

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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТИРЕНКО

«___» _____ 2021 р.

Дипломний проект

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»**

на тему: Багатокористувацька система редагування програмного коду

Виконала: студентка 4 курсу, групи ІВ-72

Парубець Любов Миколаївна

Керівник асистент Іваніщев Б.В.

Консультант (нормоконтроль)

Рецензент

Засвідчую, що у цьому дипломному
проекті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

«Комп'ютерні системи та мережі»

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТИПЕНКО
«__» _____ 2021 р.

ЗАВДАННЯ

на бакалаврський дипломний проект студента

Парубець Любові Миколаївни

1. Тема проекту Багатокористувацька система редагування програмного коду
керівник проекту асистент Іваніщев Б. В.

Затверджені наказом по університету від «11» травня 2021 року № 1139-с

2. Термін здачі студентом закінченого проекту (роботи) 1 червня 2021 року

3. Вихідні дані до проекту технічна документація, теоретичні та статистичні дані;

4. Зміст розрахунково-пояснювальної записки Опис предметної області, обґрунтування оптимального варіанту реалізації мети цієї роботи, розробка додатку: вибір технологій та їх обґрунтування, основні рішення з реалізації додатку. Висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): принципова схема, функціональна схема, структурна схема

6. Консультанти проекту, з вказівкою розділів проекту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25 вересня 2020 року

Календарний план

№ п/п	Найменування етапів дипломного проекту	Терміни виконання етапів проекту	Примітки
1.	<i>Затвердження теми проекту</i>	<i>10.12.2021-15.12.2021</i>	
2.	<i>Вивчення та аналіз завдання</i>	<i>15.12.2021-15.03.2021</i>	
3.	<i>Розробка архітектури та загальної структури системи</i>	<i>15.03.2021-25.03.2021</i>	
4.	<i>Розробка структур окремих підсистем</i>	<i>25.03.2021-5.04.2021</i>	
5.	<i>Програмна реалізація системи</i>	<i>5.04.2021-15.04.2021</i>	
6.	<i>Оформлення пояснювальної записки</i>	<i>15.04.2021-20.05.2021</i>	
7.	<i>Захист програмного продукту</i>	<i>25.04.2021</i>	
8.	<i>Передзахист</i>	<i>23.05.2021</i>	
9.	<i>Захист</i>	<i>17.06.2021</i>	

Студент

Парубець Л. М.

Керівник проекту

Іваніщев Б. В.

АНОТАЦІЯ

У даній бакалаврській дипломній роботі було реалізовано багатокористувацьку систему редагування програмного коду та досліджено її функціональні можливості у порівнянні з системами-аналогами.

Система дозволяє користувачу зареєструватися, авторизуватися та писати програмний код.

Програма являє собою веб-додаток, розроблений з використанням таких технологій: JavaScript бібліотеки React та CodeMirror, а також платформи розробки додатків Firebase.

ABSTRACT

In this bachelor's thesis, a multi-user system for editing program code was implemented and its functionality was investigated in comparison with analog systems.

The system allows the user to register, log in and write program code.

The program is a web application developed using technologies such as React and CodeMirror JavaScript-libraries, as well as the Firebase application development platform.

Опис альбому

до дипломного проекту

на тему: «Багатокористувацька система редагування програмного коду»

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№ екземпляру	Примітки	
	A4		Завдання на дипломний проєкт	2			
	A4	ІАЛЦ.467200.001 ОА	Багатокористувацька система редагування програмного коду Опис альбому	1			
	A4	ІАЛЦ.467200.002 ТЗ	Багатокористувацька система редагування програмного коду Технічне завдання	3			
	A4	ІАЛЦ.467200.00 ПЗ	Багатокористувацька система редагування програмного коду Пояснювальна записка	57			
	A4	ІАЛЦ.467200.004 Д1	Додаток 1 Схема структурна	1			
	A4	ІАЛЦ.467200.005 Д2	Додаток 2 Схема функціональна	1			
	A4	ІАЛЦ.467200.006 Д3	Додаток 3 Схема принципова	1			
	A4	ІАЛЦ.467200.006 Д4	Додаток 4 Лістинг програми	19			
			ІАЛЦ.467200.001 ОА				
Зм	Арк.	№ докум	Підпис	Дата			
Розроб.	Парубець Л.М.				Багатокористувацька система редагування програмного коду Опис альбому		
Первір.	Іваніщев Б. В.						
Н.Контр.	Сімоненко В. П.						
Затверд.							
					Літ.	Арк.	Аркуші в
					Т	1	1
					КПІ ім. Ігоря Сікорського ФІОТ, ІВ-72		

Технічне завдання

до дипломного проекту

на тему: «Багатокористувацька система редагування програмного коду»

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ.....	2
2. ПРИЧИНИ ДЛЯ РОЗРОБКИ.....	2
3. ЦІЛЬ ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
4. ДЖЕРЕЛА РОЗРОБКИ.....	2
5. ТЕХНІЧНІ ВИМОГИ.....	3
5.1. ВИМОГИ ДО ПРОДУКТУ, ЩО РОЗРОБЛЯЄТЬСЯ	3
5.2. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	3
5.3. ВИМОГИ ДО АПАРАТНОЇ ЧАСТИНИ	3
6. ЕТАПИ РОЗРОБКИ	4

					ІАЛЦ.467200.002 ТЗ						
Зм.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Парубець Л.М.			Багатокористувацька система редагування програмного коду Технічне завдання			Лит.	Арк.	Аркушів	
Перевір.		Іваніщев Б.В.								1	3
								КПІ ім. Ігоря Сікорського ФІОТ, ІВ-72			
Н. Контр.		Сімоненко В.П.									
Затверд.											

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: Багатокористувацька система редагування програмного коду.

Область застосування: сфера інформаційних технологій.

2. ПІДСТАВИ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка багатокористувацької системи редагування програмного коду, що буде доступна і зрозуміла для всіх користувачів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в мережі Інтернет з даних питань, довідники з програмування.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до розробленого продукту

- Простий і зрозумілий інтерфейс веб-сервісу.
- Актуальна інформація.

5.2 Вимоги до програмного забезпечення

- Windows/Linux/macOS
- Будь-який Браузер

5.3 Вимоги до апаратної частини

- Комп'ютер на базі процесора Intel Pentium 166 та вище

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

- Оперативної пам'яті не менше 64 Мбайт
- Підключення до Інтернету

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	12.12.2020
Складання і узгодження технічного завдання	09.01.2021
Аналіз структури програмного забезпечення	11.02.2021
Створення модулів розроблюваної системи	16.03.2021
Тестування окремих модулів системи	17.04.2021
Доопрацювання, налагодження і виправлення помилок	05.05.2021
Оформлення документації дипломної роботи	12.05.2021

Пояснювальна записка

до дипломного проекту

на тему: «Багатокористувацька система редагування програмного коду.»

Київ – 2021

ЗМІСТ

ВСТУП.....	3
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	4
РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ СИСТЕМ РЕДАГУВАННЯ ПРОГРАМНОГО КОДУ	5
1.1 Опис предметного середовища	5
1.2 Теоретичні відомості	7
1.3 Огляд існуючих аналогів.....	13
ВИСНОВКИ ДО РОЗДІЛУ 1.....	21
РОЗДІЛ 2. ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ БАГАТОКОРИСТУВАЦЬКОЇ СИСТЕМИ РЕДАГУВАННЯ ПРОГРАМНОГО КОДУ	23
2.1 Огляд призначення системи, що буде розроблюватись.....	23
2.2 Огляд інструментів для реалізації багатокористувацької системи редагування програмного коду	24
2.2.1 Visual Studio Code	24
2.2.2 JavaScript та React.....	25
2.2.3 CodeMirror.....	28
2.2.4 Firebase	29
ВИСНОВКИ ДО РОЗДІЛУ 2.....	31
РОЗДІЛ 3. ОПИС РОЗРОБКИ БАГАТОКОРИСТУВАЦЬКОЇ СИСТЕМИ РЕДАГУВАННЯ ПРОГРАМНОГО КОДУ	32
3.1 Ініціалізація проекту	32
3.2 Реалізація авторизації та автентифікації.....	33
3.3 React components та React router.....	34
3.4 Збереження даних користувача	39
ВИСНОВКИ ДО РОЗДІЛУ 3.....	42

					ІАЛЦ.467200.003 ПЗ					
Зм.	Арк.	№ докум.	Підпис	Дата	<i>Багатокористувацька система редагування програмного коду Пояснювальна записка</i>			Лит.	Арк.	Аркушів
<i>Розроб.</i>		<i>Парубець Л.М.</i>								
<i>Перевір.</i>		<i>Іваніщев Б.В.</i>							<i>1</i>	<i>57</i>
								<i>КПІ ім. Ігоря Сікорського ФІОТ, ІВ-72</i>		
<i>Н. Контр.</i>		<i>Сімоненко В.П.</i>								
<i>Затверд.</i>										

РОЗДІЛ 4. ВИКОРИСТАННЯ СИСТЕМИ.....	43
4.1 Реєстрація в системі	43
4.2 Вхід до системи.....	44
4.3 Вікно зміни паролю.....	46
4.4 Сторінка профілю користувача	48
4.5 Вікно оновлення профілю	48
4.6 Вікно редактора програмного коду	49
ВИСНОВКИ ДО РОЗДІЛУ 4.....	54
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	57

ВСТУП

На сьогоднішній день ми маємо можливість спостерігати стрімкий розвиток інформаційних технологій, що в свою чергу породжує великий попит на кваліфікованих спеціалістів. За результатами прогнозів ринку праці найближчим часом цей попит буде тільки зростати. Таким чином з'являється потреба у програмному забезпеченні, яке б дозволяло працювати та навчатися в ІТ-сфері максимально продуктивно.

Останні декілька років серед розробників програмного забезпечення все більшу популярність набирають системи редагування програмного коду. Особливо зручно використовувати онлайн-редактори коду, адже користувач не прив'язаний до конкретного девайсу чи локальної машини. Отже, дана система є інструментом, що знаходиться на віддаленому сервері, доступ до якого можна отримати за допомогою браузера.

Сьогодні на просторах Інтернету таких редакторів можна знайти безліч, але вони відрізняються за своїми функціональними можливостями, прив'язкою до тих чи інших мов програмування, зручністю та простотою використання. Деякі онлайн-редактори програмного коду мають функціонал, схожий на функціонал звичайного текстового редактора, інші ж нагадують повноцінні IDE. Тому кожне рішення певною мірою користується попитом.

Отже, зважаючи на вище зазначені фактори, у даній дипломній роботі була поставлена наступна задача: розробити програмне забезпечення – доступну систему редагування програмного коду, яка у свою чергу надасть можливість одночасно взаємодіяти безлічі користувачів і у той же час не потребуватиме додаткових маніпуляцій та плагінів для комфортної роботи з програмним кодом, а саме написання, редагування та збереження.

					ІАЛЦ.467200.003 ПЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IT	Інформаційні технології
IDE	Integrated Development Environment
ПЗ	Програмне забезпечення
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
VS Code	Visual Studio Code
UI	User Interface
MVC	Model-View-Controller
API	Application programming interface
BaaS	Backend as a Service

					ІАЛЦ.467200.003 ПЗ	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1

ОГЛЯД ІСНУЮЧИХ СИСТЕМ РЕДАГУВАННЯ ПРОГРАМНОГО КОДУ

1.1 Опис предметного середовища

Веб-застосунок (або веб-додаток) – це будь-яка комп'ютерна програма, що виконує певні функції, використовуючи при цьому веб-браузер у якості клієнта. Dodatok може бути таким же простим, як контактна форма на веб-сайті чи дошка оголошень, а може бути складним, як наприклад, текстовий процесор чи мобільний ігровий додаток для декількох гравців, який користувач завантажує на свій мобільний телефон.

«Клієнт» використовується у середовищі клієнт-сервер для позначення програми, котру людина використовує для запуску додатку. Клієнт-серверне середовище – це середовище, в якому декілька комп'ютерів обмінюються інформацією, наприклад, вводять інформацію в базу даних. «Клієнт» – це додаток, який використовується для вводу інформації, а «сервер» – це додаток, який використовується для збереження інформації. [\[1\]](#)

Для доступу до веб-додатку потрібне лише підключення до Інтернету. Користувач використовує веб-браузер, такий як Safari, Mozilla Firefox або Google Chrome, щоб підключитися до своєї програми. Існує три елементи, які веб-додаток вимагає для роботи: веб-сервер для обробки запитів від клієнта, сервер додатків для виконання запитуваних завдань та база даних для зберігання інформації.

Розробники кодують веб-застосунки двома типами мов програмування. Як правило, веб-додаток використовує комбінацію сценарію на стороні сервера та сценарію на стороні клієнта. Сценарій на стороні сервера займається зберіганням та отриманням інформації і потребує таких мов, як

					ІАЛЦ.467200.003 ПЗ	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

PHP, ASP, Python або Java. Розробники програмують серверну частину для створення сценаріїв, які використовуватиме веб-програма. Для клієнтського сценарію потрібні такі мови, як JavaScript, CSS та HTML. Ці мови підтримуються браузерами і потрібні для запуску програми в браузері. Сценарій на стороні клієнта займається поданням інформації користувачеві.

Більшість веб-програм мають короткий цикл розробки та можуть створюватися невеликими командами. Деякі веб-застосунки не потребують обробки на стороні сервера і є статичними. Інші ж вимагають обробки на стороні сервера. Їх називають «динамічними».

Веб-додаток працює таким чином:

1. Користувач створює запит до веб-сервера через Інтернет через користувацький інтерфейс програми.
2. Веб-сервер надсилає цей запит серверу веб-додатку.
3. Сервер веб-додатку виконує запитане завдання, а потім генерує результати необхідних даних.
4. Сервер веб-додатку надсилає ці результати назад веб-серверу (запитувана інформація або оброблені дані).
5. Веб-сервер передає запитувану інформацію клієнту (планшет, мобільний телефон або десктоп).
6. Запитана інформація відображається на дисплеї користувача.

Актуальність веб-додатків росте з кожним днем, вони активно витісняють десктопні програми. Така тенденція спричинена рядом переваг веб-додатку:

- користувачу не потрібно встановлювати його на жорсткий диск, тому це не спричиняє обмеження простору на диску;
- веб-додаток вимагає меншої підтримки та обслуговування з боку бізнесу та технічних вимог до комп'ютера користувача;

					ІАЛЦ.467200.003 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

- веб-додаток зменшує витрати як для кінцевого користувача, так і для бізнесу;
- веб-додатки завжди оновлені, оскільки оновлення застосовуються централізовано;
- усі користувачі можуть отримати доступ до однієї версії, що усуває будь-які проблеми сумісності;
- ви можете отримати доступ до веб-додатку в будь-якому місці за допомогою веб-браузера;
- поки браузер сумісний, веб-програми можуть працювати на декількох платформах, незалежно від операційної системи або пристрою;
- розробник звільняється від відповідальності за створення клієнта, сумісного з певним типом комп'ютера або певної операційної системи, тому будь-хто може користуватися додатком, маючи доступ до Інтернету;
- веб-додатки зменшують піратство програмного забезпечення у додатках на основі передплати. [\[3\]](#)

1.2 Теоретичні відомості

Система редагування програмного коду – це інструмент, що дозволяє користувачеві писати та корегувати програмний код. Часто такі системи представлені або як десктоп-програма, або у вигляді веб-додатків.

Серед десктоп-програм варто виділити IDE (інтегроване середовище розробки) – це комплекс програмних засобів, який використовується програмістами для розробки ПЗ. Тобто це не лише текстовий редактор. У той час як текстові редактори для коду, такі як Sublime або Atom, пропонують безліч зручних функцій, які користувач може налаштовувати, відштовхуючись від власних потреб, таких як підсвічування синтаксису, розташування елементів

					ІАЛЦ.467200.003 ПЗ	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

інтерфейсу та розширені засоби навігації, вони дозволяють тільки писати код. Для створення функціонуючих програм потрібні як мінімум компілятор і відладчик.

IDE включає в себе ці компоненти, як і ряд інших. Деякі з них поставляються з додатковими інструментами для автоматизації, тестування і візуалізації процесу розробки. Термін «інтегроване середовище розробки» означає, що надаються всі необхідні елементи для перетворення коду в функціонуючі програми. [\[11\]](#)

Отже, інтегроване середовище розробки зазвичай включає в себе:

- текстовий редактор;
- транслятор (компілятор і / або інтерпретатор);
- засоби автоматизації збірки;
- відладчик.

Іноді IDE містить також засоби для інтеграції з системами управління версіями і різноманітні інструменти для спрощення конструювання графічного інтерфейсу користувача. Багато сучасних середовищ розробки також включають браузер класів, інспектор об'єктів і діаграму ієрархії класів - для використання при об'єктно-орієнтованій розробці ПЗ. IDE зазвичай призначені для декількох мов програмування - такі як IntelliJ IDEA, NetBeans, Eclipse, Qt Creator, Geany, Embarcadero RAD Studio, Code :: Blocks, Xcode або Microsoft Visual Studio, але є і IDE для однієї конкретної мови програмування - як, наприклад, Visual Basic, Delphi, Dev-C ++.

Існує також окремий вид IDE - середовища візуальної розробки, які включають в себе можливість наочного редагування інтерфейсу програми.

IDE набули широкого застосування серед сучасних розробників ПЗ, адже вони відносно швидкі, їх можна завантажити на свою локальну машину та налаштувати під свої потреби елементи інтерфейсу, збірку проектів та різноманітні плагіни для полегшення використання програми і спрощення

					ІАЛЦ.467200.003 ПЗ	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

написання програмного коду. І, хоча IDE – це доволі потужний багатофункціональний інструмент, воно не є універсальним рішенням для усіх користувачів у ІТ-сфері та має свої недоліки.

У зв'язку з поширенням популярності проведення онлайн-співбесід деякі всесвітньовідомі корпорації почали шукати альтернативу встановленим локально інтегрованим середовищам розробки. Так сталося і з Microsoft та їхнім продуктом Skype. У веб-версію Skype був інтегрований вбудований редактор, що підтримує 7 популярних мов програмування: C++, C, C#, JavaScript, Python, Java і Ruby. У процесі співбесіди користувач має змогу змінювати обрану мову програмування. Також наявне підсвічування синтаксису задля запобігання синтаксичних помилок під час співбесіди.

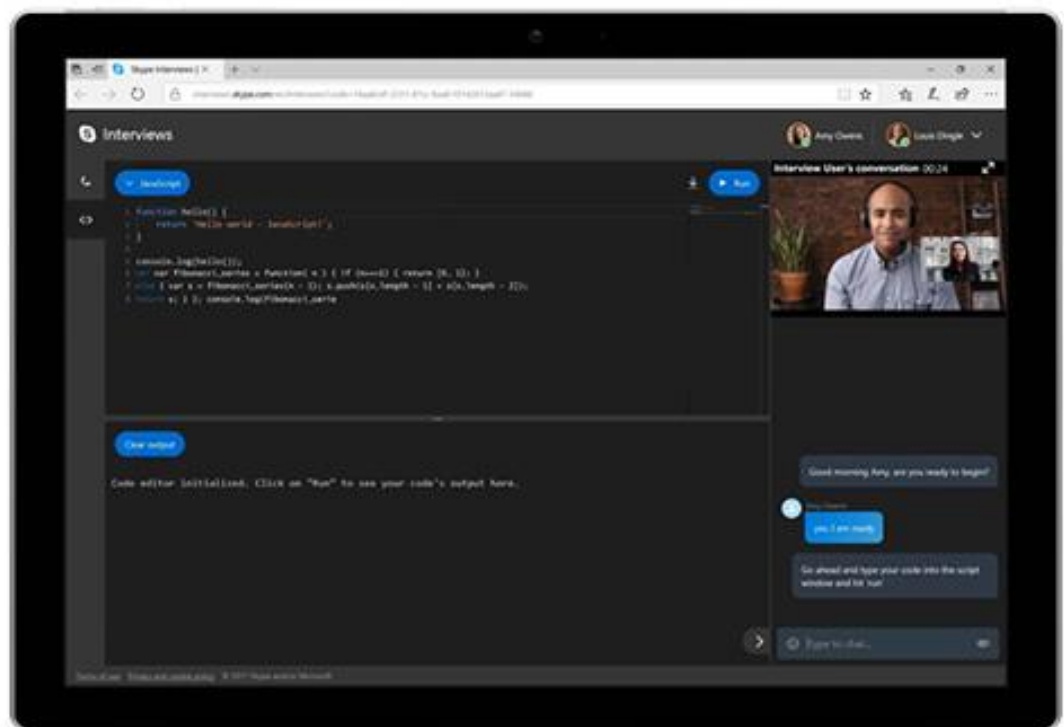


Рис. 1.1: Інтерфейс сервісу Skype Interviews.

Отже, на перший погляд, сервіс Skype Interviews виглядає досить привабливим рішенням, адже це взагалі прискорить і спростить процес співбесіди, тому що під час розмови тепер не потрібно запускати сторонній

					ІАЛЦ.467200.003 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

додатковий інструмент, щоб оцінити вміння кандидатів писати код. Але, Skype Interviews має і свої недоліки. Перш за все це те, що ця функція працює тільки у веб-версії браузерів Microsoft Edge і Google Chrome (версія 32 і вище) та лише у браузерах з англійською локалізацією. Також незважаючи на те, що даний інструмент має деякі функції IDE, і є справді зручним для написання коду без використання IDE, все ж тут відсутня можливість збереження прогресу, тобто по закінченню сесії записи будуть втрачені. Інший вагомий недолік такої системи – це те, що вона вузько направлена. Тобто її не доцільно і не раціонально використовувати для інших цілей, окрім співбесід та комунікації розробників, коли потрібно разом вирішити поставлену задачу в режимі реального часу.

Саме тому варто розглянути ще один вид систем редагування програмного коду.

Якщо раніше розробники програмного забезпечення надавали перевагу інтегрованому середовищу розробки, встановленому на власній локальній машині, то останнім часом кількість користувачів онлайн-редакторів стрімко зростає. Розробники почали розуміти, що для певних задач онлайн-системи редагування програмного коду можуть бути більш зручними та доцільними для використання. Безперечно, поки що не існує універсальних рішень, які б повністю замінили та витіснили десктопні IDE, але онлайн-редактори надають користувачеві більше можливостей з точки зору мобільності, адже доступ до них можна отримати відносно швидко за допомогою браузера. І оскільки такі редактори постійно примножують свої функціональні можливості, зараз вони можуть скласти гідну конкуренцію десктоп-додаткам.

Загалом практично усі сервіси такого виду зараз підтримують багатокористувацький режим. Це означає, що така програма дозволяє її використання декількома користувачами одночасно. Тобто декілька розробників мають змогу писати та редагувати свій код на різних робочих

					ІАЛЦ.467200.003 ПЗ	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

станціях. Таким чином робочі процеси можливо розділити між кількома користувачами.

Онлайн-системи редагування програмного коду використовують для таких цілей:

- Співпраця. Налагодити спільну роботу з іншими людьми, використовуючи IDE, доволі складно, а онлайн-редактори спрощують цей процес. Робота з ними нагадує роботу з Google Docs, де кілька користувачів одночасно мають доступ та можуть вносити зміни до документу.
- Можливість поділитися кодом. За допомогою онлайн-редакторів можна поділитися програмним кодом з колегами, викладачами та і взагалі з ким-завгодно. Користувач може розміщувати вбудовані уривки коду чи посилання на «пісочницю» (ізольоване середовище) в документації, інструкціях та на інших веб-сайтах.
- Швидке розміщення нових додатків. Деякі редактори коду мають інтеграцію з популярними хостингами, тому користувач може розробити простий додаток і буквально за лічені хвилини розмістити його на хостингу, щоб протестувати.
- Онлайн-співбесіди та найм розробників. З допомогою онлайн-редакторів коду можна проводити співбесіди. Також стає можливим проведення онлайн-співбесід, коли кандидат та інтерв'юер знаходяться на відстані та організовують бесіду використовуючи спеціальні додатки для так званих онлайн-конференцій, що набуло особливої популярності в останні кілька років. Таким чином інтерв'юер може побачити як кандидат підходить до вирішення конкретних задач і як він працює з кодом.
- Прототипування. Ідеї варто тестувати наскільки можливо раніше і наскільки можливо частіше. Перевіряючи свої ідеї, розробник може отримувати миттєвий фідбек від користувачів. Функціонал для

					ІАЛЦ.467200.003 ПЗ	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

спільної роботи та функціонал, призначений для того, щоб ділитися кодом, допоможе в цьому.

- Навчання. У сучасній веб-розробці потрібно постійно слідкувати за останніми новинами, а ще краще – експериментувати з новинками в реальних умовах. Онлайн-редактори коду дозволяють це робити достатньо швидко і без іноді зайвих зусиль, витрачених на конфігурацію. Надзвичайно привабливо виглядає можливість створення окремої приватної «пісочниці» для навчання та експериментів, чим активно користуються не тільки досвідчені користувачі, а й розробники-новачки, адже щоб почати писати код не потрібно налаштовувати додаткового програмного забезпечення для розробки, варто лише підключитися до мережі Інтернет та перейти за відповідним посиланням у браузері. З такою метою можна використовувати практично будь-який браузер, оскільки зазвичай такі системи є веб-додатками, що підтримуються усіма сучасними браузерами.

Загалом системи редагування програмного коду відрізняються за своїм функціональним спектром та рівнем складності, але зазвичай спільною рисою є те, що такі додатки діляться на кілька робочих областей, у яких відображається сам код, а також результат виконання програмного коду.

Переваги використання додатків-редакторів коду:

- Не потрібно нічого додатково налаштовувати. Користувачу не доведеться завантажувати, встановлювати та налаштовувати IDE.
- Можливість поділитися кодом і працювати над ним спільно з іншими людьми.
- Редакторами можна користуватися безкоштовно або за відносно невелику плату. Більшість таких сервісів мають безплатну версію, функціоналу котрих буде достатньо для звичайних щоденних задач.

Недоліки онлайн-редакторів коду:

					ІАЛЦ.467200.003 ПЗ	Арк.
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

- Обмеженість функціоналу. У деяких редакторах коду відсутній навіть базовий функціонал, тому їх навіть складно назвати редакторами коду.
- Проблеми з продуктивністю.
- Відсутність контролю версій. Більшість онлайн-редакторів коду не мають інтеграцій з інструментами git.
- Залежність розробки від наявності та якості Інтернет-з'єднання.

1.3 Огляд існуючих аналогів

У даному розділі будуть розглянуті системи-аналоги, а саме веб-додатки, що дозволяють писати та редагувати програмний код.

HTML Code Editor (рис. 1.2) – популярний редактор коду, що підходить виключно для редагування HTML. Він абсолютно безкоштовний. Серед його функцій варто відзначити:

- автодоповнення коду;
- базові теги в хедері;
- tag wizard;
- пошук і заміна тегів;
- підсвічування коду;
- широкий спектр функцій очищення.

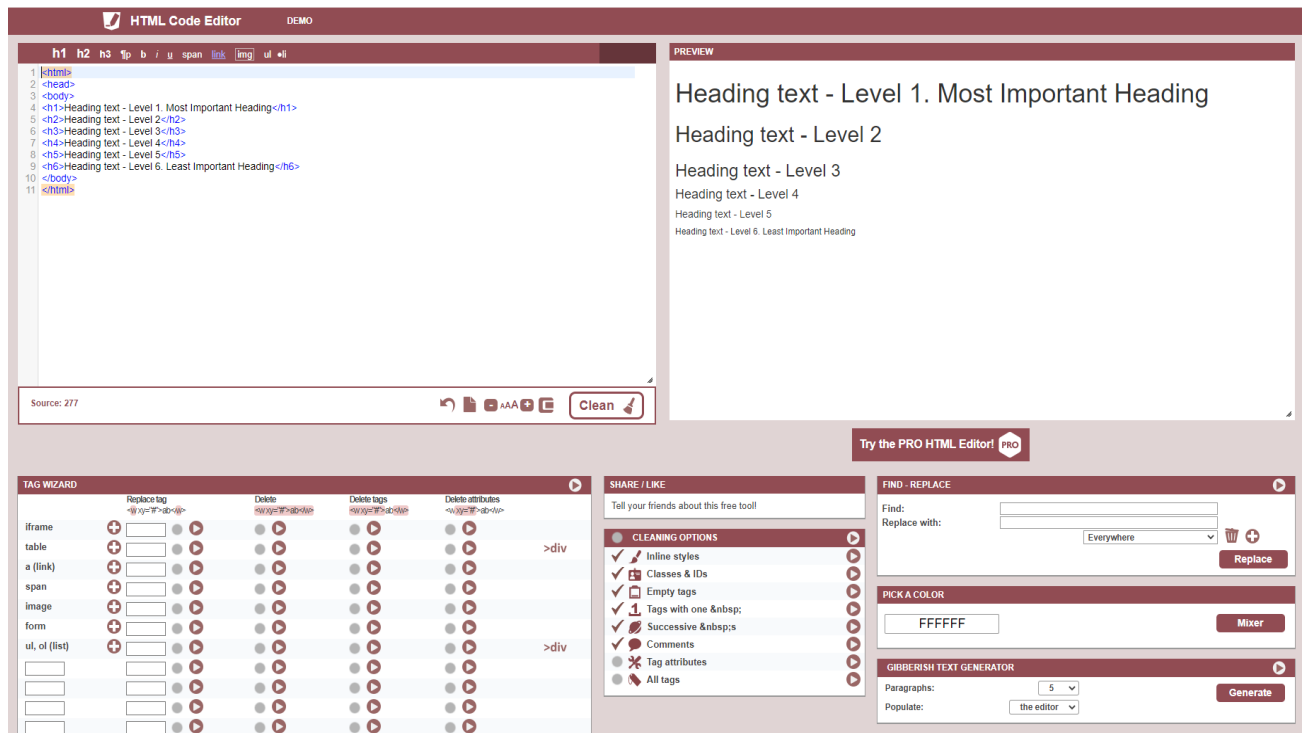


Рис. 1.2: Інтерфейс сервісу HTML Code Editor.

Серед недоліків програми:

- істотним недоліком цього редактора є такий аспект: оскільки сервіс безкоштовний, на веб-сайті присутня велика кількість реклами, що часто відволікає користувачів від основної задачі;
- незважаючи на те, що редактор має велику кількість зручних інструментів, вони можуть бути застосовані тільки для редагування HTML.

iCoder.Uz (рис. 1.3) — форум для програмістів, основною метою якого є навчання аудиторії інтернету програмуванню на різних мовах. iCoder.Uz являє собою веб-сайт, що надає користувачам можливість користуватися такими утилітами, як різноманітні конвертери, генератори, валідатори, інструменти домена та IP, утиліти для шифрування та перевірки адаптивності веб-сайтів, приклади коду, а також серед можливостей сервісу можна обрати вбудований онлайн редактор коду.

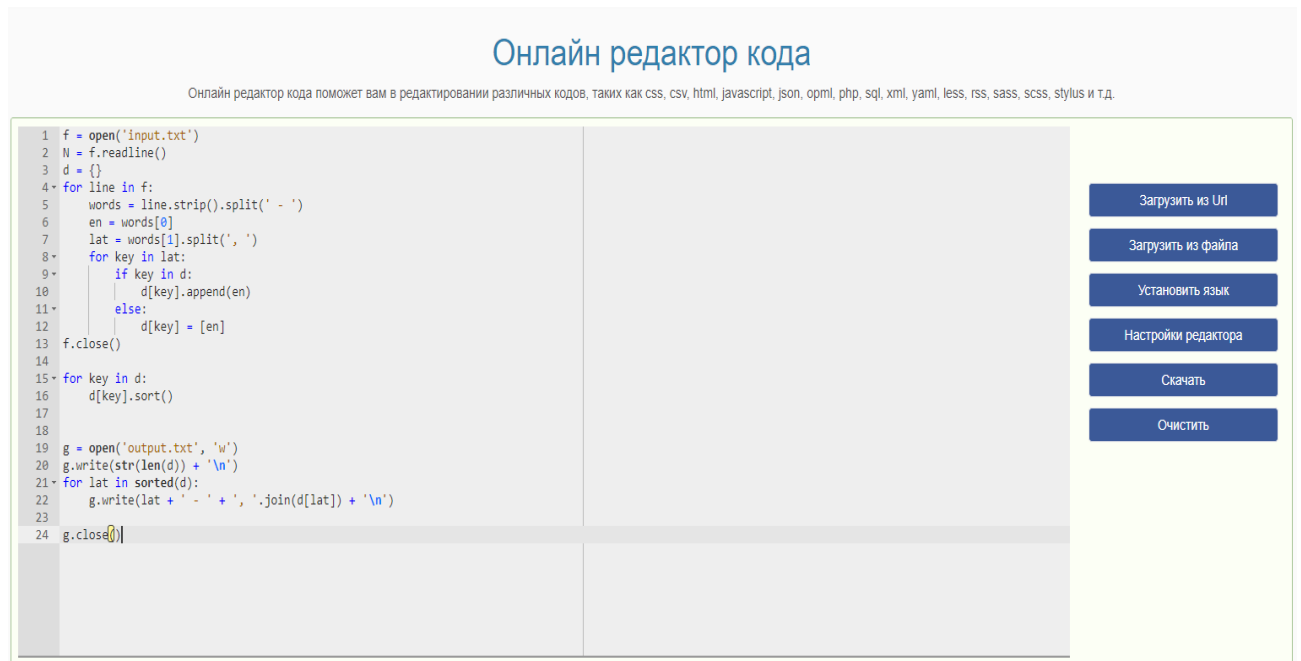


Рис. 1.3: Интерфейс вбудованого онлайн редактору коду для веб-сайту iCoder.Uz.

Переваги даного онлайн редактору:

- присутній вибір широкого спектру мов програмування;
- безкоштовний сервіс;
- завантаження коду з файлу;
- автодоповнення синтаксису;
- підсвічування коду;
- можливість завантажити написаний код у файл.

Серед недоліків:

- неможливо додати зображення;
- не надто сучасний та на інтуїтивному рівні зрозумілий інтерфейс;
- відсутність наглядності, відображення та перегляду результатів;
- відсутність базової файлової системи;
- хоча автодоповнення синтаксису присутнє, відкриті лапки та дужки не закриваються автоматично;

- прогрес (дані, що були введені користувачем) не зберігається в редакторі, потрібно щоразу писати код заново, копіювати або завантажувати з існуючого файлу.

Repl.it (рис. 1.4) — це один з найпотужніших онлайн-інструментів для розробки. Він підтримує широкий спектр мов і фреймворків, включаючи популярні зараз Haskell і Kotlin, а також широко відомі Javascript, C, C++, Ruby, Python, Golang, C# та ін.

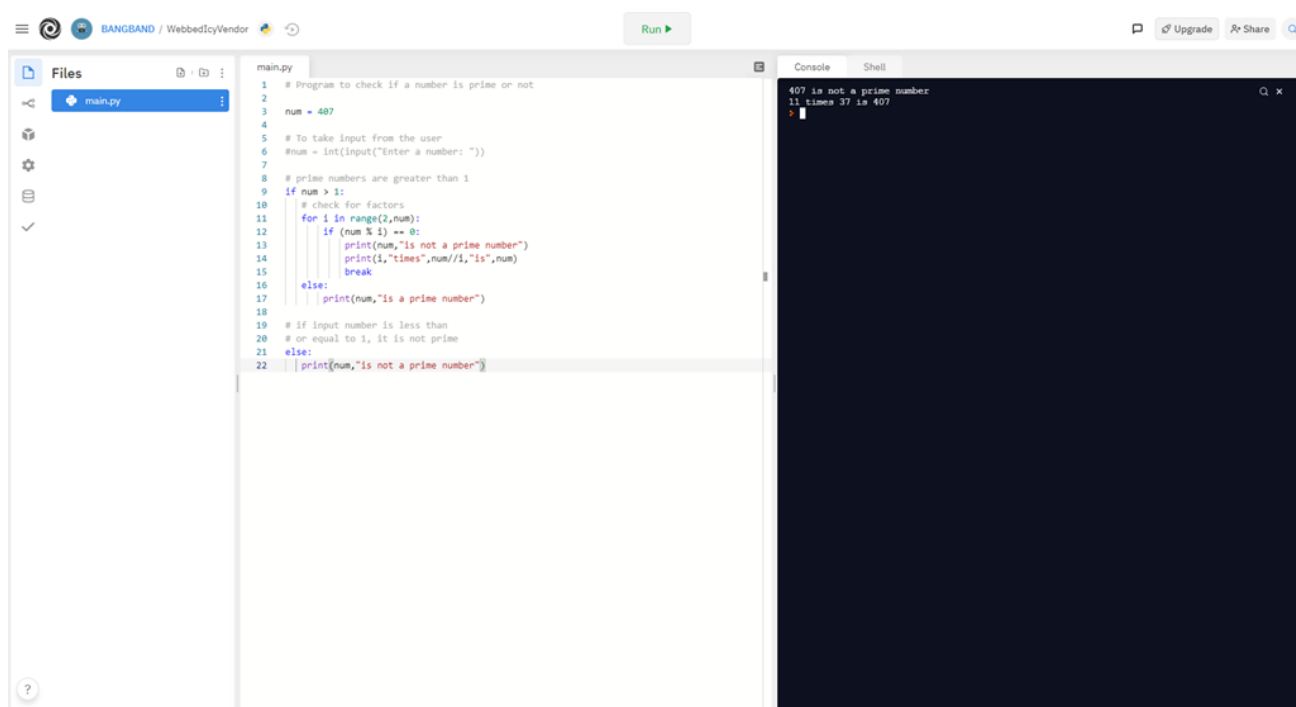


Рис. 1.4: Інтерфейс сервісу Repl.it.

На головній сторінці вам спочатку запропонують вибрати мову, а потім одну з категорій: веб-розробка, геймдев та ін. Після цього ви створюєте проект - repl.

Серед переваг:

- наявність файлової структури;
- підсвічування коду;
- у налаштуваннях можна редагувати макет, тему, розмір шрифту і відступів;

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		16

- що стосується можливості поділитися кодом, це можна зробити за допомогою вбудованого або звичайного посилання і інтеграції зі спільнотою dev.to; Також є функціонал соціальної мережі під назвою «talk», де розробники можуть ділитися кодом і отримувати фідбек;
- Repl.it має ще одну важливу особливість - Classrooms (класні кімнати). Користувачі можуть створювати ці кімнати, запрошувати в них учнів і відслідковувати прогрес.

Недоліки Repl.it:

- хоча додаток має безкоштовну версію, більш широкий спектр можливостей доступний саме у платній версії;
- відсутність автодоповнення коду;
- для того, щоб почати писати код, потрібно зареєструватися.

compilejava.net (рис. 1.5) – ще один спосіб уникнути використання прив'язаного до локальної машини середовища розробки. Широко використовується серед Java-розробників початківців та більш досвідчених фахівців. Вдало підійде в якості редактора для лайв-кодингу під час співбесіди, а також швидкої демонстрації невеликих прикладів коду.

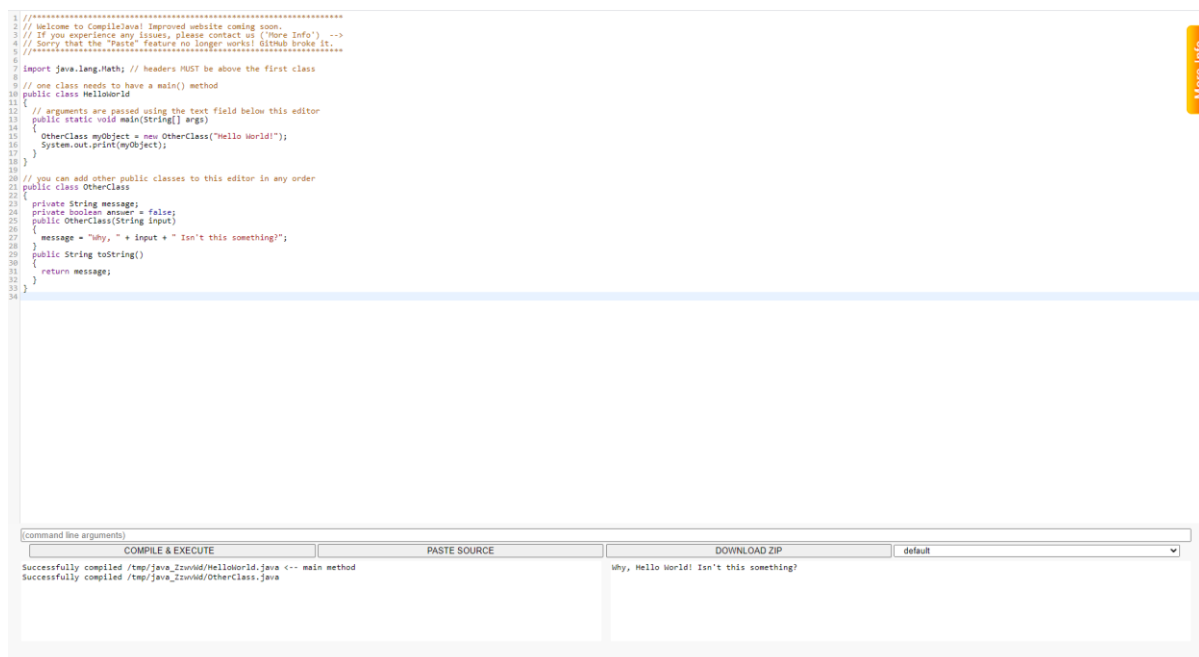


Рис. 1.5: Інтерфейс сервісу compilejava.net.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		17

Перевагами даного сервісу є:

- простий інтерфейс;
- можливість компілювання та виконання фрагменту коду;
- підсвічування синтаксису;
- можливість коментування;
- наявна можливість збереження архіву з написаним кодом;
- можливість змінити вигляд інтерфейсу (обрати тему).

Основними недоліками є:

- підтримує лише мову програмування java;
- відсутність автодоповнення коду;
- відсутність контролю версій;
- відсутність файлової системи;
- хоча можливість збереження архіву з написаним кодом присутня, іноді це є найзручнішим способом збереження прогресу (написаного програмного коду).

Codepen (рис. 1.6) – сервіс, створений для тестування та розміщення робочих фрагментів коду (сніппетів) на основі HTML, CSS та JavaScript. Це одна з найбільш відомих та найбільш популярних платформ на ринку редакторів програмного коду. Має зручний користувацький інтерфейс, користувачі за потреби можуть зареєструватися або продовжити роботу в гостьовому режимі. Авторизованому користувачеві доступно більше можливостей системи, ніж не авторизованому.

					ІАЛЦ.467200.003 ПЗ	Арк.
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

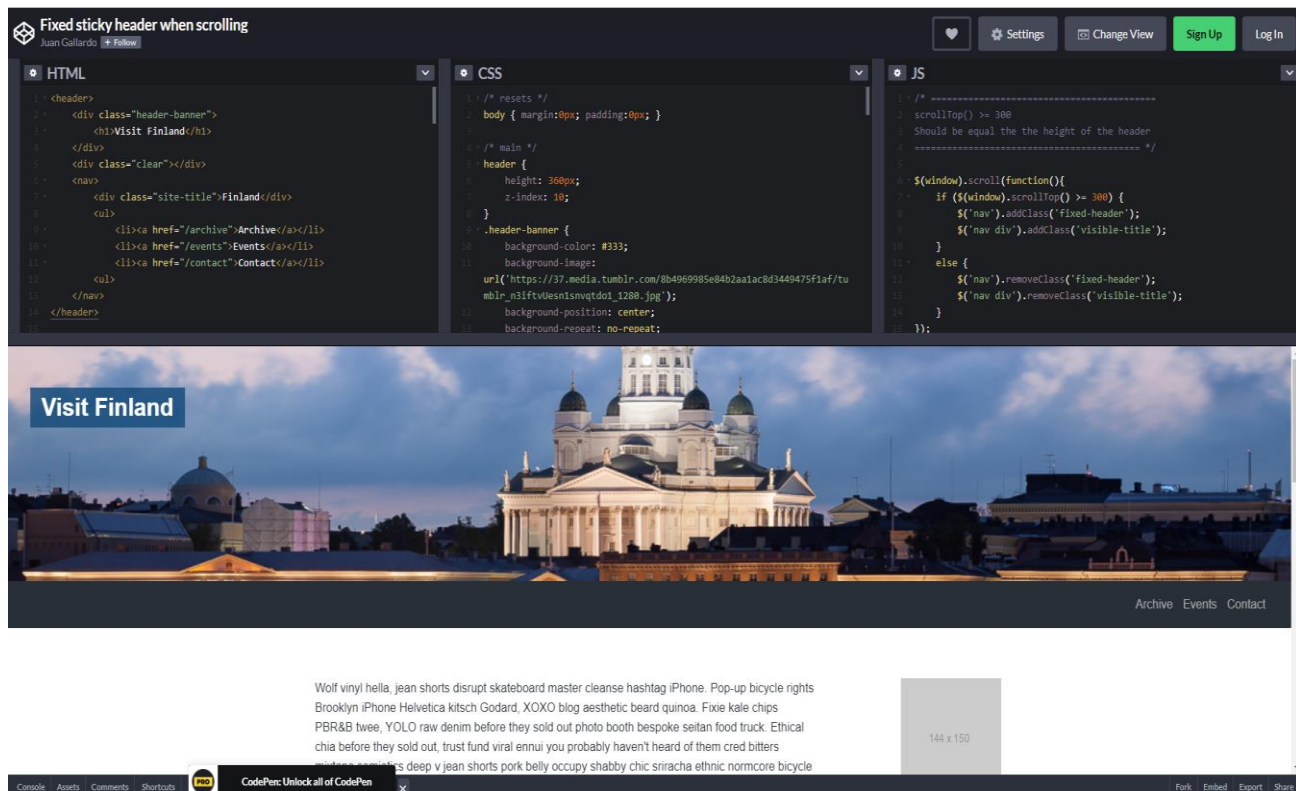


Рис. 1.6: Інтерфейс сервісу Codepen.

Цей інструмент певним чином поступово почав переростати в соціальну мережу, завдяки якій розробники мають змогу розповсюджувати серед інших користувачів свої доробки і отримувати схвалення від інших членів спільноти.

Серед властивостей системи, що забезпечують таку популярність серед розробників, варто відзначити:

- однією з найважливіших особливостей даної платформи є режим презентації коду, це особливо корисно для демонстрації написання коду в режимі реального часу при проведенні конференцій, онлайн-зустрічей та мітапів;
- коментування коду;
- автодоповнення коду;
- перегляд результатів, наглядність;
- деплоймент сайтів;
- режим спільної роботи;
- наявність базової файлової системи;

					ІАЛЦ.467200.003 ПЗ	Арк.
						19
Зм.	Арк.	№ докум.	Підпис	Дата		

- реалізований продуманий функціонал пошуку, тут користувач може шукати проекти, шаблони і уривки програмного коду на будь-яку тему, що його цікавить.

Також профілі на сайті Codepen можна розглядати, як цілком наглядні резюме веб-розробників, що прагнуть справити враження на потенційного роботодавця чи інвестора. Таким чином сайт Codepen – це чудова нагода оперативно обговорити код прямо в процесі його написання. Крім того, це зручний і універсальний навчальний посібник для веб-розробників початківців. Іншими словами, Codepen – комплексна багатофункціональна платформа з великою спільнотою.

Але вона має і недоліки:

- за замовчуванням користувачам Codepen доступна безкоштовна обмежена версія продукту. Для того, щоб отримати повний доступ до функціоналу додатку, потрібно буде сплачувати за доступ певну суму щомісяця;
- оскільки система має досить широкий спектр функціональних можливостей, часто користувачі (особливо нові) роблять вибір на користь менш комплексної та «перенавантаженої» платформи, де інтерфейс зрозумілий на інтуїтивному рівні;
- підтримує тільки HTML, CSS та JavaScript;
- неможливо додати файл (наприклад, зображення), лише посилання на нього.

					ІАЛЦ.467200.003 ПЗ	Арк.
						20
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі був проведений аналіз теоретичних даних, пов'язаних з імплементацією системи редагування програмного коду, а також порівняння та огляд багатокористувацьких систем редагування програмного коду, наявних наразі на ринку.

Огляд сучасних багатокористувацьких систем редагування програмного коду показав:

- На сьогоднішній день існує велика кількість схожих рішень для написання програмного коду. Переважна частина таких систем має спільні риси, але є і окремі рішення, що відрізняються за рахунок певних функцій, що користуються більшим попитом у розробників ПЗ.
- Основне завдання таких систем – максимально спростити та зробити зручним для користувача процес написання та редагування програмного коду.
- Після аналізу готових реалізацій систем редагування програмного коду стали очевидними такі недоліки:
 - обмежений доступ до функціоналу додатку, щоб отримати повний доступ до функціоналу користувач має внести додаткову плату;
 - «перенавантажений» інтерфейс системи;
 - підтримка лише однієї певної мови програмування;
 - відсутність наглядності, відображення та перегляду результатів;
 - відсутність автодоповнення коду;
 - відсутність коментування коду;
 - відсутність підсвічування синтаксису;
 - дані, що були введені користувачем, не зберігаються в редакторі;
 - велика кількість реклами у безкоштовних додатках.

- Зважаючи на перелічені вище недоліки, були визначені основні функціональні можливості багатокористувацької системи редагування програмного коду:
 - безкоштовний необмежений доступ до функцій сервісу;
 - відсутність реклами;
 - зручний та зрозумілий на інтуїтивному рівні користувацький інтерфейс;
 - автодоповнення коду;
 - коментування коду;
 - підсвічування синтаксису;
 - відображення та перегляд результатів;
 - збереження введених користувачем даних;
 - підтримка HTML, CSS та JavaScript, що дозволяє користування як окремою мовою, так і їх комбінацією.

					ІАЛЦ.467200.003 ПЗ	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2

ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ БАГАТОКОРИСТУВАЦЬКОЇ СИСТЕМИ РЕДАГУВАННЯ ПРОГРАМНОГО КОДУ

Надзвичайно важливим аспектом при розробці програми є вибір технологій та програмних засобів, що дозволять спростити розробку програмного продукту і в той же час зможуть надати потрібні інструменти для імплементації поставленої задачі. На виході розробник має отримати продукт, що повністю задовольнятиме користувача цього продукту.

2.1 Огляд призначення системи, що буде розроблюватись

У даній дипломній роботі буде розроблена багатокористувацька система редагування програмного коду. Ця система надає користувацький інтерфейс для написання програмного коду з використанням популярних на сьогоднішній день інструментів веб-розробки, а саме мови програмування JavaScript, каскадних таблиць стилів (CSS) та стандартизованої мови розмітки гіпертексту (HTML).

Користувачам буде надана можливість зареєструватися в системі, заповнивши поля «електронна пошта» та «пароль». Зареєстрований користувач може авторизуватися для подальшої роботи з системою, використовуючи значення електронної пошти та паролю, що були надані під час реєстрації. Після того, як користувач авторизувався, надається можливість змінити пошту або перейти до «робочої області», де, власне, і буде відбуватися написання програмного коду. По закінченню роботи користувач може вийти з системи.

					ІАЛЦ.467200.003 ПЗ	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

Також у разі, якщо користувач забув свій пароль чи бажає його змінити, наявна функція зміни паролю.

2.2 Огляд інструментів для реалізації багатокористувацької системи редагування програмного коду

Дана система потребує сучасних технологій для реалізації програмного рішення. Розглянемо інструменти, що були використані під час її розробки.

2.2.1 Visual Studio Code

Для написання даного проекту в якості інтегрованого середовища розробки був обраний Visual Studio Code.

Visual Studio Code - редактор вихідного коду, що був розроблений корпорацією Microsoft для Windows, Linux та macOS. Позиціонується як «легкий» редактор коду для кросплатформенної розробки веб- та хмарних додатків. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense та засоби для рефакторингу. Також присутні широкі можливості для кастомізації: користувацькі теми, комбінації клавіш та файли конфігурації. [\[4\]](#)

VS Code – безкоштовний та дуже популярний редактор. Свою популярність він здобув завдяки великій кількості вбудованих можливостей, що спрощують процес розробки.

Графічний інтерфейс програмного середовища Visual Studio Code можна побачити на Рисунку 2.1.

					ІАЛЦ.467200.003 ПЗ	Арк.
						24
Зм.	Арк.	№ докум.	Підпис	Дата		

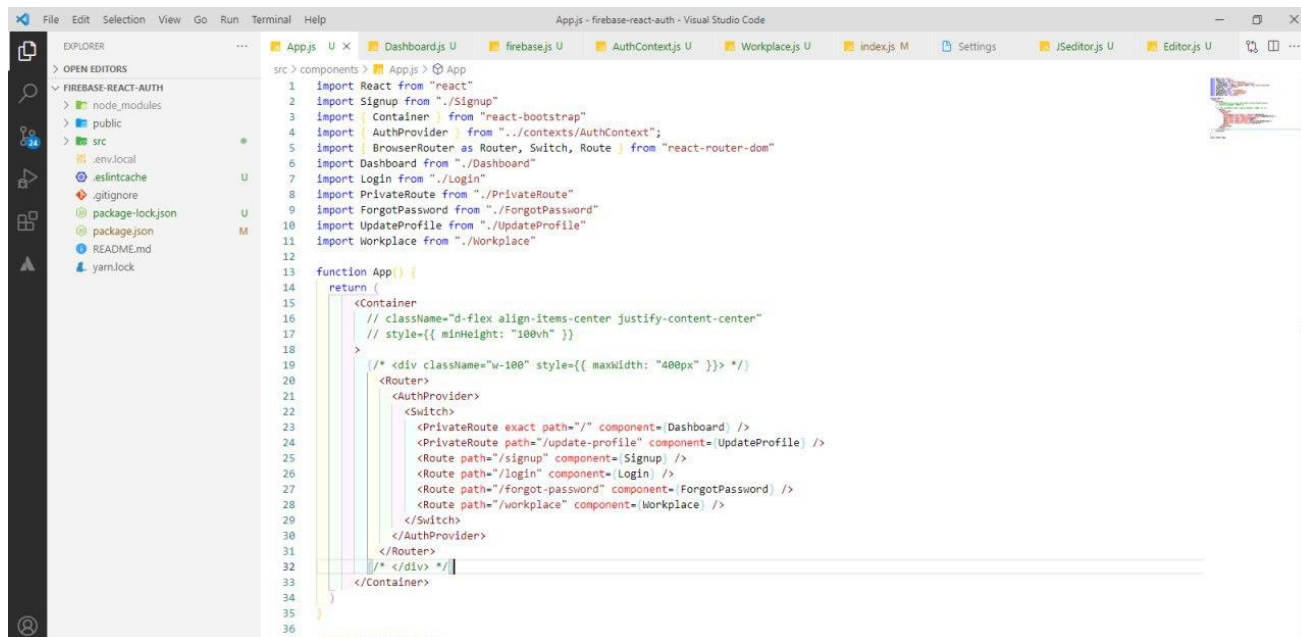


Рис. 2.1: Інтерфейс Visual Studio Code.

2.2.2 JavaScript та React

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript наразі є однією з найпопулярніших мов програмування в інтернеті. В перші роки існування, більшість професійних програмістів скептично ставилися до мови, а її цільова аудиторія складалася з програмістів-аматорів. Але подальші модифікації за стандартами ES6+ внесли багато корисних можливостей, яких не вистачало для ефективного програмування. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером. [\[6\]](#), [\[7\]](#), [\[10\]](#)

Для реалізації системи, описаної в даній роботі, використовується мова програмування JavaScript, а точніше — її бібліотека React.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		25

React — це декларативна, ефективна і гнучка JavaScript бібліотека для створення користувацьких інтерфейсів. Вона дозволяє збирати складний UI з маленьких ізольованих шматочків коду, званих «компонентами», а також покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. [\[5\]](#)

Кілька причин, чому розробники роблять вибір на користь програмування за допомогою React:

- React швидкий. Додатки, створені в React, можуть обробляти складні оновлення, але при цьому залишаються швидкими і «responsive».
- React модульний. Замість того, щоб писати великі файли коду, ви можете писати декілька менших файлів, які можна багаторазово використовувати. Модульність React може бути вдалим рішенням проблем з підтримкою JavaScript.
- React є масштабованим. React найкраще працює у великих програмах, які відображають багато змінних даних.
- React гнучкий. React можна використовувати для цікавих проектів, які не мають нічого спільного із створенням веб-програми. Потенціал React все ще з'ясовується та досліджується.
- React популярний. Хоча ця причина, очевидно, мало пов'язана з якістю React, правда полягає в тому, що розуміння React зробить розробника більш конкурентноспроможним.

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або у великих фреймворках MVC, як наприклад AngularJS.

					ІАЛЦ.467200.003 ПЗ	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

Модель–вид–контролер (MVC, Модель–представлення–контролер) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Схема взаємодії між компонентами у шаблоні MVC представлена на Рисунку 2.2.

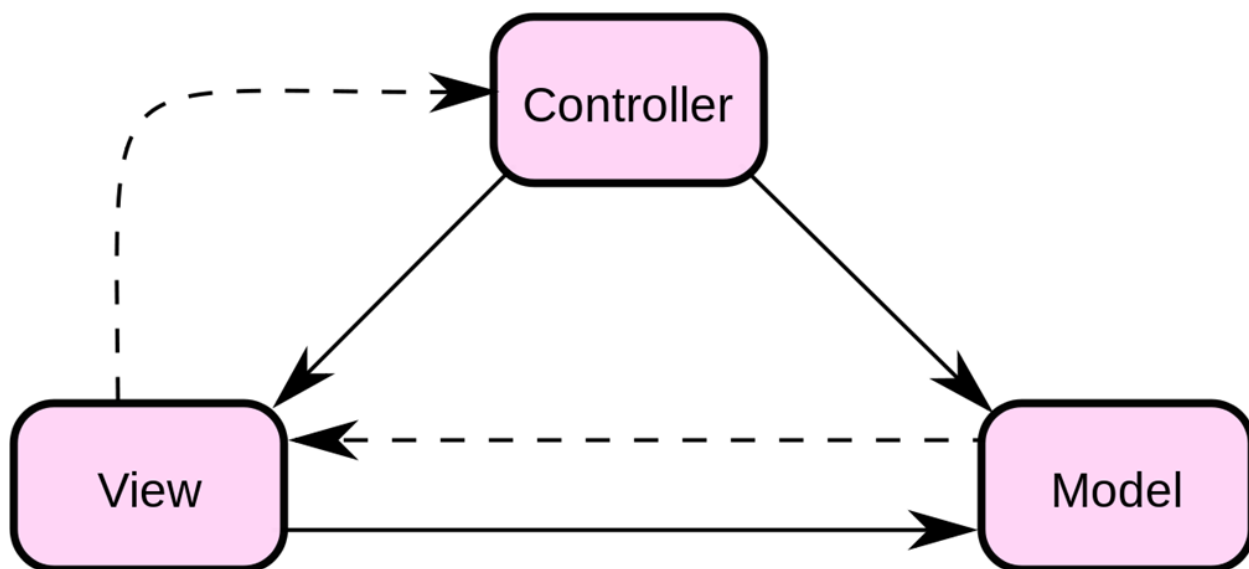


Рис. 2.2: Діаграма взаємодії між компонентами шаблону MVC.

Мета шаблону — гнучкий дизайн програмного забезпечення, призначений для полегшення подальших змін чи розширення програм, а також надання можливості повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності. [\[2\]](#)

2.2.3 CodeMirror

Оскільки система передбачає введення користувачем послідовності команд у робочій області, їх потрібно буде представити у вигляді не просто тексту, а програмного коду. Для такої цілі було обрано JavaScript бібліотеку CodeMirror.

CodeMirror — це універсальний текстовий редактор, реалізований в JavaScript для браузера і який можна вбудовувати у веб-сторінки. Він спеціалізується на редагуванні коду та постачається з низкою мовних режимів та доповнень, що реалізують розширені функції редагування. Режими — це програми JavaScript, які допомагають виділяти кольором (і, за бажанням, відступами) текст, написаний даною мовою.

Також для розробників доступний багатий API програмування та система тем CSS для налаштування CodeMirror відповідно до конкретної програми та розширення її за допомогою нових функціональних можливостей.

Основна бібліотека забезпечує лише компонент редактора, без супровідних кнопок, автодоповнення чи інших функцій IDE, але водночас вона надає обширний API, поверх якого така функціональність може бути реалізована.

Бібліотека підтримується більшістю найпопулярніших на сьогодні браузерів, а саме: Firefox (версії 4.0 і вище), Chrome (будь-яка версія), Safari (версії 5.2 і вище), Internet Explorer/Edge (версії 8.0 і вище), Opera (версії 9.0 і вище). [\[8\]](#)

Також оскільки CodeMirror може взаємодіяти та надавати широкий спектр мовних режимів, це відкриває нові можливості для подальшого розвитку та масштабування системи без додаткових інших технологій та без зайвих зусиль.

					ІАЛЦ.467200.003 ПЗ	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

2.2.4 Firebase

Для збереження даних користувача була обрана така технологія як Firebase.

Firebase — це одне з BaaS-рішень; платформа розробки додатків з величезним функціоналом, а конкретніше — хмарна база даних, яка дозволяє користувачам зберігати і отримувати збережену інформацію, а також має зручні засоби і методи взаємодії з нею.

Firebase є водночас сервером, базою даних, хостингом та аутентифікацією в одній платформі. Так, Firebase Realtime Database надає розробникам API, який синхронізує дані додатки між клієнтами і зберігає їх у хмарному сховищі.

Firebase зберігає текстові дані в JSON форматі і надає зручні методи для читання, оновлення та вилучення даних. Також, Firebase може допомогти з реєстрацією та авторизацією користувачів, зберіганням сесій (авторизовані користувачі), медіафайлів до яких з легкістю надає доступ завдяки Cloud Storage. Firebase особливо зручно використовувати для SPA або мобільних додатків, де основні маніпуляції стосуються саме текстових значень. У таких випадках Firebase можна використовувати як основну базу даних.

Серед служб і рішень для розробки Firebase виділяють наступні:

- Firebase Analytics
- Firebase Cloud Messaging
- Realtime Database
- Firebase Auth
- Firebase Storage
- Firebase Hosting та Functions
- ML Kit

У даному випадку для імплементації системи нас цікавитиме Firebase Auth та Realtime Database. Firebase Auth — це служба, яка може

					ІАЛЦ.467200.003 ПЗ	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

аутентифікувати користувачів, використовуючи лише код на стороні клієнта. Вона підтримує соціальні логін-провайдери Facebook, GitHub, Twitter, Google (і Google Play Games). Крім того, вона включає в себе систему управління користувачами, за допомогою якої розробники можуть увімкнути автентифікацію користувача за допомогою входу з електронної пошти та пароля, що зберігаються у Firebase.

Найосновніші і найбільш затребувані функції реєстрації, авторизації та зберігання тексту доступні кожному після реєстрації в системі.

Щодо Realtime Database, Firebase надає в режимі реального часу базу даних та бекенд як службу. Ця служба надає розробникам застосунків API, який дозволяє синхронізувати дані застосунків між клієнтами та зберігати їх у хмарі Firebase. Компанія також надає клієнтські бібліотеки, які дозволяють інтеграцію із застосунками Android, iOS, JavaScript / Node.js, Java, Objective-C, Swift. База даних також доступна через REST API та прив'язки до декількох сценаріїв JavaScript, таких як AngularJS, React, Ember.js та Backbone.js. REST API використовує протокол подій із сервером, який є інтерфейсом для створення HTTP-з'єднань для отримання push-повідомлень від сервера. Розробники, які використовують Realtime Database, можуть захищати свої дані за допомогою правил безпеки, що застосовуються на сервері. [\[9\]](#)

					ІАЛЦ.467200.003 ПЗ	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ ДО РОЗДІЛУ 2

У даному розділі був наведений опис і проведений аналіз технологій та інструментів, що використовуються для створення багатокористувацької системи редагування програмного коду.

Було розглянуто середовище для розробки Visual Studio Code та представлено короткий перелік його функцій та допоміжних інструментів.

Для проектування системи було обрано декларативну ефективну і гнучку JavaScript бібліотеку React, оскільки вона дозволяє збирати складний UI з компонентів, які можна використовувати повторно для уникнення повторення коду.

Також для представлення введених команд користувачем у вікні редактору у вигляді програмного коду було обрано JavaScript бібліотеку CodeMirror, оскільки вона надає розробникам API програмування та систему тем CSS для налаштування CodeMirror відповідно до конкретної програми та розширення її за допомогою інших функціональних можливостей.

Для збереження і отримання збереженої інформації було обрано хмарну базу даних Firebase. Платформа Firebase надає служби та рішення для авторизації, аутентифікації користувачів системи, а також для читання, запису та оновлення даних.

					ІАЛЦ.467200.003 ПЗ	Арк.
						31
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3

ОПИС РОЗРОБКИ БАГАТОКОРИСТУВАЦЬКОЇ СИСТЕМИ РЕДАГУВАННЯ ПРОГРАМНОГО КОДУ

У рамках даного розділу буде представлена розробка багатокористувацької системи редагування програмного коду.

3.1 Ініціалізація проекту

Щоб почати роботу з React перш за все необхідно створити React-проект за допомогою команди `npm create-react-app code-editor`. Далі, щоб користуватися можливостями Firebase, створюємо у консолі на офіційному сайті Firebase новий проект, це надасть нам об'єкт конфігурації, який буде збережений у окремому файлі `firebase.js`, що знаходиться у папці `src`. Цей об'єкт конфігурації буде експортований для подальшої роботи. Приклад експорту наведений на рисунку 3.1.

```
import firebase from "firebase/app"
import "firebase/auth"

const app = firebase.initializeApp({
  apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
  authDomain: process.env.REACT_APP_FIREBASE_AUTH_DOMAIN,
  databaseURL: process.env.REACT_APP_FIREBASE_DATABASE_URL,
  projectId: process.env.REACT_APP_FIREBASE_PROJECT_ID,
  storageBucket: process.env.REACT_APP_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: process.env.REACT_APP_FIREBASE_MESSAGING_SENDER_ID,
  appId: process.env.REACT_APP_FIREBASE_APP_ID
})

export const auth = app.auth()
export default app
```

Рис. 3.1: Вміст файлу `firebase.js`.

Маючи цю конфігурацію, створимо ще один файл, де буде ініціалізовано наш додаток firebase за допомогою цієї конфігурації, а потім експортовано для подальшого повторного використання в коді.

3.2 Реалізація авторизації та автентифікації

Основні методи, потрібні для авторизації та автентифікації, знаходяться у межах функції AuthProvider. А саме signup, login, logout, resetPassword, updateEmail та updatePassword, що приймають значення email та/або password.

```
export function AuthProvider({ children }) {
  const [currentUser, setCurrentUser] = useState()
  const [loading, setLoading] = useState(true)

  function signup(email, password) {
    return auth.createUserWithEmailAndPassword(email, password)
  }

  function login(email, password) {
    return auth.signInWithEmailAndPassword(email, password)
  }

  function logout() {
    return auth.signOut()
  }

  function resetPassword(email) {
    return auth.sendPasswordResetEmail(email)
  }

  function updateEmail(email) {
    return currentUser.updateEmail(email)
  }

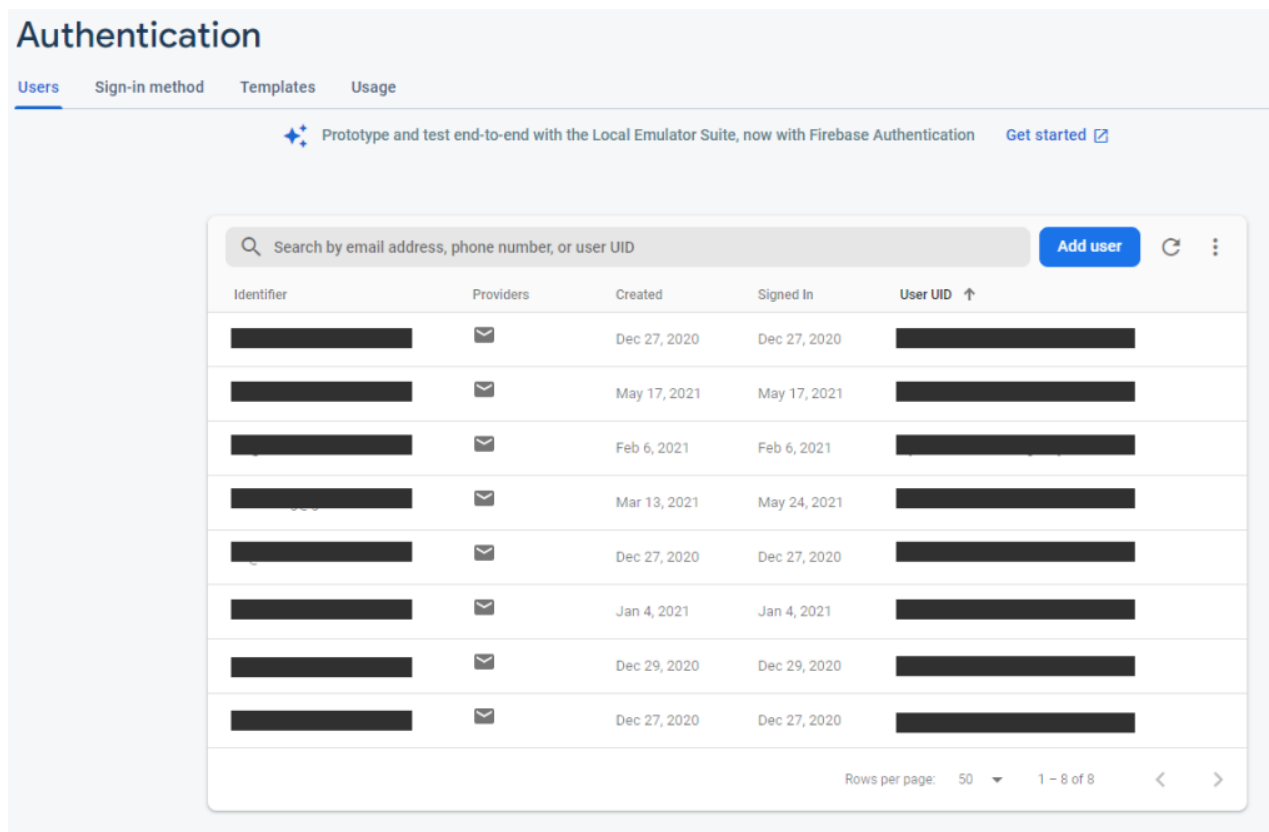
  function updatePassword(password) {
    return currentUser.updatePassword(password)
  }

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged(user => {
      setCurrentUser(user)
      setLoading(false)
    })

    return unsubscribe
  }, [])
}
```

Рис. 3.2: Функція AuthProvider.

Щоразу, коли користувач реєструється в системі, у створеному Firebase-проекті у секції Authentication з'являється запис про користувача. Запис містить такі дані, як ідентифікатор, провайдери (у даному випадку це електронна пошта), дату реєстрації користувача, дату останнього входу до системи та унікальний User UID.



The screenshot shows the 'Authentication' section of the Firebase console, specifically the 'Users' tab. It features a search bar at the top with the text 'Search by email address, phone number, or user UID'. Below the search bar is a table with the following columns: Identifier, Providers, Created, Signed In, and User UID. The table contains 8 rows of user data. At the bottom right of the table, there is a pagination control showing 'Rows per page: 50' and '1 - 8 of 8'.

Identifier	Providers	Created	Signed In	User UID ↑
[REDACTED]	[Email]	Dec 27, 2020	Dec 27, 2020	[REDACTED]
[REDACTED]	[Email]	May 17, 2021	May 17, 2021	[REDACTED]
[REDACTED]	[Email]	Feb 6, 2021	Feb 6, 2021	[REDACTED]
[REDACTED]	[Email]	Mar 13, 2021	May 24, 2021	[REDACTED]
[REDACTED]	[Email]	Dec 27, 2020	Dec 27, 2020	[REDACTED]
[REDACTED]	[Email]	Jan 4, 2021	Jan 4, 2021	[REDACTED]
[REDACTED]	[Email]	Dec 29, 2020	Dec 29, 2020	[REDACTED]
[REDACTED]	[Email]	Dec 27, 2020	Dec 27, 2020	[REDACTED]

Рис. 3.3: Таблиця з даними про користувача у секції Authentication.

3.3 React components та React router

Важливою складовою кожного React-додатку є компоненти. Компоненти схожі на функції, які повертають HTML-елементи.

Компоненти - це незалежні та багаторазові біти коду, що дозволяють уникнути копіювання коду завдяки їх повторному використанню. Вони виконують ті самі цілі, що і функції JavaScript, але працюють ізольовано і повертають HTML через функцію render().

При створенні ім'я компонентів React має починатися з великої літери. Компонент повинен наслідувати `React.Component` і надавати конкретному компоненту доступ до функцій `React.Component`.

Основним компонентом даного React-додатку є `App`, що знаходиться у JavaScript-файлі `App.js`. Він містить контейнер, що включає в себе список інших компонентів, до яких користувач матиме доступ у залежності від обраного шляху.

У даному випадку цей список включає такі компоненти, як:

- `Dashboard` (відповідає за сторінку профілю користувача);
- `UpdateProfile` (відповідає за сторінку оновлення даних профілю користувача);
- `Signup` (відповідає за сторінку реєстрації користувача);
- `Login` (відповідає за сторінку входу в систему);
- `ForgotPassword` (відповідає за сторінку зміни паролю);
- `Workplace` (відповідає безпосередньо за вікно редактора програмного коду).

Усі ці компоненти містяться у окремих файлах з розширенням `.js` та мають бути імпортовані перед використанням їх у функції `App`.

Структура компоненту `App`, що знаходиться у JavaScript-файлі `App.js`, зображена на рисунку 3.4. Усі вище згадані компоненти знаходяться у межах тегу `<Switch>`.

```

import React from "react"
import Signup from "../Signup"
import { Container } from "react-bootstrap"
import { AuthProvider } from "../contexts/AuthContext";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom"
import Dashboard from "../Dashboard"
import Login from "../Login"
import PrivateRoute from "../PrivateRoute"
import ForgotPassword from "../ForgotPassword"
import UpdateProfile from "../UpdateProfile"
import Workplace from "../Workplace"

function App() {
  return (
    <Container>
      <Router>
        <AuthProvider>
          <Switch>
            <PrivateRoute exact path="/" component={Dashboard} />
            <PrivateRoute path="/update-profile" component={UpdateProfile} />
            <Route path="/signup" component={Signup} />
            <Route path="/login" component={Login} />
            <Route path="/forgot-password" component={ForgotPassword} />
            <Route path="/workplace" component={Workplace} />
          </Switch>
        </AuthProvider>
      </Router>
    </Container>
  )
}

export default App;

```

Рис. 3.4: Вміст файлу App.js.

У React є своя система маршрутизації, яка дозволяє зіставляти запити до додатка з певними компонентами. Ключовою ланкою в роботі маршрутизації є модуль `react-router`, який містить основний функціонал по роботі з маршрутизацією.

Тож у даному випадку навігація між компонентами додатку відбувається за допомогою `react-router`.

У системі маршрутизації кожен маршрут зіставляється з певним компонентом. Router визначає набір маршрутів і, коли до додатка, приходить запит, то Router виконує зіставлення запиту з маршрутами. І якщо якийсь

					ІАЛЦ.467200.003 ПЗ	Арк.
						36
Зм.	Арк.	№ докум.	Підпис	Дата		

маршрут збігається з URL запиту, то цей маршрут вибирається для обробки запиту.

І також для вибору маршруту визначено об'єкт Switch. Він дозволяє вибрати перший-ліпший маршрут і його використовувати для обробки. Без цього об'єкта Router може використовувати для обробки одного запиту теоретично кілька маршрутів, якщо вони відповідають рядку запиту.

Кожен маршрут представляє об'єкт Route. Він має ряд атрибутів. Зокрема, тут для маршруту встановлюються два атрибути:

1. path: шаблон адреси, з яким буде зіставлятися запитана адреса URL;
2. component - компонент, який відповідає за обробку запиту по цьому маршруту.

Причому в даному випадку один маршрут виступає в якості кореневого. Він зіставляється з адресою "/" і обробляється компонентом Dashboard.

Варто зазначити, що даний маршрут використовує кваліфікатор exact. Цей кваліфікатор допускає тільки точний збіг маршруту з рядком запиту. Наприклад, рядок запиту може представляти шлях "/" або шлях "/signup", шлях "/login" тощо, і всі ці шляхи будуть відповідати маршруту з шаблоном адреси "/". Тобто маршруту, який обробляється компонентом Dashboard. Але слово exact дозволяє розглядати точний збіг, тобто коли шаблоном "/" відповідає тільки рядок запиту "/".

```
<Router>
  <AuthProvider>
    <Switch>
      <PrivateRoute exact path="/" component={Dashboard} />
      <PrivateRoute path="/update-profile" component={UpdateProfile} />
      <Route path="/signup" component={Signup} />
      <Route path="/login" component={Login} />
      <Route path="/forgot-password" component={ForgotPassword} />
      <Route path="/workplace" component={Workplace} />
    </Switch>
  </AuthProvider>
</Router>
```

Рис. 3.5: Router, Switch та Route.

					ІАЛЦ.467200.003 ПЗ	Арк.
						37
Зм.	Арк.	№ докум.	Підпис	Дата		

У свою чергу посилання всередині компонентів на інші компоненти є також можливою і досить часто використовуваною практикою серед React-розробників.

Саме такий підхід було використано при проектуванні Workplace компоненту. Оскільки робоча область редактора коду має три однотипні вікна для написання коду (HTML, CSS та JavaScript), було створено окремий компонент, який можна буде перевикористати та для якого можна встановити такі параметри, як мова (language), відображувана назва (displayName) та значення (value).

Ще одна частина компоненту Workplace має відображати вивід для введеного коду, для цього варто вказати параметр sandbox зі значенням allow-scripts.

```
return (  
  <>  
    <div style={{ display: "flex", height: "50vh", backgroundColor: "#3c3e44" }} className="pane top-pane">  
      <Editor  
        language="xml"  
        displayName="HTML"  
        value={html}  
        onChange={setHtml}  
      />  
      <Editor  
        language="css"  
        displayName="CSS"  
        value={css}  
        onChange={setCss}  
      />  
      <Editor  
        language="javascript"  
        displayName="JS"  
        value={js}  
        onChange={setJs}  
      />  
    </div>  
    <div className="pane" style={{ display: "flex", height: "50vh" }}>  
      <iframe  
        srcDoc={srcDoc}  
        title="output"  
        sandbox="allow-scripts"  
        frameborder="0"  
        width="100%"  
        height="100%"  
      />  
    </div>  
  </>  
)
```

Рис. 3.6: Фрагмент компоненту Dashboard, що міститься у файлі Dashboard.js.

					ІАЛЦ.467200.003 ПЗ	Арк.
						38
Зм.	Арк.	№ докум.	Підпис	Дата		

Основний функціонал редактора визначається за допомогою опцій в елементі `ControlledEditor`, що міститься у кожному компоненті `Editor`. Тут за допомогою можливостей JavaScript-бібліотеки `Codemirror` встановлюються такі опції, як `lineWrapping`, `lint`, `mode` (що визначає мову програмування), `theme` (що визначає тему для візуального оформлення), `lineNumbers`, `smartIndent`, `foldGutter`, `gutters`, `autoCloseTags`, `keymap`, `matchBrackets`, `autoCloseBrackets` та `extraKeys` (тут вказуємо комбінацію клавіш `Ctrl+Space`, що зробить можливим автоматичне доповнення коду та вибір із запропонованого списку можливих варіантів доповнення).

```
<ControlledEditor
  onBeforeChange={handleChange}
  value={value}
  className="code-mirror-wrapper"
  options={{
    lineWrapping: true,
    lint: true,
    mode: language,
    theme: 'material',
    lineNumbers: true,
    smartIndent: true,
    foldGutter: true,
    gutters: ["CodeMirror-linenumbers", "CodeMirror-foldgutter"],
    autoCloseTags: true,
    keyMap: "sublime",
    matchBrackets: true,
    autoCloseBrackets: true,
    extraKeys: {
      "Ctrl-Space": "autocomplete"
    }
  }}
/>
```

Рис. 3.7: Фрагмент компоненту `Editor`, що міститься у файлі `Editor.js`.

3.4 Збереження даних користувача

Для збереження введених користувачем даних використовується база даних `Realtime Database`. База даних `Firebase Realtime Database` - це хмарна база

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		39

даних NoSQL, яка дозволяє зберігати та синхронізувати дані між своїми користувачами в режимі реального часу.

У даному випадку було створено таблицю users, у якій зберігаються записи про введені кожним користувачем дані у вікнах редактору HTML, CSS та JS. Відповідно, у кожного користувача має бути власний унікальний ідентифікатор та такі поля: css, html, js, до яких буде мати доступ лише користувач, у якого співпав ідентифікатор.

Схема цієї таблиці наведена на рисунку 3.8.

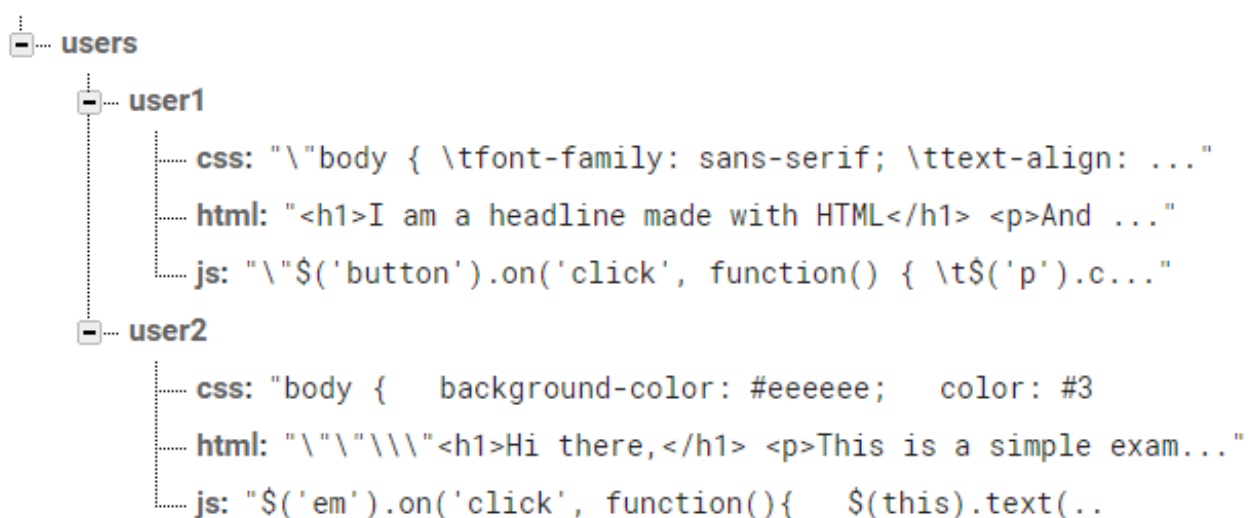


Рис. 3.8: Схема таблиці users.

Для взаємодії з базою даних було створено функцію, в якій перевіряється, чи існує користувач з певним ідентифікатором у базі даних, якщо так, то ми повернемо його дані, а якщо немає, створимо їх. Фрагмент коду, що ілюструє взаємодію з базою даних, наведено на рисунку 3.9.

```

import { db } from "lib/firebase";

const getUserFiles = async (userId) => {
  const doc = await db.collection("users").doc(userId).get();

  if (doc.exists) {
    console.log("User found in database");
    const snapshot = await db
      .collection("users")
      .doc(doc.id)
      .collection("files")
      .get();

    let userFiles = [];
    snapshot.forEach((file) => {
      let { name, content } = file.data();
      userFiles.push({ id: file.id, name: name, content: content });
    });
    return userFiles;
  } else {
    console.log("User not found in database, creating new entry...");
    db.collection("users").doc(userId).set({});
    return [];
  }
}

```

Рис. 3.9: Взаємодія з базою даних.

ВИСНОВКИ ДО РОЗДІЛУ 3

У даному розділі було наведено стислий огляд реалізації клієнтської та серверної частини багатокористувацької системи редагування програмного коду, взаємодію між ними, а також деталі реалізації основного функціоналу системи.

У якості технології для імплементації клієнтської частини було обрано фреймворк React. У розділі наведено опис основних компонентів, з яких був побудований користувацький інтерфейс системи.

Серверна частина була розроблена з використанням платформи Firebase, адже вона надає розробникам гнучкі інструменти для того, щоб у повній мірі реалізувати функції системи.

У результаті було реалізовано ефективну серверну та клієнтську частину системи, що є гнучкою та можливою для подальшого розширення та масштабування. У той же час система відповідає вимогам, що були представлені раніше.

					ІАЛЦ.467200.003 ПЗ	Арк.
						42
Зм.	Арк.	№ докум.	Підпис	Дата		

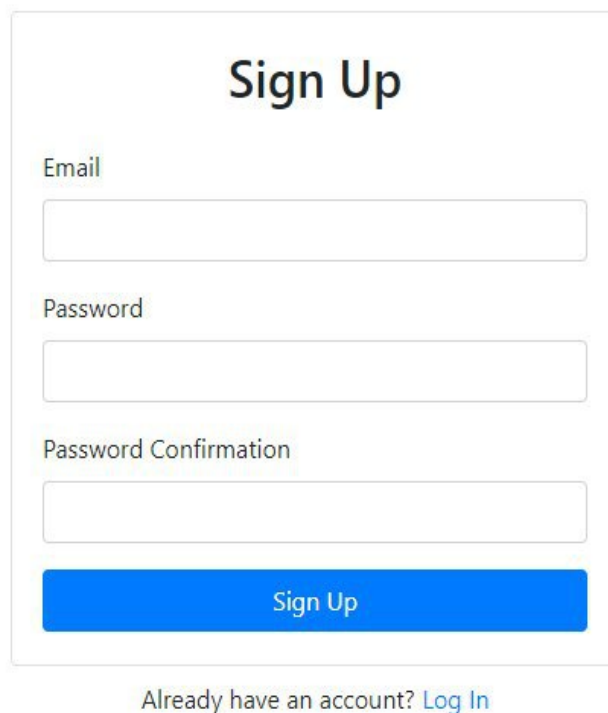
РОЗДІЛ 4

ВИКОРИСТАННЯ СИСТЕМИ

Даний розділ є інструкцією користувача із зазначеними можливими сценаріями використання реалізованої системи редагування програмного коду. Далі у розділі буде наведений огляд користувацького інтерфейсу та функціоналу програми.

4.1 Реєстрація в системі

Для того, щоб почати розботу з системою перш за все потрібно зареєструватися. Для цього потрібно заповнити значення електронної пошти, паролю та повторити пароль. Після натиснення конпки “Sign Up” дані користувача будуть збережені в системі. З вікна реєстрації є можливість перейти до вікна входу в систему. Для цього варто натиснути на “Log In”.



Sign Up

Email

Password

Password Confirmation

Sign Up

Already have an account? [Log In](#)

Рис. 4.1: Вікно реєстрації.

					ІАЛЦ.467200.003 ПЗ	Арк.
						43
Зм.	Арк.	№ докум.	Підпис	Дата		

Якщо при спробі реєстрації в системі паролі не співпадають, користувач бачить відповідне повідомлення.

The image shows a web form titled "Sign Up". At the top, there is a red error message box that says "Password do not match". Below this, the form has three input fields: "Email" with the value "test34502@test.com", "Password" with masked characters "*****", and "Password Confirmation" also with masked characters "*****". At the bottom of the form is a blue button labeled "Sign Up". Below the form, there is a link that says "Already have an account? Log In".

Рис. 4.2: Повідомлення при помилці реєстрації в системі.

4.2 Вхід до системи

Якщо користувач уже зареєстрований, він може увійти до системи заповнивши значення електронної пошти та паролю і натиснувши на кнопку “Log In”. З вікна входу в систему є можливість перейти до вікна зміни паролю та до вікна реєстрації. Для цього варто натиснути на “Forgot Password?” та “Sign Up” відповідно.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		44

Log In

Email

Password

Log In

[Forgot Password?](#)

Need an account? [Sign Up](#)

Рис. 4.3: Вікно входу в систему.

Log In

Failed to log in

Email

Password

Log In

[Forgot Password?](#)

Need an account? [Sign Up](#)

Рис. 4.4: Повідомлення при помилці входу в систему.

4.3 Вікно зміни паролю

У разі, якщо користувач забув свій пароль або бажає його змінити, він може скористатися функцією зміни паролю. Для цього варто ввести свою електронну пошту та натиснути на кнопку “Reset Password”, після цього на вказану пошту буде надіслано лист з посиланням на для зміни паролю.

Із вікна зміни паролю є можливість перейти до вікна входу в систему та до вікна реєстрації. Для цього варто натиснути на “Log In” та “Sign Up” відповідно.

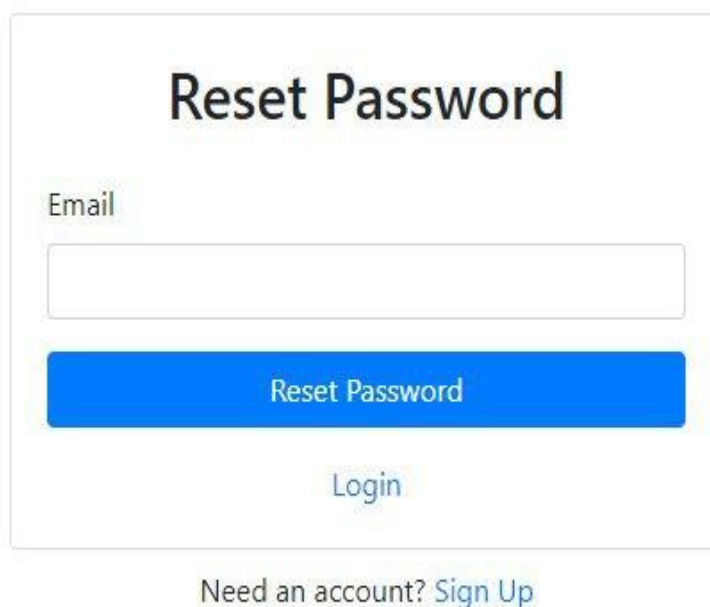
A screenshot of a web form titled "Reset Password". It features a text input field labeled "Email" with a light blue border. Below the input field is a prominent blue button with the text "Reset Password" in white. Underneath the button is a link labeled "Login" in a light blue font. At the bottom of the form, there is a link that reads "Need an account? Sign Up" in a light blue font.

Рис. 4.5: Вікно зміни паролю.

Reset Password

Check your inbox for further instructions

Email

Reset Password

[Login](#)

Need an account? [Sign Up](#)

(a)

Reset Password

Failed to reset password

Email

Reset Password

[Login](#)

Need an account? [Sign Up](#)

(б)

Рис. 4.6: Повідомлення при успішному (а) та не успішному (б) відправленні листа на вказану електронну пошту.

4.4 Сторінка профілю користувача

Після успішного входу в систему, користувачу стає доступна сторінка профілю. Тут відображається електронна пошта користувача та є опції: оновити профіль та перейти до робочої області.

Зі сторінки профілю користувача є можливість вийти з системи, натиснувши на “Log Out”.

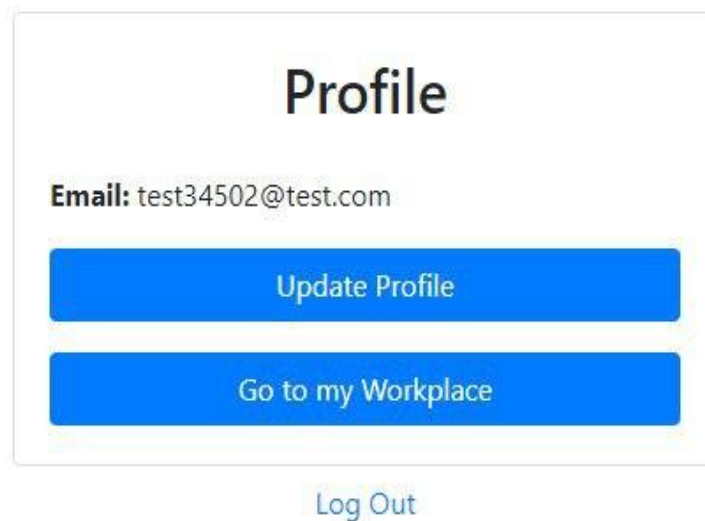


Рис. 4.7: Сторінка профілю користувача.

4.5 Вікно оновлення профілю

У вікні оновлення профілю користувач може змінити значення електронної пошти та пароля.

Є можливість повернутися назад до сторінки профілю користувача, натиснувши на “Cancel”.

Update Profile

Email

Password

Password Confirmation

Update

Cancel

Рис. 4.8: Вікно оновлення профілю.

4.6 Вікно редактора програмного коду

При переході зі сторінки профілю користувача до робочої області, стане доступним редактор коду. Він складається з трьох вікон для написання HTML, CSS та JavaScript коду, а також області, де буде відображатися результат написаних скриптів. У кожному вікні присутня нумерація строк та підсвічування синтаксису.

У кожному вікні редактора є можливість коментування строк коду. Приклади коментування HTML, CSS та JavaScript коду показані на рисунку 4.11.

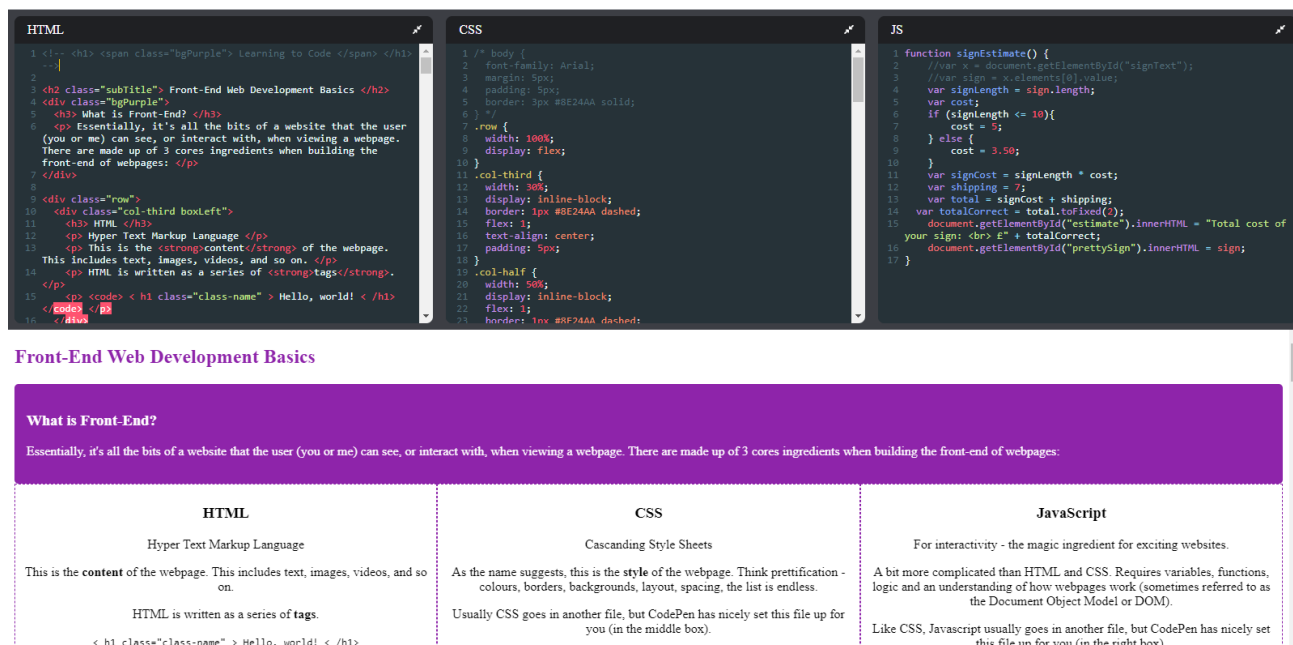


Рис. 4.11: Вигляд закоментованого та незакоментованого HTML, CSS та JavaScript коду.

У редакторі присутня така функціональна можливість, як автодоповнення коду. Щоб побачити можливі варіанти доповнення, потрібно натиснути комбінацію клавіш Ctrl+Space і після цього обрати потрібний варіант. Вигляд списку можливих варіантів доповнення зображений на рисунку 4.12.

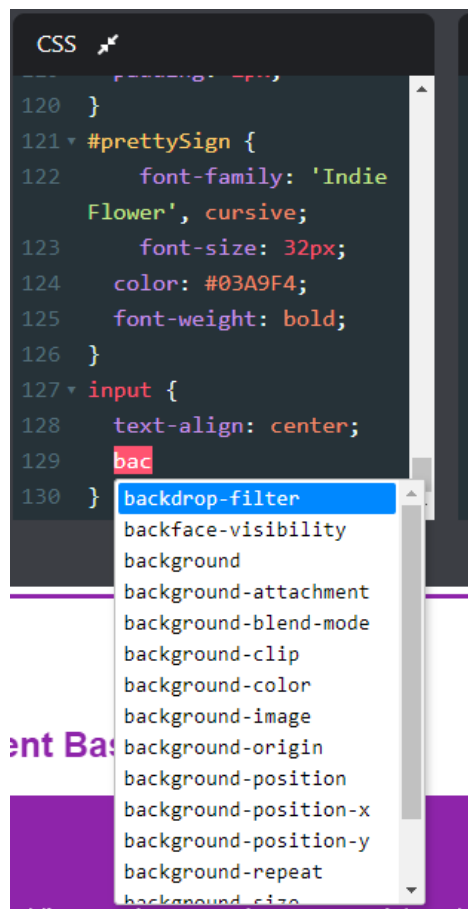
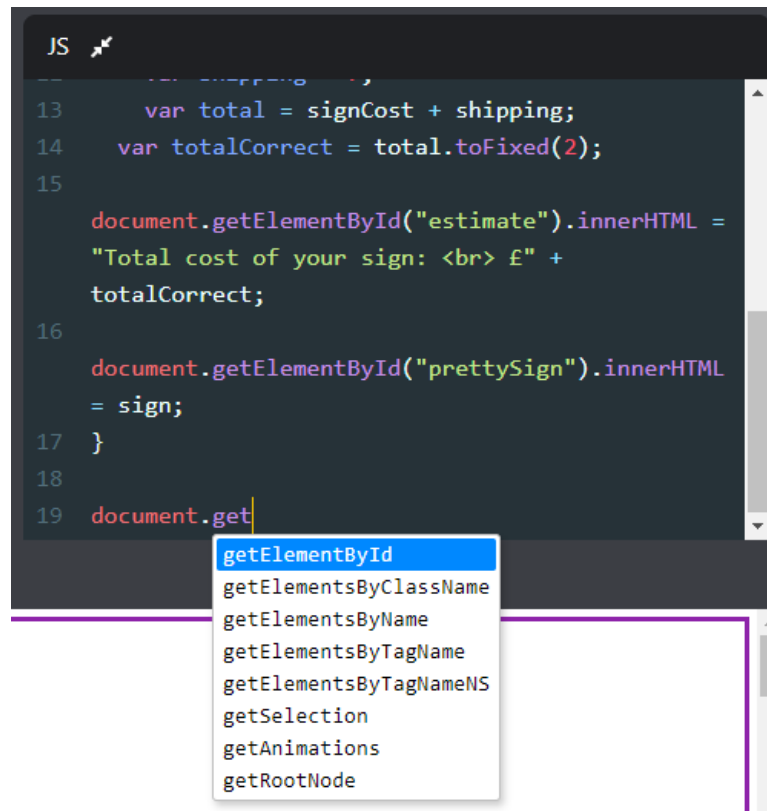


Рис. 4.12: Вигляд списку можливих варіантів доповнення.

Також у вікнах редактора є можливість згорнути та розгорнути великі строки коду, як наведено на рисунку 4.13.

```

HTML ✖
212 ▾ <div class="row">
213 ▾   <div class="col-half">
214 ▾     <p> Click the button on the right,
        for a very simple example. All the code for
        is in the right box, in a function
        called signEstimate. This does some
        calculations and changes the HTML on the
        website (interactivity!)</p>
215 ▾     <p> A <strong>function</strong> is a
        bit of code that you reuse over and over
        again. You could build a function to
        multiply two numbers easily.</p>

```

(a)

```

HTML ✖
212 ▾ <div class="row">
213 ▾   <div class="col-half">
214 ▾     <p><=></p>
215 ▾     <p> A <strong>function</strong> is a
        bit of code that you reuse over and over
        again. You could build a function to
        multiply two numbers easily.</p>
216 ▾     <p> Sometimes you won't know which
        numbers you want to multiple, so we might
        make afunction to multiple x and y. In
        this case, x and y are called
        <strong>variables</strong> (because they

```

(б)

Рис. 4.13: Вигляд згорнутої (а) та розгорнутої (б) строки коду.

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі були представлені скріншоти користувацького інтерфейсу, інструкція користувача та приклади використання багатокористувацької системи редагування програмного коду, що була реалізована в рамках цієї бакалаврської дипломної роботи.

Було розглянено сценарії реєстрації користувача, входу в систему, зміни паролю, оновлення профілю, виходу з системи, а також функціональні можливості безпосередньо редактора коду, а саме: нумерація строк, підсвічування синтаксису, автодоповнення коду, коментування та згортання строк коду, масштабування вікон для написання HTML, CSS та JavaScript коду, а також відображення у окремій області результатів виконання коду і збереження введених користувачем даних.

Отже, наведена у цьому розділі інформація демонструє, що у даному дипломному проєкті був реалізований весь заявлений функціонал, а система має сучасний та на інтуїтивному рівні зрозумілий користувацький інтерфейс, що має позитивний вплив на досвід та враження користувача.

ВИСНОВКИ

Основною метою даного бакалаврського дипломного проєкту є розробка багатокористувацької системи редагування програмного коду. У зв'язку зі стрімким ростом популярності веб-розробки стала актуальною поява зручних систем, що дозволяють розробникам писати та редагувати програмний код без прив'язки до конкретної локальної машини, що надасть користувачам більше можливостей з точки зору мобільності.

У першому розділі було наведено теоретичні відомості та описано область предметного середовища, а також проведено аналіз існуючих програмних рішень, що на сьогоднішній день представлені для використання діячам ІТ-сфери. На основі систем-аналогів було сформовано ряд функціональних можливостей системи, що розроблювалась.

Другий розділ являє собою огляд сучасних технологій, за допомогою яких було реалізовано дану систему. Для імплементації клієнтської частини додатку було обрано популярну в останні роки мову програмування JavaScript, а саме JavaScript-бібліотеки React та CodeMirror. Для серверної частини використовувався Firebase, що також демонструє потужні функціональні можливості та наразі набирає все більшої популярності не тільки серед веб-розробників, а й серед розробників мобільних додатків та інших програмних рішень.

Основний зміст третього розділу полягає у розкритті деталей реалізації багатокористувацької системи редагування програмного коду та описі взаємодії серверної частини програми з клієнтською.

Останній четвертий розділ містить у собі інструкцію користувача та наглядно демонструє користувацький інтерфейс, функціональні можливості реалізованої системи, а також користувацькі сценарії.

					<i>ІАЛЦ.467200.003 ПЗ</i>	Арк.
						55
Зм.	Арк.	№ докум.	Підпис	Дата		

Отже, таким чином багатокористувацька система редагування програмного коду, описана у першому розділі та створена у рамках цього дипломного проєкту, комбінує у собі переваги існуючих рішень та певною мірою вирішує недоліки систем-аналогів.

Система є безкоштовною, має сучасний дизайн, зручна у використанні та має основні можливості редакторів програмного коду.

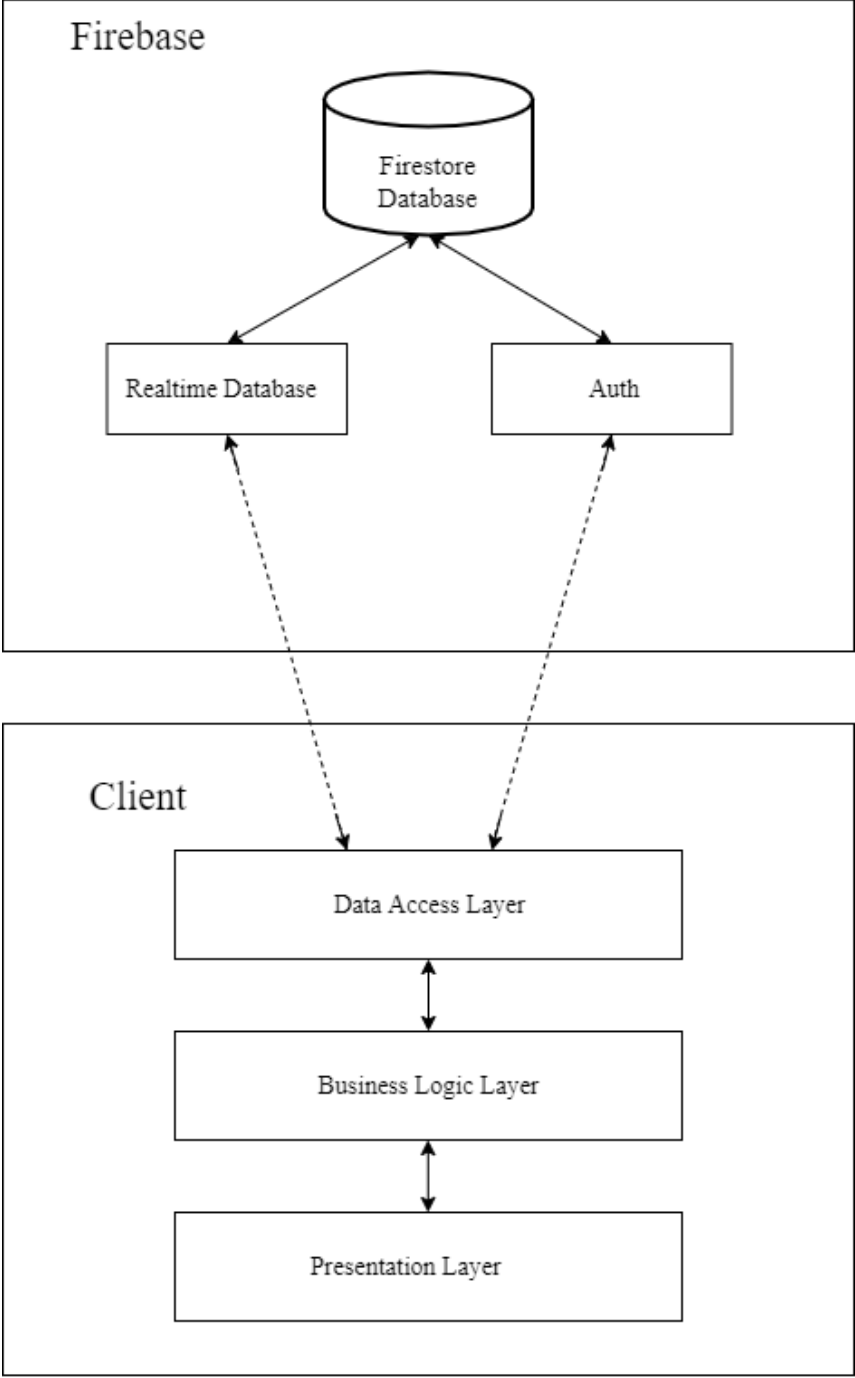
Зрештою, завдяки архітектурі додатку та обраним технологіям, стає можливим подальший розвиток та масштабування даної системи, що підвищить її актуальність та конкурентоспроможність.

					<i>ІАЛЦ.467200.003 ПЗ</i>	Арк.
						56
Зм.	Арк.	№ докум.	Підпис	Дата		

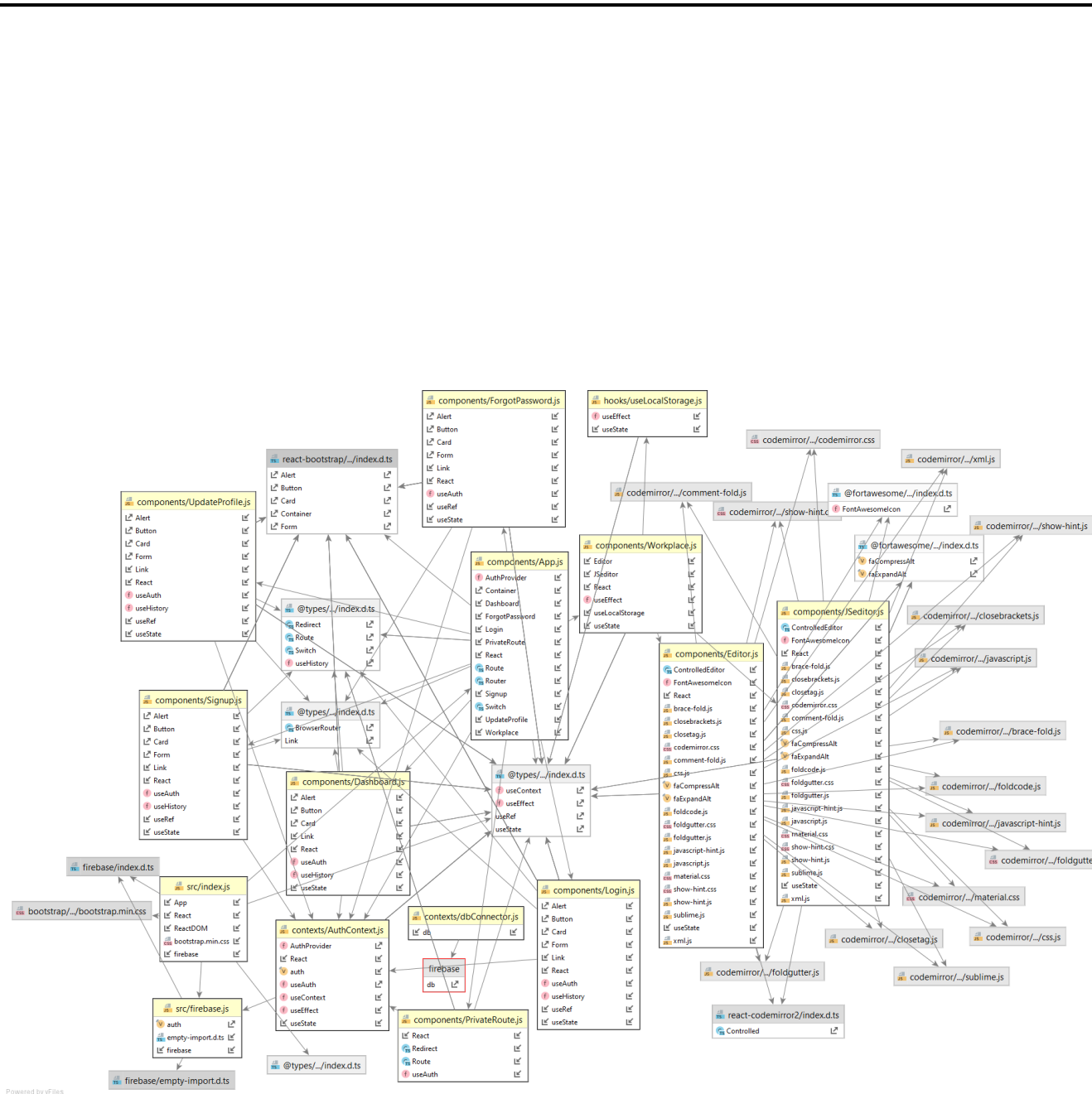
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. «Що таке веб-додаток?» [Електронний ресурс] – Режим доступу до ресурсу: <https://ukr.4meahc.com/what-exactly-is-web-application-50384>
2. «Модель-вид-контролер» [Електронний ресурс] – Режим доступу до ресурсу:
<https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80>
3. «ЯК: Що точно таке веб-додаток? – 2021» [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.go-travels.com/73084-what-is-a-web-application-3486637-6454152>
4. «Visual Studio Code» [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Visual_Studio_Code
5. reactjs.org [Електронний ресурс] – Режим доступу до ресурсу: <https://ru.reactjs.org/>
6. «Web технології та web дизайн» [Електронний ресурс] – Режим доступу до ресурсу: <https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti>
7. «JavaScript» [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/JavaScript>
8. codemirror.net [Електронний ресурс] – Режим доступу до ресурсу: <https://codemirror.net/>
9. «Firebase» [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Firebase>
10. «JavaScript» [Електронний ресурс] – Режим доступу до ресурсу: <https://www.wikizero.com/uk/JavaScript>
11. «Інструменти і середовища розробки мобільних додатків» [Електронний ресурс] – Режим доступу до ресурсу: <https://en.ppt-online.org/341525>

					ІАЛЦ.467200.003 ПЗ	Арк.
						57
Зм.	Арк.	№ докум.	Підпис	Дата		

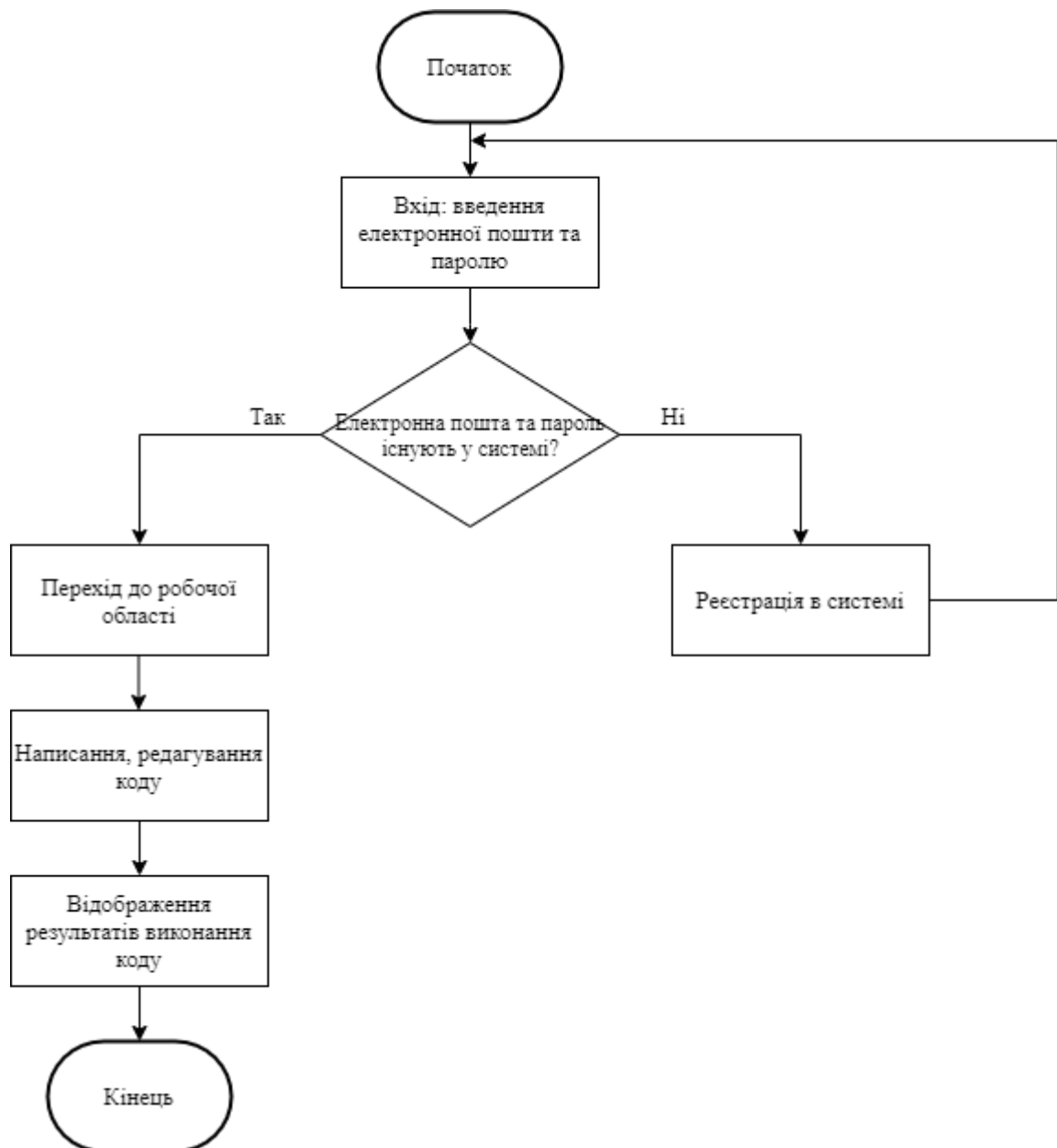


					ІАЛЦ.467200.004 Д1											
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна					Літ.		Аркуш	Аркушів			
Розробила		Парубець Л.М.										1		1		
Перевірів		Іваніщев Б.В.										НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-72				
Н.контр.		Сімоненко В.П.														
Затв.																
					Додаток 1											



Powered by yFiles

					ІАЛЦ.467200.005 Д2								
Зм.	Арк.	№ докум.	Підп.	Дата	Схема функціональна				Літ.	Аркуш	Аркушів		
Розробила	Парубець Л.М.											1	1
Перевірів	Іваніщев Б.В.												
Н.контр.	Сімоненко В.П.												
Затв.					Додаток 2				НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-72				



					<i>ІАЛЦ.467200.006 ДЗ</i>				
Зм.	Арк.	№ докум.	Підп.	Дата	Схема принципова Додаток 3				
Розробила	Парубець Л.М.								
Перевішив	Іваніщев Б.В.								
Н.контр.	Сімоненко В.П.								
Затв.					Літ. Аркуш Аркушів 1 1 НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-72				

index.js:

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './components/App';
import "bootstrap/dist/css/bootstrap.min.css";
import * as firebase from 'firebase';
```

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
)
```

firebase.js:

```
import firebase from "firebase/app"
import "firebase/auth"
```

```
const app = firebase.initializeApp({
  apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
  authDomain: process.env.REACT_APP_FIREBASE_AUTH_DOMAIN,
  databaseURL: process.env.REACT_APP_FIREBASE_DATABASE_URL,
  projectId: process.env.REACT_APP_FIREBASE_PROJECT_ID,
  storageBucket: process.env.REACT_APP_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: process.env.REACT_APP_FIREBASE_MESSAGING_SENDER_ID,
  appId: process.env.REACT_APP_FIREBASE_APP_ID
})
```

```
export const auth = app.auth()
export default app
```

					ІАЛЦ.467200.007 Д4						
Зм.	Арк.	№ докум.	Підп.	Дата							
Розробила	Парубець Л.М.				Лістинг програми			Літ.	Аркуш	Аркушів	
Перевірів	Іваніщев Б.В.									1	19
Н.контр.	Сімоненко В.П.				Додаток 4			НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-72			
Затв.											

AuthContext.js:

```
import React, { useContext, useState, useEffect } from 'react'
```

```
import { auth } from '../firebase'
```

```
const AuthContext = React.createContext()
```

```
export function useAuth() {
```

```
  return useContext(AuthContext)
```

```
}
```

```
export function AuthProvider({ children }) {
```

```
  const [currentUser, setCurrentUser] = useState()
```

```
  const [loading, setLoading] = useState(true)
```

```
  function signup(email, password) {
```

```
    return auth.createUserWithEmailAndPassword(email, password)
```

```
  }
```

```
  function login(email, password) {
```

```
    return auth.signInWithEmailAndPassword(email, password)
```

```
  }
```

```
  function logout() {
```

```
    return auth.signOut()
```

```
  }
```

```
  function resetPassword(email) {
```

```
    return auth.sendPasswordResetEmail(email)
```

```
  }
```

```
  function updateEmail(email) {
```

```
    return currentUser.updateEmail(email)
```

```
  }
```

```
  function updatePassword(password) {
```

					ІАЛЦ.467200.007 Д4	Арк.
						2
Зм.	Арк.	№ докум.	Підп.	Дата		

```

    return currentUser.updatePassword(password)
  }

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged(user => {
      setCurrentUser(user)
      setLoading(false)
    })

    return unsubscribe
  }, [])

  const value = {
    currentUser,
    login,
    signup,
    logout,
    resetPassword,
    updateEmail,
    updatePassword
  }

  return (
    <AuthContext.Provider value={value}>
      {!loading && children}
    </AuthContext.Provider>
  )
}

```

App.js:

```

import React from "react"
import Signup from "../Signup"
import { Container } from "react-bootstrap"
import { AuthProvider } from "../contexts/AuthContext";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom"

```

					ІАЛІЦ.467200.007 Д4	Арк.
						3
Зм.	Арк.	№ докум.	Підп.	Дата		

```

import Dashboard from "./Dashboard"
import Login from "./Login"
import PrivateRoute from "./PrivateRoute"
import ForgotPassword from "./ForgotPassword"
import UpdateProfile from "./UpdateProfile"
import Workplace from "./Workplace"

function App() {
  return (
    <Container
      className="d-flex align-items-center justify-content-center"
      style={{ minHeight: "100vh" }}
    >
      <div className="w-100" style={{ maxWidth: "400px" }}>
        <Router>
          <AuthProvider>
            <Switch>
              <PrivateRoute exact path="/" component={Dashboard} />
              <PrivateRoute path="/update-profile" component={UpdateProfile} />
              <Route path="/signup" component={Signup} />
              <Route path="/login" component={Login} />
              <Route path="/forgot-password" component={ForgotPassword} />
              <Route path="/workplace" component={Workplace} />
            </Switch>
          </AuthProvider>
        </Router>
      </div>
    </Container>
  )
}

export default App;

```

Dashboard.js:

					ІАЛІЦ.467200.007 Д4	Арк.
						4
Зм.	Арк.	№ докум.	Підп.	Дата		

```

import React, { useState } from "react"
import { Card, Button, Alert } from "react-bootstrap"
import { Link, useHistory } from "react-router-dom"
import { useAuth } from '../contexts/AuthContext'

export default function Dashboard() {
  const [error, setError] = useState("")
  const { currentUser, logout } = useAuth()
  const history = useHistory()

  async function handleLogout() {
    setError("")

    try {
      await logout()
      history.push('/login')
    } catch {
      setError('Failed to log out')
    }
  }

  return (
    <div style={{ height: "100vh", display: "flex", flexDirection: "column", alignItems: "center",
justifyContent: "center" }}>
      <Card style={{ maxWidth: "400px" }}>
        <Card.Body>
          <h2 className="text-center mb-4">Profile</h2>
          {error && <Alert variant="danger">{error}</Alert>}
          <strong>Email:</strong> {currentUser.email}
          <Link to="/update-profile" className="btn btn-primary w-100 mt-3">
            Update Profile
          </Link>
          <Link to="/workplace" className="btn btn-primary w-100 mt-3">
            Go to my Workplace
          </Link>
        </Card.Body>
      </Card>
    </div>
  )
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						5
Зм.	Арк.	№ докум.	Підп.	Дата		

```

        </Card.Body>
      </Card>
    <div
      // className="w-100 text-center mt-2"
      style={{ maxWidth: "400px" }}>
        <Button variant="link" onClick={handleLogout}>Log Out</Button>
      </div>
    </div>
  )
}

```

Editor.js:

```

import React, { useState } from 'react'
import 'codemirror/lib/codemirror.css'
import 'codemirror/theme/material.css'
import 'codemirror/mode/xml/xml'
import 'codemirror/mode/javascript/javascript'
import 'codemirror/mode/css/css'
import { Controlled as ControlledEditor } from 'react-codemirror2'
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import { faCompressAlt, faExpandAlt } from '@fortawesome/free-solid-svg-icons'

import "codemirror/lib/codemirror.css";
import "codemirror/theme/material.css";
import "codemirror/mode/javascript/javascript";
import "codemirror/addon/hint/show-hint";
import "codemirror/addon/hint/javascript-hint";
import "codemirror/addon/hint/show-hint.css";
import "codemirror/keymap/sublime";
import "codemirror/addon/edit/closebrackets";
import "codemirror/addon/edit/closetag";
import "codemirror/addon/fold/foldcode";
import "codemirror/addon/fold/foldgutter";
import "codemirror/addon/fold/brace-fold";
import "codemirror/addon/fold/comment-fold";

```

					ІАЛЦ.467200.007 Д4	Арк.
						6
Зм.	Арк.	№ докум.	Підп.	Дата		

```

import "codemirror/addon/fold/foldgutter.css";

export default function Editor(props) {
  const {
    language,
    displayName,
    value,
    onChange
  } = props
  const [open, setOpen] = useState(true)

  function handleChange(editor, data, value) {
    onChange(value)
  }

  return (
    <div style={{ display: "flex", flexDirection: "column", flexBasis: "0", flexGrow: "1",
    backgroundColor: "#3c3e44", padding: ".5rem" }} className={`editor-container ${open ? " :
'collapsed'}`}>
      <div style={{ display: "flex",
        justifyContent: "space-between",
        backgroundColor: "#1f2023",
        color: "#fff",
        padding: ".5rem .5rem .5rem 1rem",
        borderTopLeftRadius: ".5rem",
        borderTopRightRadius: ".5rem" }}
        className="editor-title">
        {displayName}
      <button
        type="button"
        className="expand-collapse-btn"
        style={{ marginLeft: ".5rem", background: "none", border: "none", color: "#fff", cursor:
"pointer" }}
        onClick={() => setOpen(prevOpen => !prevOpen)}

```

					ІАЛЦ.467200.007 Д4	Арк.
						7
Зм.	Арк.	№ докум.	Підп.	Дата		

```

        >
        <FontAwesomeIcon icon={open ? faCompressAlt : faExpandAlt} />
    </button>
</div>
<ControlledEditor
  onBeforeChange={handleChange}
  value={value}
  style={{ flexGrow: "1",
    borderBottomRightRadius: ".5rem",
    borderBottomLeftRadius: ".5rem",
    overflow: "hidden" }}
  className="code-mirror-wrapper"
  options={{
    lineWrapping: true,
    lint: true,
    mode: language,
    theme: 'material',
    lineNumbers: true,
    smartIndent: true,
    foldGutter: true,
    gutters: ["CodeMirror-linenumbers", "CodeMirror-foldgutter"],
    autoCloseTags: true,
    keyMap: "sublime",
    matchBrackets: true,
    autoCloseBrackets: true,
    extraKeys: {
      "Ctrl-Space": "autocomplete"
    }
  }}
/>
</div>
)
}

```

ForgotPassword.js:

					ІАЛІЦ.467200.007 Д4	Арк.
						8
Зм.	Арк.	№ докум.	Підп.	Дата		

```

import React, { useRef, useState } from "react"
import { Form, Button, Card, Alert } from "react-bootstrap"
import { useAuth } from "../contexts/AuthContext"
import { Link } from "react-router-dom"

export default function ForgotPassword() {
  const emailRef = useRef()
  const { resetPassword } = useAuth()
  const [error, setError] = useState("")
  const [message, setMessage] = useState("")
  const [loading, setLoading] = useState(false)

  async function handleSubmit(e) {
    e.preventDefault()

    try {
      setMessage("")
      setError("")
      setLoading(true)
      await resetPassword(emailRef.current.value)
      setMessage('Check your inbox for further instructions')
    } catch {
      setError('Failed to reset password')
    }

    setLoading(false)
  }

  return (
    <>
    <Card>
      <Card.Body>
        <h2 className="text-center mb-4">Reset Password</h2>
        {error && <Alert variant="danger">{error}</Alert>}
        {message && <Alert variant="success">{message}</Alert>}
      </Card.Body>
    </Card>
  )
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						9
Зм.	Арк.	№ докум.	Підп.	Дата		

```

    <Form onSubmit={handleSubmit}>
      <Form.Group id="email">
        <Form.Label>Email</Form.Label>
        <Form.Control type="email" ref={emailRef} required />
      </Form.Group>
      <Button disabled={loading} className="w-100" type="submit">
        Reset Password
      </Button>
    </Form>
    <div className="w-100 text-center mt-3">
      <Link to="/login">Login</Link>
    </div>
  </Card.Body>
</Card>
<div className="w-100 text-center mt-2">
  Need an account? <Link to="/signup">Sign Up</Link>
</div>
</>
)
}

```

Login.js:

```

import { Alert } from 'bootstrap'
import React, { useRef, useState } from "react"
import { Form, Button, Card, Alert } from "react-bootstrap"
import { useAuth } from "../contexts/AuthContext"
import { Link, useHistory } from "react-router-dom"

export default function Login() {
  const emailRef = useRef()
  const passwordRef = useRef()
  const { login } = useAuth()
  const [error, setError] = useState("")
  const [loading, setLoading] = useState(false)
  const history = useHistory()

```

					ІАЛЦ.467200.007 Д4	Арк.
						10
Зм.	Арк.	№ докум.	Підп.	Дата		

```

async function handleSubmit(e) {
  e.preventDefault()

  try {
    setError("")
    setLoading(true)
    await login(emailRef.current.value, passwordRef.current.value)
    history.push("/")
  } catch {
    setError('Failed to log in')
  }

  setLoading(false)
}

return (
  <>
    <Card>
      <Card.Body>
        <h2 className="text-center mb-4">Log In</h2>
        {error && <Alert variant="danger">{error}</Alert>}
        <Form onSubmit={handleSubmit}>
          <Form.Group id="email">
            <Form.Label>Email</Form.Label>
            <Form.Control type="email" ref={emailRef} required />
          </Form.Group>
          <Form.Group id="password">
            <Form.Label>Password</Form.Label>
            <Form.Control type="password" ref={passwordRef} required />
          </Form.Group>
          <Button disabled={loading} className="w-100" type="submit">
            Log In
          </Button>
        </Form>
      </Card>
    </>
  )
}

```

					ІАЛІЦ.467200.007 Д4	Арк.
						11
Зм.	Арк.	№ докум.	Підп.	Дата		

```

        <div className="w-100 text-center mt-3">
            <Link to="/forgot-password">Forgot Password?</Link>
        </div>
    </Card.Body>
</Card>
<div className="w-100 text-center mt-2">
    Need an account? <Link to="/signup">Sign Up</Link>
</div>
</>
)
}

```

PrivateRoute.js:

```

import React from 'react'
import { Route, Redirect } from 'react-router-dom'
import { useAuth } from '../contexts/AuthContext'

export default function PrivateRoute({ component: Component, ...rest }) {
    const { currentUser } = useAuth()

    return (
        <Route
            {...rest}
            render={props => {
                return currentUser ? <Component {...props} /> : <Redirect to="/login" />
            }}
        ></Route>
    )
}

```

SignUp.js:

```

import React, { useRef, useState } from 'react'
import { Form, Button, Card, Alert } from 'react-bootstrap'
import { useAuth } from '../contexts/AuthContext'
import { Link, useHistory } from "react-router-dom"
// import * as firebase from 'firebase';

```

Зм.	Арк.	№ докум.	Підп.	Дата

ІАЛЦ.467200.007 Д4

Арк.

12

```

export default function Signup() {
  const emailRef = useRef()
  const passwordRef = useRef()
  const passwordConfirmRef = useRef()
  const { signup } = useAuth()
  const [error, setError] = useState("")
  const [loading, setLoading] = useState(false)
  const history = useHistory()
  // const usersRef = firebase.database().ref("users/");

  async function handleSubmit(e) {
    e.preventDefault()

    if (passwordRef.current.value !== passwordConfirmRef.current.value) {
      return setError('Password do not match')
    }

    try {
      setError("")
      setLoading(true)
      await signup(emailRef.current.value, passwordRef.current.value)

    } catch {
      setError('Failed to create an account')
    }

    setLoading(false)
  }

  return (
    <>
      <Card>
        <Card.Body>

```

					ІАЛЦ.467200.007 Д4	Арк.
						13
Зм.	Арк.	№ докум.	Підп.	Дата		

```

<h2 className="text-center mb-4">Sign Up</h2>
{error && <Alert variant="danger">{error}</Alert>}
<Form onSubmit={handleSubmit}>
  <Form.Group id="email">
    <Form.Label>Email</Form.Label>
    <Form.Control type="email" ref={emailRef} required />
  </Form.Group>
  <Form.Group id="password">
    <Form.Label>Password</Form.Label>
    <Form.Control type="password" ref={passwordRef} required />
  </Form.Group>
  <Form.Group id="password-confirm">
    <Form.Label>Password Confirmation</Form.Label>
    <Form.Control type="password" ref={passwordConfirmRef} required />
  </Form.Group>
  <Button disabled={loading} className="w-100" type="submit">
    Sign Up
  </Button>
</Form>
</Card.Body>
</Card>
<div className="w-100 text-center mt-2">
  Already have an account? <Link to="/login">Log In</Link>
</div>
</>
)
}

```

UpdateProfile.js:

```

import React, { useRef, useState } from 'react'
import { Form, Button, Card, Alert } from 'react-bootstrap'
import { useAuth } from '../contexts/AuthContext'
import { Link, useHistory } from "react-router-dom"

export default function UpdateProfile() {

```

					ІАЛЦ.467200.007 Д4	Арк.
						14
Зм.	Арк.	№ докум.	Підп.	Дата		

```

const emailRef = useRef()
const passwordRef = useRef()
const passwordConfirmRef = useRef()
const { currentUser, updateEmail, updatePassword } = useAuth()
const [error, setError] = useState("")
const [loading, setLoading] = useState(false)
const history = useHistory()

function handleSubmit(e) {
  e.preventDefault()

  if (passwordRef.current.value !== passwordConfirmRef.current.value) {
    return setError('Password do not match')
  }

  const promises = []
  setLoading(true)
  setError("")

  if (emailRef.current.value !== currentUser.email) {
    promises.push(updateEmail(emailRef.current.value))
  }
  if (passwordRef.current.value) {
    promises.push(updatePassword(passwordRef.current.value))
  }

  Promise.all(promises).then(() => {
    history.push('/')
  }).catch(() => {
    setError('Failed to update account')
  }).finally(() => {
    setLoading(false)
  })
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						15
Зм.	Арк.	№ докум.	Підп.	Дата		

```

}

return (
  <>
    <Card>
      <Card.Body>
        <h2 className="text-center mb-4">Update Profile</h2>
        {error && <Alert variant="danger">{error}</Alert>}
        <Form onSubmit={handleSubmit}>
          <Form.Group id="email">
            <Form.Label>Email</Form.Label>
            <Form.Control type="email" ref={emailRef} required
              defaultValue={currentUser.email} />
          </Form.Group>
          <Form.Group id="password">
            <Form.Label>Password</Form.Label>
            <Form.Control type="password" ref={passwordRef}
              placeholder="Leave blank to keep the same" />
          </Form.Group>
          <Form.Group id="password-confirm">
            <Form.Label>Password Confirmation</Form.Label>
            <Form.Control type="password" ref={passwordConfirmRef}
              placeholder="Leave blank to keep the same" />
          </Form.Group>
          <Button disabled={loading} className="w-100" type="submit">
            Update
          </Button>
        </Form>
      </Card.Body>
    </Card>
    <div className="w-100 text-center mt-2">
      <Link to="/">Cancel</Link>
    </div>
  </>
)

```

Зм.	Арк.	№ докум.	Підп.	Дата

ІАЛІЦ.467200.007 Д4

Арк.

16

```
}
```

Workplace.js:

```
import React, { useState, useEffect } from "react"
```

```
import JSeditor from './JSeditor'
```

```
import Editor from './Editor'
```

```
import useLocalStorage from '../hooks/useLocalStorage'
```

```
function Workplace() {
```

```
  const [html, setHtml] = useLocalStorage('html', "")
```

```
  const [css, setCss] = useLocalStorage('css', "")
```

```
  const [js, setJs] = useLocalStorage('js', "")
```

```
  const [srcDoc, setSrcDoc] = useState("")
```

```
  useEffect(() => {
```

```
    const timeout = setTimeout(() => {
```

```
      setSrcDoc(`
```

```
        <html>
```

```
          <body>${html}</body>
```

```
          <style>${css}</style>
```

```
          <script>${js}</script>
```

```
        </html>
```

```
      `)
```

```
    }, 250)
```

```
    return () => clearTimeout(timeout)
```

```
  }, [html, css, js])
```

```
  return (
```

```
    <>
```

```
    <div style={{ display: "flex", height: "50vh", backgroundColor: "#3c3e44" }}>
```

```
    className="pane top-pane">
```

```
      <Editor
```

```
        language="xml"
```

```
        displayName="HTML"
```

					ІАЛЦ.467200.007 Д4	Арк.
						17
Зм.	Арк.	№ докум.	Підп.	Дата		

```

        value={html}
        onChange={setHtml}
    />
    <Editor
        language="css"
        displayName="CSS"
        value={css}
        onChange={setCss}
    />
    <Editor
        language="javascript"
        displayName="JS"
        value={js}
        onChange={setJs}
    />
</div>
<div className="pane" style={{ display: "flex", height: "50vh" }}>
    <iframe
        srcDoc={srcDoc}
        title="output"
        sandbox="allow-scripts"
        frameborder="0"
        width="100%"
        height="100%"
    />
</div>
</>
)
}

export default Workplace;

dbConnector.js:
import { db } from "lib/firebase";

```

					ІАЛІЦ.467200.007 Д4	Арк.
						18
Зм.	Арк.	№ докум.	Підп.	Дата		

```

const getUserFiles = async (userId) => {
  const doc = await db.collection("users").doc(userId).get();

  if (doc.exists) {
    console.log("User found in database");
    const snapshot = await db
      .collection("users")
      .doc(doc.id)
      .collection("files")
      .get();

    let userFiles = [];
    snapshot.forEach((file) => {
      let { name, content } = file.data();
      userFiles.push({ id: file.id, name: name, content: content });
    });
    return userFiles;
  } else {
    console.log("User not found in database, creating new entry...");
    db.collection("users").doc(userId).set({});
    return [];
  }
};

```

					ІАЛЦ.467200.007 Д4	Арк.
						19
Зм.	Арк.	№ докум.	Підп.	Дата		