня в більшості додатків.

Якщо розробник вирішить провести тестування доступності, він виявить, що виявлення статичних проблем, таких як контрасти і розміри сенсорних цілей, можливо за допомогою автоматизованих середовищ тестування і сканера доступності Google. Однак тестування проблем динамічного доступу, таких як навігація за допомогою клавіатури і виявлення дій, заснованих на часі, займає набагато більше часу. Це означає, що хоча тестування доступності можливо при невеликій конфігурації, тестування для всього діапазону динамічного доступу

Проблеми часто не завершені. Багато з цих типів принципів деталізовані і потрібні у відповідності з керівними принципами, такими як WCAG 2.0, і вважаються важливими для підвищення доступності програмного забезпечення. Крім того, інтерфейси і методи звітності, що використовуються для передачі результатів доступності розробникам, можуть бути недостатніми або важкими для аналізу. Наприклад, деякі платформи повідомляють про одну й ту ж проблему доступності кілька разів, якщо елемент призначеного для користувача інтерфейсу з цією проблемою

повторюється з одного визначення, хоча в цьому випадку існує тільки одна справжня коренева проблема.

Мета розроблюваної системи полягає в тому, щоб подолати обидві ці перешкоди; забезпечити надзвичайно просту у використанні інфраструктуру тестування доступності, яка підтримує широкий спектр тестів доступності, але вимагає мінімальної конфігурації. Простота настройки і використання інфраструктури системи стимулюватиме розробників оцінювати свої додатки на предмет доступності та приводити до розробки більш доступних мобільних додатків. Розробити середу для тестування динамічної та статичної доступності на мобільних пристроях в цілому, яка визначала б зміни в контексті виключно за допомогою інформації про користувальницький інтерфейс, а також матиме автоматичну навігацію по додатку практично без участі розробника. Середа матиме алгоритм автоматичної генерації і скорочення простору станів призначеного для користувача інтерфейсу в контексті конкретного аналізу та процес тестування проблем динамічної доступності, таких як клавіатурна навігація і спонтанні зміни в контексті програми, шляхом створення кінцевого автомата, що представляє додаток.

Висновки до розділу

Проведено аналіз сфери доступності веб-додатків, а саме мобільних додатків для смартфонів, а також розвиток протоколів та стандартів доступності веб-сервісів та порівняно основні з них.

В розділі проаналізовані особливості існуючих рішень для тестування доступності в мобільних додатках й визначено загальні вимоги до проектовано системи

Досліджені основні недоліки сучасних автоматизованих систем доступності, які необхідно врахувати при проектуванні власної автоматизованої системи.

РОЗДІЛ 2. АНАЛІЗ ПРОБЛЕМ ПРИ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТЕСТУВАННЯ ДОСТУПНОСТІ

2.1 Загальні відомості про прототип

Будь то в контексті тестування доступності або інших способів аналізу, уявлення призначеного для користувача інтерфейсу надзвичайно важливо при виявленні різних аспектів призначеного UI. Подання досліджуваного додатку повинно виявити важливі чинники, які полегшують його аналіз.

У цьому розділі обговорюються деякі сучасні підходи до подання призначених для користувача інтерфейсів і подробиці, чому вони не підходять при тестуванні додатків на доступність. Щоб виправити деякі з цих проблем, я також представлю нову мову для опису користувацьких інтерфейсів в контексті тестування доступності.

Дотримуючись парадигмі Model-View-Controller (зазвичай використовується уявлення призначених для користувача інтерфейсів), стан призначеного для користувача інтерфейсу визначається його базовою моделлю, яка відображається через оновлення властивостей елемента користувальницького інтерфейсу, яке керується контролером. , Коли користувач взаємодіє з призначеним для користувача інтерфейсом, базова модель може змінитися, і, отже, будуть відображати інформацію для користувача, також може змінитися.

Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») – схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно.

− модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан;

− подання (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі;

− контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін [16].

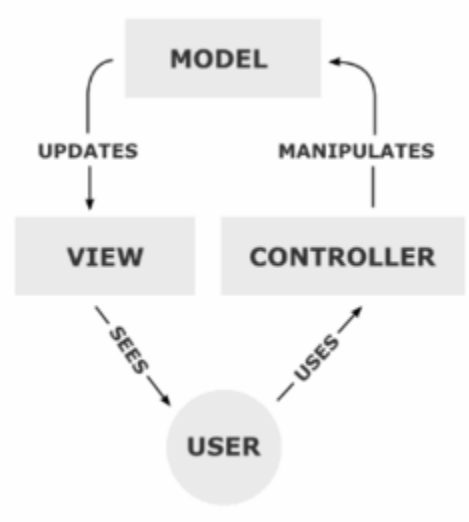


Рисунок 2.1. Зображення циклу Model-View-Controller

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (уявлення, виду). За рахунок такого поділу підвищується можливість повторного використання коду. Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах і/або з різних точок зору. Зокрема, виконуються наступні завдання:

До однієї моделі можна приєднати кілька видів, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми;

Не торкаючись реалізацію видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних) – для цього досить використовувати інший контролер;

Ряд розробників спеціалізується тільки в одній з областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому можливо добитися того, що програмісти, які займаються розробкою бізнес-

логіки (моделі), взагалі не будуть обізнані про те, яке уявлення буде використовуватися.

Концепція MVC дозволяє розділити модель, уявлення і контролер на три окремих компоненти:

Модель. Модель надає дані і методи роботи з ними: запити до бази даних, перевірка на коректність. Модель не залежить від уявлення (не знає як дані візуалізувати) і контролера (не має точок взаємодії з користувачем) просто надаючи доступ до даних і управління ними.

Модель будується таким чином, щоб відповідати на запити, змінюючи свій стан, при цьому може бути вбудовано повідомлення «спостерігачів». Модель, за рахунок незалежності від візуального представлення, може мати кілька різних уявлень для однієї «моделі».

Подання. Подання відповідає за отримання необхідних даних з моделі і відправляє їх користувачеві. Подання і не виконує жодних введені дані користувача [17].

Контролер. Контролер забезпечує «зв’язок» між користувачем і системою. Контролює і направляє дані від користувача до системи і навпаки. Використовує модель і уявлення для реалізації необхідного дії.

Оскільки MVC не має суворої реалізації, то реалізований він може бути по-різному. Немає загальноприйнятого визначення, де повинна розташовуватися бізнес-логіка. Вона може знаходитися як в контролері, так і в моделі. В останньому випадку, модель буде містити всі бізнес-об’єкти з усіма даними і функціями.

Деякі фреймворки жорстко задають де повинна розташовуватися бізнес-логіка, інші не мають таких правил.

Також не вказано, де повинна знаходитися перевірка введених користувачем даних. Проста валідація може зустрічатися навіть у поданні, але частіше вони зустрічаються в контролері чи моделлю.

Інтернаціоналізація і форматування даних також не має чітких вказівок по розташуванню.

Для реалізації схеми «Model-View-Controller» використовується досить велика кількість шаблонів проектування (в залежності від складності архітектурного рішення), основні з яких – «спостерігач», «стратегія», «компоновщик» [17].

Найбільш типова реалізація – в якій уявлення відокремлено від моделі шляхом встановлення між ними протоколу взаємодії, що використовує «апарат подій» (позначення «подіями» певних ситуацій, що виникають в ході виконання програми, – і розсилка повідомлень про них всім тим, хто підписався на отримання): при кожному особливому зміні внутрішніх даних в моделі (позначеному як «подія»), вона сповіщає про нього ті залежать від неї уявлення, які передплатили такого оповіщення – і уявлення оновлює я. Так використовується шаблон «спостерігач».

При обробці реакції користувача – уявлення вибирає, в залежності від реакції, потрібний контролер, який забезпечить ту чи іншу зв’язок з моделлю. Для цього використовується шаблон «стратегія», або замість цього може бути модифікація з використанням шаблону «команда».

Для можливості однотипного поводження з підоб’екти складно-складеного ієрархічного виду – може використовуватися шаблон «компоновщик». Крім того, можуть використовуватися і інші шаблони проектування – наприклад, «фабричний метод», який дозволить задати за замовчуванням тип контролера для відповідного виду.

2.2 Аналіз проблем прототипу та їх рішень

Проте, цей метод недостатньо точно представляє стан додатки з точки зору користувача. Базова модель або дані можуть відображатися для користувача за допомогою змін властивостей призначеного для користувача інтерфейсу, але користувач має обмежені знання про все базовому стані. Якщо частина даних зберігається в моделі для подальшого використання, і ці дані не відображаються в інтерфейсі, то стан призначеного для користувача інтерфейсу нічим не відрізняється від того ж призначеного для користувача

інтерфейсу без цієї частини прихованих даних (щонайменше, з точки зору користувача, за умови, що користувач не знає про ці приховані даних). Коли справа доходить до моделювання стану призначеного для користувача інтерфейсу через доступність, корисно відстежувати тільки те, що ми очікуємо від користувача – все, що фактично відображається в інтерфейсі. Спостерігаючи тільки цю інформацію, ми отримуємо більш точний опис того, на що користувач може відповісти.

«Модель» аспект цієї парадигми не єдина частина, яка зазнає невдачі при спробі перевірки доступності. Компонент «Перегляд» також не вистачає, особливо в мобільних додатках. Коли хтось обговорює призначений для користувача інтерфейс з точки зору його уявлень в Android, він найчастіше посилається на компоненти користувальницького інтерфейсу, які відображаються і відображаються на екрані – такі елементи, як кнопки, текст і зображення. Проте, аналіз тільки цих уявлень залишає велику частину можливостей всього призначеного для користувача інтерфейсу. Наприклад, кнопки на пристрої, що підключаються клавіатури, аудіовихід, фізичні перемикачі, гіроскопи і вібрації є прикладами компонентів введення і виведення призначеного для користувача інтерфейсу, які повністю ігноруються цими уявленнями. З огляду на обмеження, що накладаються цією виставою уявлення призначених для користувача інтерфейсів в Android, корисно визначити новий спосіб опису цих користувальницьких інтерфейсів.

Реалізація системи, яку буде обговорено в цьому розділі, зроблена для Android (хоча вона може бути розширена до iOS й інших платформ). Як видно з вищевикладеного, буде складно обговорювати призначені для користувача інтерфейси Android, що використовують тільки уявлення в якості базової термінології, і тому в розділі буде представлено нову термінологію. Для цілей роботи ця мова буде називатися Universal Design Language (UDL).

Поточний спосіб опису користувацьких інтерфейсів добре відомий дизайнерам і розробникам програмного забезпечення, найчастіше використовують такі концепції:

− подання, компоненти або елементи – основні об’єкти, з яких може складатися призначений для користувача інтерфейс. Ці об’єкти мають стилі і властивості;

− ієрархія і структура – організація цих уявлень, компонентів і елементів.;

− кінцеві автомати – спосіб представлення стану призначеного для користувача інтерфейсу в будь-який момент, який також визначає, як користувач може переходити з одного стану в інший.

Щоб вловити цю ідею елементів призначеного для користувача інтерфейсу, які не визначені виключно як елементи, які відображаються для відображення, а також щоб врахувати цю ідею, що призначений для користувача елемент визначається виключно на основі того, як користувач його інтерпретує, буде використано такі терміни:

Сприйняття (perceptifer) – це носій речей, які може сприймати користувач. Наприклад, кнопка – це сприймач; це компонент візуального інтерфейсу, який потенційно може сприйматися користувачем. Іншим прикладом є аудіокліп, оскільки він передає звуки, які потенційно може сприймати користувач. Користувач іноді може взаємодіяти з перцептиваторами.

Percept – предмет сприйняття, або щось, що сприймається. Сприймач буде передавати сприйняття уздовж пов’язаних з ними вихідних каналів на пристрої (тобто візуального каналу, такого як екран, аудіоканалу, такого як динамік, або рухового каналу, такого як двигун вібрації). Сприйняття сприймаються та обробляються користувачем, якщо той користувач здатний приймати це сприйняття по правильному вхідному каналу (наприклад, екран, якщо користувач не сліпий). Перцепція містить фактичну інформацію, яка дозволяє користувачеві міркувати про сприймач.

Літеральний інтерфейс – сукупність перцепторів і сприйняття, яке вони відображають, а також набір вихідних каналів, про які передаються ці сприйняття.

Користувач/Персона – фактичний агент, який взаємодіє з користувальницьким інтерфейсом, сприймаючи сприйняття та приймаючи рішення щодо взаємодії з користувальницьким інтерфейсом. У випадку розроблюваної системи персона є віртуальною та автоматизованою, приймаючи рішення, виходячи з сприйняття, яке вона споживає.

Внутрішній інтерфейс – внутрішнє розуміння та представлення буквального інтерфейсу, сприйнятого конкретним користувачем або персоналом. Наприклад, буквальний інтерфейс буде переведений на інший внутрішній інтерфейс для сліпого користувача порівняно з інтерфейсом для несліпого користувача.

Цільова мета цієї термінології – спростити перехід інформації з користувальницького інтерфейсу до користувача, охоплюючи при цьому всі аспекти користувальницького інтерфейсу, які не були б доступні виключно за допомогою обговорення Переглядів чи наданих компонентів, а також формалізації різниці між визначеним інтерфейсом і тим, що користувач може насправді зрозуміти або сприйняти з цього користувальницького інтерфейсу. Цей останній аспект буде надзвичайно важливим при врахуванні обмежень, які впливають на інтерпретацію користувачів та взаємодію з користувальницьким інтерфейсом.

Як приклад цієї термінології в дії розглянемо два приклади. Перший приклад, уявлення кнопки в UI. Кнопку можна описати цією термінологією наступним чином:

− button – це сприйняття (perceptifer);

− текст, який він передає, колір фону, колір тексту, колір рамки, радіус рамки, розмір шрифту, керування шрифтом, сімейство шрифтів, позиція на сторінці та аудіо, який можна прочитати на екрані, – це приклади предметів сприйняття;

− його сприйняття передається через візуальний канал (екран) і аудиоканал (програма читання з екрану і динамік).

Можна помітити, що цей перцептіфікатор кнопки не має такого сприйняття, як «дозволяє клікати». Це пов’язано з тим, що той факт, що кнопка забезпечує клікабельність, не є притаманним самому сприйняттю, а скоріше інтерпретується користувачем. Наприклад, якщо користувач раніше ніколи не використовував програмний інтерфейс, він може не розуміти, що на цю кнопку можна натиснути.

Однак в разі кнопки найбільш ймовірно припустити, що описані сприйняття (колір фону, кордону і т.д.) дозволять користувачеві інтерпретувати цей об’єкт як кнопку. Щоб спростити це поняття, дозволено визначати спеціальний тип сприйняття, званий віртуальним сприйняттям. Віртуальне сприйняття передає інформацію, таку як клікабельність, фокусованість та специфічний для Android тип сприйняття пристроїв (наприклад, TextView). Без віртуальних сприйняття персона повинна була б надати всю логіку, щоб інтерпретувати цей набір сприйнятть як кнопку – це просто ярлик, який слід використовувати з обережністю. Ідеально реалізована персона ігнорувала б все віртуальні сприйняття на користь інтерпретації всіх реальних сприйнятть.

Іншим прикладом такого використання термінології є звуковий запис, відтворений через динаміки пристрою, який описується в такий спосіб:

− audio clip – це сприйняттч;

− його предмет сприйняття передається через аудиоканал (динаміки пристрою);

− сприйняття, яке несе цей перцептор, є інформацією, такою як частота в часі і гучність аудіосигналу;

− віртуальне сприйняття – це такий елемент, як мовний контент аудіозапису.

Відзначимо, що така інформація, як слова, що міститься в аудіо- та ніколи не передається через сприйняття, оскільки сигнал може

інтерпретуватися по-різному для різних користувачів. Швидше, реалізація системи може забезпечити віртуальний перцептор в цих випадках для того, яким може бути читане людиною вміст цього звуку.

Тепер, коли ця термінологія існує, ми можемо почати описувати призначені для користувача інтерфейси, використовуючи базовий набір сприйнятть і віртуальних сприйняття. Основний список сприйняття, які можна відстежити в розроблюваній системі для даного сприймає пристрою:

− текст – візуалізований текст;

− місце розташування – піксельні координати верхнього лівого кута цього сприймає;

− розмір – загальна ширина і висота (або обмежувальна рамка);

− колір тексту – список кольорів, які використовуються в видимій частині тексті;

− колір фону – список кольорів, які використовуються в якості фону;

− виконання прокрутки – кількість прокручуваних пікселів, якщо цей перцептор є прокручувати елементом;

− альфа – рівень непрозорості всього сприймає;

− розмір шрифту – розмір відображуваного тексту;

− font family – тип відображуваного шрифту або сімейства шрифтів;

− font kerning – інтервал між символами в видимій частині тексті;

− міжрядковий інтервал – інтервал між рядками тексту, що відображається;

− стиль шрифту – індикатор тексту, виділеного жирним, курсивом, підкресленим або підкресленим;

− фізична кнопка – логічне значення, яке вказує, чи є ця кнопка фізичної кнопкою на пристрої, наприклад, регулятором гучності;

− сфокусованість – віртуальне сприйняття, яке описує, чи має цей перцептор в даний час фокус;

− аудіоконтент – віртуальне сприйняття, яке описує вміст аудиопотоку, наприклад слова або певні створені шуми;

− ім’я – віртуальне сприйняття, яке описує ім’я цього елемента користувальницького інтерфейсу для конкретної платформи;

− вміст програми читання з екрану – віртуальне сприйняття, що містить будь-який текст, який система передасть програмі читання з екрану для оголошення;

− isroot – віртуальне сприйняття, яке є просто логічним значенням, яке, якщо воно істинне, означає, що цей сприймає елемент є кореневим елементом всього призначеного для користувача інтерфейсу;

− clickable – віртуальне сприйняття, яке є логічним значенням, яке, якщо true, вказує, що цей компонент для користувача інтерфейсу може бути натиснуто;

− дочірні просторові відносини – дерево або порядок сприйнятть, що представляє ієрархію безпосередніх нащадків цього сприймає, якщо цей сприймає є контейнером інших сприймають.

Компонентів, що містяться в призначених для користувача інтерфейсів додатки, недостатньо для опису цього призначеного для користувача інтерфейсу. Додатки реагують і реагують на введення і, отже, змінюються з часом. Таке динамічне поведінку неймовірно важливо і необхідно для опису повної функціональності мобільного додатка. В контексті доступності цей аспект програми також необхідний для аналізу потенційного зручності використання для користувачів з руховими порушеннями і когнітивними порушеннями.

2.3 Вибір і обґрунтування оптимальності технічних рішень поставлених задач

Призначені для користувача інтерфейси і мобільні додатки є невід’ємною частиною стану і, як такі, часто описуються при створенні прототипів як кінцевий автомат або автомати. Наприклад, Sketch для Mac дозволяє дизайнерам створювати екрани для додатка, а потім робити їх інтерактивними, створюючи стрілки, які визначають руху між екранами при

натисканні елементів (див. рис. 2.2). У розробці для iOS розкадровка часто використовується для створення потоку логіки призначеного для користувача інтерфейсу, який по суті є автоматом, що описує різні стани додатки і то, як користувач може переходити між ними (див. рис. 2.3).

В рамках розроблюваної системи додатки також представлені як автомати і кінцеві автомати. Однак, щоб зробити систему надзвичайно простою у використанні, розробникам не потрібно визначати цей автомат заздалегідь. Швидше, інфраструктура надає персону, яка розумно орієнтується в інтерфейсі, створюючи уявлення додатки як автомата, що складається з станів, які містять колекцію перцепторов.

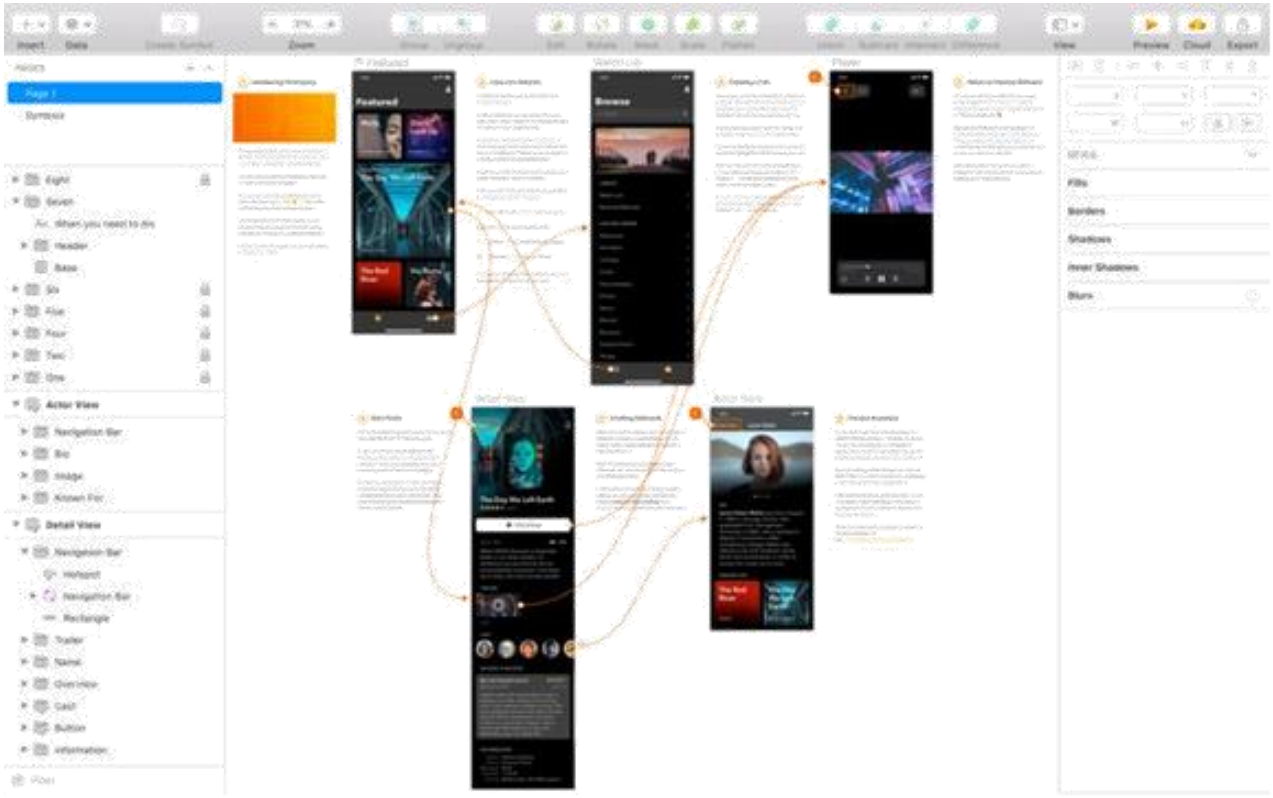


Рисунок 2.2. Прототип проекту програми

Рисунок 2.3. Розкадровка для додатка

При створенні кінцевого автомата для додатку на основі змодельованих дій користувача та інформації про користувальницький інтерфейс стає важко вирішити, що являє собою зміну стану призначеного для користувача інтерфейсу. Різні типи аналізу можуть вимагати відстеження різних властивостей в інтерфейсі. Наприклад, якщо конкретний екран прокручується вниз на один піксель, а видимі елементи не змінюються по змісту (змінюються лише їхні позиції на сторінці), це може не означати значимого зміни стану, але все ж є властивістю призначеного для користувача інтерфейсу (прогрес прокрутки) змінився.

З новою мовою розбиття призначеного для користувача інтерфейсу на сприйняття стає легко не тільки відслідковувати важливі частини призначеного для користувача інтерфейсу на основі типу аналізу, але також стає легко створювати хеш, який представляє призначений для користувача інтерфейс. Створення цього хеша призначеного для користувача інтерфейсу дозволяє нам вибірково вибирати, які аспекти призначеного для користувача інтерфейсу важливі, а також дозволяє Система швидко порівнювати два призначених для користувача інтерфейсу, щоб визначити, чи є вони «схожими» у цих варіантах.

Наприклад, в контексті доступності платформа система пропонує базовий набір сприйнятть для відстеження, які вважаються важливими для визначення стану програми. Сприйняття вибираються наступним чином: якщо сприйняття може впливати на статичну доступність додатка, воно включається в якості властивості при визначенні стану. Наступні сприйняття вибираються як ті, які можуть впливати на доступність програми: *ALPHA, BACKGROUND_COLOR, FONT_SIZE, FONT_STYLE, LINE_SPACING, VIRTUAL_NAME, VIR-TUAL_FOCUSABLE і TEXT_COLOR*. Цей вибір сприйняття по суті говорить про те, що зміни в стані не відбуваються, коли змінюється вміст або навіть змінюється кількість вмісту; скоріше, зміна стану відбувається, коли новий стан потенційно може вплинути на доступність додатка. Крім того, відстежуються віртуальні сприйняття, такі як порядок дочірніх елементів в контейнерах.

Розглянемо алгоритм, використовуючи приклад призначеного для користувача інтерфейсу на рис. 2.4. У верхньому рядку показано розбиття елементів призначеного для користувача інтерфейсу на контейнери і їх дочірні елементи аж до листя ієрархії уявлення. Нижній ряд показує «хеш» кожного елемента, параметризовані відстежувати уявленнями, як згадано вище.

Наприклад, в розділі, позначеному як «A», аналізований сприймає елемент являє собою ImageView, який хешіруется на основі імені уявлення (тобто ImageView) і рівня альфа (тобто непрозорість 1). Потім розділи, помічені як «B», хешіруются на основі відслідковується сприйняття кольору тексту, розміру шрифту, типу шрифту, кернинга, альфа-каналу і кольору фону. Будь-які текстові елементи в одних і тих же властивостях хешіруются з однаковим значенням (в даному випадку «B»). Розділ «C» є результатом хеширования одних і тих же типів сприйняття, причому цей хеш враховує стилі та розміри шрифтів, які відрізняються від хеш-функції «B».

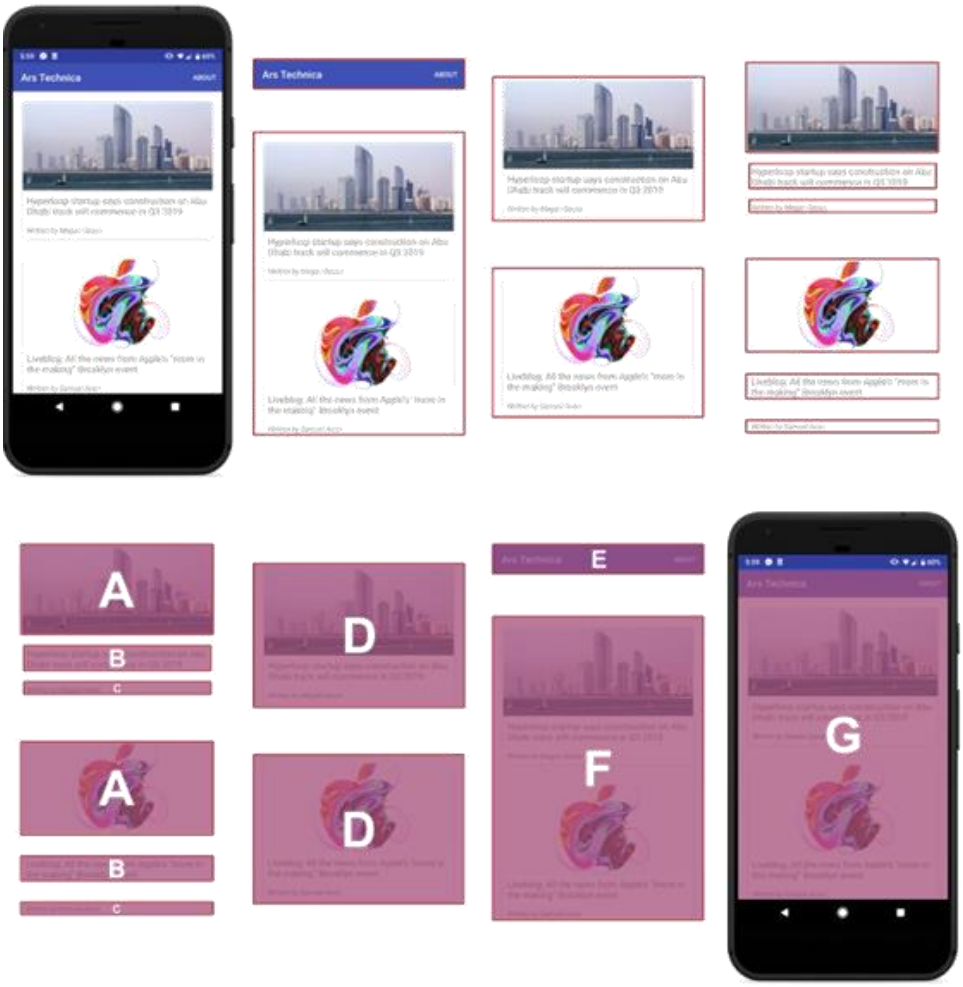


Рисунок 2.4. Розкладання призначеного для користувача інтерфейсу на частини, які представляють аналогічні компоненти, виявлені в інтерфейсі

Розділ «D» представляє хеш контейнера, який містить ImageView і TextView, представлені хешамі A, B і C. Хеш контейнера – це просто порядковий хеш хешей його дочірніх елементів, де визначається порядок на відстані від джерела контейнера до дочірнього елемента. Це хешування контейнера відбувається для кожного контейнера, причому хеш кореневого контейнера являє собою хеш всього призначеного для користувача інтерфейсу (в даному прикладі позначений як «G»).

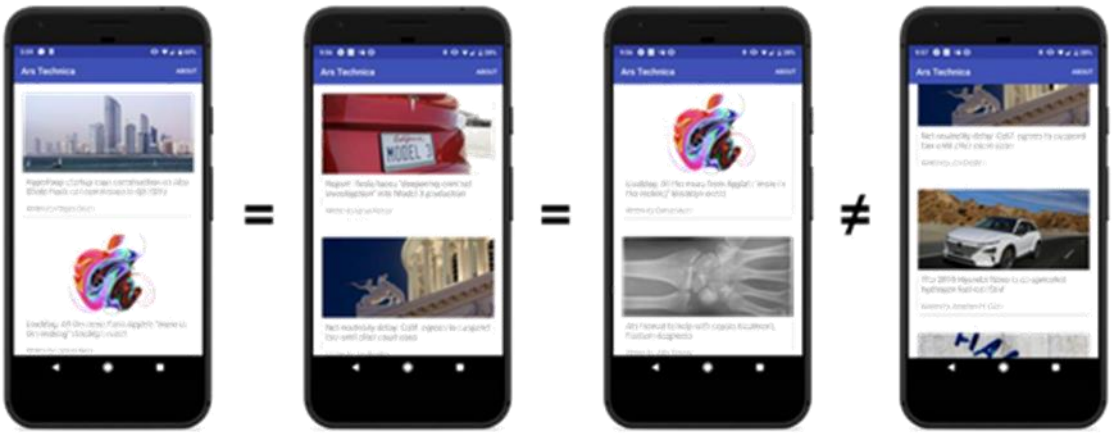


Рисунок 2.4. Перші три стану призначеного для користувача інтерфейсу рівні відповідно до базової технікою хешування

Обговорюваний вище алгоритм для хеширования призначеного для користувача інтерфейсу в єдине значення, що представляє його композицію важливих аспектів, спеціально викладено в алгоритмі.

Алгоритм 1: Хешування користувацького інтерфейсу

Функція getAccessicистемаHashes( ):

Дані: Колекція всіх перцепторів в прямому інтерфейсі.

Результат: Зіставлення хеш призначеного для користувача інтерфейсу зі списком ідентифікаторів сприймають, у яких є цей хеш ℎ, зіставлення ID ідентифікаторів сприймають з хешамі і зіставлення ідентифікаторів сприймають з сприймають.

← хеш-карта ідентифікатора сприйняття для сприймає;

← getRoot(*перцептіфікатори*); // getRoot повертає перцептіфікатор, найвищий в ієрархії View

ℎ ← ініціалізувати хешмап рядків до цілих чисел;

performPerceptiferHash(, , ℎ, ,); ← зворотний h, ланцюжки конфліктів;

повертає , ℎ, *and* ;

Функція performPerceptiferHash(, , ℎ , ,):

Дані є perceptifer, являє собою карту ідентифікаторів сприймають об’єктів для об’єктів perceptifer, h являє собою карту ідентифікаторів сприймають елементів для хеш perceptifer, є логічним значенням і являє собою функцію, яка приймає perceptifer і повертає набір сприйнять

Результат: Повертає хеш перцептора, а також рекурсивно заповнює h користувальницьким інтерфейсом хешування результатів усіх дочірніх перцепторів.

← transformFn( );

ℎ ← getIdsOfChildren( );

Якщо ℎ *не порожній* **тоді**

```
 ℎ        ℎ   ← карта  ℎ          в
| performPerceptiferHash( [  ℎ    ], ,     ℎ , , ); Якщо
| тоді
|
|  | ℎ       ℎ  ←  ℎ        ℎ  із видаленими дублікатами
| кінець
| ℎ  ℎ← Objects.hash( ℎ         ℎ  , ); // Визначається Java ℎ
| .put( .id, ℎ  ℎ);
| повертає ℎ  ℎ;
кінець
ℎ  ℎ← .hashCode(); // Визначається Java
ℎ .put( .id, ℎ  ℎ);
повертає ℎ  ℎ;
```

Зверніть увагу, що функція перетворення *transformFn* – це функція, яку можна використовувати для зміни того, які сприйняття враховуються при хешуванні. Наприклад, для хеша доступності згадані вище сприйняття, пов’язані з доступністю, повертаються, коли передається перцептор. Також зверніть увагу, що функція *getIdsOfChildren* приймає *perceptifer* і повертає впорядкований список ідентифікаторів його дочірніх елементів, зазначених віртуальним перцепт *CHILDREN_SPATIAL_RELATIONS*. Використовуючи карти, які повертаються *getAccessibilityHashes*, можна знайти запис для кореневого контейнера цього призначеного для користувача інтерфейсу, в якому зберігається хеш для цього призначеного для користувача інтерфейсу.

Здатність не обмежується виключно відстеженням сприйняття, які пов’язані з безпосередньою доступністю для користувача інтерфейсу. Компоненти відстеження, які впливають на доступність, є відмінним способом уникнути надмірного повідомлення про статичних проблеми доступності, і, хоча це хороша евристика для виявлення різних динамічних станів додатки, кожне застосування може мати певні стани, які необхідно відстежувати на основі різних типів. сприйняття.

2.4 Деталізація постановки задачі

Реалізувавши реєстратор станів таким чином, щоб розробнику було дуже легко розширювати і змінювати відстеження станів, що можна зробити декількома способами. По-перше, розробник може просто вказати, які базові сприйняття слід відстежувати. Наприклад, за замовчуванням текст не відстежується як дифференціатор стану, оскільки текст часто генерується динамічно. Якщо додаток розробника залежить від тексту відстеження (тобто мітки з написом «Так» або «Ні»), користувач може включити відстеження тексту.

По-друге, розробник може створювати свої власні сприйняття, специфічні для їх застосування. Наприклад, якщо базове стан даних надзвичайно важливо для відстеження стану користувача, розробник може додати віртуальне сприйняття до сприймаючого. Це віртуальне сприйняття може потім використовуватися для розрізнення станів.

По-третє, розробник може захотіти виконувати більш складні операції, такі як «відстежувати текст, якщо він говорить« так »або« ні », але в іншому випадку ігнорувати текст». Система також вирішує цю можливість, дозволяючи користувачеві перевизначати як функцію хешування, так і функцію перетворення сприйняття.

Метою платформи є виявлення більш широкого кола проблем доступності. Відстежуючи динамічні проблеми в інтерфейсі, ми повинні збалансувати компроміс між покриттям стану і вибухом простору станів. Мій запропонований настроюється підхід до відстеження станів в узагальненому вигляді дозволяє розробникам коригувати цей компроміс, при цьому пропоновані сприйняття доступності виступають в якості першого базового рівня в тестуванні доступності.

Висновки до розділу

В даному розділі проведено аналіз проблем при реалізації автоматизованої системи тестування доступності. Також описано загальні відомості про прототип та парадигми циклу Model-View-Controller. Проаналізовано проблеми прототипу та виявлено шляхи для їх рішень. Обрано і обґрунтовано оптимальності технічні рішення поставлених задач, та створено прототип проекту, а також його розкадровка для різних операційних систем. Наприкінці запропоновано та описано алгоритм хешування користувальницької інтерфейсу.

РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТЕСТУВАННЯ ДОСТУПНОСТІ МОБІЛЬНИХ ДОДАТКІВ

3.1 Архітектура проекту

Існуючі інструменти аналізу доступності (в першу чергу, Google Accessicистема Scanner і інструменти для малювання Android Studio) використовують пристрій Android або інтегроване середовище розробки. Оскільки середовище тестування Система фокусується на виявленні проблем доступності в динамічному режимі і в режимі реального часу, вона використовує як пристрій Android (емулювати або фізична), так і хост-комп'ютер (або сервер) для запуску тестів доступності.

Зовнішній сервер використовується для розвантаження обчислень, які в іншому випадку довелося б виконувати на мобільному пристрої під час тестування. Пристрій Android обробляє запити навігації від сервера і відправляє інформацію про інтерфейс користувача (у вигляді JSON) і знімки екрану (у вигляді PNG) назад на сервер для аналізу. Рис. 3.1 показує цю архітектуру більш докладно.

Для простоти розгортання сервер бази даних, сервер веб-сайту і сервер maven створюються і обслуговуються як контейнери Docker.

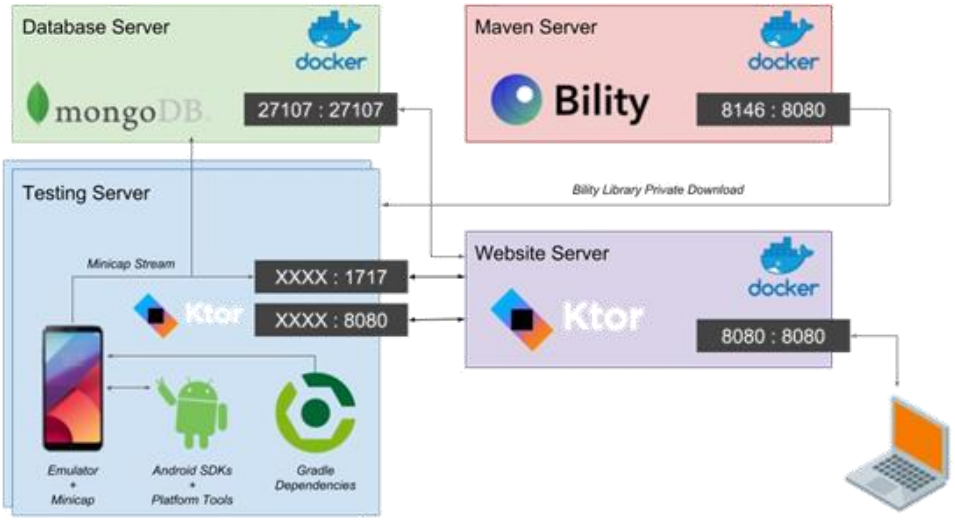


Рисунок 3.1. Архітектура запуску тестового сервера та тестових пристроїв

Проте сервер тестування керується та розгортається за допомогою сценарію, який можна виконати на будь-якій машині, що базується на Linux або Mac. Тестовий сервер не працює на Docker через обмеження роботи емульованих пристроїв Android на екземплярах Docker, викликаних апаратним відображенням [18].

3.1.1 Тестування сервера

Сервер тестування – це машина, яка управляє пристроєм Android. Поточна реалізація написана на Kotlin і Java, використовуючи Ktor1, оскільки це базова структура для HTTP-з'єднань. Цей комп'ютер також містить Android SDK і інструменти платформи, необхідні для взаємодії з пристроєм Android.

Android-пристрій може бути емулятором або реальним пристроєм. Він повинен бути підключений безпосередньо до тестового сервера, так як зв'язок відбувається через HTTP, а не через Android Device Bridge.

HTTP-сервер Ktor надає кінцеві точки для отримання інформації про інтерфейс користувача від пристрою, а також кінцеві точки для відправки на пристрій даних про командах і діях, якщо доступні нові дії. Цей процес зациклення можна знайти на рис..

Тестовий сервер також інформує розробника та користувача про результати тестування через декілька результатів. Для одного, він записує всі результати тестів на сервер бази даних - це сервер MongoDB, який працює в контейнері Docker. Тестовий сервер пропонує кінцеві точки для сервера веб-сайту (головний портал для розробника) для отримання тестової інформації. Нарешті, бібліотека Minicap [19] використовується для передачі відео з пристрою Android на сервер веб-сайтів, щоб користувач міг переглядати тестування в режимі реального часу в браузері.

Зауважимо, що тестовий сервер не обмежується наявністю однотонного екземпляра. Тестова основа підтримує можливість копіювання тестувального сервера для тестування на декількох пристроях та машинах

одночасно. Сервер веб-сайтів, в свою чергу, управляє наявністю цих машин. Це дозволяє протестувати доступність з декількома додатками та паралельно декількома конфігураціями.

3.1.2 Сервер веб-сайту

Сервер веб-сайту виступає в якості основного інтерфейсу, з яким взаємодіє розробник в процесі тестування. Один із принципів інфраструктури тестування Система – зробити тестування доступності максимально простим і інформативним для розробника. Як видно на рис. 3.2, інтерфейс інтуїтивно зрозумілий і простий. Користувач може легко завантажити проект Android і налаштувати тест. Параметри конфігурації включають в себе специфікації, такі як максимальна кількість дій, які може зробити персона. Персона також підтримує настройку типів дій, які вона може виконувати, типи проблем, які вона може виявити, і час очікування між діями (використовується для повільних пристроїв і мережевих підключень).

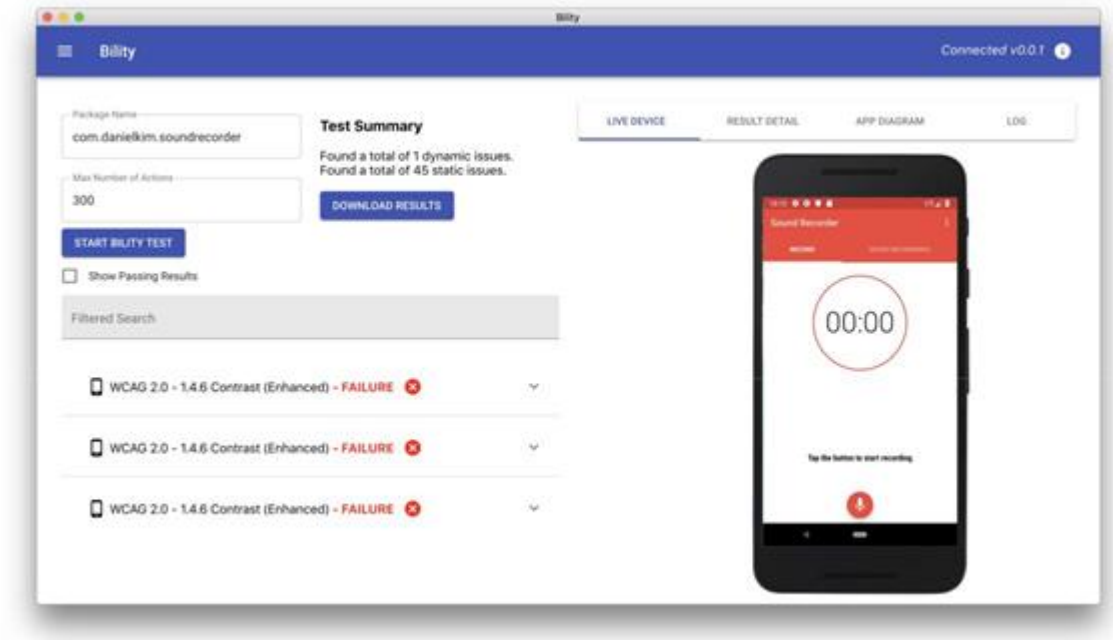


Рисунок 3.2. Додаток, що показує поточний тест зі звітами про проблеми в реальному часі

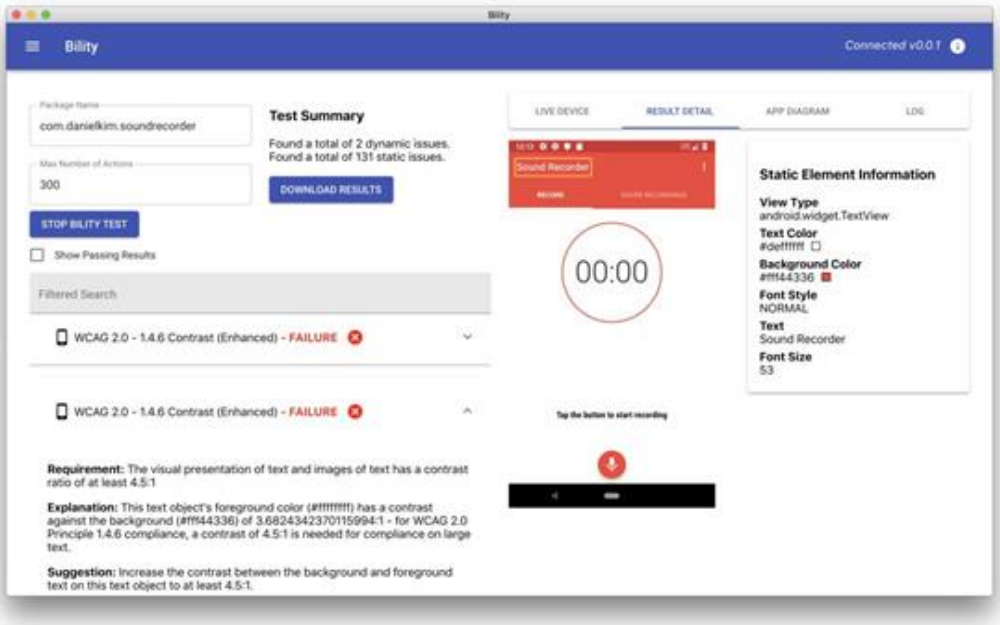


Рисунок 3.3. Проблема контрасту

Як тільки користувач запускає тест доступності, йому надається призначений для користувача інтерфейс, який дозволяє йому як дивитися тест, так і переглядати результати тестування доступності. Як видно на рис. 3.3, ці результати багаті інформацією – будь-яка знайдена проблема включає посилання на основну проблему доступності, детально описану в WCAG 2.0, пояснення, чому ця проблема була виявлена, підрахунок кількості випадків цієї проблеми знайшов, і навіть пропозиції про те, як вирішити цю проблему. Для довідки також показаний знімок екрана компонента користувальницького інтерфейсу, якщо такий є.

Інтерфейс також надає користувачеві можливість переглядати динамічні проблеми і інформацію про їх застосування. Наприклад, в лівій частині рис. 3.4 показана вкладка «Діаграма додатки», на якій представлена діаграма станів, що відображає стану і дії, що вживаються для переміщення між станами. З графіком можна взаємодіяти, масштабувати і включати інформацію в формі типів дій, предметів дій і знімків екрану.

Рисунок 3.4. Кінцевий автомат, що відображає екрани і дії, підтримувані цим додатком

Крім того, докладне уявлення про проблеми динамічного доступу показує зв'язок між двома станами. Наприклад, в правій частині рис.3.4 показаний детальний вид проблеми доступності навігації за допомогою клавіатури, що показує, що немає стану, яке може досягти стану призначення, використовуючи тільки клавіатуру.

Результати, представлені користувачеві в режимі реального часу, також зберігаються на сервері бази даних, що дозволяє отримувати цю інформацію після тестування. Крім того, користувач може завантажити результати тесту за допомогою кнопки «Завантажити результати», яка повертає об'єкт JSON з усіма знайденими проблемами і всіма параметрами, використаними для запуску тесту доступності.

3.1.3 Сервер Maven

Замість того щоб упевнитися, що на кожному сервері тестування встановлено останню версію код, доступний для бібліотеки Система, все залежно Java, необхідні для виконання тестів, знаходяться на окремому сервері Maven, розміщеному з використанням Artifactory. [20] Кожного разу, коли тестує сервер починає компілювати додаток для тестування, він

спочатку перевіряє наявність нових версій бібліотеки Система на цьому віддаленому сервері Maven, що дозволяє існуючим і працюючих тестових серверів бути в курсі всіх нових методів стеження за вадами. і типи результатів.

3.2 Опис інтерфейсу

Використовуючи архітектуру і концепції, описані в попередніх розділах, моя основна реалізація включає логіку і процеси для аналізу і оцінки доступності мобільних додатків. У цьому розділі я обговорюю конкретні алгоритми і правила для виявлення статичних і динамічних проблем доступності, а також алгоритм аналізу і виконання дій користувача на основі поточної і минулої інформації призначеного для користувача інтерфейсу.

3.2.1 Збір інформації про інтерфейс користувача

Пристрій Android відповідає за видалення інформації призначеного для користувача інтерфейсу і відправку її на сервер для обробки. Існує безліч методів, які можна використовувати для очищення призначеного для користувача інтерфейсу, наприклад, Android Device Bridge надає команди для вивантаження ієрархії уявлення і основних властивостей у вигляді XML [21]. Інший метод полягає в реалізації користувальницьких уявлень Android, які можуть реєструвати інформацію перегляду [22]. Бібліотека Система запускається з використанням інструменту інструментального тестування Android, який дозволяє отримати доступ до будь-яких активним в даний момент дій, вікнам і уявленням.

Рисунок 3.5. Цикл Observe-Process-React

Використовуючи відображення Java для доступу до всіх доступних вікон (повний код цієї функції можна знайти в додатку А.1), нам надається доступ до такої інформації, яка виявиться корисною при аналізі доступності:

– кореневі ViewGroup(и) цього призначеного для користувача інтерфейсу, надаючи всю інформацію View. Це включає в себе як коріння дій, контекстні меню і діалоги;

– які кнопки пристрою доступні;

– існування і функціонування різних інструментів доступності, таких як TalkBack.

З цією інформацією, в будь-який момент часу призначений для користувача інтерфейс Android може бути аналізується і перетворюється в його сприйняття і розлад сприйняття.

3.3 Спрощення станів

Після того, як інтерфейс був відправлений з пристрою Android на сервер тестування за допомогою запиту *POST методи*, згадані в розділі 2.3.2, використовуються для хешування буквальний інтерфейс в одне значення. Це єдине значення, а також скріншот призначений для користувача інтерфейс і посилання на буквальний інтерфейс зберігаються в автоматі. Об’єкт *CondensedState* містить всі ці посилання і властивості і реалізує функцію рівності на основі обчисленого хеша.

Коли *CondensedState* додається до кінцевого автомату, виконується попереднє дію, щоб вказати на цей стан. Якщо цей стан вже знайдено всередині кінцевого автомата (тобто хеш цього стану дорівнює існуючому стану), виконується попереднє дію, щоб вказати на це вихідне стислий стан.

Перед прийняттям рішення про наступний дії кінцевий автомат також оновлюється і включає нові «порожні дії». Виявляючи інтерактивні компоненти поточного призначеного для користувача інтерфейсу (тобто визначаючи, які елементи є натискати, що зчитуються і підтримують натискання клавіш), ребра поміщаються в кінцевий автомат, який представляє ці дії, кожне з яких вказує на нульовий стан. При визначенні того, які стану не досліджені, це просто перевірка, щоб визначити, які стану все ще мають ребра з нульовими адресатами.

3.4 Дії прийняття рішень

Як тільки поточний стан призначеного для користувача інтерфейсу було оброблено, кінцевий автомат оновлюється, щоб бути в правильному стані. В цей час персона може використовувати чотири різних підходи, щоб вирішити, яким має бути наступна дія. Система, за замовчуванням, використовує персону, яка намагається зробити всі можливі дії, щоб повністю досліджувати простір станів. По-перше, він може вирішити зробити дію *QUIT*, яке завершує тест. Ця дія виконується, коли або більше немає

станів з можливими діями, які потрібно спробувати, коли було виконано максимальну кількість дій, або коли було досягнуто цільовий стан.

Якщо не використовується дію *QUIT*, персона може прийняти рішення про виконання призначеного для користувача призначеного для користувача дії, якщо воно визначено. Ця опція корисна при первинному вивченні призначеного для користувача інтерфейсу, дозволяючи персонажу досягти більшості станів, перш ніж зважитися провести стрес-тест з діями, пов’язаними з доступністю. Наприклад, деякі програми можуть зажадати, щоб персона пройшла через екран входу в систему, який вимагає певних вхідних даних. Виконання призначеного для користувача дії визначається наявністю і відсутністю певних сприйнятть в поточному інтерфейсі (наприклад, виконати дію входу в систему, якщо присутні два текстових поля з вмістом «Ім’я користувача» і «Пароль»).

Якщо для користувача дію не вказано, персона автоматично спробує виконати одну з двох дій. Якщо з поточного стану доступний перехід, який ще не досліджений (тобто в поточному стані доступний порожній край), то автомат спробує виконати цю дію. Якщо недосліджений перехід недоступний з поточного стану, то персона буде шукати найближчий шлях до стану з недослідженим переходом, планувати шлях дій для досягнення цього стану, а потім робити це перша дія для переходу в цей стан, Цей пошук виконується з використанням алгоритму Дейкстри [23] , причому ваги ребер відповідають часу, який буде потрібний для виконання дій, які були виконані раніше.

Через характеру процесу спрощення стану, можливо, що стан ніколи не буде знову досяжно. У цих випадках розробник повинен визначити настроюється дію, щоб виявити це, і вживати відповідні заходи. В іншому випадку, персона виявить збої в досягненні стану призначення (за допомогою несподіваних обходів і повторних дій) і позначить це стан як недоступне. Ці відмічені стану можуть бути проігноровані або включені в остаточну тестування через конфігурацію тесту.

Як тільки тип дії визначено, параметри для цього дії генеруються при необхідності. Наприклад, для дії *SWIPE* приймається рішення про те, яку відстань повинна пройти персонажа і в якому напрямку повинно статися це рух. Визначаються функції, які генерують випадкові параметри для кожної дії (тобто, скільки потрібно провести пальцем, де всередині елемента, на який можна натискати, персонажу слід змоделювати клацання, яке натискання клавіші виконати і т.д.).

Якщо персона виявляється не в змозі виконати дію (наприклад, елемент призначеного для користувача інтерфейсу переміщається або зникає до того, як дія може мати місце, як у випадку деяких анімацій), персона повертається до випадкового дії. Незалежно від того, яку дію зроблено, дія зберігається як посилання при додаванні наступного стану в кінцевий автомат в формі об’єкта *UserAction* (який містить тип дії і параметри цієї дії).

3.4.1 Відповідь на запити дій

Одна ітерація циклу Observe-Process-React завершується відповіддю на рішення, прийняте бібліотекою Система, яке докладно описано вище. Кінцева точка /api/getNextAction надається сервером Система і діє як чергу, яку пристрій Android може запросити для дії. Як тільки дія запрошено і отримано пристроєм Android, змінна дії в черзі очищається. Якщо пристрій Android не читає нічого або нуль з цієї кінцевої точки, воно просто чекає і запитує на регулярному інтерв’ю, поки не буде надано чинність.

Після того як дію надано, бібліотека GSON використовується для приведення відповіді JSON в об’єкт *UserAction*. Як пояснювалося раніше, цей об’єкт містить тип дії, яку необхідно виконати, і параметри для завершення цієї дії, такі як час гортання або екранні координати. Система використовує об’єкт *UiDevice* для управління виконанням дії [24]. Наприклад, цей об’єкт надає наступні корисні методи:

− *swipe* – метод, який приймає початкову (x, y) координату, кінцеву (x, y) координату і кількість кроків. Робиться свайп, який починається з

початкової координати і закінчується у кінцевій координати, причому кількість кроків визначає, скільки часу має пройти свайп (кожен крок займає 5 мілісекунд). За замовчуванням Система використовує 100 кроків, що вказує на час натискання 1/2 секунди;

− *click* – метод, який приймає координати (x, y) і моделює клацання по цій позиції на екрані;

− *pressKeyCode* – метод, який приймає код ключа і моделює цей код ключа на пристрої.

Як тільки дія виконана, система чекає одну секунду, поки не прочитає наступну дію. З мого тестування часто буває достатньо часу, щоб пристрій міг реагувати на будь-який виконану дію. Цей цикл повторюється до тих пір, поки пристрій не отримає дію QUIT, яке завершує тест системи.

3.4.2 Перемикач стилю тексту

Як приклад використання цієї інфраструктури я розробив простий додаток з кількома проблемами доступності. Додаток називається «Перемикач стилів тексту» і являє собою простий інтерфейс з трьома кнопками і текстовим полем. Натискання кожної кнопки змінює фон і гарнітуру текстового поля. Зокрема, «Make Regular» дає текстовому вікна зелений фон і нормальний шрифт, кнопка «Make Italic» дає текстовому вікна синій фон з курсивом, а кнопка «Make Bold» дає текстовому вікна червоний фон. жирним шрифтом На рис. 3.5 показані три основних стану цього додатка. Зверніть увагу, що під час запуску програми стані з зеленим і звичайним текстовим полем шрифту.

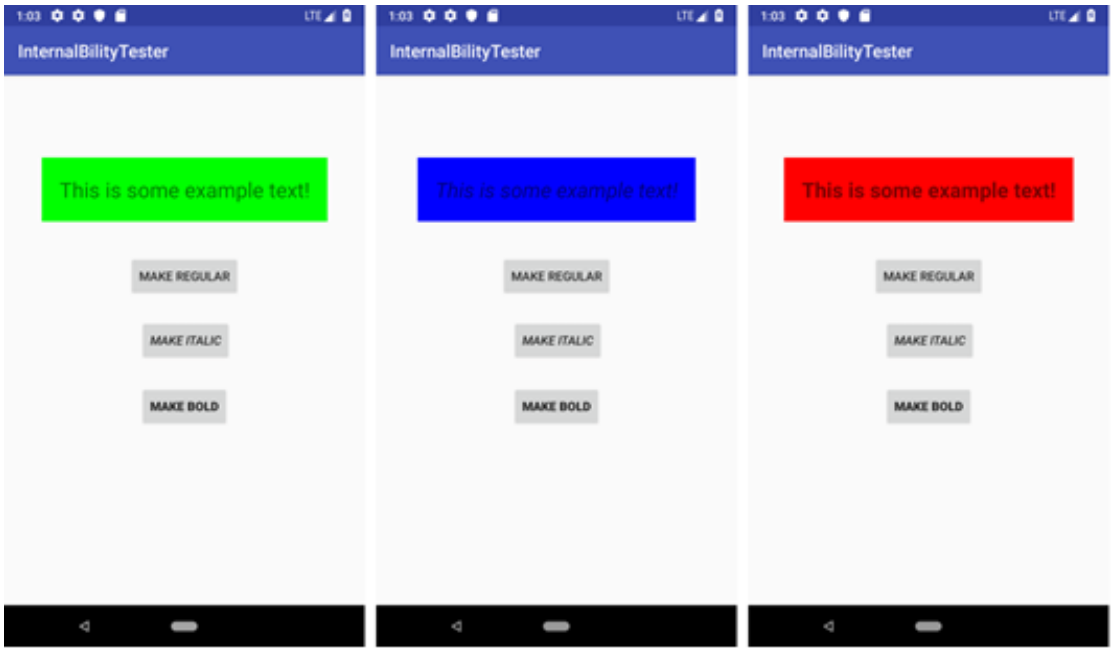


Рисунок 3.5. Три основні стани програми для переключення тексту, кожен з яких переходить до використання кнопок

У цій програмі є такі проблеми навмисної доступності:

– кожен колір (червоний, зелений синій) не відповідає мінімальному контрасту тексту, як вимагає WCAG 2.0 Принцип 1.4.3 [25];

– коли фокус надійде на першу кнопку (тобто ЗРОБИТИ РЕГУЛЯРНУ), намічена дія виконується автоматично. Іншими словами, текст зміниться на звичайний.

48 шрифту та зеленого фону, коли ця кнопка отримує фокус. Це порушення Принципу 3.2.1 WCAG 2.0, який вимагає, щоб жодна зміна контексту не відбувалася, коли компонент отримує фокус [25].

Слідуючи цикл Observe-Process-React, спостерігаємо ітерацію одного кроку для цього користувальницького інтерфейсу. По-перше, інтерфейс користувача перекладається на UDL, це означає, що видимий в даний час екран розбивається на його сприймачі. Коли відображається зелене поле з нормальним текстом, витяг розкладу видно нижче:

LiteralInterfaceMetadata(id=1d11527a-db62-4747-b2ad-f1d1fd3dd83d)

Perceptifer w/ ID e50d32b4-3d2d-4edf-888f-b36dae3d8986

(R) Percept(type=FONT_SIZE, info=49.0)

(R) Percept(type=TEXT, info=Make Bold)

(R) Percept(type=LINE_SPACING, info=57.0)

(R) Percept(type=LOCATION, info={left=542.0, top=1675.0})

(R) Percept(type=ALPHA, info=1.0)

(R) Percept(type=FONT_STYLE, info=BOLD)

(R) Percept(type=SIZE, info={height=168.0, width=355.0})

(R) Percept(type=TEXT_COLOR, info={colorHex=de000000})

(R) Percept(type=BACKGROUND_COLOR, info=Color(color=FFFFFFFF))

(V) Percept(type=VIRTUAL_NAME, info=android.widget.Button)

(V) Percept(type=VIRTUALLY_CLICKABLE, info=true)

(V) Percept(type=VIRTUAL_FOCUSABLE, info=false)

(V) Percept(type=VIRTUAL_IDENTIFIER, info=2.131165217E9)

Можна інтерпретувати з цього опису, показаний вище сприймач – це перцептор, що представляє кнопку «Bold». Його сприйняття складається із спостережуваних властивостей (R, що означає реально), таких як розмір шрифту, кольори та розташування на екрані, а також віртуальних сприймань (V), таких як назва об’єкта та додає він чи ні натисніть дію.

Тепер, коли цей інтерфейс був перетворений в буквальний інтерфейс всередині UDL, він може бути перетворений у конденсований стан. Використовуючи техніку хешування, кожен контейнер і кожен об’єкт хешують, в результаті чого загальний хеш для всього користувальницького інтерфейсу. У цьому випадку розмір шрифту, інтервал між рядками, альфа, стиль шрифту, колір тексту, колір тла та ім’я використовуються в хеші. Цей хеш завершується за допомогою стандартних методів хешування на Java, збираючи ці уявлення в набір, а потім хешуючи цей набір.

Після створення конденсованого стану за допомогою цих хешей персонал аналізує конденсований стан, щоб спочатку визначити, чи раніше ця програма була досягнута. Оскільки інтерфейс представлений як хеш, це

проста перевірка. У цьому прикладі це перший стан, що зіткнувся, і тому цей стан просто додається до автомата як стартовий стан. Однак, якщо було вжито дію для досягнення цього стану, то цей стан додається (або знайдеться) в автоматичному режимі, і здійснюється перехід з попереднього стану в цей поточний стан, використовуючи дію, яка була останньо зроблена.

Нарешті, персона вирішує, які дії є доступними та які дії вона хотіла б зробити. Використовуючи персонаж, кожне можливе натискання клавіш, сприймаючий клік та сприймач додається як потенційна дія з поточного стану, якщо перехід не було здійснено раніше. Ці дії легко знайти, оскільки людина просто фільтрує перцептори для правильного сприйняття *VIRTUAL_FOCUSABLE* та *SCROLL_PROGRESS*. Використовуючи алгоритм, описаний у попередньому розділі, тоді персона вибирає дію, яку слід здійснити з цього стану, в черзі на кінцеву точку /api/getNextAction. Після того, як бібліотека Android Система запитає на цю дію, виконується дія, аналізуючи тип дії та параметри дії від запиту (наприклад, якщо натиснути, ось координати для натискання). Потім цикл продовжується, аналізуючи новий інтерфейс користувача на UDL.

Після досягнення дії *QUIT* у цій програмі, графічне зображення цього користувальницького інтерфейсу зберігається. На рис. 3.6 показаний вихід для програми переключення тексту (без включення натискання клавіш, оскільки ця схема досить велика для друку). Ця state-машина генерується автоматично, не вводячи розробника; розробник просто вказує Система на їх застосування, запускає тест і чекає завершення тесту.

Рисунок 3.6. Проста машина стану, що представляє програму перемикання тексту

3.5 Цикл виявлення проблем доступності

Цикл Observe-Process-React найкраще намагається навігацію користувальницького інтерфейсу, який тестується. Під час цього процесу будується машина стану, що містить як статичну, так і динамічну інформацію про функціонування інтерфейсу користувача. Використовуючи цю інформацію, тепер можна виявити проблеми з доступністю.

Найпопулярнішим стандартом сьогодні для тестування доступності є WCAG 2.0 [26], який визначає правила задоволення доступності на веб-сайтах. Хоча створено для Інтернету, багато стандартів, викладених у WCAG 2.0, можна проаналізувати на мобільних пристроях.

Оскільки система є основою, на якій може відбуватися тестування доступності, я застосував кілька принципів WCAG на вершині системи як приклад його універсальності та корисності. Зокрема, Система включає реалізацію для виявлення наступних статичних та динамічних проблем, визначених принципами WCAG [25]:

Статичні питання:

− нетекстовий вміст: Увесь нетекстовий вміст, який представлений користувачеві, має текстову альтернативу, яка відповідає рівнозначному призначенню, за винятком кількох винятків;

– контрастність (мінімум): Візуальне представлення тексту та зображень тексту має коефіцієнт контрасту принаймні 4,5: 1, за винятком великого тексту, який вимагає контрасту принаймні 3: 1. (Рівень AA);

– контрастність (посилена): Візуальне представлення тексту та зображень тексту має контрастне співвідношення щонайменше 7: 1, за винятком великого тексту, який вимагає контрасту принаймні 4,5: 1. (Рівень AAA);

– розмір цілі: розмір цілі для входів вказівника не менше 44 на 44 пікселів CSS. (Рівень AAA, зауважте, що це частина WCAG 2.1).

Динамічні питання:

Клавіатура: Усі функціональні можливості вмісту функціонують через інтерфейс клавіатури, не вимагаючи конкретних синхронізувань для окремих натискань клавіш, за винятком випадків, коли основна функція вимагає введення даних, що залежить від шляху руху користувача, а не лише від кінцевих точок. (Рівень A)

Про фокусування: Коли будь-який компонент отримує фокус, він не ініціює зміну контексту. (Рівень A)

Зміни під час запиту: Зміни контексту ініціюються лише за запитом користувача або є механізм для вимкнення таких змін. (Рівень AAA).

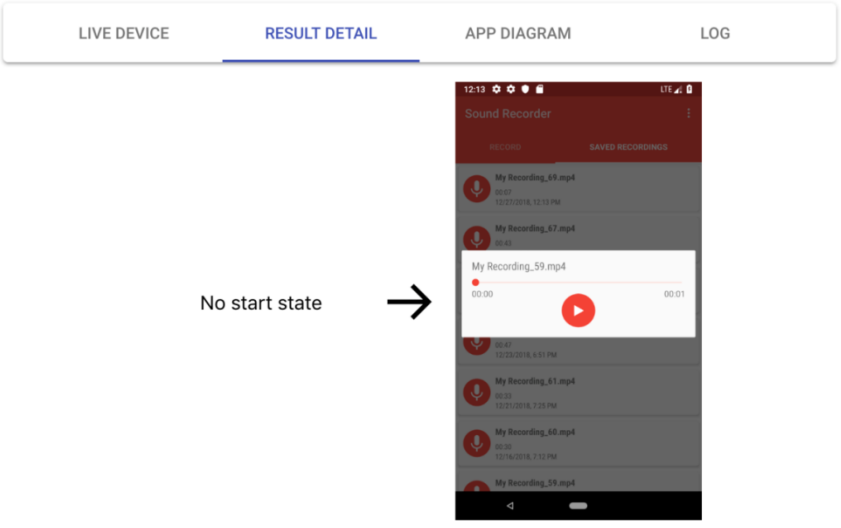


Рисунок 3.7. Статичні питання доступності

3.6 Використані методи виявлення проблем доступності

3.6.1 Відсутні текстові альтернативи

Поширена методика підвищення доступності програми вимагає встановлення текстових альтернатив для нетекстових елементів. Наприклад, кожне зображення в інтерфейсі користувача, будь то значок чи фотографія, повинно мати певну текстову альтернативу, яка пояснює або значення цього значка, або вміст цього зображення.

У Android це робиться шляхом встановлення опису вмісту на зображеннях та інших об’єктах View. Оскільки цей вміст зазвичай не сприймається під час прямої взаємодії з користувальницьким інтерфейсом, є два різні варіанти виявлення цих описів вмісту – увімкнення TalkBack і запис звукових сприймань або зчитування описів вмісту та запис віртуальних уявлень, що представляють вміст, про який можна говорити Відповісти. Останній варіант швидше і простіше виявити на практиці, і як така Система використовує віртуальне сприйняття *VIRTUAL_SCREEN_READER_CONTENT* для виявлення таких проблем.

З огляду на перцептор, процес відбувається наступним чином:

− якщо цей сприймач містить *TEXT* сприймання, а текст не є порожнім рядком, повідомте про відсутність проблеми. Якщо рядок порожній, повідомте про проблему, якщо не знайдено непустого сприйняття *VIRTUAL_SCREEN_READER_CONTENT*;

− якщо цей сприймач містить сприйняття *MEDIA_TYPE* типу IMAGE, і не знайдено порожнього *VIRTUAL_SCREEN_READER_CONTENT* сприйняття, повідомте про проблему;

− якщо цей сприймач має невідчутне сприйняття, не повідомляйте про проблему. Невидимий вказує, що цей елемент є або контейнером, або декоративним.

За допомогою цієї логіки можна перевірити, чи доступна програма стосовно WCAG.

У випадку, коли видимий текст доступний для зчитування з екрана зчитувача, тоді про жодну проблему не повідомляється. Якщо текстовий вміст будь-якого, що слід прочитати на екрані зчитувача, є порожнім рядком, то зчитувач екрана може нічого не повідомити, що означає, що цей елемент повинен повідомлятися про помилку. Нарешті, якщо елементом є зображення або недекоративний елемент без тексту, повинен бути доступний опис зчитування не порожнього екрана.

На додаток до цієї логіки також робиться швидка перевірка, щоб визначити, чи всі сприйняття вмісту зчитувача екрана в екземплярі буквального інтерфейсу різні. Якщо на одному екрані виявлено сприймання вмісту з численним екраном з одним і тим же текстом, повідомляється про проблему, оскільки ці елементи не можуть бути диференційовані сліпим користувачем.

3.6.2 Контраст тексту та піктограм

Контрастність обчислюється шляхом обчислення відносної яскравості кольору переднього плану до кольору фону. Реалізація Android UDL забезпечує сприйняття кольорів тексту для текстового вмісту, сприйняття кольорів тла для всіх представлень, здатних утримувати колір тла, та сприйняття кольорів переднього плану для зображень (наприклад, піктограм).

З огляду на перцептив, для отримання сприйняття типу *TEXT_COLOR*, *FOREGROUND_COLOR* та *BACKGROUND_COLOR* використовується функція помічника *getPerceptsOfType*, яка приймає тип сприйняття та перцептив. Колір тексту використовується як передній план, якщо він є, інакше використовується колір переднього плану.

Далі обчислюється яскравість кожного кольору [27], визначена наступними формулами, з урахуванням значення RGB, коли кожен кольоровий компонент знаходиться в діапазоні від 0 до 255:

$$
\begin{aligned}
&= \quad \textit{component normalized from 0 to 1} \\
&= \quad \textit{component normalized from 0 to 1} \\
&= \textit{component normalized from 0 to 1} \\
&= \quad \frac{}{12.92}\ \textit{if} \quad <= 0.03928\ \textit{else}\ \left(\frac{+\,0.055}{1.055}\right)^{2.4} \\
&= \quad \frac{}{12.92}\ \textit{if} \quad <= 0.03928\ \textit{else}\ \left(\frac{+\,0.055}{1.055}\right)^{2.4} \\
&= \quad \frac{}{12.92}\ \textit{if} \quad <= 0.03928\ \textit{else}\ \left(\frac{+\,0.055}{1.055}\right)^{2.4} \\
&= \quad 0.2126 \quad + 0.7152 \quad + 0.0722
\end{aligned}
\tag{1}
$$

Ці формули виводяться з інтенсивності, з якою кожен колір сприймається людським оком.

Після обчислення яскравості фонового кольору () та яскравості кольору переднього плану () контраст обчислюється наступним чином:

$$= \frac{+\,0.05}{+\,0.05}\ if \quad > \quad else\ \frac{+\,0.05}{+\,0.05} \tag{2}$$

Отримане значення може бути в діапазоні від 1 до 21, яке тепер можна використовувати для оцінки набору кольорів за критеріями контрасту. Наприклад, WCAG 2.0 Принцип 1.4.3 вимагає, щоб контраст між переднім планом та фоном був не менше 4,5: 1 (тобто> = 4,5). Хоча Принцип 1.4.6 WCAG 2.0 вимагає, щоб контраст між переднім планом та фоном був не менше 7: 1 (тобто> = 7). Однак якщо колір переднього плану є текстом, а текст вважається великим (тобто шрифт 18 точок або жирний шрифт 14 точок), потрібно контрастне співвідношення лише 3 та 4,5 відповідно. Іншими словами, як тільки кольори фону та переднього плану визначаються, визначати контрастні ситуації, які спричиняють недоступність, досить просто.

Однак обчислення кольорів переднього плану та фонових елементів в інтерфейсі користувача не завжди є простим. Прозорість кольорів може вимагати знання основних елементів і кольорів, а кольори можуть бути не суцільними або суцільними (у випадку градієнтів). WCAG 2.0 вимагає чіткого оголошення кольорів, ігнорування питань прозорості [28]. Однак

можна вирішити деякі ці проблеми шляхом змішування кольорів переднього плану та фонових кольорів за допомогою методів альфа-змішування [29]. Припускаючи білий базовий колір фону, функція *blendColors* приймає стопку кольорів від фону до переднього плану, повертаючи значення кольору, яке дорівнює реальному композитному кольорові після врахування ефектів кожного альфа-каналу (повний код для цієї функції можна знайти в додатку як лістинг А.3).

Як показано на рис. 3.8, Система знайшов проблему доступності контрасту, посилаючись на принцип WCAG 2.0 1.4.3. Наведіть курсор на предмет проблеми, розроблювана система представляє знімок екрана, про який йдеться, а також виділену межу навколо порушення. Наведіть курсор на цей компонент інтерфейсу, ми побачимо, що це TextView з кольором тексту # 8a000000 та кольором тла # ff0000ff. Система також представляє кілька відомостей про властивості шрифту цього тексту.

Випуск повідомляє про колір переднього плану # ff000075, який на тлі має розрахунковий контраст приблизно 1,94. Це менше, ніж потрібна контрастність 3: 1 для великого тексту (який цей TextView кваліфікується як), і тому повідомляється про проблему, надаючи інформацію та ресурси, які розробник може використовувати для пошуку та виправлення проблеми.

Рисунок 3.7. Приклад порушення WCAG 2.0 принципу

Подібно до Принципу 1.4.3,система знайде та повідомить про подібні порушення для Принципу 1.4.6, який вимагає вищих контрастів для задоволення рівня AA.

Якщо розмір сенсорної цілі елемента інтерфейсу користувача дуже малий, у користувача можуть виникнути проблеми з взаємодією з ним, якщо він має інвалідність двигуна, як, наприклад, Паркінсон [30]. WCAG 2.1 вирішує це, вимагаючи сенсорного розміру цілі принаймні 44 пікселів на 44 пікселі для всіх елементів, на які можна натиснути. У рамках Система це так само просто, як і фільтрація для всіх сприймачів, які мають VIRTUALLY_CLICKABLE сприйняття, а потім стверджувати, що кожен з цих елементів має SIZE сприймання розміром не менше 44 пікселів на 44 пікселі. Якщо це не так, то повідомляється про проблему. Наприклад, див. Рисунок 4-5 для прикладу цього виявлення.

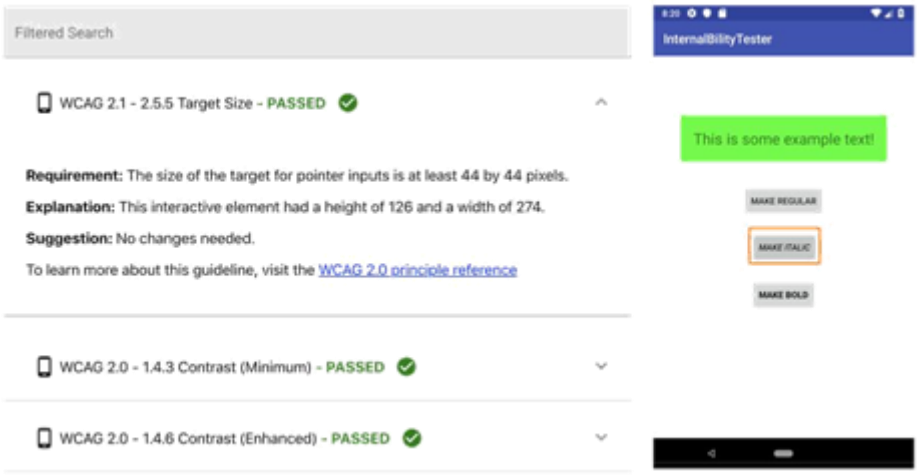


Рисунок 3.9 – Приклад принципу WCAG 2.5.5

3.7 Питання динамічної доступності

У той час як більшість мобільних пристроїв взаємодіють із сенсорним екраном, деякі користувачі з руховими порушеннями взаємодіють із своїми пристроями за допомогою клавіатур та комутаторів. Визначення того, чи інтерфейс є навігаційним та функціональним, використовуючи лише

пристрій, схожий на клавіатуру, є важливим для цих користувачів. Принцип 2.1.1 WCAG 2.0 вимагає, щоб користувальницькі інтерфейси були повністю функціональними, використовуючи лише клавіатуру (якщо тільки для введення не потрібно введення, не клавіатура) [25].

Запевнивши, що користувальницький інтерфейс можна керувати за допомогою лише клавіатури, можна зробити, видаливши з автомата крайки, які не є натисканнями клавіатури, а потім переконавшись, що всі стани все ще доступні, використовуючи лише ці дії. Це по суті означає, що для повного переходу по користувальницькому інтерфейсу можна використовувати лише натискання клавіатури. Якщо є оригінальним автоматом і є автомат зі всіма видаленими ребрами, які не є взаємодією клавіатури, то повинно бути істинним: якщо стан в досяжному від стану, то стан повинен бути доступним зі стану в автоматі через деякий шлях. Простіше кажучи, це означає, що якщо користувач може перейти з одного стану в інший, використовуючи типові методи взаємодії, він також повинен мати можливість потрапляти між тими ж станами за допомогою клавіатури.

Є одне виняток із цього правила, оскільки елементи користувальницького інтерфейсу зосереджуються під час використання клавіатури та комутатора. У випадку введення з клавіатури, у користувальницького інтерфейсу часто не буде сфокусованих елементів до того, як буде надісланий будь-який вклад з клавіатури. Після використання клавіатурного введення в деякий перцептивний інтерфейс користувача буде фокусувати включений сприймач. Після введення цього стану переміщення фокусу між елементами інтерфейсу користувача неможливо видалити цей фокус за допомогою натискання клавіш (тобто зазвичай єдиний спосіб видалити цей фокус – це натиснути або провести інтерфейс). Оскільки реалізоване хешування користувацьких інтерфейсів відстежує цілеспрямовані елементи, це означає, що стан машини, який видаляє всі краї, крім клавіш натискання, може мати стан, який здається "недосяжним", який насправді є лише недосяжним через те, що вони не орієнтовані елементів. У

випадку, якщо стан не може досягти стану в автоматиці, але шлях існує в автоматі, спочатку перевіряємо, що h $(h)!$ = h (h), де h h – хеш-функція, аналогічна хеш-функції доступності, представлена рано, але виключає відстеження зосереджених елементів. Якщо і є рівними за цією мірою, то вони просто різняться через зосереджений елемент, і доступність клавіатури ігнорується (інакше повідомляється про цей недолік шляху від до).

Приклад такого виявлення ми можемо побачити на рис. 3.10, який представляє додаток з новою модифікацією. У цій версії видалено логіку, коли фокусування на "звичайній" кнопці змінило б тип тексту та фон, і замість цього відключила можливість зосередитись на "жирній" кнопці. У вершині машини побачите, що кожен стан доступний від кожного іншого за допомогою декількох серій клацань або натискань клавіш. State-машина знизу видалила всі дії клацання і залишила лише натискання клавіш, такі як Вліво, Вправо, Вгору, Вниз, ТАБ та ВВІД. Ця state-машина виявляє, що не кожна держава доступна від будь-якої іншої держави. По-перше, існує зелений, червоний і синій стан, у якого всі краї не входять. Як вже згадувалося раніше, ці стани – це стани, які представляють стан "до введення фокусу на клавіатурі". Вони виявляються за допомогою логіки, згаданої вище, і ігноруються як проблеми. Однак у клацання червоних станів у нижній лівій частині немає ребер, що надходять із групи синіх та зелених станів, що означає, що логіка жирної кнопки ніколи не може бути застосована при запуску із синього чи зеленого стану. Використовуючи такий алгоритм, як Алгоритм Флойда-Варшалла [31], Розроблювана система може виявити цю відсутність шляху від синього та зеленого станів до червоних станів, і повідомить про кожну пару вихідного призначення як про порушення принципу WCAG 2.1.1 .

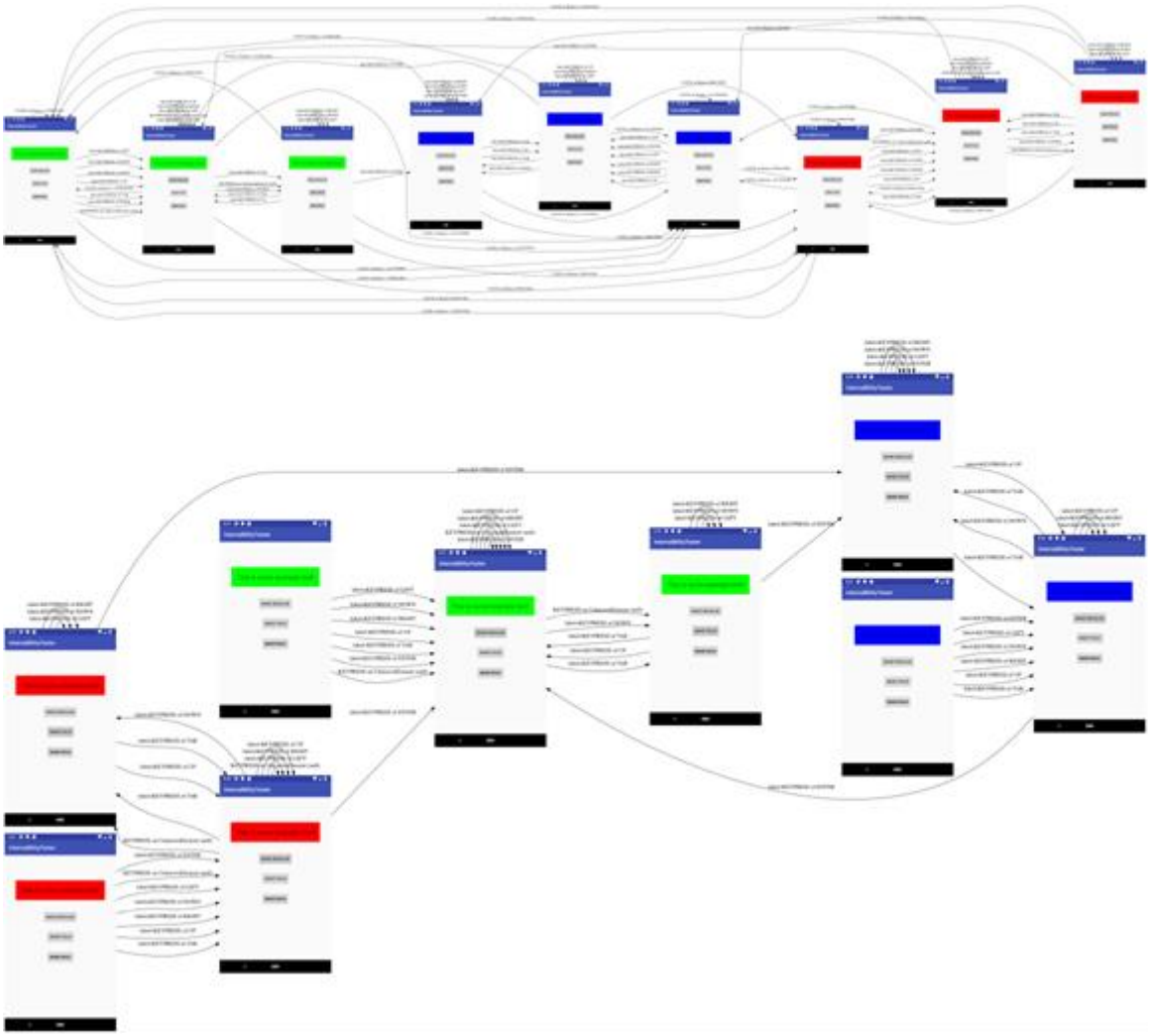


Рисунок 3.10. Діаграма станів

Діаграма стану вгорі показує всі типи дій та стани, досягнуті за допомогою цих дій. На цій діаграмі є щонайменше один край, що йде в кожен стан. Нижня діаграма – це та сама машина стану, але лише з ребрами, які представляють натискання клавіш. Зауважте, що три штати (один червоний, один зелений та один синій) не мають у них ребер. Фокус не викликає зміни контексту

Як уже згадувалося раніше, використання клавіатур часто призводить до того, що елементи зосереджуються або елементи, такі як кнопки, текст та зображення стають «вибраними», дозволяючи користувачеві взаємодіяти з ними, імітуючи дії, такі як клацання (часто натисканням кнопки «введіть кнопку» на їх клавіатурі).

Принцип 3.2.1 WCAG 2.0 вимагає, що коли елемент отримує фокус, зміна контексту не відбудеться [25]. Зміна контексту визначається як будь-яка зміна інтерфейсу користувача, що є більшою мірою, ніж зміна вмісту і є зміною, яка може дезорієнтувати користувача. Наприклад, перехід на новий інтерфейс користувача або зміна макета поточного інтерфейсу – це обидві зміни контексту.

State-машина, побудована особою, добре підходить для виявлення цієї проблеми. Спочатку видаляються всі ребра, крім ребер, що індукують фокус. Іншими словами, залишиться лише край *KEYPRESS*. Далі нам потрібно підтвердити, що стан призначення кожного краю не є зміною контексту від початкового стану для цього краю. За допомогою Система ми просто використовуємо процедуру зміни хешування контексту – якщо хеші двох станів рівні, цей перехід не є зміною контексту. Якщо ці хеші не рівні, Система повідомляє про порушення принципу WCAG 2.0 3.2.1.

Зміна режиму хешування контексту схожа на процедуру хешування доступності, в якій відстежуються альфа, колір тла, розмір шрифту, стиль шрифту, інтервал між рядками, іменування, фокус та сприйняття кольорів тексту. З іншого боку, режим хешування контексту відстежує ті самі уявлення, крім сприйняття фокусу. Іншими словами, зміна рутинного режиму хешування контексту перевіряє, чи два стани рівні відповідно до хеша доступності, але ігнорує будь-які зосереджені елементи.

Жодні дії не призводять до змін.

Принцип 3.2.5 WCAG 2.0 вимагає, щоб зміни контексту були ініційовані виключно за запитами користувачів (або щоб був передбачений механізм їх вимкнення). Це означає, що якщо користувач не вчинить жодних дій, у програмі мало що повинно змінитися.

Розробник може налаштувати систему на очікування перед тим, як здійснити будь-яку дію (або по суті створити недіючу дію). За замовчуванням ця дія робиться випадковим чином з певною частотою

базовою персоною, а також відбувається, коли обчислення на тестовому сервері повільні.

Враховуючи стан машини, що представляє додаток, кожен край, який не є *NONE*-дією, видаляється. Після цього для кожного краю *NONE* перевіряється, що це власне ребро (тобто це цикл самозапуску, що починається і закінчується в одному стані) або якщо цей край не є саморубіжним, що два стани однакові як визначено змінами контексту. Подібно до перевірки того, що зміна фокусу не змінює контекст, тут використовується однакова техніка хешування для визначення того, чи є два стани одного контексту. Якщо їх немає, то було виявлено порушення принципу WCAG 2.0 3.2.5.

Висновки до розділу

У даному розділі розглянуто архітектуру майбутньої системи, а саме було обрано тестовий сервер, сервер веб-сайту та сервер Maven. Зібрано інформацію про користувацький інтерфейс та описано інтерфейс користувача. Описано процес спрощення станів та процес прийняття рішень системою (створено варіації відповідей на запити дій та перемикач стилю тексту). Проаналізовано цикл виявлення проблем доступності та описано використані методи, в ситуації відсутності текстових альтернатив та контрасту тексту й піктограм. Вирішено питання динамічної доступності.

РОЗДІЛ 4. ОЦІНКА РОБОТИ СИСТЕМИ

Система оцінювалась двома способами – порівнюючи його результати зі сканером доступності Google і користувацьким дослідженням, щоб з'ясувати, які проблеми доступності можуть бути визначені розробниками людини без використання будь-яких інструментів.

Було обрано три програми з відкритим кодом, знайдені на GitHub, щоб пройти тестування доступності: додаток для запису звуку під назвою SoundRecorder, додаток для супроводу подорожей під назвою Travel-Mate і додаток для музичного плеєра під назвою Timber. Кожен додаток витягнуто з GitHub, складеного Android Studio та встановленого на пристрої Pixel 2 XL. Кожен додаток проходив тестування за допомогою сканера доступності Google, людського тестера та бібліотеки система, результати підсумовуються у кожному розділі.

В цілому було виявлено, що система відслідковував широкий спектр проблем із доступністю (три додаткові типи випусків) порівняно зі сканером доступності Google, уникав проблем із доступністю для звітування (іноді із вилученням повторних проблем у вісім-дев'ять разів менше проблем), і надав більше інформації щодо проходження та невдачі тестів на доступність (таких як детальна інформація про перегляд). Було також встановлено, що люди краще знаходили когнітивні проблеми в порівнянні з Система та сканером доступності Google, наприклад, гарантуючи, що етикетки та заголовки розділів точно описували їхній вміст та функції.

4.1 Результати сканера доступності Google

Сканер доступності Google використовувався для тестування кожного екрана, знайденого в програмі. Процес використання доступності Google для пошуку проблем із доступністю визначали наступним чином:

− перейдіть до стану інтерфейсу користувача, який ще не був протестований;

− натисніть кнопку Сканер доступності Google і поділіться результатами з тестером (у цьому випадку електронною поштою);

− повторюйте пункти 1 і 2, поки випробувач не переконається, що всі стани охоплені;

− після закінчення складіть результати в повний корисний звіт.

У випадку кроку 3 вихідний код програми та знання попередніх взаємодій із додатками використовувались для покриття якомога більшої кількості простору стану додатків.

4.1.1 SoundRecorder

Додаток SoundRecorder – це проста програма Android, яка підтримує запис і відтворення аудіо. Він містить сторінку про, діалогове вікно відтворення та інформацію про попередні та активні записи.

Під час навігації у програмі SoundRecording я виявив п’ять різних станів додатків, пов’язаних із запуском запису, активним записом, переглядом існуючих записів, налаштуваннями перегляду та переглядом інформації. Загалом, сканер доступності Google виявив 10 проблем. Три з цих питань стосувалися принципу WCAG 2.0 1.1.1, який вимагає текстових альтернатив для нетекстового контенту (у цьому випадку для кнопок і повзунків). Решта сім питань стосувалися принципу WCAG 2.0 1.4.3, який вимагає певного рівня контрасту між текстом переднього плану та фоном. Однак одне з цих питань повідомляє про рівень контрасту між піктограмою та її фоном. Скріншоти цих питань можна знайти на рис. 4.1.

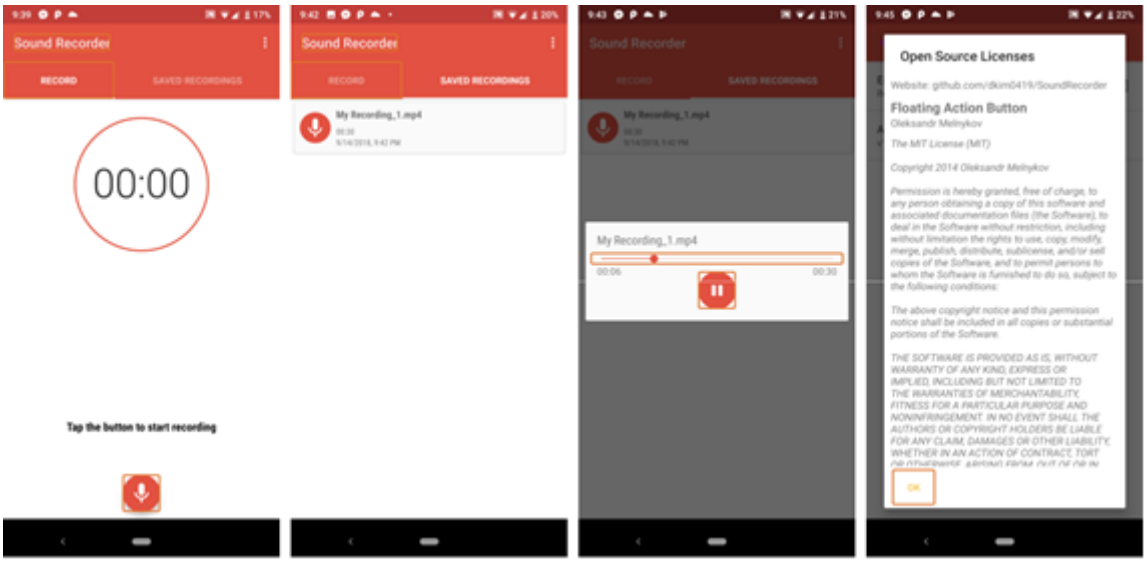


Рисунок 4.1. Скріншоти деяких проблем із доступністю SoundRecorder, про які повідомляється сканером доступності Google

Загалом, для програми SoundRecording сканер доступності Google зміг знайти відсутні мітки зчитувача екрана, поганий контраст тексту та поганий контраст зображення.

4.1.2 Travel-Mate

Додаток Travel-Mate – це багатофункціональний додаток для супутника подорожей [32], який має можливість планування поїздок, перегляду погоди, пошуку визначних пам’яток тощо.

У додатку Travel-Mate є набагато більше екранів, ніж у програмі SoundRecorder. Загалом у сканері доступності Google оцінюється 20 екранів, причому кожен екран відрізняється від додатка (наприклад, екран сповіщень, екран входу, екран погоди, та ін.) Кілька з цих екранів із виділеними проблемами можна побачити на рис. 4.2.

Загалом, програма Travel-Mate повідомила про загальну кількість:

– 32 проблеми із низькою контрастністю тексту до фону;

– 53 проблеми з сенсорними цілями занадто малі (менше 48dp в ширину або висоту);

– 18 питань, де вміст зчитувачів екрана для елементів повторювався серед елементів інтерфейсу.

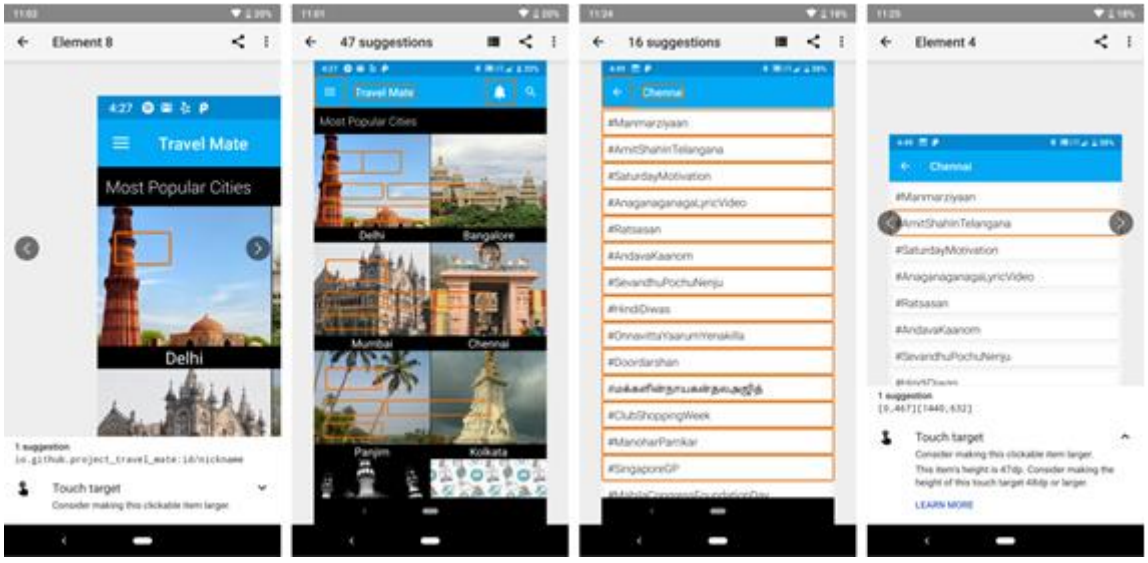


Рисунок 4.2. Скріншоти детального звіту про випуск програми Travel-Mate у сканері доступності Google

Виявлені проблеми включають невеликі цільові дотики та поганий контраст.

– 9 питань, коли елемент інтерфейсу не був сумісний із методами виявлення сканера доступності Google;

– 8 питань, де елементи, які можна натиснути, перекриваються;

– 13 питань, у яких контраст зображення або значка з фоном був низьким.

Issues 2 випуски, де на зображеннях чи піктограмах відсутні описи вмісту.

Більш глибокий аналіз цих результатів виявляє три обмеження або помилки зі сканером доступності Google. По-перше, Сканер не зміг розпізнати ситуації, коли елементи інтерфейсу користувача були приховані іншими елементами інтерфейсу користувача. З облямованих областей на рис.

4.3 ви можете побачити кілька проблем, що виявляються з елементами, які, схоже, приховані під зображеннями розташування. Ці елементи не сприймаються при нормальному використанні програми, і тому такі їх питання не стосуватимуться користувачів програми. Ці додаткові проблеми, що повідомляються, додають часу та часу зусиллям розробника щодо поліпшення доступності програми. Після подальшої перевірки коду, здається, що ці представлення є частиною користувацького перегляду, який використовується для вказівки різних аспектів головного екрана при першому використанні

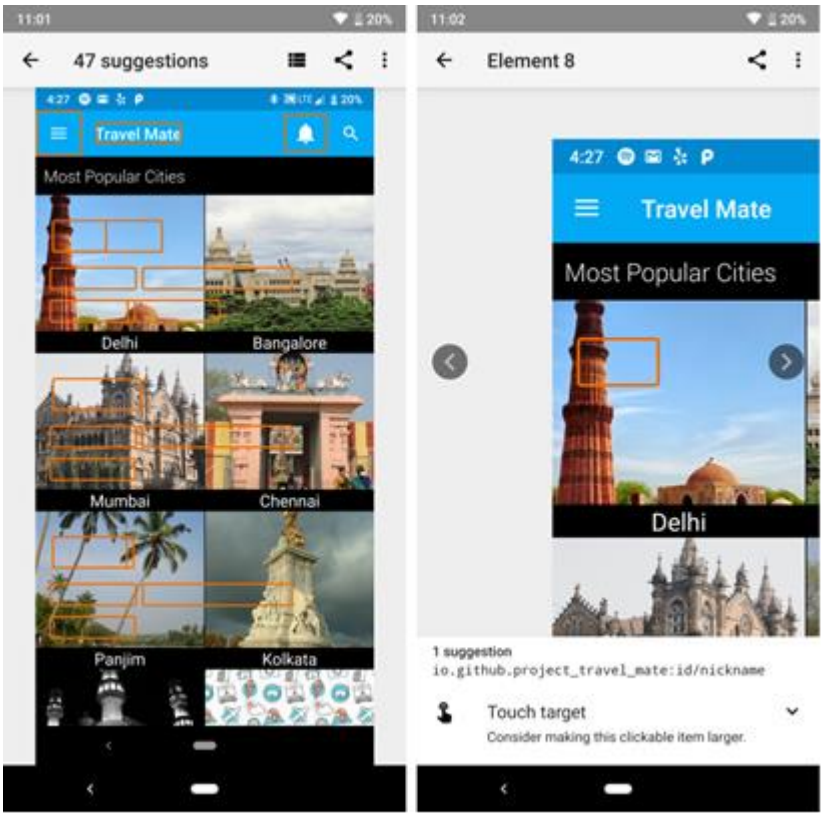


Рисунок 4.3. Скріншоти сканера доступності Google, який повідомляє про перекриваються об’єкти, що можна натиснути в додатку Travel-Mate

Іншим очевидним обмеженням сканера доступності Google є його нездатність реалізувати повторювані проблеми в динамічно повторюваному елементі. Наприклад, візьміть список повторних елементів, які можна натиснути на рис. 4.4. У цьому прикладі інтерактивний TextView, який

можна натиснути, програмно повторюється з одного базового примірника декларації TextView, або створеного безпосередньо з об’єкта TextView, або шляхом надування окремого файлу XML. Однак, хоча чотирнадцять випусків повідомляється розробнику, існує дійсно лише один екземпляр випуску, який при виправлення обробляє всі 14 випусків. Ця надмірна звітність може ввести в оману розробників і створити появу накладних витрат, що може призвести розробника до відмови від спроб збільшити доступність.

Кілька інших обмежень, виявлених у цих результатах, полягають у тому, що сканер доступності має певні проблеми із користувацькими типами перегляду. Наприклад, ця програма використовує TextInputLayout, про який сканер повідомляє наступне повідомлення: Тип цього елемента TextInputLayout може не вирішуватися службами доступності.

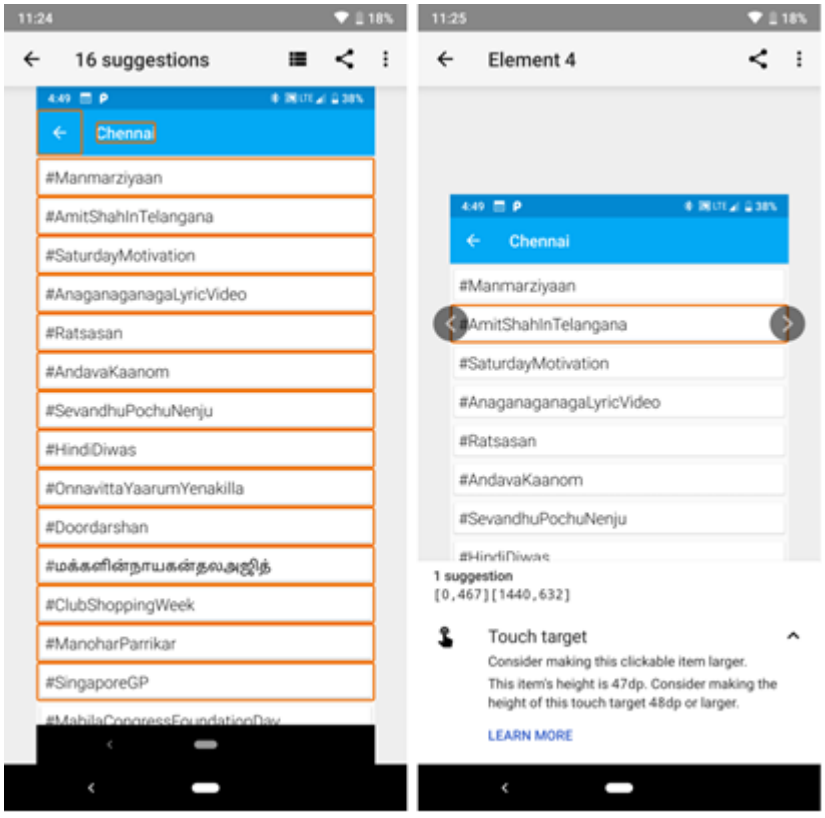


Рисунок 4.4. Повідомлення про проблеми з цільовим розміром для повторного елемента інтерфейсу

Сканер доступності Google покладається на служби доступності та вузли доступності, які не завжди доступні для користувацьких представлень.

Однак у цій програмі я виявив, що сканер доступності здатний повідомляти, коли вміст зчитувачів екрана різних компонентів повторюється на одному екрані. Повторне опис може спричинити плутанину під час навігації по екрану з пристроями зчитування екрана (наприклад, повторний елемент з непараметризованим описом матиме однаковий вміст для кожного елемента). Це ускладнює користувачеві розрізнення різних елементів один від одного.

4.1.3 Timber

Timber – це програма для зворотного зв’язку з музикою та звуком [33], створена для відображення різних компонентів та моделей дизайну, знайдених у матеріальному дизайні Google. У ньому є кілька екранів вибору та відтворення аудіо.

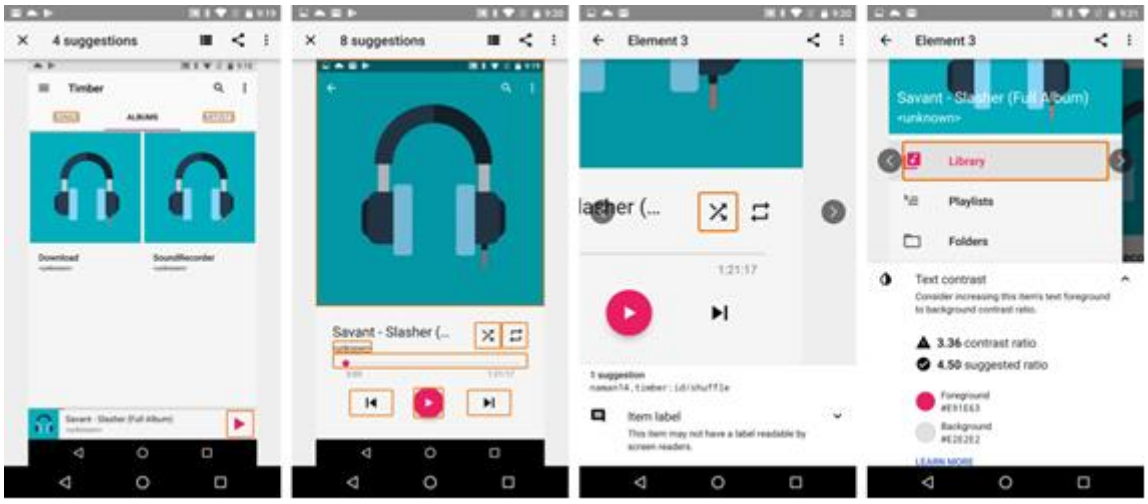


Рисунок 4.5. Приклад результатів сканера доступності Google для програми Timber

Додаток Timber оцінювався сканером доступності Google на 16 екранах, залежно від стану (наприклад, музика призупинена, відновлення

музики) та теми (наприклад, темна тема, світла тема). Загалом, сканер доступності повідомив про загальну кількість:

− 17 питань із низькою контрастністю тексту до фону;

− 56 проблем із сенсорними цілями занадто малі (менше 48dp в ширину або висоту);

− 2 проблеми, де вміст зчитувача екрана для елементів повторювався серед елементів інтерфейсу;

− 10 питань, в яких контраст зображення або значка з фоном був низьким;

− 73 питання, де на зображеннях чи піктограмах відсутні описи вмісту.

Додаток Timber значною мірою недоступний через відсутність описів вмісту на іконах, а також малого розміру цих піктограм. Однак багато з цих питань – це просто те саме питання, що повторюється на динамічних елементах, як це видно на рис. 4.6. У цьому звіті повідомляється про 9 випусків для піктограми невеликих опцій для кожного елемента (піктограма трьох вертикальних точок).

Рисунок 4.6. Скріншот результатів сканера доступності Google для сторінки з динамічно повторюваними елементами

Зверніть увагу, що кожна помаранчева підсвітка вважається однією проблемою, хоча більшість питань меню опцій походять з одного визначення View. Однак цей елемент інтерфейсу користувача – це справді одне визначення, повторене кілька разів, і як такий розробник має вирішити лише одне питання.

4.2 Оцінка роботи Google сканера

Загалом, сканер доступності Google виявив широкий спектр проблем статичної доступності (таких як невелика цільова сенсорність, контрастність, повторний вміст зчитувача екрана та відсутність вмісту зчитувача екрану), але кількість повідомлених проблем є досить великою через неможливість обробляти динамічну генерацію переглядів.

Сканер доступності Google використовувався для оцінки доступності трьох реальних програм у Play Store (а також доступних як проекти з відкритим кодом). В цілому було встановлено, що сканер доступності Google зміг досягти наступних функціональних можливостей: testing Тестування доступності для складених програм без файлів проекту. Виявлення описів відсутніх вмістів (тісно пов’язаних з принципом WCAG 2.0 1.1.1). Виявлення поганого контрасту тексту та піктограм на їхньому тлі (тісно пов’язане з принципами WCAG 2.0 1.4.3 та 1.4.6) · Виявлення невеликих розмірів цільового дотику (тісно пов’язане з WCAG 2.1 Принцип 2.5.5). Виявлення повторних описів зчитувачів екрану. Виявлення елементів, що перекриваються користувальницьким інтерфейсом, які можуть заважати здібностям клацання. Однак існує також кілька обмежень. Проблеми з елементами, які динамічно повторюються, визнаються окремими питаннями. На екрані за екраном виявляються лише статичні проблеми – динамічне функціонування програми не перевіряється на доступність. Автоматичне тестування не доступне для сканера доступності – розробник повинен орієнтуватися в програмі, провести тест і надіслати результати на свій комп’ютер. Виявлено, що сканер доступності Google дуже добре знаходив

статичні проблеми, пов’язані з кольорами та текстом. Однак йому не вистачало динамічного виявлення проблем із доступністю та було втомливим для використання (як зусиль, необхідних для запуску тесту, так і зусиль, необхідних для розуміння та обробки результатів).

4.3 Результати розробленої системи

Було протестовано кожну програму, використовуючи рамки тестування системи. Завантаживши вихідний код кожної програми через GitHub, вставив у програму бібліотеку Система Android, створив тестовий файл (див. додаток А.2 для прикладу тестового файлу) та запустив сервер тестування Система. Для кожної програми налаштування займало не більше 5 хвилин. В цілому, я виявив, що Система не тільки змогла знайти проблеми зі статичною доступністю, подібні до тих, що були знайдені сканером доступності Google, але вона також мала більш стислий звіт про ці проблеми. Система також знайшов додаткові проблеми доступності, пов’язані з динамічним функціонуванням програми, такі як навігація по клавіатурі та зміни в контексті програми.

4.3.1 SoundRecorder

Рамка Система створила короткий звіт із статичними та динамічними проблемами, знайденими в програмі. Загалом, Система знайшов таку кількість унікальних проблем:

− порушення WCAG 1.1.1 нетекстовий вміст (загалом знайдено 66 примірників);

− порушень WCAG 1.4.6 контрастність (посилений) (загалом виявлено 38 випадків);

− 2 порушення WCAG 3.2.1 на фокус.

До порушень WCAG 1.1.1 були включені кнопки запису, відтворення та паузи, знайдені в усій програмі, а також прапорець на екрані налаштувань

для забезпечення високої якості записів. Порушення WCAG 1.4.6 містилося в панелі дій чи панелі інструментів у верхній частині екрана, а також вкладках для навігації між екранами запису та збереженими записами. Ці порушення також включали різні фокуси та вибір цих вкладок (не лише вкладки без взаємодії чи введення).

Одним із корисних аспектів представлення цих питань у системі є те, що вони не будуть переоцінювати проблеми, які динамічно повторюються. Наприклад, див. рис. 4.7, де повідомляється, що кнопка "Відтворити" не містить опису вмісту для екранного зчитувача. Замість того, щоб повідомляти про цю проблему для кожної кнопки Play на цьому екрані, вона повідомляється лише один раз, визнаючи, що інші екземпляри цієї проблеми на цій сторінці, ймовірно, викликані тим самим динамічно завищеним ресурсом View.

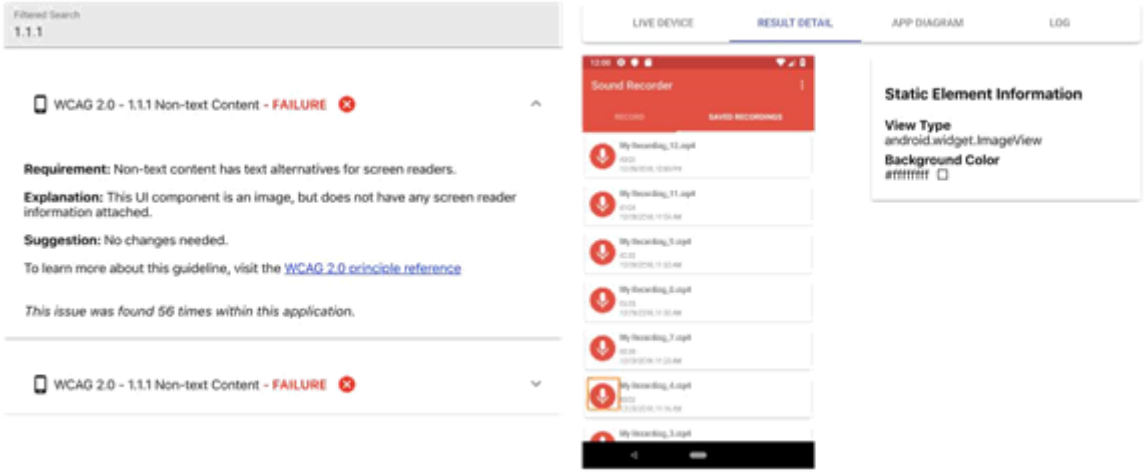


Рисунок 4.7. Екземпляр порушення нетекстового вмісту для кнопки Відтворення на екрані збережених записів

Щоб отримати уявлення про те, скільки існує подібних проблем, Система також містить ряд подібних примірників у нижній частині опису випусків. Нарешті, Система також знайшов два питання щодо динамічної доступності із порушеннями принципу 3.2.1. Як показано на рис. 4.8, Система виявив, що зміна фокусу може призвести до переключення

ViewPager, який тримає екран запису та збережених записів, між двома станами. Якщо користувач використовує клавіатуру, то зміна фокусу може змусити користувача перейти на зовсім інший екран, що може переплутати або заплутати його. Розробник може використовувати цю інформацію для відключення змін у фокусі між екранами ViewPager та змусити користувача використовувати вкладки у верхній частині екрана для навігації під час використання клавіатури.

4.3.2 Travel-Mate

Додаток Travel-Mate було протестовано на Система із повідомленнями про наступні проблеми:

− 11 порушень WCAG 1.1.1 Нетекстовий вміст (загалом знайдено 68 примірників);

− 21 порушення WCAG 1.4.3 Контрастність (загалом знайдено 143 випадки);

− 33 порушення WCAG 1.4.6 Контрастність (посилений) (знайдено 221 примірник).

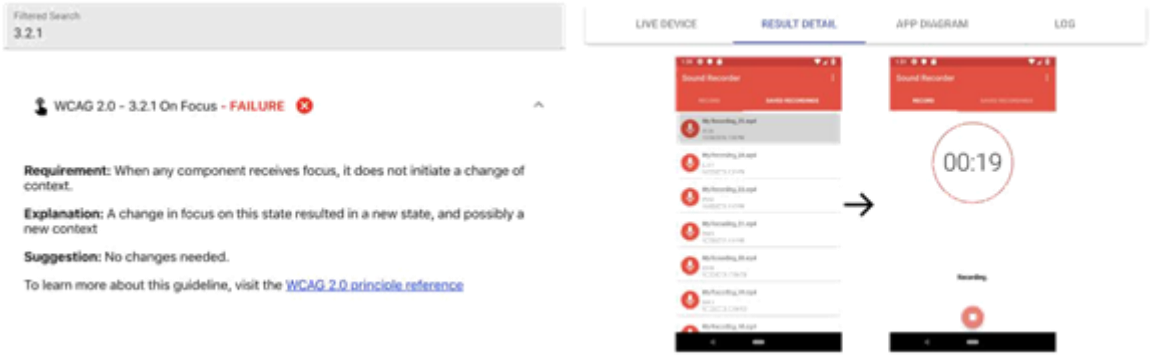


Рисунок 4.8. Використання клавіатури на екрані

Це може заплутати деяких користувачів, і тому Система позначає це як проблему доступності – 3 порушення WCAG 2.1.1 Клавіатура.

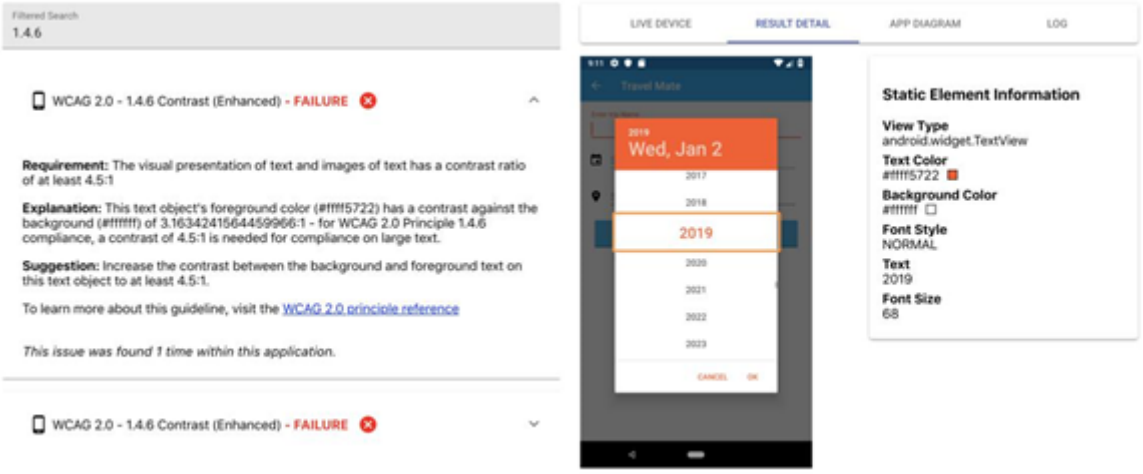


Рисунок 4.9. Білість виявилася чудовою в маніпулюванні окремими екранами в рамках програми в повному обсязі

Тут ми бачимо, як Система вибирає дату в межах вибору дати для екрана створення поїздки.

У додатку Travel-Mate є дуже великий простір стану; під час ручного тестування було встановлено, що ця програма мала щонайменше 20 різних екранів. Під час маніпулювання цими користувальницькими інтерфейсами кожен екран може мати декілька конфігурацій та внутрішніх станів, що робить надзвичайно важким тестування кожного стану вручну. Система виявився корисним у цій ситуації, досліджуючи безліч конфігурацій одного екрану, намагаючись множинні дії на одному екрані (див. рис. 4.9). Однак така розвідувальна поведінка призвела до компромісу, коли не було досліджено стільки екранів порівняно з ручними

Однак така розвідувальна поведінка призвела до компромісу, коли не було досліджено стільки екранів у порівнянні з ручною розвідкою. Система (в базовій конфігурації без введення розробника) спробує будь-яку дію на будь-якій сторінці, яка ще не була зроблена, і як така не знає, як найкраще досліджувати весь простір стану.

Насправді ця спроба найкращих зусиль дослідити додаток розглядається через 3 порушення принципу WCAG 2.1.1. Замість того, щоб бути справжніми проблемами навігації на клавіатурі, ці три штати Система

не були доступні через те, що додаток Travel-Mate зірветься до того, як система зможе повністю вивчити простір штатів. Після декількох пробних запусків, результати, які тут бачать, – це результати тестового запуску, які мали найбільше дій, здійснених під час тесту (82 дії), що представляють собою "найбільш досліджений" пробіг.

Якщо Travel-Mate вдалося знайти ці винятки, виявлені під час виконання, Система продовжуватиме працювати, поки не буде отримано акцію QUIT. Однак тестовий сервер все ще може повідомляти і обробляти збої в програмі, оскільки всі результати обробляються і зберігаються в режимі реального часу.

Що стосується статичних питань, Система знайшов досить багато питань, пов’язаних з контрастними та альтернативними вмістами вмісту. Однак Система також неправильно повідомила про деякі з цих контрастних питань. На рис. 5-10 знайдено екземпляр, де правильний колір фону для кнопки не знайдено. Це можна покращити за допомогою кращих процедур, які аналізують справжні композиції кольорів пікселів, а не лише властивості перегляду.

4.3.3 Timber

Програма Timber також була протестована за допомогою Система. Загалом, Система знайшов таку кількість питань:

− 15 порушень WCAG 1.1.1 Нетекстовий контент (загалом знайдено 53 випадки);

− 12 порушень WCAG 1.4.3 Контраст (мінімум) (із загальною кількістю 112 випадків);

− 22 порушення WCAG 1.4.6 Контрастність (посилений) (із загальною кількістю знайдено 237 примірників) .

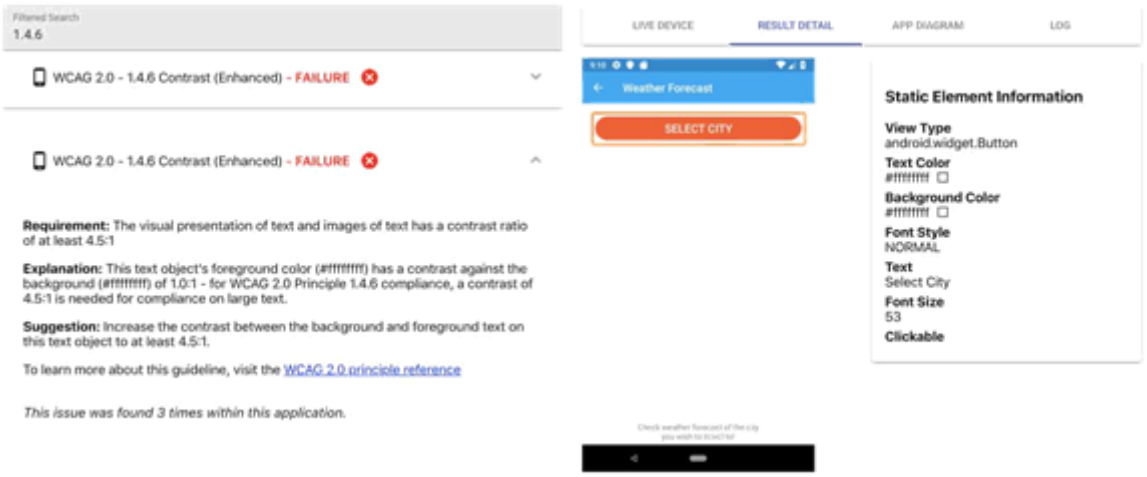


Рисунок 4.10.Порушення в додатку Timber

Система іноді матиме проблеми з пошуком правильного кольору фону для користувацьких компонентів та компонентів, які використовували подання фону, окремо від подання переднього плану . У цьому випадку кнопці надається спеціальний фон, який не повідомляє її колір як властивість View.

- порушення WCAG 2.1.1 Клавіатура;
- порушень WCAG 3.2.1 На фокус.

Персона була вражаючою своєю здатністю досліджувати простір станів, значно більшою мірою порівняно з ручним тестуванням, проведеним людьми-тестерами та під час тестів сканера доступності Google. Після дозволу людині Система здійснити 395 дій, повідомлялося, що 176 станів з невивченими діями все ще були присутніми (див. рис. 4.11). Цей величезний простір станів був спричинений не тільки широким розмаїттям базових екранів, які доступні, але й тим, що програма підтримує зміну акцентних та кольорів тла. На рис., система насправді вдалося знайти цей діалог із зміною кольору та перевірити різні акценти.

Компанія Timber також представила кілька питань щодо динамічної доступності. Наприклад, на рис. 4.13 показано виявлення помилок навігації на клавіатурі (WCAG 2.0 Принцип 2.1.1), коли деякі стани недоступні лише за допомогою клавіатури. Користуючись цією інформацією, у програмі я

підтвердив, що багато екранів не реагують на введення клавіатури, що робить неможливим охоплення деяких екранів за допомогою лише клавіатури. Потім розробник може скористатися цією інформацією, щоб переконатися, що жоден екран не відображає пастку клавіатури для користувача.

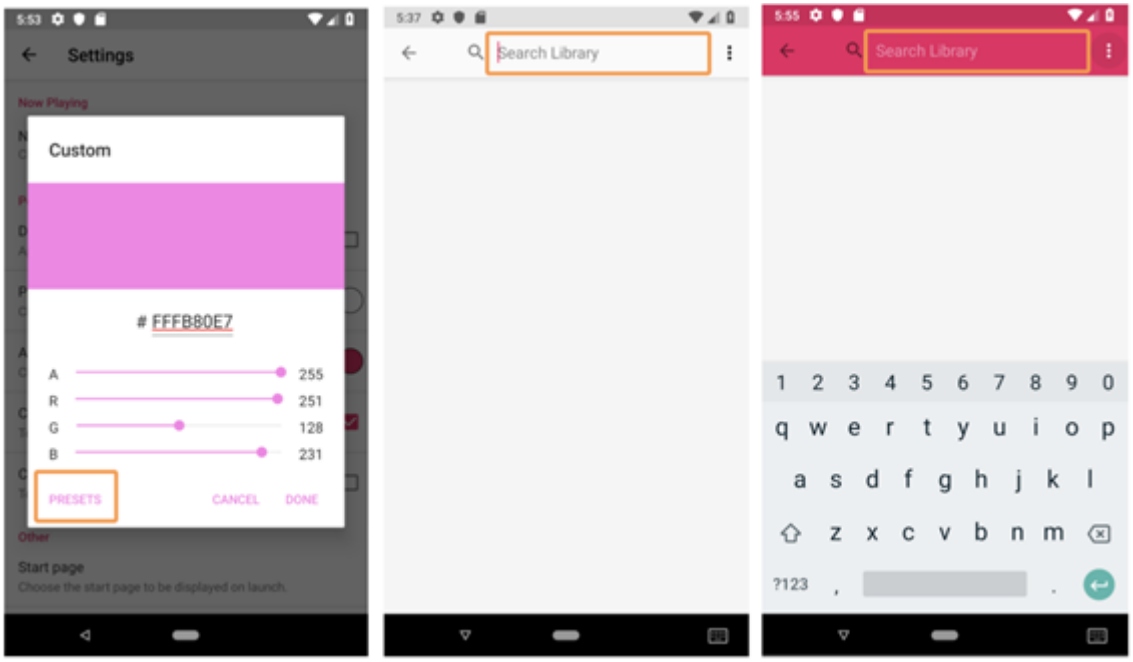


Рисунок 4.11. Приклад роботи з додатком

Перше зображення показує зустріч з діалогом зміни акценту, а на другому та третьому зображеннях відображаються різні наголоси, які використовуються у програмі. У цьому випадку пошук EditText є однаковим, але через зміну фону про будь-яку проблему з EditText повідомляється кілька разів.

Рисунок 4.12. Діаграма станів у розрізі області

Деякі штати були широко досліджені, в той час як в інших є ще багато дій для перевірки. Статичні проблеми, знайдені Система, включають відсутні описи вмісту на кнопках та зображеннях у всьому користувальницькому інтерфейсі, а також погана контрастність деяких текстових елементів користувальницького інтерфейсу. Однак тестування цього додатка за допомогою програми Система також виявило декілька обмежень програми Система, зокрема стосовно екранів та зображень.

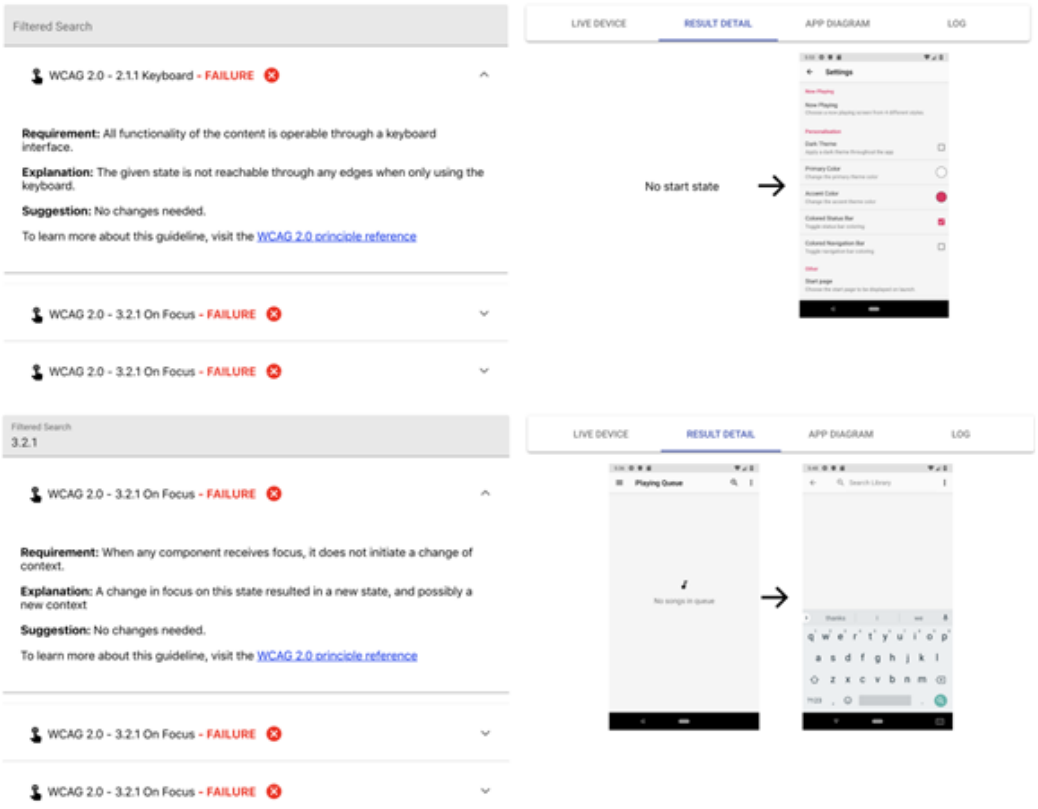


Рисунок 4.13. Недоступність екрану в динамічній клавіатури

У верхньому екрані зображено повідомлення про один екран, недоступний за допомогою клавіатури, тоді як на нижньому екрані відображається зміна контексту, викликана зміною фокусу (тобто фокусування на EditText спричиняє появу клавіатури).

Наприклад, на рис. 4.14 показано повідомлення про проблему з контрастом, оскільки TextView має білий колір, і колір фону не знайдено. Швидше фоном є зображення ImageView, яке не створено для кольорів та

кольорових областей. Типовою поведінкою Система є припущення білого базового фону, що спричиняє білосніжний екземпляр поганого контрасту. Однак альтернативним рішенням для усунення цих проблем є відключення контрастних перевірок, коли колір фону не знайдено.

4.4 Оцінка результатів розробленої системи

Використання рамки тестування система у трьох додатках виявилося корисним прикладом для оцінки можливостей системи. Виявлено, що система змогла досягти наступного:

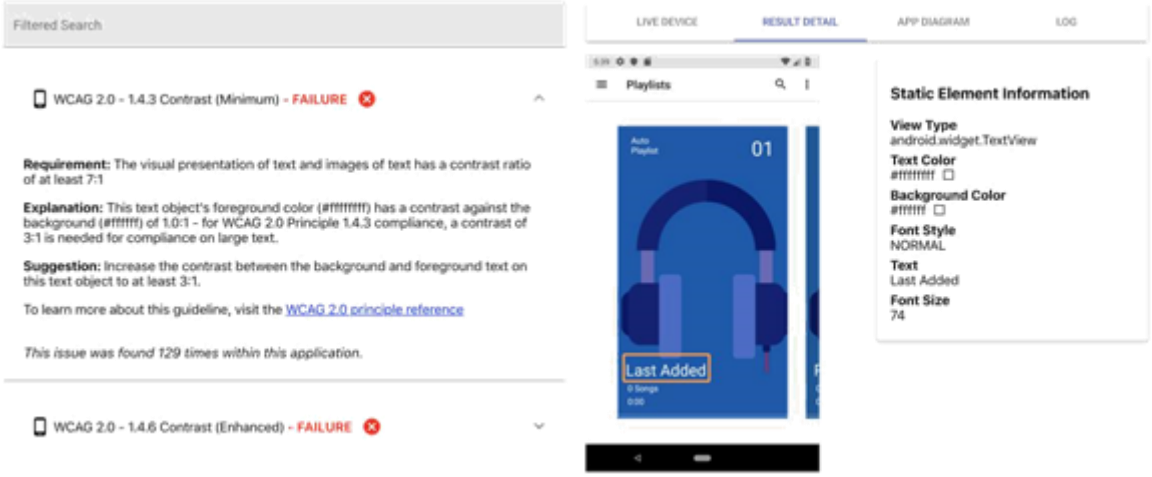


Рисунок 4.14. Повідомлення про кольорову композицію

Виділений TextView над ImageView, який не повідомляє про його кольорову композицію. Тому білий текст вважається прозорим фоном, який трактується білим.

Тому основними перевагами розробленої системи є:

Виявлення проблем статичної доступності, такі як поганий контраст, відсутні мітки зчитування екрана та невеликі розміри цільового сенсорного екрана.

Визначення проблеми динамічної доступності, такі як пастки клавіатури та зміни контексту при фокусуванні.

Автоматична перевірка програми на доступність з мінімальною конфігурацією від розробника.

Повідомляйте про проблеми доступності інформативно, не повторюючи проблеми в динамічно генерованих поданнях та макетах.

У розділі дизайну також було відмічено, що система здатна випробовувати багаторазові мітки зчитування з екрана та зміни контексту, коли не вводяться дані користувача. Хоча ця логіка реалізована, досліджувані стани цих додатків не виявили жодних проблем, пов’язаних з цими тестами доступності. Однак, під час цієї оцінки було виявлено декілька обмежень та недоліків системи:

Система не завжди змогла автоматично досліджувати простір стану програми в тій же мірі, що й людина.

Про деякі проблеми, пов’язані з контрастом, повідомлялося невірно, найчастіше пов’язані з використанням власних представлень та спеціального фону. Більш детальне пояснення та погляд на сильні та слабкі сторони системи можна знайти в розділі Порівняння результатів.

4.4.1 Аналіз результатів оцінки

Порівняння Система, сканера доступності Google, бібліотеки тестування Еспресо та інструментів зв’язування Android Studio можна знайти в Таб. 4.1, де порівнюється кожне програмне рішення на основі різних критеріїв, що мають значення під час тестування розробниками.

Таблиця 4.1. Порівняння інших інструментів з розроблюваною системою

	Розроблювана система	**Google Accessibility Scanner**	**Espresso**	**Lint**
Автоматизована	+		+	+
Перевірка статичного коду				+
Статична перевірка часу виконання	+	+	+	

Закінчення таблиці 4.1

Виявлення низької контрастності	+	+	+	
Виявлення мітки зчитування екрана	+	+	+	+
Виявлення пасток клавіатури	+			
Мультиплатформна розширювана версія	+			
Тестування на налагоджувальній версії APK	+		+	
Тестування на виробництво APK		+		
Необов’язкова конфігурація тесту	+			
Звіт про випуск в реальному часі	+	+		+
Усунення повторних питань	+			
Перегляд прохідних екземплярів	+			

У таб. 4.2 порівнюються результати кожної методики, оціненої в розділі Оцінка, щодо критеріїв успішності, викладених у WCAG 2.0 та 2.1. Загалом, зрозуміло, що Система перевершує інші рамки як у здатності виявляти проблеми з динамічною доступністю, так і у здатності виявляти проблеми доступності з мінімальними конфігураційними та обробними зусиллями розробника. Використовуючи результати навігації програми, Система здатний проаналізувати структуру даних про стан машини в знайти проблеми доступності, пов’язані з використанням клавіатур та несподіваними змінами контексту.

Таблиця 4.2. Порівняння щодо принципів WCAG та критеріїв успіху

Принцип	**Система**	**Google Accessibility Scanner**	**Людина**
1.1.1 Нетекстовий вміст	+	+	
1.2.1 Тільки для аудіо та відео			+
1.2.2 Підписи (попередньо записані)			+
1.2.3 Альтернатива для медіа			+

Закінчення таблиці 4.2

1.4.3 Контрастність (мінімальна)	+	+	
1.4.6 Контрастність (посилена)	+	+	
2.1.1 Клавіатура	+		
2.2.1 Регулювання часу			+
2.4.6 Заголовки та етикетки			+
2.5.5 Розмір цілі	+	+	
3.2.1 Про фокус	+		
3.2.5 Зміни на запит	+		
3.3.2 Мітки або інструкції			+

Інші бібліотеки наразі не можуть цього зробити, оскільки аналізується лише поточний користувальницький інтерфейс, який не відображається та не знає попередніх станів. Крім того, це динамічне дослідження проводиться з невеликою конфігурацією або зусиллями розробника. Для таких бібліотек, як Еспресо, розробник повинен написати навігаційний код для дослідження програми, тоді як сканер доступності Google вимагає від розробника вручну клацнути програму. З іншого боку, Система просто вимагає від розробника скопіювати та вставити один тестовий файл у свою програму, який буде обробляти всю автоматичну навігацію. Використовуючи нову техніку хешування користувальницького інтерфейсу, Система також може уникнути повідомлення про проблеми, про які ефективно повідомлялося раніше. Поширеною технікою в розробці Android є динамічне створення представлення з одного ресурсу або декларації; Білість здатна визначити ситуації, коли проблема доступності, можливо, є тією, яка вже охоплюється попередньою ідентифікацією тієї самої проблеми в аналогічному режимі. Однак сканер доступності Google виконує більш високу точність для деяких типів проблем із доступністю. Наприклад, Система іноді не вдається точно визначити кольори тла, коли кольори фону вбудовані у власні представлення та власні фонові малюнки. Сканер доступності Google уникає цих проблем, аналізуючи малюнки для реального кольору фону або ігноруючи будь-які

проблеми, якщо спеціальний вид не підтримує пошук інформації про доступність.

Висновки до розділу

У даному розділі проведена загальна оцінка роботи системи, яка включала в себе: тестування роботи розроблюваної системи на трьох мобільних додатків та порівняння отриманих результатів з результатами тестування стороннім додатком. Також виділено сильні та слабкі сторони розробки й проаналізовано подальші шляхи для роботи.

РОЗДІЛ 5. РОЗРОБКА СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

Стартап має на меті впровадження новітніх технологій для людей з обмеженими можливостями. Основна ідея проекту наведено у таблиці 5.1

Таблиця 5.1. Опис ідеї стартап-проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Ідея проекту надати повний доступ людям з обмеженими можливостями до мобільних додатків	1. Інженерні Послуги (розробка обладнання та модернізація обладнання)	Звернувшись до нас клієнт отримує ефективну перевірку свого продукту в короткі терміни, високої якості та більш низькою ціною
	2.Дослідження та аналіз	Звернувшись до нас клієнт отримує можливість комплексно перевірити доступність розроблюваних мобільних додатків
	3.Фінансова звітність та консалтингові послуги	Звернувшись до нас клієнт отримує можливість контролювати свою фінансову звітність та отримати консультування з широкого кола питань у сфері фінансової, комерційної, технологічної, технічної діяльності
	4.Розробка програмного забезпечення	Звернувшись до нас клієнт отримує можливість отримати, якісні та сучасні розробки

Такий підхід дає можливість надати повний спектр послуг користувачу. Проведено а наліз потенційних техніко-економічних переваг ідеї:

− порівняно із пропозиціями конкурентів;

− визначено перелік техніко-економічних властивостей та характеристик ідеї;

− визначено попереднє коло конкурентів, що вже існують на ринку, та проведено збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів;

− проведено порівняльний аналіз показників: для власної ідеї визначено показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (таблиця 5.2).

Таблиця 5.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

п/п	**Характеристики ідеї**	**(потенційні) конкурентів**			**послуги**	**W (слабка сторона)**	**N (нейтральна сторона)**	**S (сильна сторона)**
		Мій проект	*Конкурент 1*	*Конкурент2*	*Конкурент3*			
1.	Інженерні послуги	має	має	має	має	-	-	+
2.	Дослідження та аналіз	має	немає	має	немає	-	-	+
3.	Фінансова звітність	має	має	немає	немає	-	-	+
4.	Розробка програмного забезпечення	має	немає	немає	немає	-	-	+
5.	Креативний дизайн	немає	немає	немає	має	-	+	-
6.	Послуги архітектора	немає	має	немає	має	+	-	-

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційних послуг, що є підґрунтям для формування його конкурентоспроможності.

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу проведено аудит способу, за допомогою якого можна реалізувати ідею проекту та наведено його у таблиці.

Таблиця 5.3. Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Надання інструменту я тестування доступності мобільних додатків	Створення автоматизовано х системи для тестування додатків в спектрі доступності	Ні, необхідно розробити	Так, дані технології доступні
Обрана технологія реалізації ідеї проекту: є можливою				

За результатами аналізу видно, що можливості технологічної реалізації проекту, та методи реалізації є можливими.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Основні оператори ринку:

Основними операторами ринку є міжнародні та вітчизняні продуктові компанії, які спеціалізуються на розробці мобільних додатків конструкторські відділи, які виконують роботу на аутсорс та на вітчизняні ринки послуг.

Таблиця 5.4.Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	10
2	Динаміка ринку (якісна оцінка)	Зростає
3	Наявність обмежень для входу (вказати характер обмежень)	Масштабність
4	Специфічні вимоги до стандартизації та сертифікації	ДСТУ,ГОСТ, ISO, WCGA

За результатами аналізу таблиці можна зробити висновок що ринок є привабливим для входження за попереднім оцінюванням.

Економічні та соціальні тенденції ринку:

На даний момент ринок знаходиться на стадії введення. Економічна ситуація в Україні та поступовий вхід України до Європейського Союзу вимагає перепрофілювання наших підприємств, збільшення конкурентоспроможності промисловості та введення нових інноваційних технологій та інвестицій.

Рисунок 5.1 – Ринок IT послуг в Україні

Надходження від експорту IT послуг в 2018 році за даними НБУ: $ 3,204 млрд, приріст склав 29% в порівнянні з 2017 роком.

Дохід від експорту IT послуг в 2018 році за даними Держслужби статистики: $ 1,578 млрд, приріст – 20% в порівнянні з попереднім роком.

Чисельність персоналу в експорті IT послуг в 2018 році за даними DOU 159 687 технічних фахівців на кінець року, приріст склав 26% з початку року. Всього зареєстрованих ФОП для надання IT послуг на початок 2018 року – 125 000.

З огляду на різницю між надходженнями і доходом, а також знаходження частини бізнесу в тіні, оцінюємо дохід від експорту IT послуг в 2018 році «на пальцях»:

- середня чисельність персоналу: 143 385;
- середня годинна ставка персоналу: $ 25;

− середня кількість проданих годин на рік з дисконтом від робочого часу 30%: 1 344 / чол;

− приблизний дохід: $ 4,8 млрд.

Висновки:

− IT бізнес як і раніше оптимізує податки в Україні через ФОП і роботу з нерезидентами. Різниця між статданими і реальним доходом – до 70%;

− приріст галузі видно в різних незалежних метриках (дохід, надходження, персонал) і за підсумками 2018 року перебуває в діапазоні 20-29%;

− загальне надходження від експорту товарів і послуг в 2018 році за даними НБУ – $ 59,117 млрд, тобто частка IT надходжень склала 5,4%. Для порівняння, експорт транспортних послуг приніс $ 5,923 млрд (10%).

В цілому, частку України на світовому ринку IT послуг можна порахувати як 0,5%. З іншого боку, розуміючи, що основна частина українських IT послуг – це модель аутсорсингу, а моделі аутстаффінгу і продуктової розробки займає невелику частку за чисельністю програмістів.

Аналіз ринку показує ознаки бурхливого розвитку української IT галузі. Головні з них:

− темп приросту продажів українських IT компаній перевищує загальний світовий темп в 5 разів;

− поліпшується міжнародне позиціонування українських IT компаній: залучення інвестицій в нові стартапи, великі M & A угоди, нові відомі контракти з IT гігантами, підвищення позицій в світових рейтингах;

− клієнти прагнуть урізноманітнити кількість одночасних постачальників IT послуг, тим самим підвищуючи можливості для нових продажів українських компаній;

− клієнти зміщують акценти з традиційного IT аутсорсингу на IT рішення бізнес-задач. Українські компанії підхоплюють цю хвилю, залучаючи в індустрію бізнес-аналітиків;

− держава більше втягується в розвиток IT індустрії, використовуючи тему як частину державної міжнародної PR програми;

− IT компанії інтегруються з українськими ВНЗ, створюючи попит на фахівців і коректуючи освітні програми;

− великі IT компанії розвивають власні академії;

− у 17 містах сформовані IT кластери, локально об’єднують учасників ринку.

Надалі визначаємо потенційні групи клієнтів, їх характеристики, та формуємо орієнтовний перелік вимог до товару для кожної групи (таблиця 5.5).

Таблиця 5.5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1.	Потреба в інноваційних високоефективних проектах	Аутсорс та аутстафф IT компанії	Необхідніст виготовлення продукції у відповідностідо різни норм та стандартів	- якість, - швидкість - доступність

Таблиця 5.6. Підсумкова таблиця факторів політико-правового середовища

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми чи реалізації можливості**
	Можливості	**Загрози**	
Закони України «Про основи національної безпеки», «Про інформацію», «Про захист інформації в інформаційно-телекомунікаційних системах»	Безпека праці, захист майна та інтелектуаль ної власності	Зміна законодавств а України в негативну сторону	Вчасна оплата податків. Вибір адекватної влади на виборах та активна участь в контролю влади
Конвенція Ради Європи про кіберзлочинність	Захист інформації	Невідомо	Багатоетапна організація та планування праці, для зменшення кількості помилок в роботі

Закінчення таблиці 5.6

Закони України про приватну власність	Захист інтересів	Несправедливе законодавство	Захист від недобросовісних конкурентів. Боротьба за справедливість правовими методами
WCAG 2.0	Нові клієнти в Америці	Не відповідність законів	відмова до співпраці в країнах де можуть бути несправедливе правове забезпечення
Європейські стандарти доступності, EN 301-549	Нові клієнти в Європі	Не відповідність законів, несправедливе законодавство	Знання міжнародного та Європейського законодавства
Інтелектуальна власність та закони про захист інтелектуальної власності	Захист власних розробок	Не відповідність законодавства. Викрадення та копіювання ідей.	Наймання спеціаліста з захисту інтелектуальної власності Патентування, засекречування розробок та ідей. Охорона приміщення та персональних комп'ютерів
Військові дії та нестабільна політична ситуація в Україні	Нові замовлення для військового комплексу України	Небезпека утворення нестабільної ситуації та заворушень	Універсальність запропонованих проектів
Корупція	–	Втрата майна та коштів	Захист власних інтересів

Таблиця 5.7. Підсумкова таблиця факторів економічного середовища

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми чи реалізації можливості**
	Можливості	**Загрози**	
Девальвація гривні	Переконати клієнтів виробляти обладнання власними ресурсами в Україні	Високий курс іноземних валют	Створити умови для незалежності національної валюти

Закінчення таблиці 5.7

Подорощення чи здешевіння конструкційних, робочих матеріалів	Нові проекти	Збільшення вартості наших проектів.	Створити нові проекти з мінімальним застосуванням дорогих матеріалів та їх заміна на дешевші
Нестабільна економічна ситуація (кризи)	Докази важливості нашої діяльності	Не зацікавленість багатьох клієнтіву співпраці та розвитку	Діяльність на покращення економічної ситуації в Україні та світі
Інфляція	Вигідна співпраця в іноземних валютах	Високі ціни на товари та зниження купівельної спроможності в національній валюті	Падіння економіки, не зацікавленість багатьох підприємств у співпраці
Зростання цін на енерго ресурси	Створення нових проектів енергозбереження для себе та клієнтів	Невистачання коштів на оплату енергозабезпечення діяльності підприємства	Збільшити рентабельність на енергоаудит та обрати курс на створення енергоефективних виробництв

Таблиця 5.8. Підсумкова таблиця факторів науково-технічного середовища

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми**
	Можливості	**Загрози**	
Зміна та виникнення нових тенденцій в обраній сфері діяльності чи «технологічні прориви»	Змога самому створювати нові тенденції, в обраній діяльності (машинобудування та інші)	Втрата актуальності власних розробок	Постійний пошук та моніторинг актуальних тенденцій в обраній сфері діяльності та суміжних, пошук та розробка принципово нових інновацій
Науково технічне відставання науки та техніки від провідних країн світу	Створення власних тенденцій та технологій	Не можливість забезпечити конкуренцію в даній галузі	Спроба зацікавити іноземних інвесторів в актуальності нашого проекту
Низька увага влади на інноваційну діяльність	Привернути увагу суспільства	Падіння зацікавленості потенційним и клієнтами	Спроба переконати важливість нашої діяльності

Таблиця 5.9. Підсумкова таблиця факторів демографічного середовища

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми чи реалізації можливості**
	Можливості	**Загрози**	
Низька народжуваність	Створити нові напрямки діяльності	Не достатня кількість підростаючих фахівців та споживачів послуг	Спонукати молодих людей до створення сімей. Створення вигідних умов для роботи (вільні графіки роботи та робота дома). Знайти нових клієнтів в індустрії виробництва дитячих іграшок
Високий середній вік населення (старіння населення)	Нові фахівці	Мала чисельність молодого покоління	Перекваліфікація старішого покоління на сучасний рівень знань та навичок. Обмін знаннями та навичками між старшим та молодим поколінням

Таблиця 5.10. Підсумкова таблиця факторів соціо-культурного середовища

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми чи реалізації можливості**
	Можливості	**Загрози**	
Інформатизація суспільства	Швидка комунікація з клієнтами та поставниками	Бракування живого спілкування	Створення зустрічей та виставок, де можна показати наші розробки та зацікавити споживачів наших послуг
Зміна традицій ведення бізнесу в Україні та світі	Зруйнувати недоліки минулої системи ведення бізнесу	Загрози нашій діяльност і	Спроба переконати в нашій правоті
Забуття традицій	Сприяння відновленню втрачених традицій	Втрата культурних цінностей	Сприяння відновленню втрачених традицій

Таблиця 5.11. Підсумкова таблиця факторів природного середовища

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми чи реалізації можливості**
	Можливості	**Загрози**	
Землетруси	Проекти відновлення, консалтингові та аудитні послуги	Руйнування інфраструктури	Проекти по застосуванню інновацій в будівництві
Повені	Проекти відновлення, консалтингові та аудитні послуги	Руйнування інфраструктури	Проекти по застосуванню інновацій в будівництві
Урагани	Проекти відновлення, консалтингові та аудитні послуги	Руйнування інфраструктури	Проекти по застосуванню інновацій в будівництві
Аварії техногенного походження	Проекти відновлення, консалтингові та аудитні послуги	Руйнування інфраструктури	Проекти по застосуванню інновацій в будівництві

Таблиця 5.12. Підсумкова таблиця впливу конкурентів

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми чи реалізації можливості**
	Можливості	**Загрози**	
Конкуренто спроможність підприємства та конкурентів	Виграш в конкуренції	Програш в конкуренції	Постійний пошук способів підвищення конкуренції
Насиченість ринку	Виграш в конкуренції	Програш в конкуренції	Пошук нових ринків та видів послуг: наприклад проектування обладнання для виготовлення стійких емульсій для військового комплексу чи автомобільного
Кількість послуг,	Виграш в конкуренції	Програш в конкуренції	Пошук нових послу, які ще не освоєні конкурентами
Вартість послуг	Виграш в конкуренції	Програш в конкуренції	Пошук способів Зниження вартості надання послуг
Якість послуг	Виграш в конкуренції	Програш в конкуренції	Пошук способів підвищення якості надання послуг, чи створення нових послуг, що не надаються іншими тткомпаніями
Імідж компанії	Більша зацікавленість споживачів	Втрата іміджу недобросовісна та конкуренція	Працювати на покращенням іміджу компанії Залучення засобів масової інформації

Таблиця 5.13. Підсумкова таблиця впливу постачальників

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми чи реалізації можливості**
	Можливості	**Загрози**	
Вартість послуг	Збільшення доходу	Втрата доходу	Намагатися знайти якомога дешевші послуги постачальників. Створити вигідні для обох сторін умови
Якість надання послуг	Збільшення іміджу	Зниження іміджу	Забезпечити я та швидкість постачання. кість
Кількість постачальників	Збільшення доходу	Втрата доходу	Можливість роботи по всій країні та за кордоном

Таблиця 5.14.Підсумкова таблиця впливу контактних аудиторій

Фактори	**Вплив фактору**		**Альтернативні шляхи вирішення проблеми чи реалізації можливості**
	Можливості	**Загрози**	
Зацікавленість	Збільшення зацікавленості	Зниження зацікавленості	Виявити інтерес до нашого проекту місцевих жителів, аудиторів, засобах масової інформації

Проводимо аналіз ринкового середовища: складаємо таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таб. №№ 5.15-5.16). Фактори в таблиці подавати в порядку зменшення значущості.

Таблиця 5.15. Фактори загроз

Фактор	**Зміст загрози**	**Можлива реакція компанії**
Закони України про приватну власність	Несправедливе законодавство	Захист від недобросовісних конкурентів. Боротьба за справедливість правовими методами
Військові дії та нестабільна політична ситуація в Україні	Небезпека утворення нестабільної ситуації та заворушень	Універсальність запропонованих проектів

Закінчення таблиці 5.15

Економічні криза Інфляція, Підвищення цін на сировину.	Впливає на спроможність компаній на модернізацію та переобладнання	Підвищення/пониження ціни на продукти; прив'язка до стійкої валюти
Науково технічне відставання	Не можливість забезпечити конкуренцію в даній галузі	Постійний пошук та моніторинг актуальних тенденцій в обраній та в суміжних сферах діяльності. Залучення наукових робітників до розробки обладнання
Демографічні: Зниження народжуваності та кількості населення	Зниження кількості потенційних споживачів продукції, зниження попиту. Зменшення кількості кваліфікованих кадрів для роботи проекту	Пошук нових клієнтів та проектування обладнання та вихід на міжнародні ринки
Соціо-культурні: Консервативність поглядів споживачів	Небажання споживачів впроваджувати інноваційні рішення	Пояснення споживачам, що пропоновані послуги зможуть підвищити ефективність виробництва
Природні: Пожежі, землетруси, повені, урагани	Руйнування інфраструктури	Протидія стихійним явищам шляхом облаштування захистом від стихійних явищ, пожеж тощо, проведення інструктажу з техніки безпеки

Таблиця 5.16. Фактори можливостей

Фактор	**Зміст можливості**	**Можлива реакція компанії**
Науково-технічні	Залучення молодих та перспективних кадрів та співпраця з вищими навчальними закладами	Розробка нового обладнання та конструкцій змішувальних елементів. Впровадження даної технології та декларування власної ціни на дану пропозицію
Демографічні: Зростання населення	Збільшення попиту на різні типи продукцій	Збільшення числа потенційних клієнтів в майбутньому
Соціально-культурні: Консервативність поглядів споживачів	Небажання споживачів купувати нове обладнання	Пропонувати споживачам замість купівлі нового обладнання модернізацію їх виробництва за допомогою наших послуг
Зростання цін на енерго ресурси	Створення нових проектів енергозбереження для себе та клієнтів	Збільшити рентабельність на енергоаудит та обрати курс на створення енергоефективних виробництв

Закінчення таблиці 5.16

Науково технічне відставання науки та техніки від провідних країн світу	Створення власних тенденцій та технологій	Спроба зацікавити іноземних інвесторів в актуальності нашого проекту
Високий середній вік населення (старіння населення)	Нові фахівці	Перекваліфікація старішого покоління на сучасний рівень знань та навичок. Обмін знаннями та навичками між старшим та молодим поколінням

Надалі проводимо аналіз пропозиції: визначаються загальні риси конкуренції на ринку (таб. 5.17).

Таблиця 5.17. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства**
1. Тип конкуренції Олігополія	Мала кількість спеціалістів здатних виконувати комплексну роботу.	Співпраця з технічними вузами за для покращення якості розробок
2. За рівнем конкурентної боротьби національний	Якісні інженерні послуги необхідні на кожному підприємстві)	Надання консалтингових послуг та можливість співпраці з нашим інженерним відділом)
3. За галузевою ознако ю міжгалузева	На підприємстві працюють працівники здатні надавати широкий спектр послуг	Наша компанія охоплює майже усі напрямки розробок і може надати якісні консалтингові послуги
4. Конкуренція за видами товарів товарно – видова між бажаннями	У нас є типові конструкції та є можливість розробки індивідуальних апаратів	Підприємство орієнноване на малий, середній та великий бізнес. І має можливості проектувати відповідне обладнання
5. За характером конкурентних переваг цінова	Наша компанія буде мати як і конкуренцію по низькій ціні так і на якості продукції. Наша мета робити якісне і дешеве обладнання	Дасть можливість зайняти нішу якісного дешевого обладнання в Україні та світі
6. За інтенсивністю – марочна	Наша мета зробити всесвітньо відомим	Це дасть можливість надавати послуги по усьому світу

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таб. 5.18).

Таблиця 5.18. Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товари-замінники**
Складові аналізу	Всі підприємства які займаються консалтингом і аутсорсингом промислових підприємств	Конкурентом може стати підприємство яке почне співпрацювати з технічними вузами	Постачальники мають малий вплив на наш основний напрям (консалтинг аутсорсинг) але має вплив на напрям по продажу товарів (комплектуючих виробів та ін.)	- не конкурентно спроможне обладнання -високі ціни на товари -не якісні послуги - без інноваційне	–
Висновки:	На даному етапі розвитку в Україні дуже мало підприємств які можуть провести якісну оцінку роботи підприємства та вказати їхні недоліки	На сьогодні будь який мислячий на перспективу інвестор може стати нашим конкурентом почавши співпрацювати з університетами	Так від постачальника буде залежить час поставки комплектуючих та його мінімальна вартість	Клієнту завжди необхідне: конкурентно спроможне обладнання за низькою ціною якісне та інноваційне	Даний пункт не є актуальним для нашого підприємства через те що ми надаємо комплексний спектр послуг

На основі аналізу конкуренції, проведеного в (таб. 5.18), а також із урахуванням характеристик ідеї проекту (таб. 5.4), вимог споживачів до товару (таб.5.5) та факторів маркетингового середовища (таб. № 5.15- 5.16) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за таб. 5.19.

Таблиця 5.19. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Низька ціна	Так як ми будемо співпрацювати з науково технічною базою університету ми зможемо залучати молодих фахівців для розробки обладнання також низька ціна буде через те що обладнання буде продаватися не одному замовнику а буде продаватися на сайті і кожен підприємець зможе його купити
2	Якість(швидкість та надійність)	При замовленні документації покупець отримувати якісну документацію та матиме можливість звернутися до нас консультаціями
3	Комплексний підхід	Ми надаємо комплекс послуг по розробці виготовленню і монтажу а також постачаємо комплектуючі по низьким цінам

За визначеними факторами конкурентоспроможності (таб. 5.19) проводиться аналіз сильних та слабких сторін стартап-проекту (таб. 5.20).

Таблиця 5.20. Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні проектом						
			-3	*-2*	*-1*	*0*	*+1*	*+2*	*+3*
1.	Новизна впровадження проекту	6			◇	△	□		
2.	Швидкість виконання замовлення	10		◇	□		△		
3.	Асортимент послуг	7	△					□ ◇	
4.	Якість кінцевої продукції	8		△		□ ◇			

Закінчення таблиці 5.20

	Сильні сторони	Слабкі сторони
□ – www.globallogic.com	Асортимент, Низька ціна	Швидкість надання послуг, Асортимент
△ – careers.epam.ua	Якість кінцевої продукції;	Асортимент, Висока ціна
◇ – www.softserveinc.com	Асортимент,	Швидкість надання послуг, Новизна впроваджених

Таблиця 5.21. Формулювання управлінської проблеми SWOT- аналіз

Сильні сторони	Слабкі сторони
- новизна проекту; -спроможність проекту збільшити конкурентоспроможність споживачів підприємств промисловості; - низька вартість впровадження проекту (його можна створити навіть власними зусиллями); - можливо збільшити кількість наданих послуг та працювати з обладнанням в інших видах промисловості (фармацевтична, машинобудівна тощо), -більш швидкий вихід товарів на ринок	- низька дохідність проекту внаслідок низької зацікавленості споживачів; - програш іноземним компаніям; - відсутність чітких правил співпраці підприємства та аутсорсера
Можливості	**Загрози**
Науково-технічні Демографічні: Зростання населення Соціально-культурні: Консервативність поглядів споживачів Зростання цін на енерго ресурси Науково технічне відставання науки та техніки від провідних країн світу. Високий середній вік населення (старіння населення)	Закони України про приватну власність Військові дії та нестабільна політична ситуація в Україні Економічні криза Інфляція, Підвищення цін на сировину. Науково технічне відставання Демографічні: Зниження народжуваності та кількості населення. Соціо-культурні: Консервативність поглядів споживачів

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (таб. 5.22).

Таблиця 5.22. Альтернативи ринкового впровадження стартап-проекту

№ п/п	**Альтернатива (орієнтовний комплекс заходів) ринкової поведінки**	**Ймовірність отримання ресурсів**	**Строки реалізації**
1	Загарбник	Значні	Максимум рік
2	Наступник	Суттєві	Максимум рік

Після аналізу обираємо альтернативу наступник**.**

5.4 Розроблення ринкової стратегії стартап-проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 5.23).

Таблиця 5.23. Вибір цільових груп потенційних споживачів

№ п/п	**Опис профілю цільової групи потенційних клієнтів**	**Готовність споживачів сприйняти продукт**	**Орієнтовний попит в межах цільової групи (сегменту)**	**Інтенсивність конкуренції в сегменті**	**Простота входу в сегмент**
1	Малі приватні промислові підприємства	Висока	Високий	Мала	Висока
2	Великі промислові підприємства	Середня	Середній	Висока	Середня

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (табл. 5.24).

Таблиця 5.24. Визначення базової стратегії розвитку

№ п/п	**Обрана альтернатива розвитку проекту**	**Стратегія охоплення ринку**	**Ключові конкуренто-спроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку***
1	Наступник	Концентрація на потребах одного цільового сегменту	Надання інженерних консалтингових послуг малим промисловим підприємствам	Стратегія спеціалізації

Наступним кроком є вибір стратегії конкурентної поведінки (таб. 5.25).

Таблиця 5.25 – визначення базової стратегії конкурентної поведінки

№ п/п	**Чи є проект «першопрохідцем» на ринку?**	**Чи буде компанія шукати нових споживачів, або забирати існуючих конкурентів?**	**Чи буде компанія копіювати основні характеристики товару конкурента, і які?**	**Стратегія конкурентної поведінки***
	На території України проект є першопроходьцем, для малих і середніх підприємств.	В планах компанії пошук нових споживачів та розширення своєї діяльності	Копіювання популярних послуг на ринку такі як: - Розробка програмного забезпечення - Аудит підприємства - Архітектура та дизайн	Стратегія виклику лідера

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 5.5), а також в залежності від обраної базової стратегії розвитку (таб. 5.24) та стратегії конкурентної поведінки (таб. 5.25) розробляється стратегія позиціонування (таб. 5.26). що

полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну проект.

Таблиця 5.26. Визначення стратегії позиціонування

№ п/п	Вимоги до аудиторії цільової	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції власного стартап проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту*
	Якісне надання послуг	Стратегія спеціалізації	Стратегія виклику лідера	Конкуренто-спроможне та іноваціне обладнання, за низькою ціною

5.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 27 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.27 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує послуга	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Потреба в надійному та високо кваліфікованому аутсорсингу	Швидкість, надійність(надання консультацій в продовж року) та комунікабельність персоналу	Швидкість, не висока ціна, надійність та комунікабельність персоналу

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея послуги, його фізичні складові, особливості процесу його надання (таблиця 5.28). Орієнтовний перелік можливих характеристик товару

наведено у методиці [34].

До основних техніко-економічних характеристик товару відносяться:

− економічні – вартість обслуговування, експлуатації, утилізації, витратних матеріалів, ремонту, знижки;

− призначення (технічні) – показники, що визначають головний напрямок використання товару та можливу сферу його застосування: класифікаційні показники, складу і структури, технічної досконалості;

− надійності – здатність товару безвідмовно функціонувати: безвідмовність, довговічність, ремонтопридатність;

− технологічні – можливість оптимізації витрат матеріалів, праці, коштів, часу під час технологічної підготовки виробництва, виготовлення та використання товару;

− ергономічні – показники ступеню адаптованості технічних та конструктивних рішень виробу до біологічних властивостей людини та середовища використання товару: гігієнічні, антропометричні, фізіологічні та психологічні;

− органолептичні – визначають властивості товару, які людина може визначити за допомогою своїх органів чуття;

− естетичні – оцінюють зовнішній вигляд товару;

− транспортабельності – визначають пристосованість продукції до транспортування, підготовчих, початкових і кінцевих операцій перевезення;

− екологічності – характеризують рівень негативного впливу на довкілля;

− безпеки – безпечності та нешкідливості споживання товару.

Формулюємо три рівні товару: товар за задумом, товар у реальному виконанні та товар із підкріпленням. Далі розглядаємо техніко-економічні характеристики кожного рівню товару, отримані дані вносимо до таб.5.28.

Таблиця 5.28. Опис трьох рівнів моделі товару

<table>
<tr><th>Рівні товару</th><th colspan="3">Сутність та складові</th></tr>
<tr><td rowspan="2">І. Послуга за задумом</td><td colspan="3">Опис базової потреби споживача, яку задовольняє послуга (згідно концепції), її основної функціональної вигоди</td></tr>
<tr><td colspan="3">Надання надійного та високо кваліфікованому аутсорсингу, що дозволяє зменшити витрати на утримання штату працівників</td></tr>
<tr><td rowspan="6">ІІ. Послуга у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Економічності: зниження затрат на ремонт обладнання, зменшення простоїв виробництва;
2. Призначення: хімічні, харчові та будівельні компанії.
3. Технологічні: оптимізації витрат праці та часу.
4. Ергономічність: зручність та доступність до всіх елементів конструкції;
5. Безпека: відповідність нормативам;
6. Екологічність: відповідність нормативам</td><td>–/+</td><td>+/+/+/+/+</td></tr>
<tr><td colspan="3">Якість: стандарти, нормативи, параметри тестування міжнародні та вітчизняні стандарти ДСТУ, ISO, DIN та інші</td></tr>
<tr><td colspan="3">Документи виконані з логотипом підприємства</td></tr>
<tr><td colspan="3">Марка: *</td></tr>
<tr><td rowspan="2">ІІІ. Послуга із підкріпленням</td><td colspan="3">До продажу: представлення клієнту проекту</td></tr>
<tr><td colspan="3">Після продажу: гарантійні консультації</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: Використання власних запатентованих розробок та методів оптимізації, консультування та шляхів розв’язку проблеми</td></tr>
</table>

Захист буде організовано за рахунок захисту ідеї товару у патентному відомстві.

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таб. 5. 21):

Таблиця 5.29. Формування системи збуту

№ п/п	**Специфіка закупівельної поведінки цільових клієнтів**	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
1.	Замовлення проекту	Швидкість виконання, надійність(надання консультацій в продовж року)	Глибока	Власні сили

При визначенні оптимальної системи збуту було вирішено, що ми будемо проводити збут власними силами.

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таб. 5.29).

Таблиця 5.30. Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	**Канали комунікацій, якими користуються цільові клієнти**	**Ключові позиції, обрані для позиціонування**
Орієнтована на використання комунікації, що визначається особливістю галузевої приналежності	Інтернет, виставки, друкована продукція	Спеціалізовані виставки такі як: Міжнародна спеціалізована виставка ТЕХНОЛОГІЇ ЗАХИСТУ/ПОЖТЕХ; Міжнародний авіакосмічний салон АВІАСВІТ; Галузева експозиція ПЕК УКРАЇНИ; Міжнародна спеціалізована виставка гірничодобувної промисловості MINING INDUSTRY EXPO; Міжнародна спеціалізована виставка НАФТОГАЗЕКСПО; Міжнародний водний форум AQUA UKRAINE;

Закінчення таблиці 5.30

		Міжнародна спеціалізована виставка PLAST EXPO UA. На виставках буде розповсюджуватися друкована продукція. В мережі інтернет буде здійснюватися, адресна розсилка комерційних пропозицій за базою даних потенційних клієнтів, також буде розміщено рекламу на основних профільних сайтах та в соціальних мережах

Результатом пункту 5.3 є створення ринкової програми, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого буде впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

Висновки до розділу

Відповідно до проведеного аналізу перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження, стан конкуренції, конкурентоспроможність проекту поява даного проекту є актуальна так як на ринку мала кількість компаній яка надає такий спектр послуг. І має перспективи росту на ринку послуг який відновлюється.

Розроблена система для автоматизації процесу тестування користувацького інтерфейсу на предмет доступності, що буде впроваджена у даному стартап проекті.

ВИСНОВКИ

Забезпечити доступність мобільних додатків було складно через не лише відсутність широти в типах тестів, що надаються сьогодні, а й значну кількість налаштувань, часу та знань, необхідних для запуску тестів. Тестування системи стосується обох цих питань; система тестування доступності, яка тестує як статичні, так і динамічні проблеми доступності, і яку можна легко розширити для обробки інших вимог щодо доступності, які покладаються на статичні або динамічні дані про програму. Крім того, система надає простий, але інформативний інтерфейс для тестування, який вимагає розробника, що дорівнює нулю. З цим збільшенням отримавши інформацію про результати та зменшивши зусилля розробника, також має потенціал покращити доступність багатьох існуючих та майбутніх мобільних додатків.

Загалом можна зробити наступні висновки:

− проаналізовано предметну область – розробки мобільних додатків та тестування їх доступності;

− виявлено стан розвитку вирішення проблеми в Україні;

− обрано мови програмування – Java та Kotlin;

− в якості системи розробки було обрано – IntelliJ IDEA;

− для представлення схеми поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки було обрано Model-View-Controller;

− створено архітектуру системи;

− розроблено алгоритм автоматичного генерування та зменшення простору стану інтерфейсу користувача в контексті конкретного аналізу;

− створено процес тестування питань динамічної доступності, таких як навігація по клавіатурі та спонтанні зміни контексту програми через побудову державної машини, що представляє додаток;

− створена платнево-агностична мова, що використовується для опису інтерфейсів користувача, включаючи аудіо, фізичні та рухові елементи;

− автоматизовано процес запуску, навігації та запису програми Android для спроби повністю вивчити простір стану програми.

− було представлено систему автоматичного пошуку як статичних, так і динамічних проблем із доступністю в мобільних додатках, зокрема на Android;

− тестування роботи програми на декількох програмах з відкритим кодом для вирішення проблем доступності;

− результати тестування порівняно з аналогічними тестами сканера доступності Google і мануального тестування.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Accessibility testing [Електронний ресурс] - Режим доступу: https://www.w3.org/wiki/Accessibility_testing#Introduction.

2. Shell try to ensure websites conform to the "Double-A" conformance level of WCAG 1.0 [Електронний ресурс] - Режим доступу: https://www.shell.com/home/content/footer/accessibility/statement.html.

3. Звіт про виконання нормативу з працевлаштування осіб з інвалідністю [Електронний ресурс] - Режим доступу: https://uteka.ua/publication/commerce-12-zarplaty-i-kadry-3-otchityvaemsya-o-vypolnenii-normativa-po-trudoustrojstvu-lic-s-invalidnostyu.

4. The Centers for Disease Control and Prevention in Ukraine [Електронний ресурс] - Режим доступу: https://www.cdc.gov/globalhealth/countries/ukraine/default.htm.

5. Caldwell B., M. Cooper M., Reid L. G. Web content accessibility guidelines (wcag) 2.0 : WWW Consortium (W3C), 2008.

6. Macedo J., Gouveia T., Florentin F. Usability require-ments for mobile accessibility: a study on the vision impairment. 14th Inter-national Conference on Mobile and Ubiquitous Multimedia (MUM ’15): The OX Association for Computing Machinery, ACM . New York, NY, USA, 2015.

7. Android Accessibility Help. [Електронний ресурс] - Режим доступу: https://support.google.com/accessibility/ android/?hl=en#topic=6007234. Google Support.

8. Android Accessibility Help. [Електронний ресурс] - Режим доступу: https://support.google.com/accessibility/ android/?hl=en#topic=6007234.

9. Get started with Accessibility Scanner. [Електронний ресурс] - Режим доступу: https://support.google.com/ accessibility/android/answer/6376570?hl=en.

10. AccessibilityDetector.java. [Електронний ресурс] - Режим доступу: https://android.googlesource.com/platform/ tools/base/+/studio-3.0/lint/libs/lint-checks/src/main/java/com/ android/tools/lint/checks/AccessibilityDetector.java.

11. AccessibilityNodeInfo. [Електронний ресурс] - Режим доступу: https://developer.android.com/reference/android/view/accessibility/AccessibilityNodeInfo.

12. Accessibility checking. [Електронний ресурс] - Режим доступу: https://developer.android.com/training/testing/ espresso/accessibility-checking.

13. AccessibilityChecks - Robolectric. [Електронний ресурс] - Режим доступу: http://robolectric.org/javadoc/latest/ org/robolectric/annotation/AccessibilityChecks.html.

14. Project Eyes-Free. [Електронний ресурс] - Режим доступу: https://github.com/rmtheis/eyes-free.

15. M. Linares Vasquez, C. Bernal-Cardenas, K. Moran, and D. Poshyvanyk. How do Developers Test Android Applications? arXiv e-prints, January 2018.

16. Адам Фрімен. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов, 4-е издание = Pro ASP.NET MVC 4, 4th edition. — М.: «Вильямс», 2013. — 688 с.

17. Toni Sellarès. The Model View Controller: a Composed Pattern : The OX Association for Computing Machinery, ACM . New York, NY, USA, 2007.

18. Jing Li. thyrlian/AndroidSDK. [Електронний ресурс] - Режим доступу: https://github.com/thyrlian/AndroidSDK.

19. openstf. openstf/minicap. [Електронний ресурс] - Режим доступу: https://github.com/openstf/minicap.

20. Daniel Kim (dkim0419). Soundrecorder. [Електронний ресурс] - Режим доступу: https://github.com/dkim0419/SoundRecorder.

21. alex-p (StackOverflow). Is there a way to get current activity’s layout and views via adb? [Електронний ресурс] - Режим доступу:

https://stackoverflow.com/questions/26586685/is-there-a-way-to-get-current-activitys-layout-and-views-via-adb.

22. Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, UIST '17, pages 845–854, New York, NY, USA, 2017. ACM.

23. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition, page 595–601. The MIT Press, 3rd edition, 2009.

24. Android Developers. UiDevice. [Електронний ресурс] - Режим доступу: https://developer.android.com/reference/ android/support/test/uiautomator/UiDevice.

25. Web Content Accessibility Guidelines 2.0, W3C World Wide Web Con-sortium Recommendation 11 December 2008, Success Criteria 1.1.1, 1.4.3, 1.4.6, 2.1.1, 2.4.6, 2.5.5, 3.1.5, 3.2.1, 3.2.5. https://www.w3.org/TR/2008/ REC-WCAG20-20081211/. Accessed Dec 29, 2018.

26. Web content accessibility guidelines (wcag) 2.0. W3C Recommendation 11 December 2008.

27. WCAG Working Group. Relative luminance. [Електронний ресурс] - Режим доступу: https://www.w3.org/WAI/GL/ wiki/Relative_luminance.

28. W3C. WCAG 2.0 - Contrast Ratio (Note). [Електронний ресурс] - Режим доступу: https://www.w3.org/TR/WCAG21/ #dfn-contrast-ratio.

29. Thomas Porter and Tom Duff. Compositing digital images. SIGGRAPH Comput. Graph., 18(3):253–259, January 1984.

30. P. Maziewski, P. Suchomski, B. Kostek, and A. Czyzewski. An intuitive graphical user interface for the parkinson's disease patients. In 2009 4th International IEEE/EMBS Conference on Neural Engineering, pages 14–17, April 2009.

31. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition, page 558–565. The MIT Press, 3rd edition, 2009.

32. project-travel mate. Travel-mate. [Електронний ресурс] - Режим доступу: https://github.com/project-travel-mate/ Travel-Mate.

33. Naman Dwivedi (naman14). [Електронний ресурс] - Режим доступу: Timber. https://github.com/naman14/Timber.

34. Розроблення стартап-проекту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.

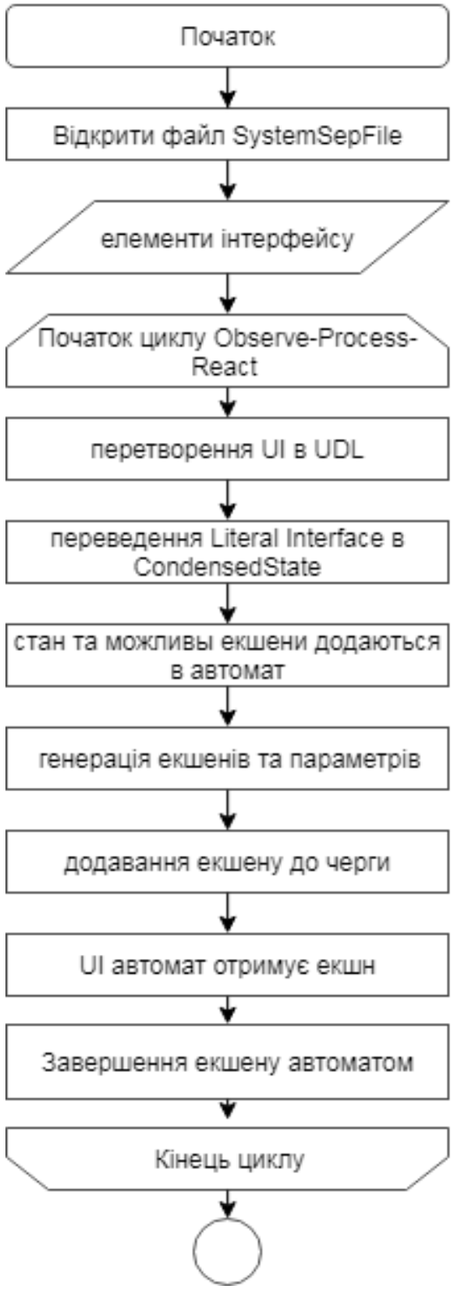
Додаток А

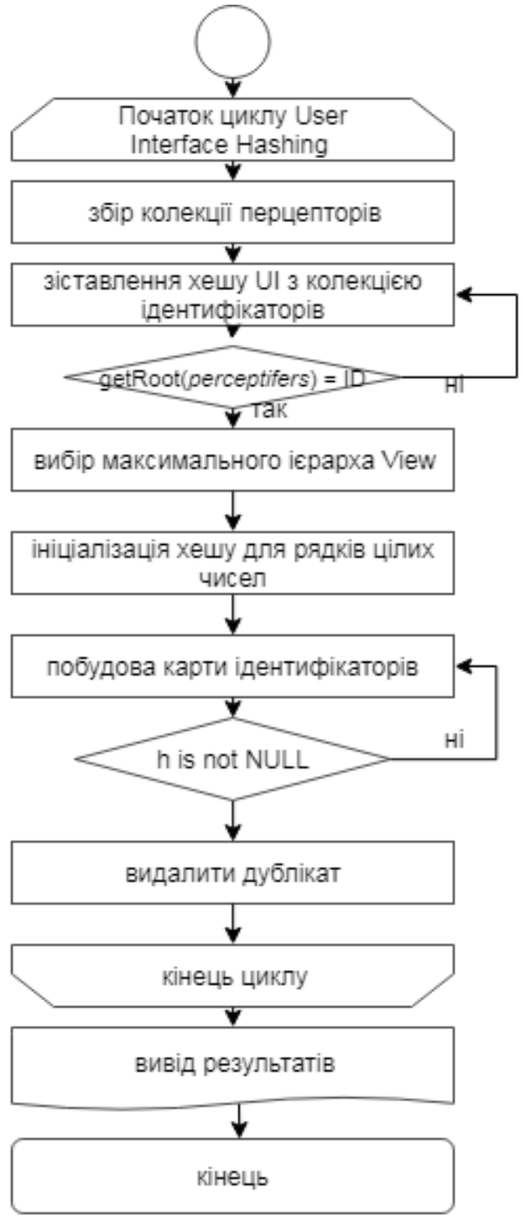
Графічні матеріали

Додаток Б

Результат перевірки на співпадіння

АЛГОРИТМ РОБОТИ СИСТЕМИ

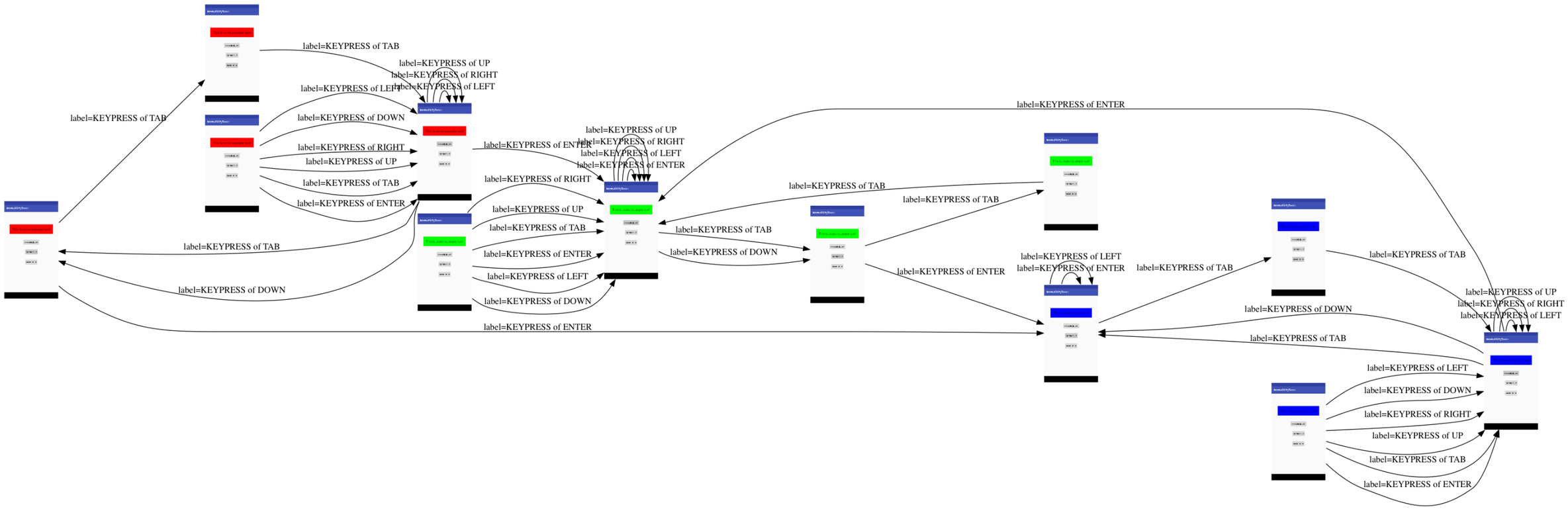




Демонстраційний плакат 1
до магістерської дисертації на тему
Автоматизована система для тестування доступності мобільних додатків

Розробила: Кострикіна Діана
Прийняв: _____

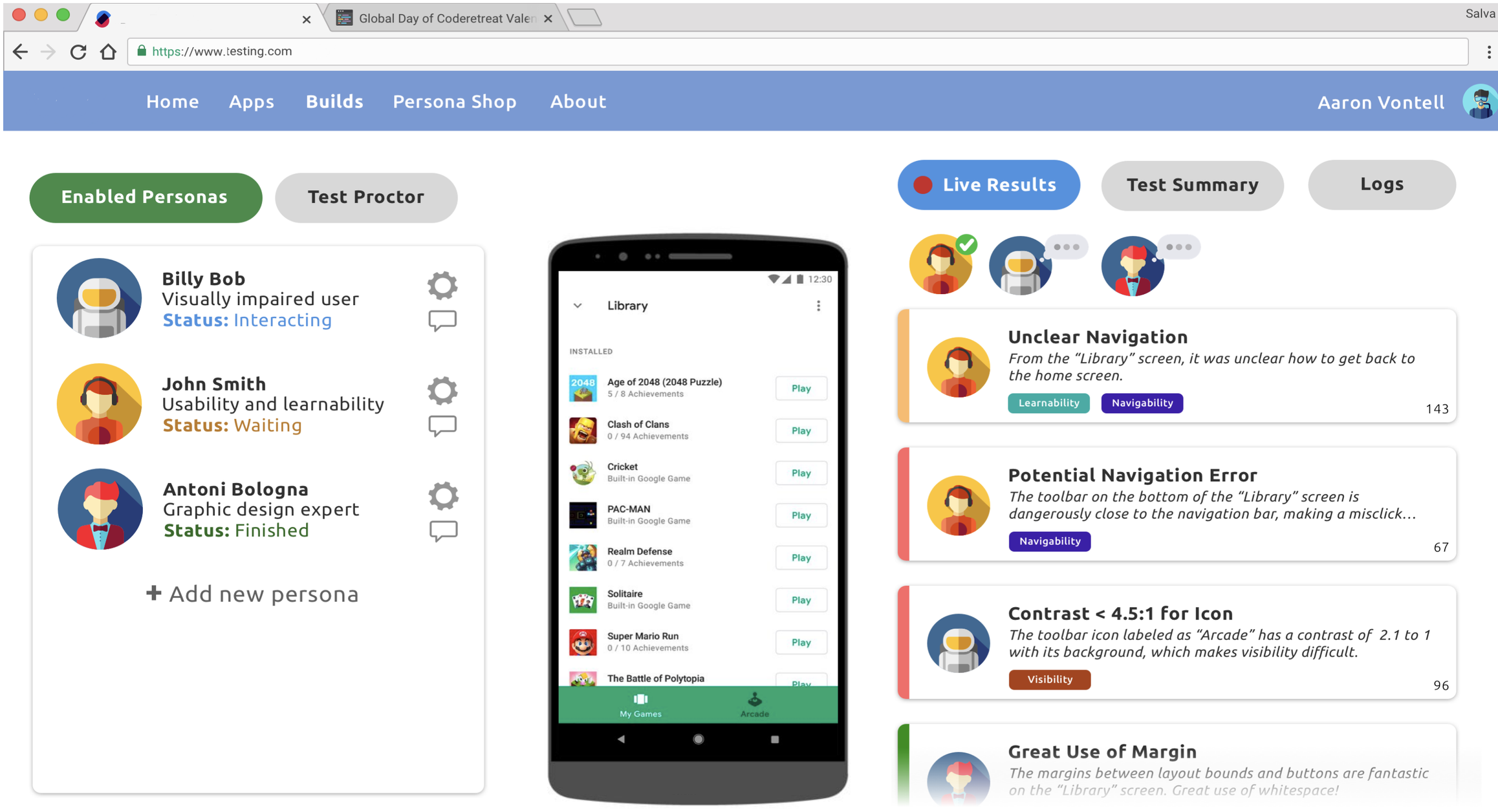
ДІАГРАМА ПЕРЕХОДІВ



Демонстраційний плакат 3
до магістерської дисертації на тему
Автоматизована система для тестування доступності мобільних додатків

Розробила: Кострикіна Діана
Прийняв: _____

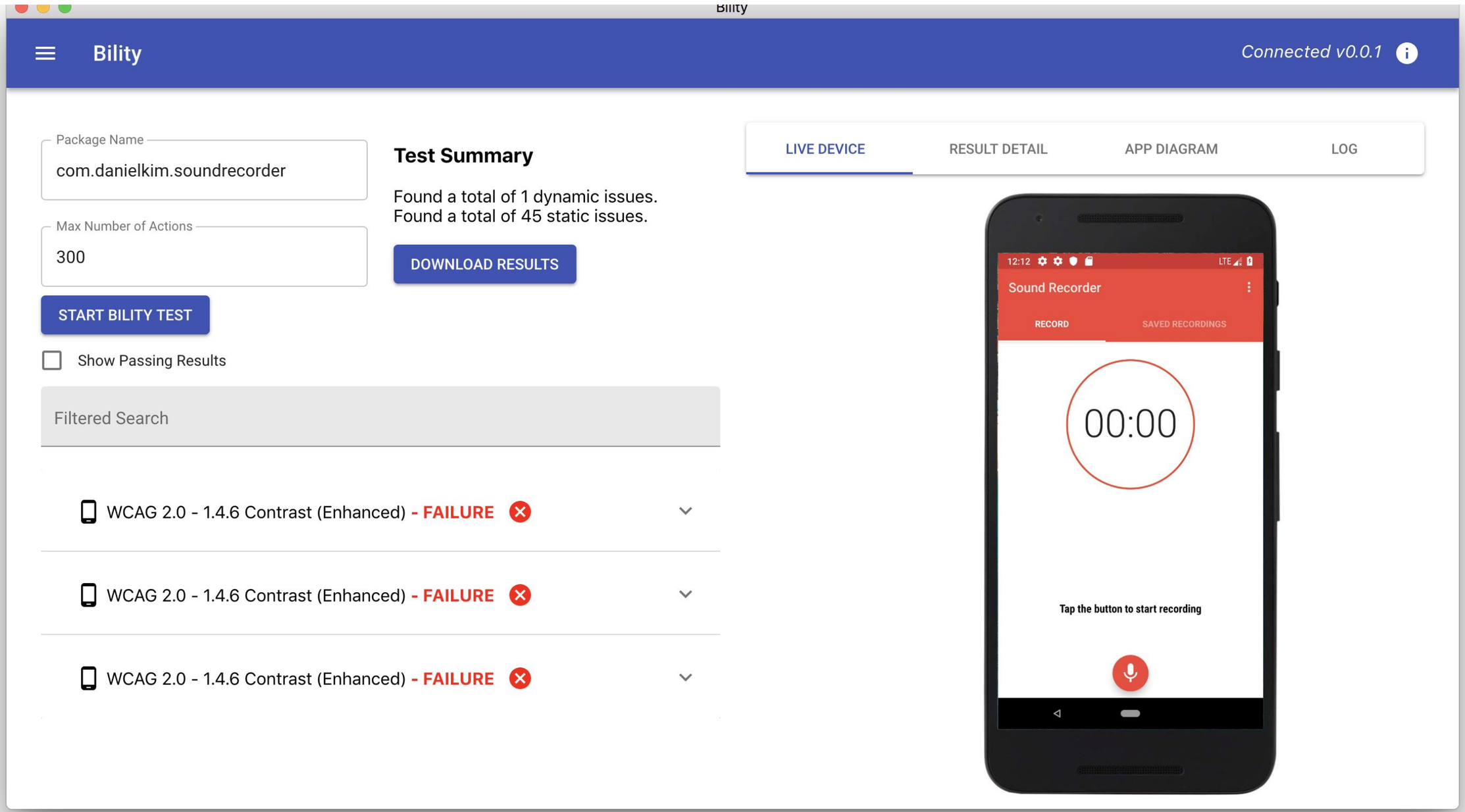
ГОЛОВНА СТОРІНКА СИСТЕМИ



Демонстраційний плакат 5
до магістерської дисертації на тему
Автоматизована система для тестування доступності мобільних додатків

Розробила: Кострикіна Діана
Прийняв: _____

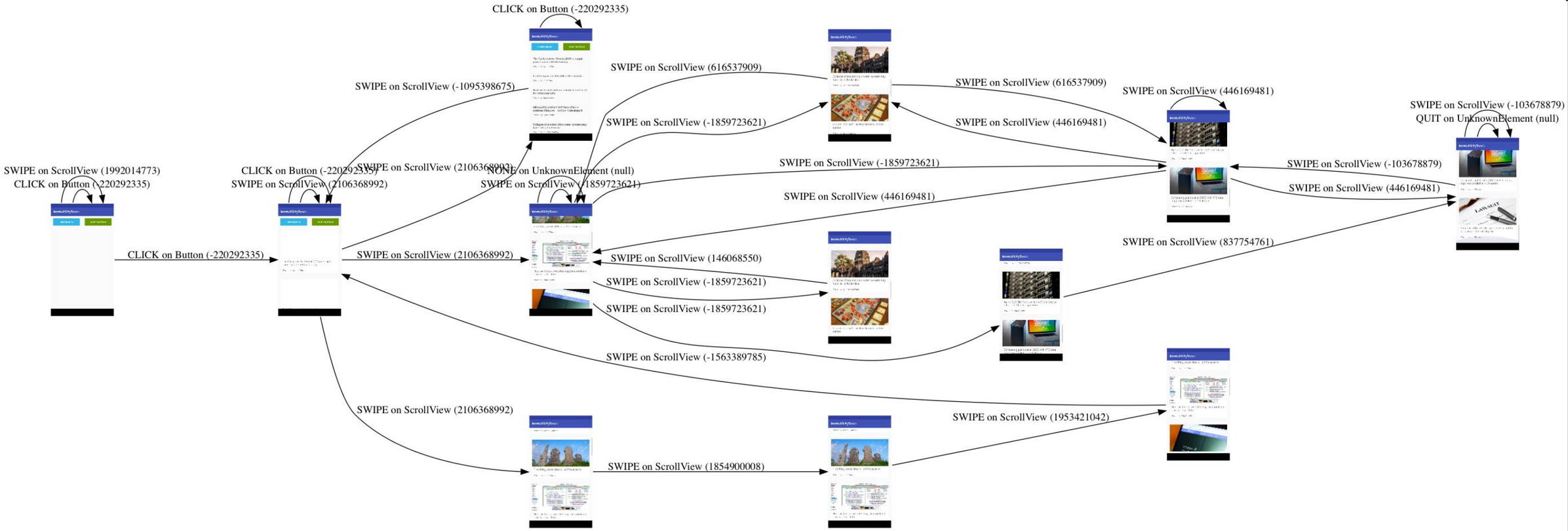
РЕЗУЛЬТАТ РОБОТИ



Демонстраційний плакат _6_
до магістерської дисертації на тему
Автоматизована система для тестування доступності мобільних додатків

Розробила: Кострикіна Діана
Прийняв: _____

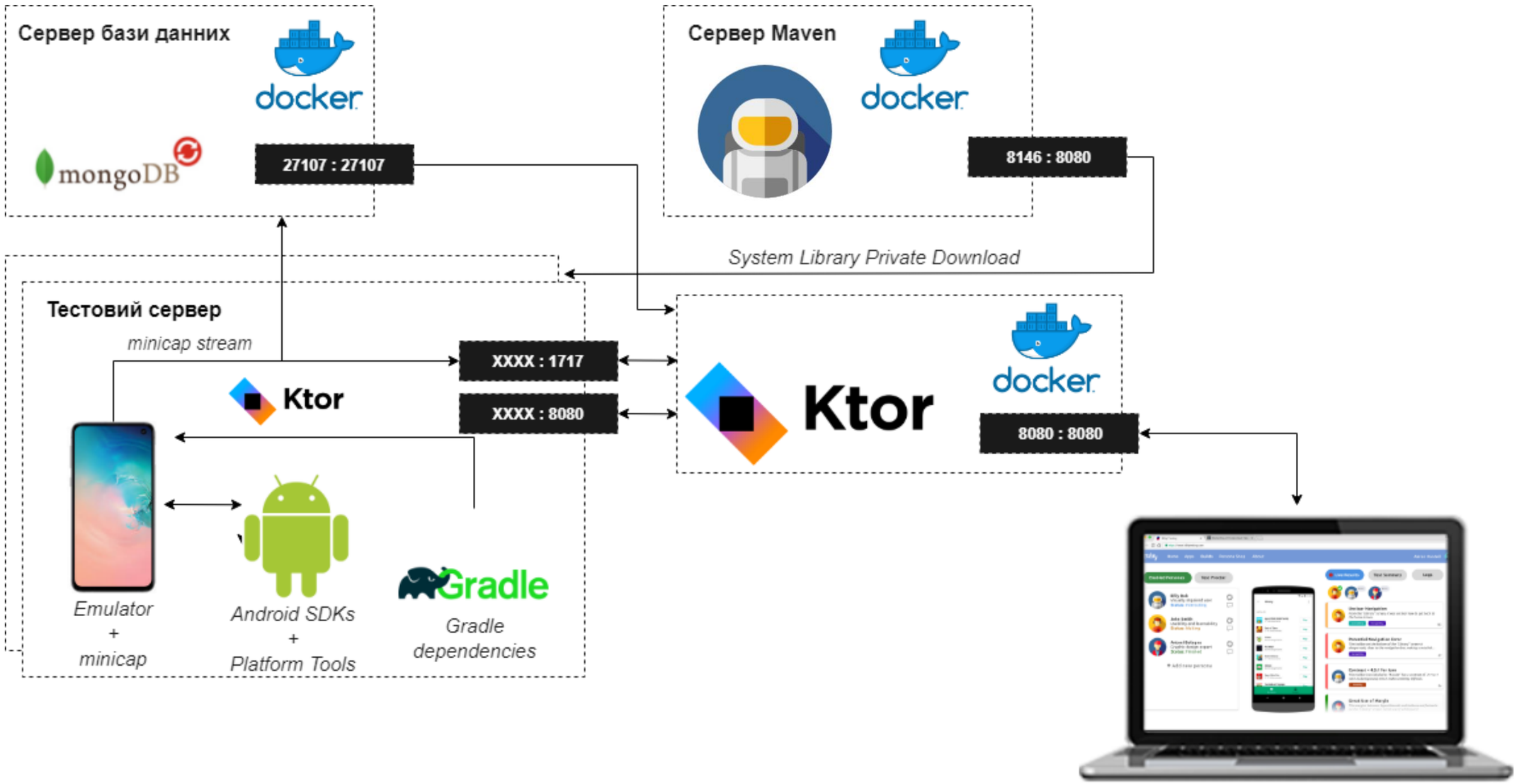
ДІАГРАМА СТАНІВ АВТОМАТА



Демонстраційний плакат 4
до магістерської дисертації на тему
Автоматизована система для тестування доступності мобільних додатків

Розробила: Кострикіна Діана
Прийняв: _____

АРХІТЕКТУРА СИСТЕМИ



Демонстраційний плакат 2
до магістерської дисертації на тему
Автоматизована система для тестування доступності мобільних додатків

Розробила: Кострикіна Діана
Прийняв: _____