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

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Наталія АУШЕВА

«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 122 «Комп'ютерні науки»

освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Система оцінки рівня сформованості компетенцій студентів»

Виконав:

Студент IV курсу, групи ТР-81

Волотівський Ілля Вікторович _____

Керівник:

доцент, кандидат військових наук

Онисько Андрій Ілліч _____

Рецензент:

_____

_____ _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2022

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший

спеціальність 122 «Комп’ютерні науки»
освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ

на дипломну роботу студенту

_____ Волотівський Ілля Вікторович _____
(прізвище, ім’я, по батькові)

1. Тема роботи: Система оцінки рівня сформованості компетенцій студентів
керівник роботи: Онисько Андрій Ілліч, кандидат військових наук, доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__” травня 2022р. № _____

2. Строк подання студентом роботи: 10 червня 2022 р

3. Вихідні дані до роботи: мова програмування Swift, середовище розробки Xcode

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): аналіз шляхів оцінки студентів, виділення критеріїв для подальшого порівняння систем, створення моделей для тестування та оцінки, написання візуальної частини та модулів для тестування, створення готового тесту.

5. Перелік ілюстративного матеріалу: «Задача класифікації рівню компетенцій студентів», «Аналіз проблеми класифікації рівню знань, аналіз поточних систем

оцінки, альтернативна система оцінки, поділ знань на рівні, аналіз необхідних знань, засоби розробки

6. Дата видачі завдання ”11” жовтня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	15 жовтня 2021 р.	
2.	Вивчення та аналіз задачі	16 жовтня – 28 грудня 2021 р.	
3.	Розробка архітектури та загальної структури системи	12 січня - 23 лютого 2022 р.	
4.	Розробка структур окремих підсистем	24 лютого – 27 березня 2022 р.	
5.	Програмна реалізація системи	01 квітня - 02 травня 2022 р.	
6.	Оформлення пояснювальної записки	02 - 29 травня 2022 р.	
7.	Захист програмного продукту	02 червня 2022 р.	
8.	Передзахист	06 червня 2022 р.	
9.	Захист	10 червня 2022 р.	

Студент

_____Волотівський Ілля Вікторович_____

(підпис) (прізвище та ініціали)

Керівники роботи:

_____ _____Онисько Андрій Ілліч _____

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою дипломної роботи є дослідження і розробка системи оцінки рівня сформованості компетенцій студенті. Аналіз поточних засобів, методів та систем оцінки, перегляд їх переваг, недоліків та специфічних відмінностей. Створення чітких критеріїв оцінки студентів для зручного контролю рівня знань для самих студентів. Результат призначений для студентів, що хочуть мати можливість чітко зрозуміти свій поточний рівень, та прогалини в знаннях.

Дипломна робота має обсяг 70 сторінок, містить 1 додаток, 16 посилань та 18 рисунків.

Ключові слова: система оцінки, рівень знань, програмування, комерційна розробка, класифікація розробників.

THE SUMMARY

The purpose of the thesis is to study and develop a system for assessing the level of competence of students. Analysis of current assessment tools, methods and systems, review of their advantages, disadvantages and specific differences. Creating clear criteria for assessing students for easy control of the level of knowledge for students themselves. The result is intended for students who want to be able to clearly understand their current level and gaps in knowledge.

Thesis has a volume of 70 pages, contains 1 appendix, 16 references and 18 figures.

Key words: evaluation system, level of knowledge, programming, commercial development, classification of developers.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SQL – стандартна мова для доступу до баз даних і маніпуляції з ними (англ. Structured Query Language)

DB – база даних (англ. database)

DBMS – система контролю баз даних (англ. Database Management System)

PM – менеджер проекту (Project Manager)

Тімлід – посада, що контролює команду розробників (Team Lead)

QA – тестувальник (англ. Quality Assurance)

Soft Architect – архітектор, що створює структуру проектів, або окремих рішень

Hard skills – навички на пряму пов’язані з розробкою

Soft skills – навички, що взаємодіють з оточуючими розробку процесами

Паттерни – шаблони проектування у написанні коду (англ. design patterns)

Git - розподілена або децентралізована система керування версіями

ООП – об’єктно орієнтоване програмування

Clear code – принципи, що до написання коду впроваджені у книзі Роберта Мартіна «Чистий код»

ВСТУП

Ціллю роботи є аналіз програми навчання технічної спеціальності. Через специфіку різних профілів навчання, таке тестування є вузько направленим, хоча за використанням паралелей можна створити аналогічні модулі для різних спеціальностей. У ході роботи будуть створені чіткі критерії потрібні для практичного та комерційного використання знань у комп'ютерних науках та програмуванні, а тестування буде перевіряти знання з пройдених дисциплін відносно цих критеріїв.

Таким чином студент зможе перевірити свій рівень не відносно конкретних дисциплін навчального закладу, а відносно однієї з професій.

Розроблена система для тестування рівню сформованості компетенцій студента, з яким він зможе чітко розуміти свої можливості на будь якому з етапів навчання, та що важливіше прогалини у знаннях, які потрібно залатати.

За поточної моделі навчання, тестування зазвичай проводиться для оцінки студентів викладачами. На відміну від стандартів, це тестування буде інструментом не лише класичної оцінки, а і самих тих хто навчається.

Зазвичай рівень розмивається за рахунок об'єднання поточного і остаточного оцінювання. Це тестування буде розрізом знань у моменті.

Кінцевий результат тестування не містить оцінку процесу вивчення, результатом є об'єднане значення компетенції студенту та перевірка цілісності знань у різних галузях.

Такого тестування для оцінки компетенцій студента з точки зору бізнесу та перевірки готовності студента у відкритому доступі у ході дослідження та пошуків знайдено не було.

1 ЗАДАЧА КЛАСИФІКАЦІЇ РІВНЮ КОМПЕТЕНЦІЙ СТУДЕНТІВ

Створення системи оцінки для контролю поточного рівня сформованості компетенцій студентів. За допомогою поєднання тестування для номінальної оцінки, та оцінки навичок окремо. Сбір та аналіз даних для характеристики студента у різних напрямах. Наочне приведення результатів такої системи, її переваг і недоліків.

Метою даної роботи є реалізація системи оцінки рівня сформованості компетенцій студентів, також проведення досліджень і порівнянь з поточними методами оцінки.

Класифікація студентів за рівнем буде поділятись за декількома системами, серед яких оцінка володіння інструментами розробки (hard skills) та володіння додатковими знаннями не пов'язаними напряму з розробкою (soft skills), також буде поділ на підсистеми. Таким чином кінцевий результат системи буде гнучкою системою оцінки рівню компетенції студента, з чіткою орієнтацією на ціль яку переслідує студент.

Завдання, які треба виконати для реалізації системи, такі:

— аналіз шляхів оцінки студентів, які використовують зараз у закладах вищої освіти;

— виділення критеріїв для подальшого порівняння систем з наведенням позитивних та негативних сторін;

— створення моделей для тестування та оцінки знань з розділенням на рівні та типи знань;

— написання візуальної частини та модулів для тестування;

— створення прикладу готового тесту у системі з кінцевими результатами;

Для першої задачі потрібно обробити великий об’єм інформації, що до оцінки студентів, також інформації про запит на знання, на які вміння та навички є попит.

Далі потрібно зробити порівняння цих даних та сортувати їх на окремі частини для послідуючого дослідження. Виділити недоліки та переваги поточних систем, дослідити їх особливості та критерії оцінки, що є важливим і який це дає кінцевий результат.

Для другої задачі потрібно відокремити дані на вхідні та очікувані з подальшим розподілом на типи. Створити чітку класифікацію, критерії оцінки. Порівняти отримані у ході дослідження дані з розподілом вхідних знань на типи, що впливають на кінцевий, очікуваний критерій оцінки.

Далі йде створення кінцевих моделей для оцінки та моніторингу з використанням усієї зібраної інформації.

Останнім йде створення оцінювання, створення тестів та складання запитань. Частину з цих завдань можна виконати за допомогою розробки. Тест для однієї з частин оцінювання реалізуватиметься на мові програмування Swift. У ній буде створено користувацький інтерфейс для тестів.

2 АНАЛІЗ ПРОБЛЕМИ КЛАСИФІКАЦІЇ РІВНЮ ЗНАНЬ

Зараз у більшості вузів використовується ECTS - European Community Course Credit Transfer System (Європейська кредитна трансферно-накопичувальна система). Ця система регулює загальне навантаження за допомогою кількості кредитів на дисципліну, та на весь час навчання. Також її перевагою є можливість використовувати кредити для переведення на інше навчання. Це забезпечує єдину процедуру міждержавного та міжвузівського оцінювання навчання, що дозволяє студентам вимірювати та порівнювати результати навчання, академічне визнання результатів навчання в різних навчальних закладах, а також сприяти заліку. Але оцінювання кожної окремої дисциплін не обмежене такими принципами, наведемо поточний формат оцінки в Україні.

2.1 Аналіз поточних систем оцінки

На разі перевірка знань та навичок студентів з кожного компонента освітньої програми реалізована через результати поточного і підсумкового контролю знань, так засвоєння програмного матеріалу оцінюється не лише за кінцевими результатами, а й за прогресом. Тобто перевірка є систематичною з використанням і поточного контролю, і підсумкового контролю. Викладач заздалегідь інформує що до критеріїв оцінки, також ознайомлює з запланованою програмою. Але загалом виділяють наступні критерії оцінювання відповідей [7]:

- Повнота розкриття питання
- Послідовність думок
- Культура мовлення (лише для усних)
- Аргументованість
- Використання джерел як програмних, так і зовнішніх
- Підведення висновків (лише для усних)

— Акуратність виконання (лише для письмових)

Підсумкова оцінка оволодіння матеріалами навчання поділяє знання на наступні рівні:

1. **Високий рівень.** Студент вільно володіє навчальним матеріалом на підставі вивченої основної та додаткової літератури, аргументовано висловлює свої думки, проявляє творчий підхід до виконання індивідуальних та колективних завдань при самостійній роботі;

2. **Достатній рівень.** Студент володіє певним обсягом навчального матеріалу, здатний його аналізувати, але не має достатніх знань та вмінь для формулювання висновків, допускає несуттєві неточності;

3. **Задовільний рівень.** Студент володіє навчальним матеріалом на репродуктивному рівні або володіє частиною навчального матеріалу, уміє використовувати знання в стандартних ситуаціях;

4. **Низький рівень.** Студент володіє навчальним матеріалом поверхово й фрагментарно;

5. **Незадовільний рівень.** Студент не володіє навчальним матеріалом. Кожну оцінку рівня досягнень студента викладач повинен аргументовано умотивувати;

Така система зручна для контролю знань загалом, але не підходить для встановлення компетенції учнів закладу. Такі оцінки показують рівень знань у процесі та наприкінці, але швидко втрачають актуальність. На разі відсутній інструмент для перевірки комплексних знань та рівню компетенції студентів.

2.2 Критерії оцінки

Поточна система оцінки не має спільного формату оцінки знань. Кожна дисципліна має свій формат як ведення, так і оцінки. Таким чином рівень знань

встановлюється не передбачувано, не систематично і не є абсолютним значенням, що відображає дійсність. Так маючи середню оцінку з одного предмету і вищу з іншого – ситуація у знаннях може буди діаметрально протилежною.

У ході перегляду засобів та результатів, аналіз визначив, що здебільшого такий огріх є необхідним, але для покращення системи оцінки повинен бути мінімізованим, або оптимізованим, де буде окремою частиною.

За основу цієї неточної оцінки було взято принципи співбесід. Стандартний формат співбесіди є наступним:

1. Обговорення деталей влаштування і запиту обидвох сторін – ця частина спрощена до простих класифікацій тестування, найважливішим у ній є чітка та зрозуміла документація до процесу, внесення ясності у критерії оцінки та напрями вивчення;

2. Технічна співбесіда з теорії – ця частина теж може бути спрощена, а саме зменшенням впливу третіх осіб і замінена на тестування з частковим опитуванням. Ціль цієї частини отримати номінальне значення знань у певному аспекті;

3. Технічна співбесіда з практики – ця частина є перевіркою знань менш теоретичних, на кшталт «У якій частини пам’яті зберігається формат», а більш практичних. У ній потрібно виявити всі недоліки, що може пропустити перша частина технічної співбесіди. У ході досліджень, було обрано 2 формати для цих цілей;

3.1. Написання конкретного модулю з теми – ця перевірка є одним з найпопулярніших методів тестування, яким користуються всюди. Цей модуль повинен бути невеликим за обсягами і конкретним. Завчасно потрібно визначити критерії за якими він буде оцінений. Критеріями, що будуть краще відповідати зазначеним цілям є: читабельність написаного коду, справність виконання, логічність та послідовність думок при написанні;

3.2. Перегляд готового модулю написаного іншою людиною - ця перевірка висвітлює досвід розробника і найкраще показує розуміння інструменту розробки;

4. Розмова про досвід – не зовсім перевірка, скоріш підведення підсумків і створення загальної картини і рівню студенту;

3 АЛЬТЕРНАТИВНА СИСТЕМА ОЦІНКИ

В процесі досліджень стало зрозуміло, що система яка повністю відокремлена від особливостей напряму навчання є не оптимальною. Тобто кореляція оцінки повинна залежати не від роботи в певний момент і з певної дисципліни, а повинна відображати поточний спектр знань.

3.1 Поділ знань на рівні

Класифікація рівнів підготовки у контексті навчання на технічній спеціальності пов’язаній з IT (Information Technology), з різним вхідним рівнем знань не є стандартною, її не можна оцінити лише числом. Також потрібно ввести опис відносин між цими «класами», далі буде наведено принципіальні відмінності між ними, що очікують від конкретного рівню. Наприклад студент зазвичай починає навчання з навичками роботи з комп’ютером і без навичок у програмуванні, або з мінімальним досвідом.

Якщо абстрагуватись від навчання, рівні розробника можна поділити на наступні [1]:

1. Без досвіду – відсутність досвіду у програмуванні;
2. "Початківець" – знання деяких з мов програмування на початковому рівні;
3. Trainee - розуміння обраного інструменту і вміння ним користуватися для повноцінної розробки, знання англійської (рівень intermediate);
4. Junior – має хоча б мінімальний досвід комерційної розробки та навички для самостійного виконання поставлених технічних задач (тобто сформованих до ТЗ на рівні розробки);
5. Strong Junior – це скоріше окремий випадок рівня Junior, але доречним буде про нього згадати, Strong Junior є перехідним етапом до Middle, ними називають тих розробників чий рівень одного чи декількох з критеріїв оцінки рівню розробника дійшов до Middle, зазвичай це розробники с

високим рівнем технічної вправності, але або маючих замало досвіду, або розуміння розробки з точки зору бізнесу;

6. Middle – має розуміння бізнес процесів, навички для самостійного виконування поставлених задач, 2 та більше років досвіду комерційної розробки, може ставити технічні задачі;
7. Senior – спеціаліст відповідальний за технічну частину, завдяки знанням та досвіду приносить мінімум втрат і максимум користі бізнесу, досвід більше 4-5 років;

Також є альтернативний шлях, стати архітектором, SCRUM-майстром, проектним менеджером або тімлідом, ці посади потребують інших знань ніж програмування і покращують роботу великих команд.

Людина з відсутнім досвідом у програмуванні за час навчання переходить до рівня початківця, це розуміння базових концепцій розробки, логічних операцій, ініціалізації змінних, зберігання даних та інших початкових знань. Цей рівень можна умовно поділити на дві стадії, на першій студент може написати простий код для чіткого технічного завдання, на другій студент може знайти рішення сам, підкоривши цей другий рівень – розробнику відчиняється шлях до Trainee.

З проведених мною опитувань, збору додаткової інформації та аналізу цього у купі з документацією навчального процесу, за час навчання у ВУЗі, без додаткових навчальних закладів чи курсів – ваш рівень підніметься до Trainee, або за певних взаємодій з кафедрою до Junior чи Strong Junior. Тобто при максимальному вичавлюванні знань та можливостей з навчального процесу – рівень технічних знань може бути досить високим, але використовувати його одразу – не є можливим.

3.2 Аналіз необхідних знань

Знання потрібно поділити на менші компоненти, адже використання числа від 0 до 100 для оцінки знань відносно певного «Рангу» не буде коректним. Коли ми говоримо про знання розробника, перший і очевидний крок поділу, це розділення на Soft skills та Hard skills.

Почнемо з hard skills, тут є розгалуження за:

1. Мовою програмування (C#, Swift, C++, JS та багато інших);

2. Платформою під розробку (Windows, Mac OS, Linux);

3. Типом остаточного продукт;

Розглянемо мови програмування ближче.

Мови умовно поділяються на сучасні та застарілі, так як галузь програмування дуже стрімко рухається вперед – ринок постійно поповнюється новими мовами, таким чином багато мов що були колись новаторськими, потужними та зручними – зараз не є раціональним рішенням задач, тому їх не використовують. Чинників такого зсуву багато, це може бути поява нової ОС (операційної системи), новий формат додатків, чи банальна програш у порівнянні з іншими мовами. Не рідким є випадок коли з’являються дві однакові за сенсом мови і одна з них швидко набирає популярність, витісняючи іншу з ринку. Також важливо згадати про оновлення мови, яскравим і лаконічним прикладом цього можна навести мову Swift та Objective-C, обидві мови були створені компанією Apple, Objective-C був мовою для написання всіх додатків під Mac OS та IOS і виконував свої задачі з 1986, але вже чуттєво відставав від інших мов, не дивлячись на постійні обновлення, тому в 2014 році компанією було випущено нову мову програмування, що містила в собі 80% функціоналу свого предку і до того ж була сумісна з ним, тому дуже швидко майже повністю замінила його, адже зручність написання коду на Swift була порядком зручніша та простіша.

Говорячи про платформи ми можемо мати на увазі, як систему де ми пишемо код, так і систему для якої пишемо код. Для зручності, не будуть згадуватись специфічні платформи, які або застаріли, або є нішевими, наприклад про побутову техніку. Здебільшого для написання використовують Windows, Mac OS та Linux. Пишуть же для Windows, Mac OS, Linux, Android, IOS або cross platform (програма що працює на декількох платформах одразу, умовно до неї можна віднести веб додатки).

Мова програмування та платформа зазвичай тісно пов'язані між собою і є інструментом для остаточного продукту, тому доречнішим далі буде використовувати саме тип додатку. Таким чином розробників можна поділити на наступні галузі (з поясненням, платформою розробки, платформою користування та мовою або технологією що лежить в основі) [3-6]

1. Game development – це розробка ігор для Windows, Android, IOS та інших, платформою для розробки зазвичай Windows чи Linux, серед мов найпопулярнішими є C# з його бібліотекою Unity та C++ для розробки власного двигуна, або супроводу вже готового (це не єдині варіанти, та найбільш поширені);

2. Mobile development – це розробка додатків для телефонів, розробка не залежить від платформи на якій їх пишуть, але може бути написана для Android або IOS (також є менш відомі, наприклад Google phone), для Android мастодонтом є мова Java, для IOS це Swift;

3. Desktop development – це розробка настільних додатків, платформа написання, та платформа використання продукту зазвичай співпадають, мов для написання таких додатків мабуть більше всього, для Windows частіше всього це C#, для Mac OS це Swift (але через свою новизну, він частково доповнюється Objective-C);

4. Web frontend development – це розробка візуальної частини веб-сторінок, розробка не залежить від платформи на якій її пишуть, такий контент доступний з усіх платформ через звичайний браузер, є мови програмування які пов'язані з Web

frontend, але здебільшого використовується таблиця стилів CSS, мова розмітки HTML та мова скриптів JS, але вже довгий час набирають популярність бібліотеки для JS, на кшталт TypeScript, Angular, React та інші;

5. Backend development – це розробка серверної частини, розробка не залежить від платформи на якій її пишуть, сервер надає доступ до інформації та оброблює її. Вибір мови для серверної архітектури також великий, зазвичай для цього використовують мову програмування Java, але також є популярними варіантами C#, JS, Python, PHP;

З переліку вище можна побачити, що зараз є досить велике різноманіття варіантів чим займатися розробникам, вже не кажучи що процес розробки не кінчається лише на написанні коду.

Говорячи про hard skills ми беремо до уваги не лише технології специфічні для мови, типу додатку або проекту, а також базові навички роботи з кодом.

1. DB (Database);
2. Debag;
3. IDE;
4. Алгоритми та патерни;
5. Git;

Ці 5 інструментів є невід’ємними частинами розробки і без них створення продуктів є або не можливим зовсім, або робить розробку повільною та неякісною.

Продовжимо з soft skills, їх можна здебільшого не розділяти, але важливо чітко позначити, що матиметься на увазі при оцінюванні під soft skills. Ці навички є питанням ресурсу розробника та його взаємодії з командою на роботодавцем. Серед них можна виділити:

— Критичне мислення;

— Навички роботи в команді;

— Лідерство;

— Креативність в знаходженні рішень;

Також поміж цими двома типами навичок – є робота з документацією, вміння пошуку інформації та таке інше, але беручи до уваги результати

Наводячи висновки з неведеної інформації – можна побачити, що знання які використовуються для оцінки рівня компетенції студентів ділять на специфічні для типу розробки та спільні або базові навички.

4 ЗАСОБИ РОЗРОБКИ

Розробка програмного забезпечення — це процес, за якого на мові програмування створюється програмне забезпечення. Цей процес поділяється на написання окремих частин коду, що разом формують функціональну складову додатків або іншого програмного забезпечення.

Для розробки було обрано інструменти від написання настільних додатків під Mac OS, мова програмування Swift з бібліотеками UIKite що працює з відображенням елементів програми, також була використана Cocoapods, адже без неї неможливою стає підтримка додаткових бібліотек. Середовищем для розробки був обраний XCode, завдяки зручному та повному функціоналі цього IDE стає простішим та безпечнішим.

Візуальна частина буде відображатись через базові елементи NSView, NSButton та такі інші, створені та налаштовані через код, без використання StoryBoard. У роботі були використані елементи Objective-C для відпрацювання елементів інтерфейсу користувача при взаємодії з ним.

Конфігурація та налаштування використані здебільшого базові і не були змінені.

Було завантажено CocoaPods через термінал, його було налаштовано та підібрано під поточну версію операційної системи. Через налаштовану конфігурацію CocoaPods – було завантажено бібліотеки для роботи з Realm, та було додано всі відповідні файли до проекту.

4.1 Обрана мова

Вибір для розробки такого додатку лежав здебільшого між Objective-C та Swift, але Objective-C має забагато застарілих технологій та особливостей, адже він побудований на класичному C. Такий вибір збільшив би кількість коду, додав би громіздкість звичайним командам, серед таких неприємних особливостей, у ньому

є купа ключових слів з символом @, що є надлишковим для написання майже всіх типів, та деяких інших ключових слів. Так як Swift не має такого прабатька, то він і не містить у собі таких старих конвенцій. Виклик методів є зменшеним за обсягами, адже у Objective-C є величезний рівень вкладеності при їх використанні. Також Swift не вимагає використання для кожного умовного оператора купи лапок, крапок з комами на кінці усіх рядків.

Xcode і його компілятор можуть з’ясовувати залежності та автоматично виконувати інкрементальні збірки в Swift Як наслідок, повторюване завдання відокремлення заголовного файлу від файлу реалізації залишилося в минулому. Swift об'єднує присутні у Objective-C заголовок (.h) та реалізацію (.m) в один файл коду (.swift). Xcode і компілятор LLVM можуть виконувати роботу за кадром, щоб зменшити навантаження на програміста.

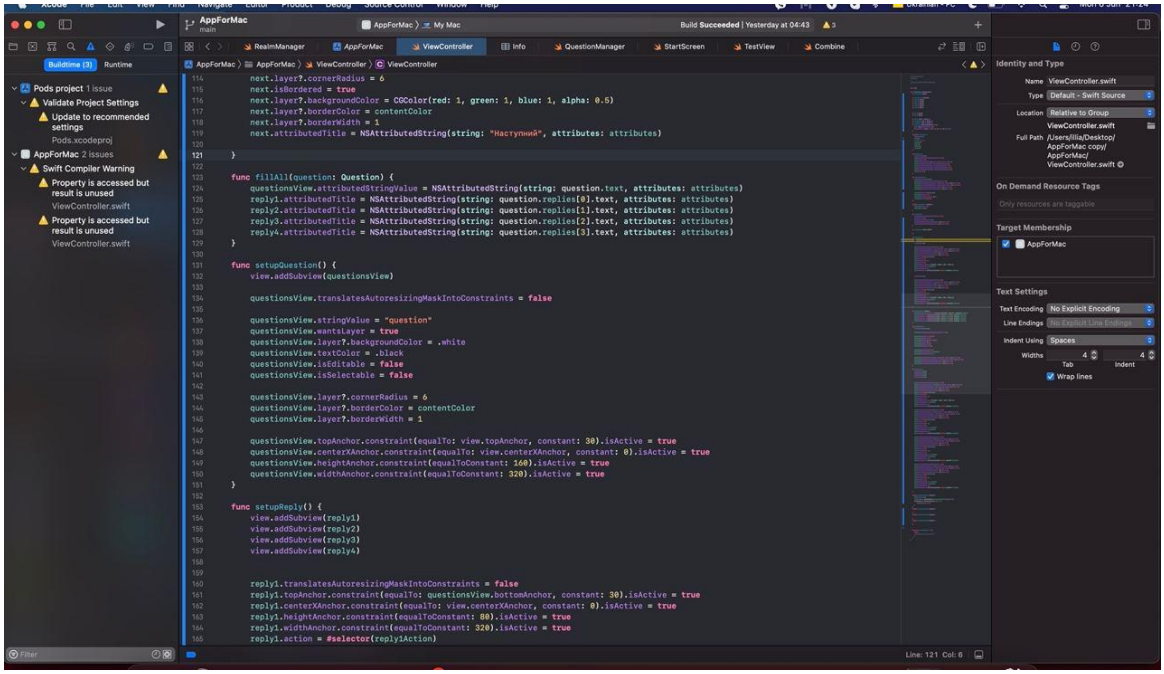


Рисунок 4.1 – Приклад коду на Swift

Мова зручно та ефективно працює зі змінними що можуть містити порожнє значення. Щоб забезпечити передбачувану поведінку, Swift ініціює збій під час виконання, якщо використовується необов’язкова змінна nil. Цей збій полегшує

процес пошуку недоліків і змушує програміста негайно виправити проблему або помилку.

Також мова надає зручні інструменти для роботи з пам’яттю, використовує Automatic Reference Counting (ARC) задля оптимального використання посилань на ділянки пам’яті і робить це автоматично. Тому розробник може не виділяти великий об’єм уваги та часу на менеджмент пам’яті, а лише дотримуючись простих настанов – повністю сконцентруватись на написанні основної логіки.

4.2 Середовище розробки

Xcode – це найпопулярніше середовище розробки на операційній системі Mac OS. Але в нього є купа переваг не лише серед своїх аналогів, а і у цілому в порівнянні з іншими IDE.

Для написання програми не потрібно витрачати купу часу на налаштування середовища, базових бібліотек, конфігурацій та такого іншого, базові налаштування майже не потребують корегування, або використання сторонніх інструментів.

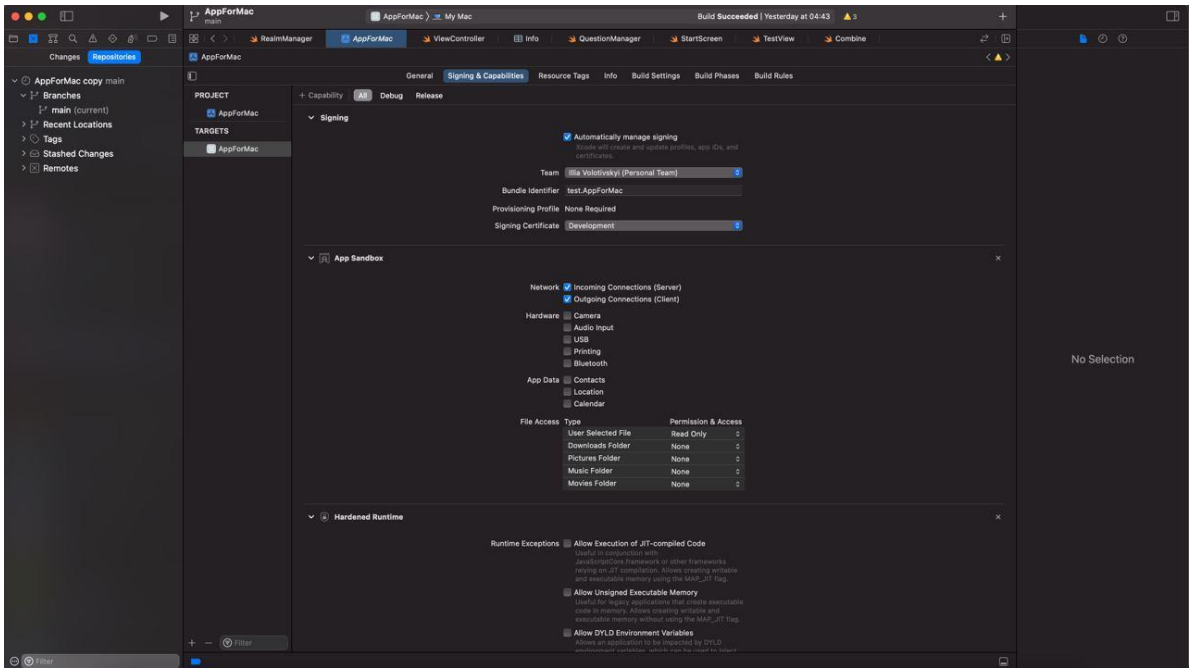


Рисунок 4.2 – Меню налаштування додатку

У Xcode влаштована власна система для роботи з контролем версій та командною розробкою, що знову ж таки допомагає відмовитись від додаткового програмного забезпечення потрібного для написання коду.

Для роботи з системою контролю версій передбачені зручні інструменти порівняння та відслідковування дат та авторів написання коду на різних етапах написання програми.

Гарячі клавіши, підказки фрагментів коду та конструкцій, зручна система налагодження коду та відлову помилок виконання програми, постійно доступна консоль, що містить помилки та влаштоване виведення тексту.

Особливої уваги заслуговує візуальний інспектор, що є одним з базових інструментів Xcode, він відображає додаток у поточному стані у якому його бачить користувач, розділяючи його на окремі елементи, через нього зручно і швидко можна знаходити та лагодити помилки.

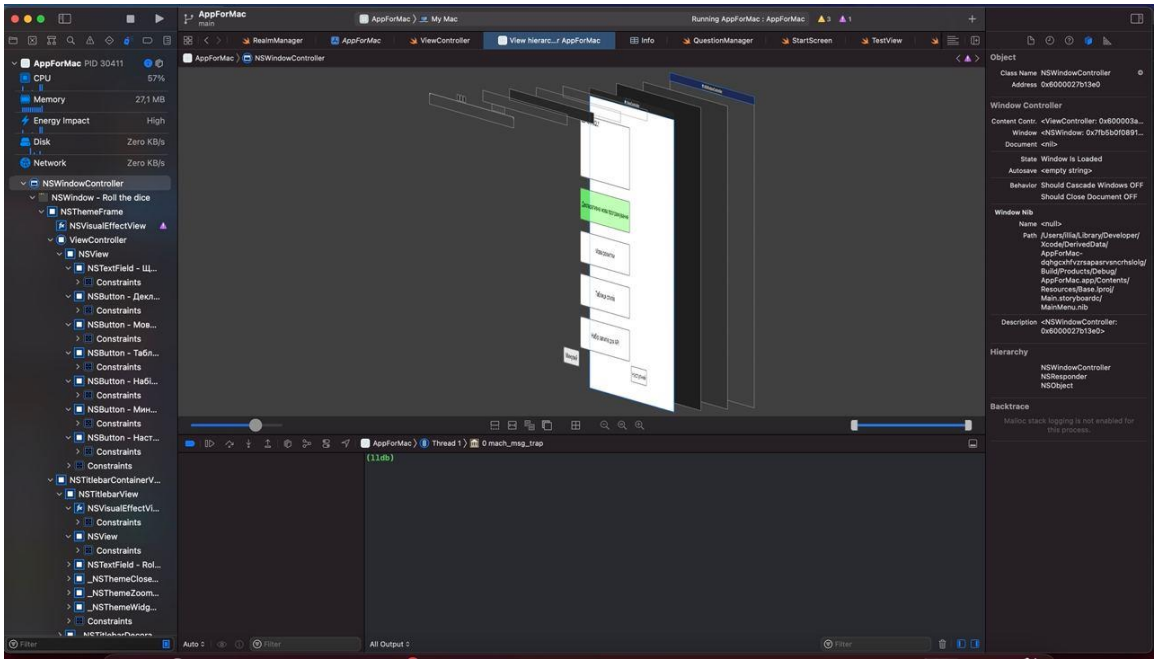


Рисунок 4.3 – Візуальний інспектор

5 ОПИС РЕАЛІЗАЦІЇ

Реалізація тестування буде через віконний додаток на Mac OS з моделями даних, будуть взаємодіяти між собою та формувати кінцеві результати. Таких тестів буде декілька, з причин що далі будуть описані.

5.1 Система оцінки

Займаючись архітектурою додатку, потрібно не лише написати моделі та інтерфейс користувача, а чітко сформувати дані які потрібні для оцінки студента. З огляду на теоретичну частину розділимо та згрупуємо дані.

1. Знання теорії мови

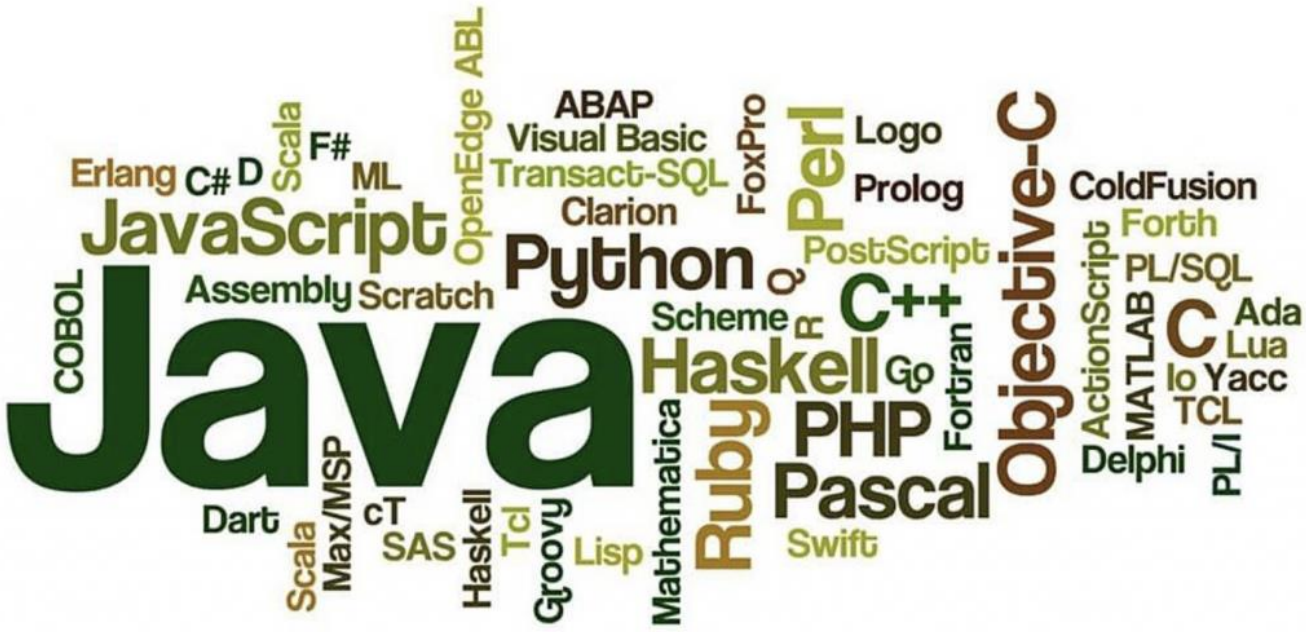


Рисунок 5.1 – Візуалізація кількості мов

2. Знання практики мови

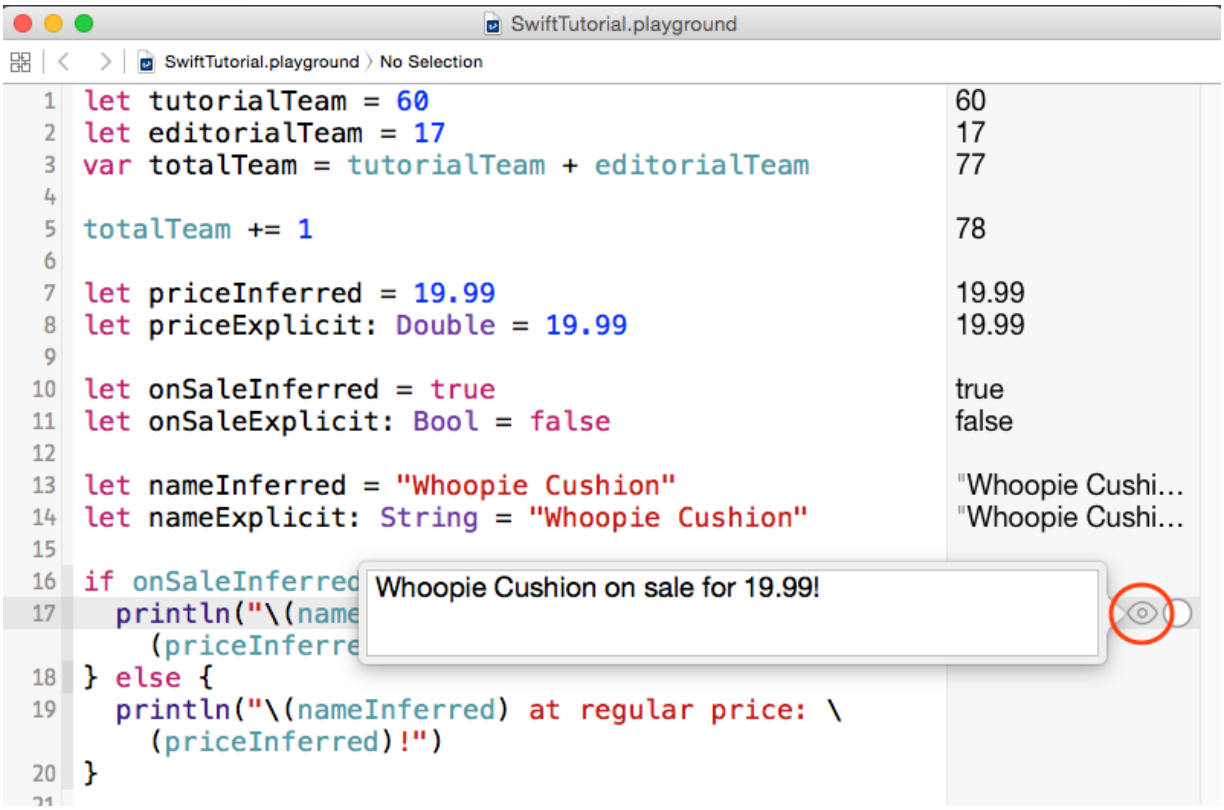


Рисунок 5.2 – Приклад написання коду

3. Знання бібліотек

Рисунок 5.3 – Наведення бібліотек для однієї з мов програмування

4.Знання патернів проектування та алгоритмів

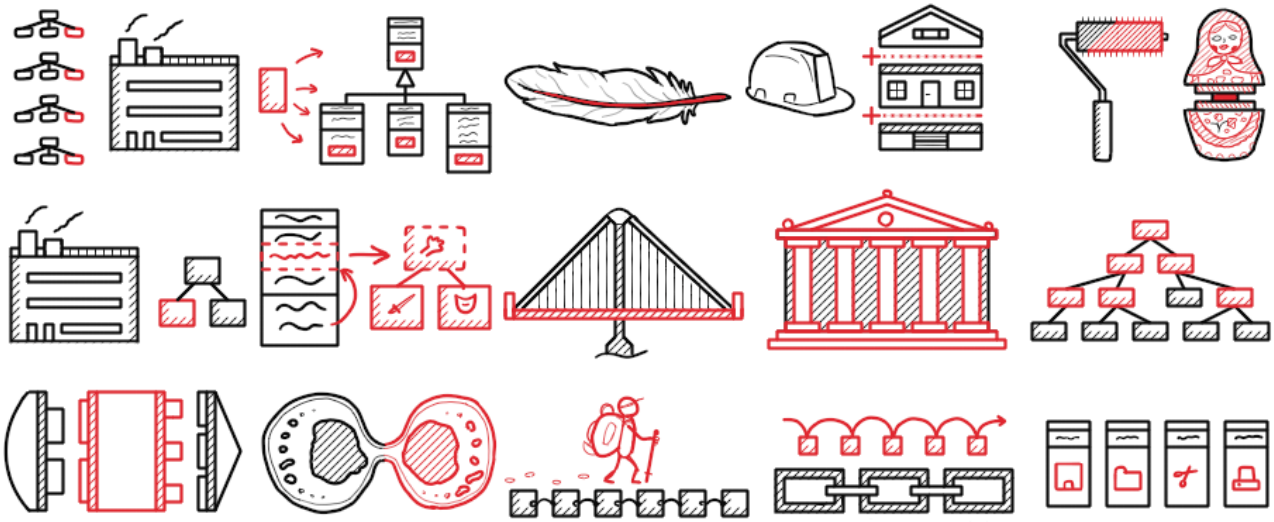


Рисунок 5.4 – ілюстрація деяких паттернів

5.Знання СКБД та SQL

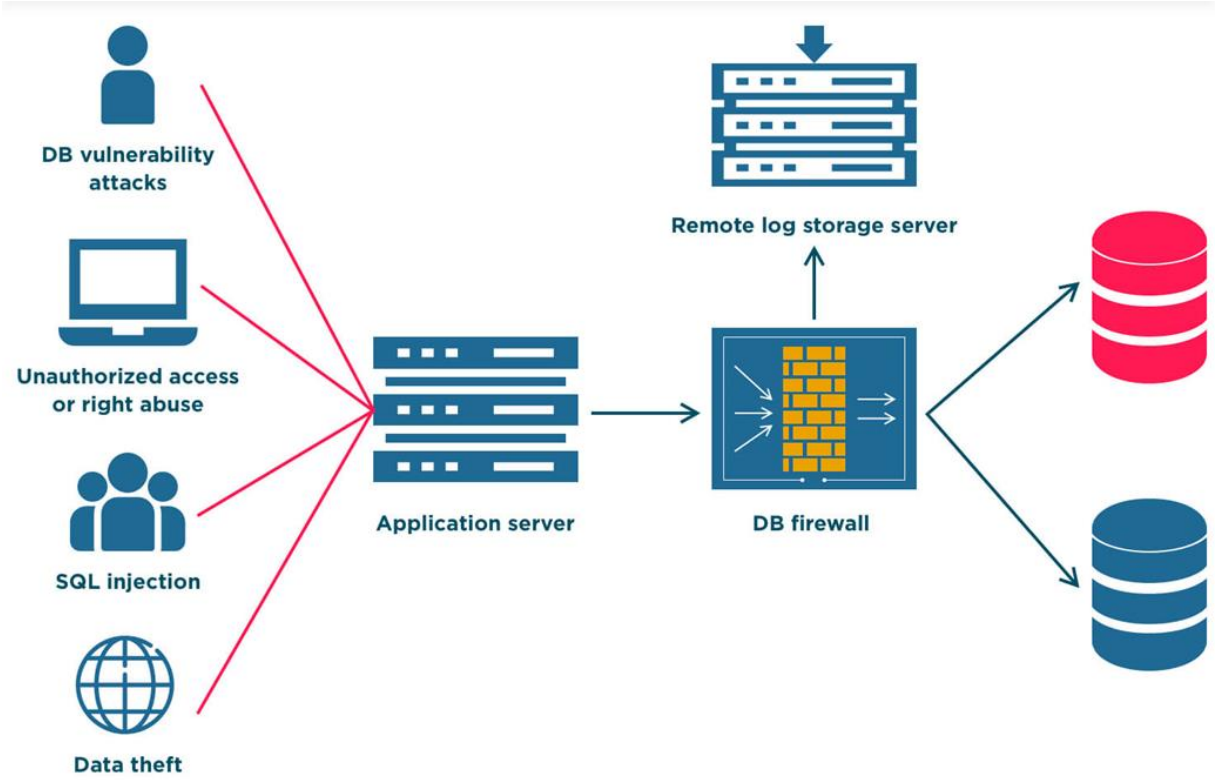


Рисунок 5.5 – Система бази даних

6.Володіння навичками Git

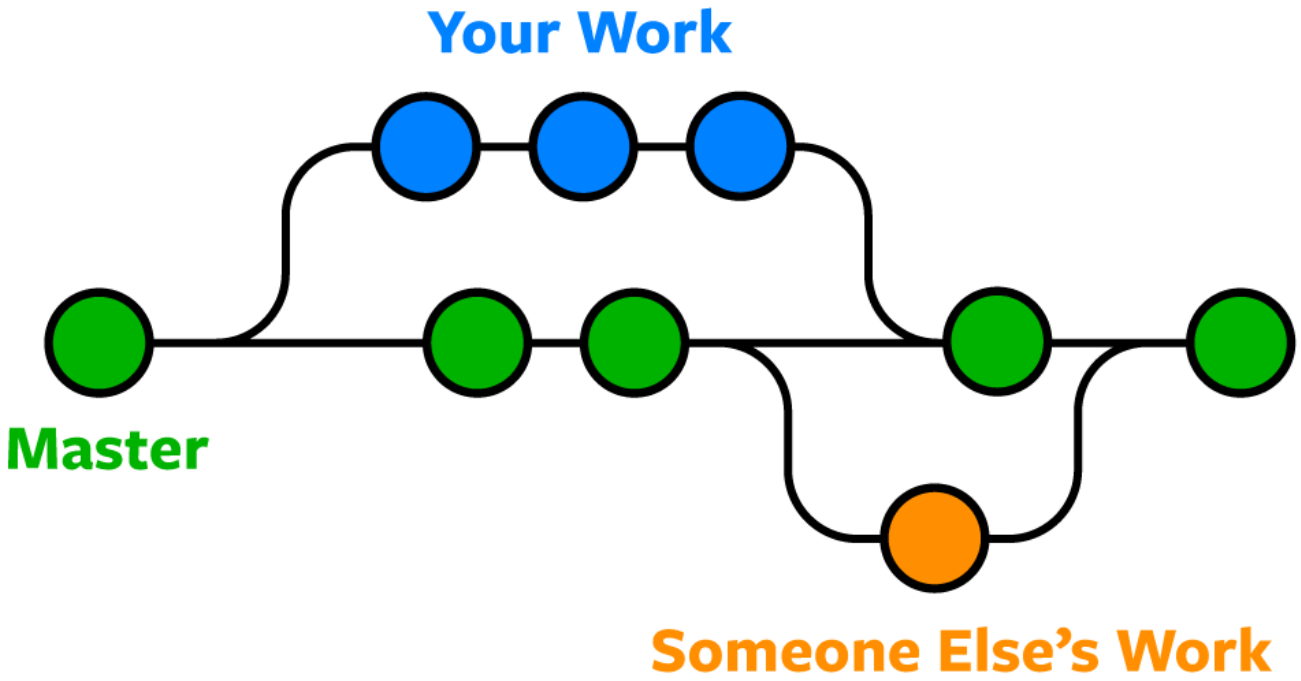


Рисунок 5.6 – Система гілок гіт

Результат технічної частини є відокремленим. Значення результатів це відсоток правильних відповідей на питання з теми, в решті отримуємо 6 значень, також їх можна навести за допомогою діаграми, для наочності було обрано пелюсткову діаграму.

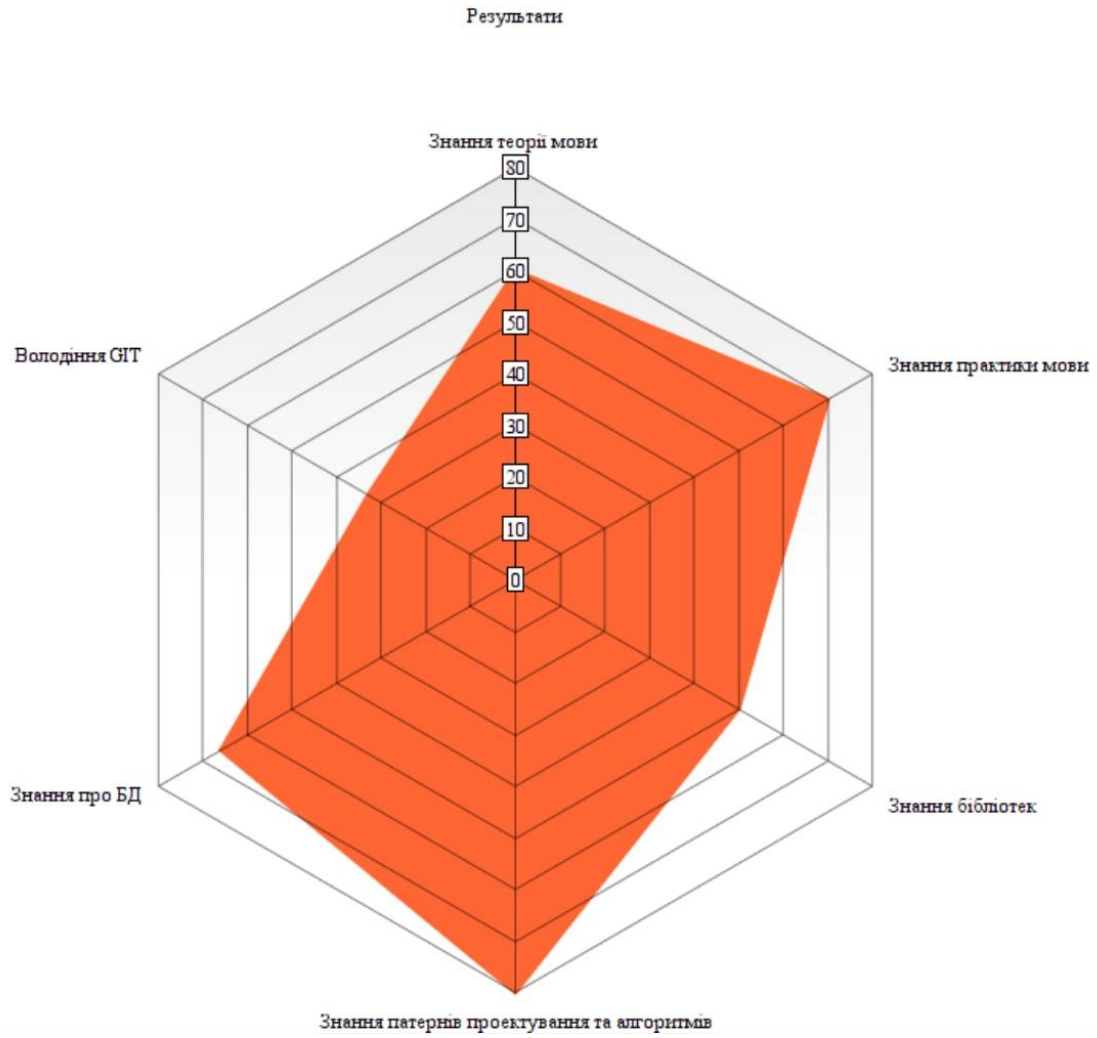


Рисунок 5.7 - Результати досвідченого у розробці студента

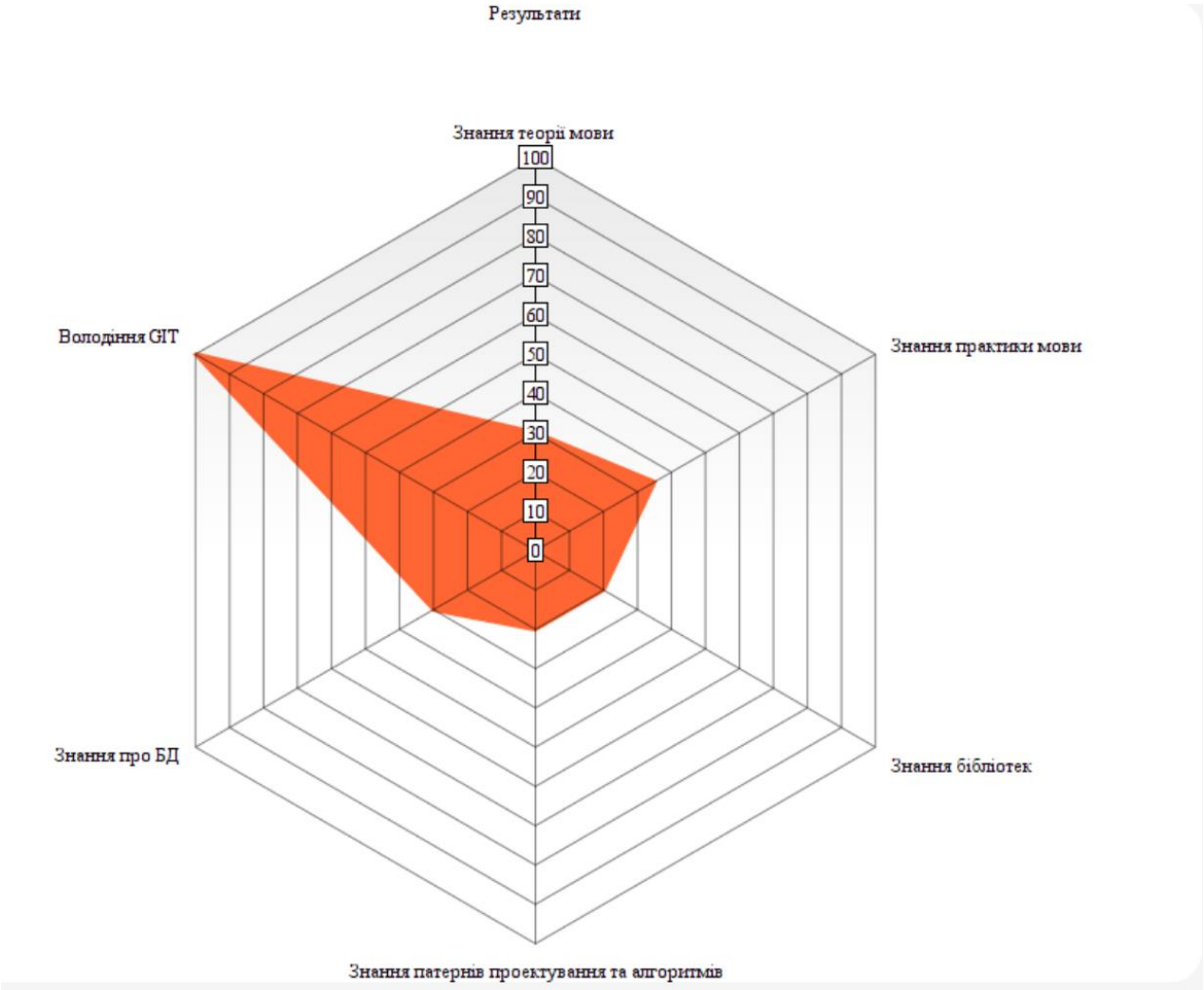


Рисунок 5.8 - Результати студента без досвіду

Рисунок 5.9 - Результати теоритично підкованого студента

Ці пункти є основними майже на будь якій співбесіді, якщо ми говоримо про технічну частину. Тому саме вони надалі будуть оцінюватись. Нажаль без перевірки усно - тестування не буде повноцінним, тому окреме оцінювання повинне бути присутнє. Далі буде створений список питань для такої співбесіди, а також приклад відповідей на них.

Продовжимо формування системи з наступним уклоном, вхідними даними все ж є не відокремлені навички, а дисципліни, що були пройдені студентами протягом навчального процесу. Тому далі візьмемо один список дисциплін з першого по четвертий курс для перегляду [2].

1. Іноземна мова - 1. Вступ до загальнотехнічної іноземної мови;
2. Аналітична геометрія та лінійна алгебра;
3. Математичний аналіз-1. Диф.числення;
4. Фізика;
5. Математичний аналіз-3.Розв'язання диф.рівнянь;
6. Іноземна мова- 2.Іноземна мова загально-технічного спрямування;
7. Інженерна графіка;
8. Чисельні методи для систем комп'терної графіки;
9. Електротехніка та електроніка;
10. WEB-орієнтована розробка програмного забезпечення;
11. Системний аналіз;
12. Системи баз даних;
13. Чисельні методи;
14. Англійська мова професійного спрямування 1. Англійська мова професійного спрямування;
15. Дослідження операцій;
16. Лінгвістичне забезпечення САПР;
17. Безпека інформаційних систем;
18. Правові основи інформаційної безпеки;
19. Комп’ютерні науки та інформаційні технології;
20. Програмування комп'ютерних мереж;
21. Підприємницьке право;
22. Системи штучного інтелекту;
23. Економіка і організація виробництва;
24. Інтелектуальний аналіз даних;
25. Геометричне моделювання та комп'ютерна графіка;
26. Моделювання систем;

27. Англійська мова професійного спрямування. Англійська мова для професійно-орієнтовного спілкування. Ділове мовлення;
28. Проектування інформаційних систем;
29. Управління ІТ-проектами;
30. Безпека життєдіяльності та цивільний захист;
31. Технології розподілених систем та паралельних обчислень;
32. Крос-платформне програмування;
33. Геометричне моделювання та комп'ютерна графіка;
34. Математичне моделювання в системах комп'ютерної графіки;
35. Теорія прийняття рішень;

Нажаль не всі дисципліни були збережені і перегляд цього списку надає нам не до кінця повну картину, але проведемо аналіз тих предметів, що були збережені.

Почнемо аналіз з поділу на групи:

1. **Дисципліни, що вивчають навички для конкретної професії**: Інженерна графіка, Геометричне моделювання та комп'ютерна графіка 1-2 **(графіка та настільні додатки)**, WEB-орієнтована розробка програмного забезпечення (**WEB**), Лінгвістичне забезпечення САПР (**створення мови розробки та компіляторів**), Програмування комп'ютерних мереж, Моделювання систем (**системний адміністратор**), Системи штучного інтелекту та Інтелектуальний аналіз даних (**розробка штучного інтелекту**).
2. **Дисципліни, що є надлишковими для професії розробника** (потрібно вточнити, що саме надлишковими і саме для розробника, адже деякі знання з цих дисциплін надалі будуть корисними, але їх відсоток занадто малий): Аналітична геометрія та лінійна алгебра, Математичний аналіз. (Диф. числення, Розв'язання диф. рівнянь), Фізика, Чисельні методи для систем комп'терної графіки та Чисельні методи (цей предмет містить корисні знання, але вони дуже відокремлені від реальних ситуацій і є гарним

прикладом надлишковості), Математичне моделювання в системах комп'ютерної графіки.

3. **Дисципліни, що є важливими і необхідними з точки зору бізнесу для розробки**: Системний аналіз, Дослідження операцій, Безпека інформаційних систем, Правові основи інформаційної безпеки, Підприємницьке право, Економіка і організація виробництва, Управління IT-проектами, Безпека життєдіяльності та цивільний захист.

4. **Дисципліни з навиками потрібними для кращого розуміння розробки як такої**: Електротехніка та електроніка, Системи баз даних, Проектування інформаційних систем, Технології розподілених систем та паралельних обчислень, Крос-платформне програмування, Теорія прийняття рішень,

5. **Англійська мова**

Перша група «**Дисципліни, що вивчають навички для конкретної професії**» це ті предмети що розвивають знання **бібліотек**, **теорії мови** та **практики мови**, вони є ключовими і відкривають основні напрями.

Друга група «**Дисципліни, що є надлишковими для професії розробника**» є корисною, якщо абстрагуватись від об’ємів часу що вона забирає. Так в деяких галузях розробки вам потрібно знати фізику чи математичний аналіз, але нажаль коли ці знання стають вам у нагоді, їх може замінити навичка пошуку інформації у інтернеті та декілька годин часу.

Третя група «**Дисципліни, що є важливими і необхідними з точки зору бізнесу для розробки**» також є ключовою, саме через неї навчання в університеті набуває такого значення, вони неймовірно підвищують розуміння процесів, та спрощують працевлаштування та подальший розвиток. Частково ці предмети впливають на **знання патернів проектування та алгоритмів.**

Четверта група «**Дисципліни з навиками потрібними для кращого розуміння розробки як такої**» корить цих дисциплін залежать від обраної галузі,

але з їх знанням вам буде простіше розвиватись як розробнику, адже ви заповните прогалини у загальному розумінні розробки та принципів роботи різних галузей. Також потрібно розуміти що та частина першої групи котра не стане для вас основною – піде до четвертої групи. Ця група ключова для **знання патернів проектування та алгоритмів, знання СКБД та SQL, володіння навичками Git.**

П’ята група виділена окремо, адже англійська мова є ключовим інструментом розробника. Всі інструменти, мови, документація, ресурси та таке інше, у переважній більшості є лише англійською, іноді є альтернативи іншими мовами, але все ж таки навіть з точки зору навчання та розробки англійська є обов’язковою.

Далі розглянемо окремо англійську мову, адже звісно, що оцінюючи компетенції студента, як розробника, обов’язковою є перевірка рівню володіння англійською.

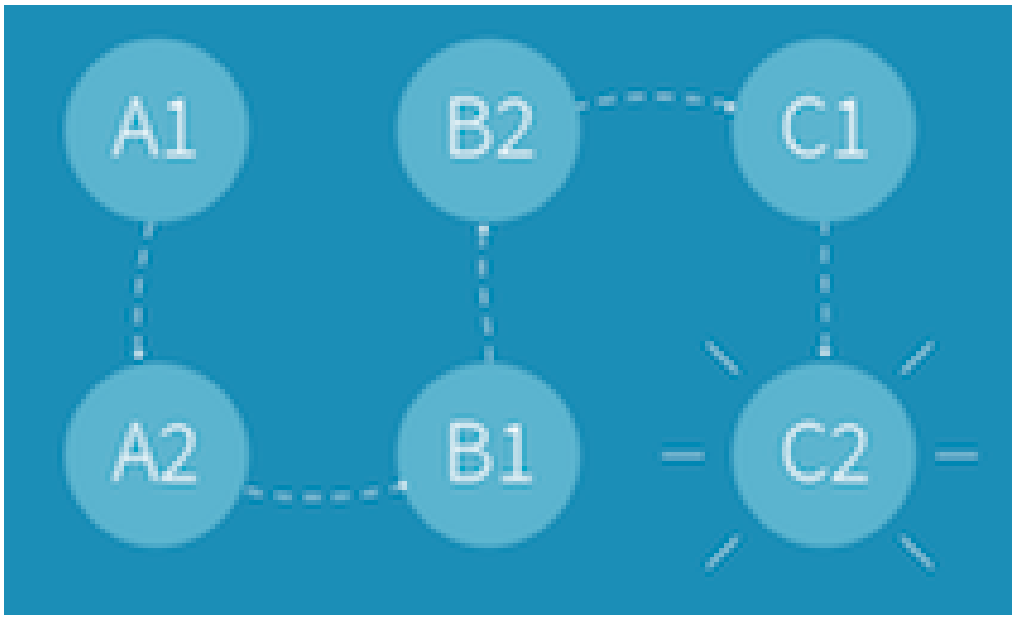


Рисунок 5.10 – Рівні англійської

1. English Basic User (A1, A2)
    1.1. A1 (Beginner)
    1.2. A2 (Elementary English)
2. English Independent User (B1, B2)

2.1. B1 (Intermediate English)

2.2. B2 (Upper-Intermediate English)

3. Proficient English User (C1, C2)

3.1. C1 (Advanced English)

3.2. C2 (Proficiency English)

A1 (Beginner) - Вміє розуміти та використовувати знайомі повсякденні вирази та найпростіші фрази, спрямовані на задоволення потреб конкретного типу. Можуть представити себе та інших, а також задавати й відповідати на запитання про особисті дані, наприклад, де хтось живе, людей, яких вони знають, і речі, які вони мають. Може взаємодіяти просто, якщо інша людина говорить повільно й чітко та готова допомогти.

A2 (Elementary English) - Може розуміти речення та часто вживані вирази, пов'язані з областями, які мають безпосереднє значення (наприклад, дуже базова особиста та сімейна інформація, покупки, місцева географія, робота). Може спілкуватися у простих і рутинних завданнях, які вимагають простого та прямого обміну інформацією щодо знайомих і рутинних справ. Може простими словами описати аспекти свого минулого, найближче оточення та справи в сферах негайної потреби.

B1 (Intermediate English) - Може розуміти основні моменти чітких стандартних вказівок щодо знайомих питань, з якими регулярно стикаються на роботі, в школі, на відпочинку тощо. Вміє розбиратися в більшості ситуацій, які можуть виникнути під час подорожі в регіоні, де розмовляють цією мовою. Може створювати простий пов'язаний текст на знайомі або особисті цікаві теми. Може описати переживання та події, мрії, надії та амбіції та коротко навести причини та пояснення думок та планів.

B2 (Upper-Intermediate) Вміє розуміти основні ідеї складного тексту як на конкретні, так і на абстрактні теми, включаючи технічні дискусії у своїй галузі спеціалізації. Може взаємодіяти з ступенем вільної та спонтанності, що робить

регулярну взаємодію з користувачами цільової мови цілком можливою, не створюючи навантаження на жодну із сторін. Може створювати чіткий, детальний текст на широкий спектр тем і пояснювати точку зору на актуальну проблему, надаючи переваги та недоліки різних варіантів.

C1 (Advanced English) - Може розуміти широкий спектр вимогливих, довших текстів і розпізнавати неявний сенс. Може висловлюватися вільно та спонтанно, не шукаючи виразів. Вміє використовувати мову гнучко й ефективно для соціальних, академічних та професійних цілей. Може створювати чіткий, добре структурований, детальний текст на складні теми, демонструючи контрольоване використання організаційних шаблонів, сполучників і зв'язних пристроїв.

C2 (Proficiency) - Може розуміти практично всі типи текстів. Уміє узагальнювати інформацію з різних усних та письмових джерел, реконструюючи аргументи та звіти у зв'язній презентації. Може висловлюватися спонтанно, дуже вільно і точно, розрізняючи тонкі відтінки сенсу навіть у складніших ситуаціях.

5.2 Створення основних модулів

Після аналізу та сортування даних можна почати створення самої системи. Як вже було зазначено потрібні:

Список знань та навичок, що очікуються від спеціаліста – за основу цієї частини будуть взяті дані з конкретної вакансії зі змінами, що були введені у ході дослідження.

Опитування на знання мови та інструментів – питання з теорії у вигляді тесту. Реалізований тест через додаток для mac os і виглядає наступним чином:

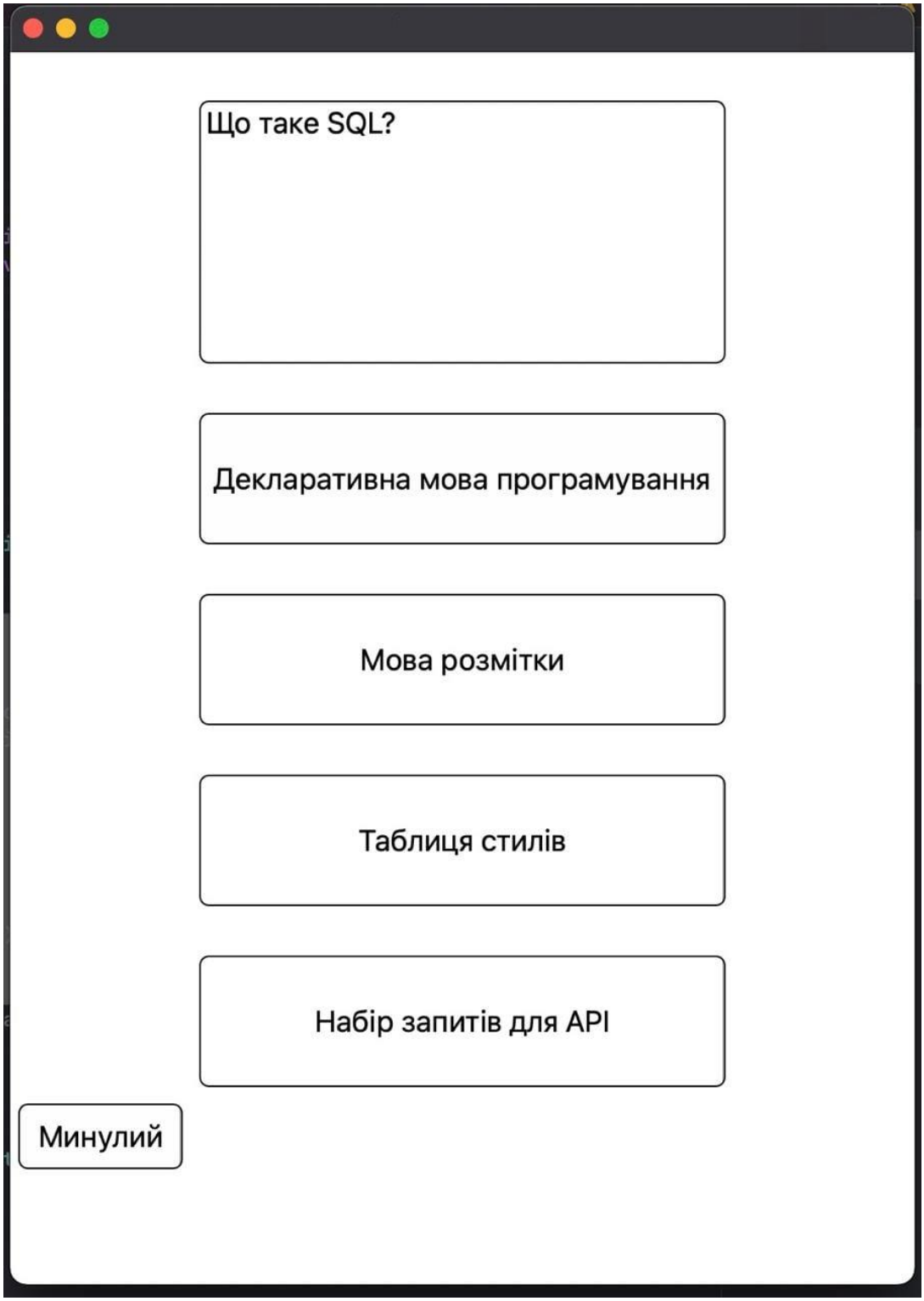


Рисунок 5.11 – Вигляд інтерфейсу тесту

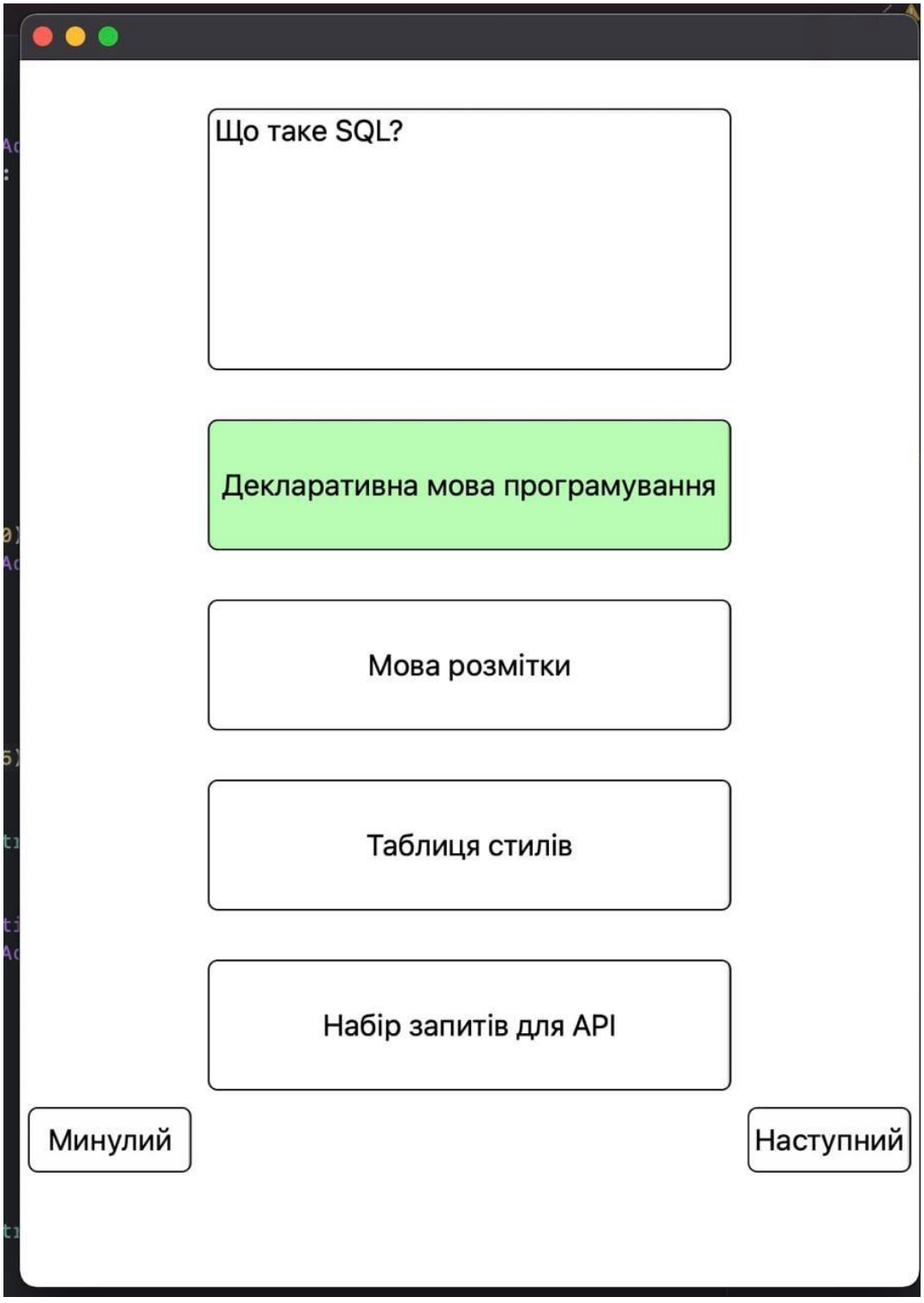


Рисунок 5.12 – Вигляд після відповіді

Перевірка навичок написання коду та вирішення логічних задач – поставлена задача для вирішення та приклад аналізу.

Код для подальшого аналізу студентом – фрагмент коду з використанням базових механізмів мови.

Список питань для опитування – питання, що до досвіду, цілей та soft skills.

Англійську можна перевірити запитавши частину питань власне англійською, але у розрахунок буде взято випадок наявності сертифіката про рівень англійської B1 і вище.

5.3 Реалізація

За приклад реалізованого тесту ми візьмемо перевірку знань на вакансію mobile IOS developer, для рівню Trainee або Junior.

Список знань та навичок, що очікуються від спеціаліста: Для проходження очікується – знання мови розробки на достатньому рівні, уміння виконувати технічні завдання, рівень англійської intermediate (досвід у комерційній розробці не є обов'язковим). Вміння працювати з базами даних, системами контролю версій, знання базових патернів. Знання бібліотек Cocoa Pods та Realm. Розуміння концепцій ООП, об'єкту, класу, багатопоточності, Auto Layout та MVVM.

Опитування на знання мови та інструментів: Для зручності питання будуть подані у вигляді таблиці і містити лише правильну відповідь.

Таблиця 5.1. Запитання до тестового завдання

Запитання	Тип питання	Відповідь
Що робить ключове слово private?	Теорія	Закриває доступ до метода або змінної класу, зовні цього класу

Яка умова використання ключового слова guard?	Теорія	Наявність return у блоці виконання else
Як можна звернутись до поля класу у ініціалізаторі цього класу, якщо ми маємо локальну зміну з такою назвою?	Теорія	Використати ключове слово self і звернутись до екземпляра
Навіщо потрібен @objc?	Теорія	Для використання коду написаному на Swift у Objective-C
Що з наведених властивостей відноситься лише до класу і не відноситься до структур?	Теорія	Успадкування
Що з наведених ключових слів є Control Transfer Statements?	Теорія	Break, Continue, Fallthrough, Throw, Return
Git – це?	Git	розподілена система керування версіями файлів та спільної роботи
Git fetch – це?	Git	Отримання даних з віддаленого репозиторія
Git pull – це?	Git	Отримання даних з віддаленого репозиторія з послідуючим злиттям
Git stash – це?	Git	Зберігає не закомічені зміни
Як називається базова гілка у системі git?	Git	master

Що не є принципом ООП?	Патерни	Композиція
Що таке делегація?	Патерни	передання завдання від зовнішнього об'єкта внутрішньому
Що таке поліморфізм?	Патерни	реалізація завдань однієї й тієї ідеї різними способами
Що таке DRY?	Патерни	1 з концепцій Clear code, не писати той самий код декілька разів
До чого має доступ ViewModel у паттерні MVVM?	Патерни	Лише до моделі
Що таке MySQL, PostgreSQL, Oracle, Microsoft SQL Server	БД	СКБД
Що таке SQL?	БД	мова структурованих запитів SQL
Які різні JOINS використовуються в SQL?	БД	INNER, LEFT, RIGHT, FULL
Первинний ключ у БД завжди	БД	Не є NULL і є унікальним
Як таблиця може посилатись на інші таблиці?	БД	Використовуючи зовнішній ключ

Цей мінімальний обсяг запитань, за для кращого результату кількість питань потрібно збільшити у 2-3 рази для кожної теми. Складність питань у такому тестуванні формує рівень, а оцінка є індикатором відповідності цьому рівню. Таким чином отримавши більше за 80% у тесті, цей етап можна вважати пройденим, у разі

результату від 60% до 80% - потрібно задати додаткові питання, у разі менших значень – тему у якій менше 50% потрібно вивчити якісніше.

Перевірка навичок написання коду та вирішення логічних задач: Мовою написання є Swift, завдання:

1. Написати функцію, що приймає у себе день тижня, та кількість днів, а повертає який буде день тижня через цю кількість днів
2. Написати функцію, що приймає у себе рядок, а повертає значення true або false, в залежності від того чи був рядок паліндромом (однаково читається в обох напрямках)

У цілому, це завдання показує рівень практичних навиків розробника, найважливішим критерієм є правильно виконане завдання, додатковими – час, структура написання, послідовність та раціональність підходу.

Таким чином, відсоток практики мови вираховується за формулою:

$$practice = (40\% * CorrectCount) + 20\% - (5\% * QualityControl) \quad (5.1)$$

Де CorrectCount – це кількість виконаних завдань, а QualityControl – це кількість зауважень

Код для подальшого аналізу студентом:. Тут буде взято класичні приклади та завдання, відповідь на них повинна бути аргументована. Допускається помилка у відповіді на будь яке з цих питань, але після другої помилки – питання вважається не зарахованим. Усі наступні рисунки є фрагментами коду, де від студенту очікується відповідь, що з цим фрагментом не так і як це виправити.

```swift
import Foundation
import Glibc

let n1: Int = 1

let n2: Float = 2.0

let n3: Double = 3.34

var result = n1 + n2 + n3
```

Рисунок 5.13 – Перетворення типів

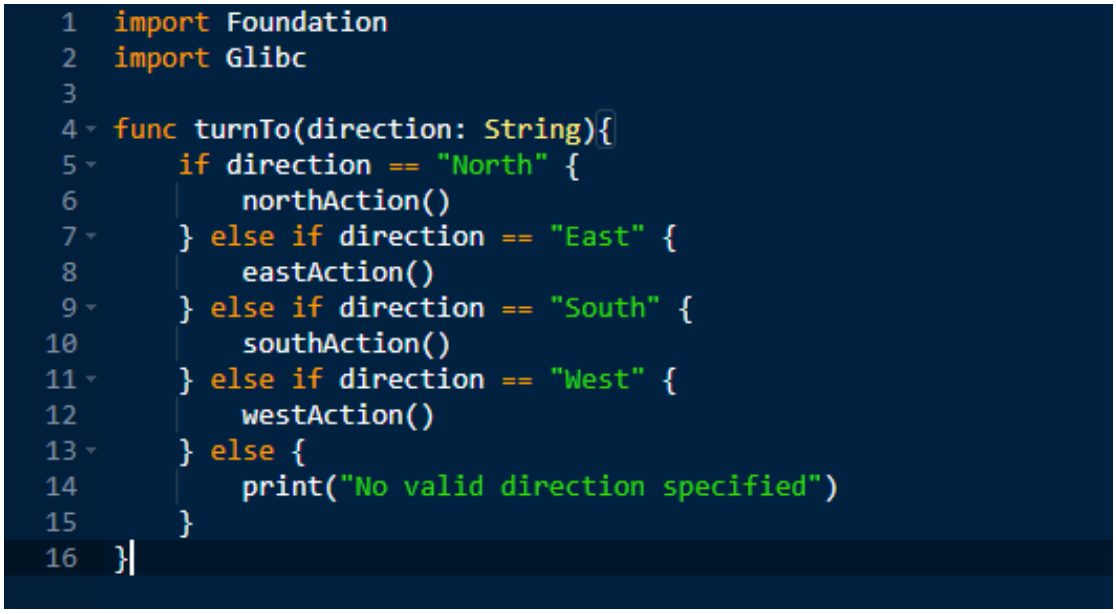
```swift
import Foundation
import Glibc

var arr1 = [1, 2, 3]
var arr2 = arr1

arr2.append(4)

var len = arr1.count
```

Рисунок 5.14 – Посилання та копіювання змінних



```
import Foundation
import Glibc

func turnTo(direction: String){
    if direction == "North" {
        northAction()
    } else if direction == "East" {
        eastAction()
    } else if direction == "South" {
        southAction()
    } else if direction == "West" {
        westAction()
    } else {
        print("No valid direction specified")
    }
}
```

Рисунок 5.15 - Використання enum

Цей етап є завершальним, у ході нього можна чіткіше зрозуміти рівень як теоретичних, так і практичних знань, тому в залежності від того, як впорався студент з цими завданнями – оцінка за практичну та теоретичну частину – може суттєво змінитись. Допустиме відхилення у 30%.

Останню частину, перевірка знань англійської, відбувається через проведення стандартних тестів на рівень англійської мови, що є у відкритому доступі.

ВИСНОВКИ

. У дипломній роботі досліджено та реалізовано систему оцінки рівня сформованості компетенцій студентів.

Було аналізовано підходи до оцінювання студентів, та знання що використовуються у розробці. Виведено класифікації, що основані на дослідженнях проведених у ході дипломної роботи, що дозволяють студенту чіткіше розуміти свій рівень знань відносно мети, та відносно критеріїв роботодавців, що допомагатиме підвищувати свій рівень знань та компетенції. Також допоможе вчасно зрозуміти, що рівень нижче за очікуваний, або достатній для мети.

— проаналізовано шляхи оцінки студентів у вищих навчальних закладах;

— виділено критерії для подальшого порівняння систем з наведенням позитивних та негативних сторін;

— створено моделі для тестування та оцінки знань з розділенням на рівні та типи знань;

— написання візуальної частини та модулів для тестування;

— створення прикладу готового тесту у системі з кінцевими результатами;

Користувачем цього додатку можуть бути студенти або інші учні під час навчання, що хочуть дізнатись свій поточний рівень компетенцій як спеціаліста.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рівні розробників у комерційній розробці. URL: https://java.lviv.ua/iyerarxiya-posad-programistiv-junior-middle-senior-lead (Дата звернення: 06.04.2022)

2. Дисципліни з 1 по 4 курс навчання на ТЕФ 122 спеціальності. URL: https://campus.kpi.ua/ (Дата звернення: 11.04.2022)

3. Інструменти та технології для веб-розробки. URL: https://www.tutch.co.uk/blog/how-to-become-a-web-developer (Дата звернення: 11.04.2022)

4. Інструменти та технології для розробки під настільні додатки. URL: https://www.indeed.com/career-advice/career-development/software-developer-skills (Дата звернення: 11.04.2022)

5. Інструменти та технології для розробки під мобільні додатки. URL: https://www.indeed.com/career-advice/resumes-cover-letters/mobile-app-developer-skills (Дата звернення: 11.04.2022)

6. Вакансії на посади розробників. URL: https://www.work.ua/jobs-kyiv-developer/ (Дата звернення: 18.04.2022)

7. Оцінка студентів в Україні. 2-6 с. URL: https://www.vspu.edu.ua/content/img/education/graph/p2.pdf (Дата звернення: 23.04.2022)

8. Оцінка студентів в Австралії. URL: https://www.education.vic.gov.au/school/teachers/teachingresources/practice/improve/Pages/eitassesscapabilities.aspx (Дата звернення: 23.04.2022)

9. Класифікація рівнів англійської мови. URL: https://tracktest.eu/english-levels-cefr/ (Дата звернення: 29.04.2022)

10. Створення наочних діаграм. URL: https://www.onlinecharts.com.ua/ (Дата звернення: 29.04.2022)

11. Бази даних. URL: https://www.w3schools.com/sql/sql_intro.asp (Дата звернення: 18.04.2022)

12. Актуальні питання з співбесід с приводу Git. URL: https://chm.org.ua/Git-interview/ (Дата звернення: 18.04.2022)

13. Популярні паттерни. URL: https://dou.ua/forums/topic/5324/ (Дата звернення: 18.04.2022)

14. Актуальні питання з співбесід с приводу мови програмування Swift. URL: https://habr.com/ru/post/659169/ (Дата звернення: 18.04.2022)

15. Що таке Git. URL: https://Git-scm.com/ (Дата звернення: 18.04.2022)

16. Наочні зображення. URL: https://www.pinterest.com/ (Дата звернення: 18.04.2022)

ДОДАТОК А

Система оцінки рівня сформованості компетенцій студентів

Текст програмного модуля

УКР.НТУУ“КПІ ім. Ігоря Сікорського”. ТР81____22Б 12-1

Аркушів 20

2022

```
    ALWAYS_EMBED_SWIFT_STANDARD_LIBRARIES = YES
    CLANG_WARN_QUOTED_INCLUDE_IN_FRAMEWORK_HEADER =
NO
    FRAMEWORK_SEARCH_PATHS = $(inherited)
"${PODS_CONFIGURATION_BUILD_DIR}/Realm"
"${PODS_CONFIGURATION_BUILD_DIR}/RealmSwift"
"${PODS_ROOT}/Realm/core"
    GCC_PREPROCESSOR_DEFINITIONS = $(inherited) COCOAPODS=1
    HEADER_SEARCH_PATHS = $(inherited)
"${PODS_CONFIGURATION_BUILD_DIR}/Realm/Realm.framework/Headers
"
"${PODS_CONFIGURATION_BUILD_DIR}/RealmSwift/RealmSwift.framewo
rk/Headers" "${PODS_XCFRAMEWORKS_BUILD_DIR}/Realm/Headers"
    LD_RUNPATH_SEARCH_PATHS = $(inherited) /usr/lib/swift
'@executable_path/../Frameworks' '@loader_path/Frameworks'
"${DT_TOOLCHAIN_DIR}/usr/lib/swift/${PLATFORM_NAME}"
    LIBRARY_SEARCH_PATHS = $(inherited)
"${DT_TOOLCHAIN_DIR}/usr/lib/swift/${PLATFORM_NAME}"
"${PODS_XCFRAMEWORKS_BUILD_DIR}/Realm" /usr/lib/swift
    OTHER_LDFLAGS = $(inherited) -l"c++" -l"z" -framework "Realm" -
framework "RealmSwift" -framework "Security"
    OTHER_SWIFT_FLAGS = $(inherited) -D COCOAPODS
    PODS_BUILD_DIR = ${BUILD_DIR}
    PODS_CONFIGURATION_BUILD_DIR =
${PODS_BUILD_DIR}/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_N
AME)
    PODS_PODFILE_DIR_PATH = ${SRCROOT}/.
    PODS_ROOT = ${SRCROOT}/Pods
```

```
    PODS_XCFRAMEWORKS_BUILD_DIR =
$(PODS_CONFIGURATION_BUILD_DIR)/XCFrameworkIntermediates
    USE_RECURSIVE_SCRIPT_INPUTS_IN_SCRIPT_PHASES = YES

    //
    //  ViewController.swift
    //  AppForMac
    //
    //  Created by Illia Volotivskyi on 19.11.2021.
    //

    import Cocoa

    class ViewController: NSViewController {
       var startButton = NSButton()

       var questionsView = NSTextField()
       var reply1 = NSButton()
       var reply2 = NSButton()
       var reply3 = NSButton()
       var reply4 = NSButton()
       let loader = NSView()


       var prev = NSButton()
       var next = NSButton()

       let bgColor: CGColor = CGColor.white
```

```
        let buttonColor: CGColor = CGColor.white
        let contentColor: CGColor = CGColor.black
        let attributes: [NSAttributedString.Key: Any] = [
            .font: NSFont.systemFont(ofSize: 18),
            .foregroundColor: NSColor(red: 255, green: 255, blue: 255, alpha: 1),
        ]

        override func viewDidLoad() {
            super.viewDidLoad()
            setupView()
            //setupStartButton()
            setupQuestion()
            setupReply()
            setupLoader()
            setupArrows()
        }

        func setupLoader() {
            view.addSubview(loader)
            loader.translatesAutoresizingMaskIntoConstraints = false
            loader.wantsLayer = true
            loader.layer?.backgroundColor = .init(red: 0, green: 0, blue: 0, alpha: 1)
            loader.topAnchor.constraint(equalTo: view.topAnchor).isActive = true
            loader.leftAnchor.constraint(equalTo: view.leftAnchor).isActive = true
            loader.rightAnchor.constraint(equalTo: view.rightAnchor).isActive =
true
            loader.bottomAnchor.constraint(equalTo: view.bottomAnchor).isActive
= true
```

```
            loader.isHidden = true
        }

        func setupStartButton() {
            self.view.addSubview(startButton)
            startButton.translatesAutoresizingMaskIntoConstraints = false
            startButton.topAnchor.constraint(equalTo: view.topAnchor, constant:
400).isActive = true
            startButton.leftAnchor.constraint(equalTo: view.leftAnchor, constant:
40).isActive = true
            startButton.heightAnchor.constraint(equalToConstant: 50).isActive =
true
            startButton.widthAnchor.constraint(equalToConstant: 320).isActive =
true

            startButton.action = #selector(onStart)
            startButton.wantsLayer = true
            startButton.layer?.backgroundColor = .black
            startButton.title = "Start test"
        }

        @objc func onStart(sender: AnyObject) {
            startButton.isHidden = true
            setupTest(with: .start)
        }

        func setupView() {
            view.translatesAutoresizingMaskIntoConstraints = false
```

```
        view.wantsLayer = true
        view.layer?.backgroundColor = bgColor
        view.heightAnchor.constraint(equalToConstant: 750).isActive = true
        view.widthAnchor.constraint(equalToConstant: 550).isActive = true
    }

    func setupTest(with section: Section) {

    }

    func setupArrows() {
        prev
        next
        view.addSubview(prev)

        prev.translatesAutoresizingMaskIntoConstraints = false
        prev.bottomAnchor.constraint(equalTo: view.bottomAnchor, constant: -
70).isActive = true
        prev.rightAnchor.constraint(equalTo: reply1.leftAnchor, constant: -
10).isActive = true
        prev.heightAnchor.constraint(equalToConstant: 40).isActive = true
        prev.widthAnchor.constraint(equalToConstant: 100).isActive = true
        prev.action = #selector(reply1Action)
        prev.wantsLayer = true
        prev.layer?.cornerRadius = 6
        prev.isBordered = true
        prev.layer?.backgroundColor = CGColor(red: 1, green: 1, blue: 1, alpha:
0.5)
```

```
        prev.layer?.borderColor = contentColor
        prev.layer?.borderWidth = 1
        prev.attributedTitle = NSAttributedString(string: "Минулий", attributes:
attributes)



        view.addSubview(next)


        next.translatesAutoresizingMaskIntoConstraints = false
        next.bottomAnchor.constraint(equalTo: view.bottomAnchor, constant: -
70).isActive = true
        next.leftAnchor.constraint(equalTo: reply1.rightAnchor, constant:
10).isActive = true
        next.heightAnchor.constraint(equalToConstant: 40).isActive = true
        next.widthAnchor.constraint(equalToCo nstant: 100).isActive = true
        next.action = #selector(reply1Action)
        next.wantsLayer = true
        next.layer?.cornerRadius = 6
        next.isBordered = true
        next.layer?.backgroundColor = CGColor(red: 1, green: 1, blue: 1, alpha:
0.5)
        next.layer?.borderColor = contentColor
        next.layer?.borderWidth = 1
        next.attributedTitle = NSAttributedString(string: "Наступний",
attributes: attributes)

    }
```

```
        func fillAll(question: Question) {
            questionsView.attributedStringValue = NSAttributedString(string:
question.text, attributes: attributes)
            reply1.attributedTitle = NSAttributedString(string:
question.replies[0].text, attributes: attributes)
            reply2.attributedTitle = NSAttributedString(string:
question.replies[1].text, attributes: attributes)
            reply3.attributedTitle = NSAttributedString(string:
question.replies[2].text, attributes: attributes)
            reply4.attributedTitle = NSAttributedString(string:
question.replies[3].text, attributes: attributes)
        }

        func setupQuestion() {
            view.addSubview(questionsView)

            questionsView.translatesAutoresizingMaskIntoConstraints = false

            questionsView.stringValue = "question"
            questionsView.wantsLayer = true
            questionsView.layer?.backgroundColor = .white
            questionsView.textColor = .black
            questionsView.isEditable = false
            questionsView.isSelectable = false

            questionsView.layer?.cornerRadius = 6
            questionsView.layer?.borderColor = contentColor
            questionsView.layer?.borderWidth = 1
```

```
        questionsView.topAnchor.constraint(equalTo: view.topAnchor,
constant: 30).isActive = true
        questionsView.centerXAnchor.constraint(equalTo: view.centerXAnchor,
constant: 0).isActive = true
        questionsView.heightAnchor.constraint(equalToConstant: 160).isActive
= true
        questionsView.widthAnchor.constraint(equalToConstant: 320).isActive
= true
    }

    func setupReply() {
        view.addSubview(reply1)
        view.addSubview(reply2)
        view.addSubview(reply3)
        view.addSubview(reply4)



        reply1.translatesAutoresizingMaskIntoConstraints = false
        reply1.topAnchor.constraint(equalTo: questionsView.bottomAnchor,
constant: 30).isActive = true
        reply1.centerXAnchor.constraint(equalTo: view.centerXAnchor,
constant: 0).isActive = true
        reply1.heightAnchor.constraint(equalToConstant: 80).isActive = true
        reply1.widthAnchor.constraint(equalToConstant: 320).isActive = true
        reply1.action = #selector(reply1Action)
        reply1.wantsLayer = true
        reply1.layer?.cornerRadius = 6
```

```
        reply1.isBordered = true
        reply1.layer?.backgroundColor = CGColor(red: 0, green: 1, blue: 0,
alpha: 0.5)
        reply1.layer?.borderColor = contentColor
        reply1.layer?.borderWidth = 1
        reply1.attributedTitle = NSAttributedString(string: "reply1", attributes:
attributes)

        reply2.translatesAutoresizingMaskIntoConstraints = false
        reply2.topAnchor.constraint(equalTo: reply1.bottomAnchor, constant:
30).isActive = true
        reply2.centerXAnchor.constraint(equalTo: view.centerXAnchor,
constant: 0).isActive = true
        reply2.heightAnchor.constraint(equalToConstant: 80).isActive = true
        reply2.widthAnchor.constraint(equalToConstant: 320).isActive = true
        reply2.action = #selector(reply2Action)
        reply2.wantsLayer = true
        reply2.layer?.cornerRadius = 6
        reply2.isBordered = true
        reply2.layer?.backgroundColor = buttonColor
        reply2.layer?.borderColor = contentColor
        reply2.layer?.borderWidth = 1
        reply2.attributedTitle = NSAttributedString(string: "reply2", attributes:
attributes)

        reply3.translatesAutoresizingMaskIntoConstraints = false
        reply3.topAnchor.constraint(equalTo: reply2.bottomAnchor, constant:
30).isActive = true
```

reply3.centerXAnchor.constraint(equalTo: view.centerXAnchor, constant: 0).isActive = true reply3.heightAnchor.constraint(equalToConstant: 80).isActive = true

reply3.widthAnchor.constraint(equalToConstant: 320).isActive = true

reply3.action = #selector(reply3Action)

reply3.wantsLayer = true

reply3.layer?.cornerRadius = 6

reply3.isBordered = true

reply3.layer?.backgroundColor = buttonColor

reply3.layer?.borderColor = contentColor

reply3.layer?.borderWidth = 1

reply3.attributedTitle = NSAttributedString(string: "reply3", attributes: attributes)

reply4.translatesAutoresizingMaskIntoConstraints = false

reply4.topAnchor.constraint(equalTo: reply3.bottomAnchor, constant: 30).isActive = true

reply4.centerXAnchor.constraint(equalTo: view.centerXAnchor, constant: 0).isActive = true

reply4.heightAnchor.constraint(equalToConstant: 80).isActive = true

reply4.widthAnchor.constraint(equalToConstant: 320).isActive = true

reply4.action = #selector(reply4Action)

reply4.wantsLayer = true

reply4.layer?.cornerRadius = 6

reply4.isBordered = true

reply4.layer?.backgroundColor = buttonColor

reply4.layer?.borderColor = contentColor

reply4.layer?.borderWidth = 1

```
        reply4.attributedTitle = NSAttributedString(string: "reply4", attributes:
attributes)

    }

    @objc func reply1Action(sender: AnyObject) {
        loader.isHidden = false
        fillAll(question: QuestionManager.shared.getQuestions(with:
.start).first!)
        DispatchQueue.main.asyncAfter(deadline: .now() + 1) {
            self.loader.isHidden = true
        }
    }
    @objc func reply2Action(sender: AnyObject) {

    }
    @objc func reply3Action(sender: AnyObject) {

    }
    @objc func reply4Action(sender: AnyObject) {

    }


    override var representedObject: Any? {
        didSet {
        // Update the view, if already loaded.
        }
```

```
    }
}
//
//  StartScreen.swift
//  AppForMac
//
//  Created by Illia Volotivskyi on 28.05.2022.
//

import Foundation
import AppKit

class StartScreen: NSView {
    var button = NSButton()
    var onStart: () -> Void

    init(onStart: @escaping () -> Void) {
        self.onStart = onStart
        super.init(frame: .zero)
        setupStartButton()
    }

    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    func setupStartButton() {
        self.addSubview(button)
```

```
            button.translatesAutoresizingMaskIntoConstraints = false
            button.topAnchor.constraint(equalTo: topAnchor, constant:
400).isActive = true
            button.leftAnchor.constraint(equalTo: leftAnchor, constant: 40).isActive
= true
            button.heightAnchor.constraint(equalToConstant: 50).isActive = true
            button.widthAnchor.constraint(equalToConstant: 320).isActive = true

            button.action = #selector(buttonAction)
            button.wantsLayer = true
            button.layer?.backgroundColor = .black
            button.title = "Start test"
        }

        @objc func buttonAction(sender: AnyObject) {
            print("action")
            onStart()
        }
    }
    //
    //  TestView.swift
    //  AppForMac
    //
    //  Created by Illia Volotivskyi on 28.05.2022.
    //

    import Foundation
    import AppKit
```

```
class TestView: NSView {
    var questionView = NSTextField()
    var repliesButtons = NSStackView()

    var question = Question(with: "Question", section: .start, replies:
[Reply(with: "Wrong"), Reply(with: "Wrong"), Reply(with: "Correct", isCorrect:
true)])

    init() {
        super.init(frame: .zero)

        translatesAutoresizingMaskIntoConstraints = false
        heightAnchor.constraint(equalToConstant: 700 * 1).isActive = true
        widthAnchor.constraint(equalToConstant: 400 * 1).isActive = true

        setupButtons()
    }

    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    func setupButtons() {
        addSubview(repliesButtons)

        repliesButtons.translatesAutoresizingMaskIntoConstraints = false
```

```
            repliesButtons.orientation = .vertical
            repliesButtons.centerXAnchor.constraint(equalTo:
centerXAnchor).isActive = true
            repliesButtons.topAnchor.constraint(equalTo: topAnchor, constant:
50).isActive = true


            for reply in question.replies {
                let button = NSButton()
                button.translatesAutoresizingMaskIntoConstraints = false
                button.heightAnchor.constraint(equalToConstant: 50).isActive = true
                button.widthAnchor.constraint(equalToConstant: 320).isActive = true
                button.stringValue = reply.text
                button.action = #selector(buttonAction)
                //reply.wantsLayer = true
                button.layer?.backgroundColor = .black
                button.title = reply.text
                repliesButtons.addArrangedSubview(button)

            }
        }


        @objc func buttonAction() {

        }
    }
    //
```

```
//  RealmManager.swift
//  AppForMac
//
//  Created by Illia Volotivskyi on 26.05.2022.
//

import Foundation
import RealmSwift

class realmReply: Object {
    @objc dynamic var qID: Int = 0
    @objc dynamic var text: String = ""
    @objc dynamic var isCorrect: Bool = false
    @objc dynamic var isSelected: Bool = false
}
class realmQuestion: Object {
    @objc dynamic var text = ""
    @objc dynamic var Replys: Data? = nil // optionals supported
}

class RealmManager {
    static let shared = RealmManager()
    func addNewReply() {
        // Use them like regular Swift objects
        let reply = realmReply()

        reply.qID = 0
        reply.text = "Reply"
```

```
reply.isCorrect = false
reply.isSelected = false

// Get the default Realm
let realm = try! Realm()

// Query Realm for all dogs less than 2 years old
let reply = realm.objects(Reply.self).filter("true")
puppies.count // => 0 because no dogs have been added to the Realm yet

// Persist your data easily
try! realm.write {
    realm.add(reply)
}

 //Queries are updated in realtime
reply.count // => 1

 //Query and update from any thread
DispatchQueue(label: "background").async {
    autoreleasepool {
        let realm = try! Realm()
        let theReply = realm.objects(Reply.self).filter("true").first
        try! realm.write {
            reply!
        }
    }
}
```

```
    }
}
//
//  Question.swift
//  AppForMac
//
//  Created by Illia Volotivskyi on 26.05.2022.
//

import Foundation

enum Section {
    case start
    case web
}

class Reply {
    //var qID: Int
    var text: String
    var isCorrect: Bool
    var isSelected: Bool = false
    init(with text: String, isCorrect: Bool = false) {
        self.text = text
        self.isCorrect = isCorrect
    }

    func choose() {
        isSelected = true
```

```
    }
}

class Question {
    //var qID: Int
    var section: Section
    var text: String
    var replies: [Reply]

    init(with text: String, section: Section, replies: [Reply]) {
        self.text = text
        self.section = section
        self.replies = replies
    }
}
//
//  AppDelegate.swift
//  AppForMac
//
//  Created by Illia Volotivskyi on 19.07.2021.
//

import Cocoa

@main
class AppDelegate: NSObject, NSApplicationDelegate {
```

```
    func applicationDidFinishLaunching(_ aNotification: Notification) {
        // Insert code here to initialize your application
    }

    func applicationWillTerminate(_ aNotification: Notification) {
        // Insert code here to tear down your application
    }


}


    ALWAYS_EMBED_SWIFT_STANDARD_LIBRARIES = YES
    CLANG_WARN_QUOTED_INCLUDE_IN_FRAMEWORK_HEADER =
NO
    FRAMEWORK_SEARCH_PATHS = $(inherited)
"${PODS_CONFIGURATION_BUILD_DIR}/Realm"
"${PODS_CONFIGURATION_BUILD_DIR}/RealmSwift"
"${PODS_ROOT}/Realm/core"
    GCC_PREPROCESSOR_DEFINITIONS = $(inherited) COCOAPODS=1
    HEADER_SEARCH_PATHS = $(inherited)
"${PODS_CONFIGURATION_BUILD_DIR}/Realm/Realm.framework/Headers
"
"${PODS_CONFIGURATION_BUILD_DIR}/RealmSwift/RealmSwift.framewo
rk/Headers" "${PODS_XCFRAMEWORKS_BUILD_DIR}/Realm/Headers"
```

LD_RUNPATH_SEARCH_PATHS = $(inherited) /usr/lib/swift '@executable_path/../Frameworks' '@loader_path/Frameworks' "${DT_TOOLCHAIN_DIR}/usr/lib/swift/${PLATFORM_NAME}"
LIBRARY_SEARCH_PATHS = $(inherited) "${DT_TOOLCHAIN_DIR}/usr/lib/swift/${PLATFORM_NAME}" "${PODS_XCFRAMEWORKS_BUILD_DIR}/Realm" /usr/lib/swift
OTHER_LDFLAGS = $(inherited) -l"c++" -l"z" -framework "Realm" -framework "RealmSwift" -framework "Security"
OTHER_SWIFT_FLAGS = $(inherited) -D COCOAPODS
PODS_BUILD_DIR = ${BUILD_DIR}
PODS_CONFIGURATION_BUILD_DIR = ${PODS_BUILD_DIR}/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)
PODS_PODFILE_DIR_PATH = ${SRCROOT}/.
PODS_ROOT = ${SRCROOT}/Pods
PODS_XCFRAMEWORKS_BUILD_DIR = $(PODS_CONFIGURATION_BUILD_DIR)/XCFrameworkIntermediates
USE_RECURSIVE_SCRIPT_INPUTS_IN_SCRIPT_PHASES = YES
"${PODS_XCFRAMEWORKS_BUILD_DIR}/Realm" /usr/lib/swift
OTHER_LDFLAGS = $(inherited) -l"c++" -l"z" -framework "Realm" -framework "RealmSwift" -framework "Security"
OTHER_SWIFT_FLAGS = $(inherited) -D COCOAPODS
PODS_BUILD_DIR = ${BUILD_DIR}
PODS_CONFIGURATION_BUILD_DIR = ${PODS_BUILD_DIR}/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)
PODS_PODFILE_DIR_PATH = ${SRCROOT}/.
PODS_ROOT = ${SRCROOT}/Pods

PODS_XCFRAMEWORKS_BUILD_DIR = $(PODS_CONFIGURATION_BUILD_DIR)/XCFrameworkIntermediates
USE_RECURSIVE_SCRIPT_INPUTS_IN_SCRIPT_PHASES = YES