

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Помічник для ведучого гри "Мафія"»

Виконав студент групи ПЗк-20-1

ГАЛАГАН Вадим Васильович

Керівник роботи:

ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:

ЄФРЕМОВ Юрій Миколайович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГАЛАГАН Вадим Васильович**

Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**

Тема роботи: **«Помічник для ведучого гри "Мафія"»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2023 р., №70с**

Вихідні дані для роботи: Об'єктом дослідження є розробка мобільного додатку спрямована на поліпшення процесу взаємодії ведучого та гравців у грі «Мафія».

Предметом дослідження є використання сучасних функціональних можливостей для виконання поставленої мети.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.,	16.02.2023	16.02.2023
2	Чижмотря О.Г.,	16.02.2023	16.02.2023
3	Чижмотря О.Г.,	16.02.2023	16.02.2023

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	16 лютого 2023 – 18 лютого 2023	виконано
2	Аналіз аналогів програмного продукту	18 лютого 2023 – 25 лютого 2023	виконано
3	Визначення вимог до додатку	25 лютого 2023 – 8 березня 2023	виконано
4	Вибір інструментальних засобів	8 березня 2023 – 10 березня 2023	виконано
5	Визначення варіантів використання	10 березня 2023 – 19 березня 2023	виконано
6	Проектування та реалізація алгоритмів системи	19 березня 2023 – 29 березня 2023	виконано
7	Написання програмного коду	29 березня 2023 – 2 травня 2023	виконано
8	Тестування роботи програмного продукту	2 травня 2023 – 11 травня 2023	виконано
9	Оформлення пояснювальної записки	11 травня 2023 – 30 травня 2023	виконано
10	Попередній захист	5 травня 2023	виконано

РЕФЕРАТ

Випускна робота бакалавра складається з мобільного додатку помічник для ведучого гри «Мафія» та пояснювальної записки. Пояснювальна записка до випускної роботи містить 76 сторінок, 27 ілюстрацій, 3 таблиці, 6 додатків та 30 використаних джерел.

Метою роботи є розробка та реалізація мобільного додатку «Помічник для ведучого гри "Мафія"». В роботі було визначено основні завдання на розробку додатку, проаналізовано аналогічні програмні продукти, визначено основний функціонал та вимоги до системи. Також було обрано технології для розробки додатку, спроектовано та реалізовано алгоритми роботи системи. Програмний продукт був повністю створений та протестований.

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ДОДАТОК, ГРА «МАФІЯ», REACT NATIVE

					ІПЗ.КР.Б – 121 – 23 - ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		В.В. Галаган			Пояснювальна записка Помічник для ведучого гри «Мафія»			Літ.	Арк.	Аркушів	
Керівник.		О.Г. Чижмотря								4	76
Реценз.		Ю.М. Єфремов						<i>Житомирська політехніка, група ІПЗк-20-1</i>			
Зав. Каф..		А.В. Морозов									

ABSTRACT

The bachelor's thesis consists of a mobile application called "Assistant for the Game Master in the 'Mafia' Game" and an explanatory memorandum. The explanatory memorandum for the bachelor's thesis includes 76 pages, 27 illustrations, 3 tables, 6 appendices, and 30 cited sources.

The aim of this work is to develop and implement a mobile application, "Assistant for the Game Master in the 'Mafia' Game." The main tasks for developing the application were defined, similar software products were analyzed, and the core functionality and system requirements were determined. Technologies for application development were also chosen, and the system algorithms were designed and implemented. The software product was fully created and tested.

KEYWORDS: Mobile Application, "Mafia" Game, React Native.

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП.....	7
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	9
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	10
1.1. Технічне завдання на розробку системи	10
1.2. Аналіз аналогів програмного продукту	12
1.3. Інструментальні засоби реалізації програмного продукту	17
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ДОДАТКУ	21
2.1. Визначення варіантів використання системи	21
2.2. Алгоритми роботи системи.....	24
2.3. Структура компонентів системи	27
2.4. Реалізація додатку.....	28
РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ	39
3.1. Архітектура та розгортання додатку.....	39
3.2. Інтерфейс та порядок роботи з системою	40
3.3. Тестування роботи додатку.....	48
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	55
ДОДАТКИ.....	57

ВСТУП

Актуальність теми: Гра «Мафія» є популярною інтелектуальною грою, яка здатна забезпечити цікавість та емоційне напруження серед учасників. Проте, проведення гри вимагає наявності ведучого, який виконує роль організатора, контролює процес гри та забезпечує її правильний хід. З метою полегшення цього процесу та створення зручного інструменту для ведучого гри «Мафія», мобільний додаток, розроблений на базі сучасних технологій, стає актуальним завданням.

Мета даної роботи: розробка та реалізація мобільного додатку «Помічник для ведучого гри "Мафія"». Цей додаток призначений для спрощення та автоматизації процесу проведення гри «Мафія», надаючи ведучому необхідні інструменти та функціонал для контролю та керування грою. Головною метою розробки є створення зручного та ефективного інструменту для ведучих гри «Мафія», який забезпечить правильний хід гри та підвищить її якість та насолоду для учасників.

Об'єкт дослідження: є процес проведення гри «Мафія» та взаємодія ведучого з учасниками. Розробка мобільного додатку спрямована на поліпшення цього процесу та забезпечення зручних інструментів для ведучого.

Предмет дослідження: є функціональні можливості мобільного додатку «Помічник ведучого гри "Мафія"». Додаток буде включати такі функції, як вибір кількості учасників гри, розподіл ролей учасникам гри, контроль ходу гри та інші інструменти, необхідні для успішного проведення гри «Мафія».

Методи дослідження: у процесі розробки мобільного додатку будуть використані наступні методи дослідження: аналіз вимог до функціоналу додатку, проектування архітектури та інтерфейсу додатку, розробка та тестування програмного забезпечення. Крім того, для забезпечення коректності та ефективності додатку будуть використані методи тестування.

Загальний результат дослідження буде представлений у вигляді функціонального мобільного додатку «Помічник ведучого гри "Мафія"», який

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

забезпечить полегшення та автоматизацію процесу проведення гри «Мафія» та покращить якість та комфортність гри для її учасників.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IOS – iPhone Operating System (Операційна система Айфон).

JS – JavaScript (Мова програмування).

RN – React Native (Фреймворк мови програмування JS).

JSX - JavaScript XML (Розширення синтаксису JavaScript).

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Технічне завдання на розробку системи

Аналіз проблематики включає визначення актуальності теми, пояснення необхідності розробки мобільного додатку «Помічник ведучого гри "Мафія"» та виявлення основних проблем з якими стикаються ведучі та учасники гри «Мафія» [5].

Далі розглядаються бізнес-вимоги, вимоги користувачів, функціональні та нефункціональні вимоги, системні вимоги, бізнес-правила та антивимоги. Це необхідно для з'ясування, які конкретні функції має виконувати додаток, які вимоги мають користувачі та бізнес, а також які обмеження і вимоги повинні бути враховані при розробці системи.

Розділ також включає аналіз проекту та тестування системи. Проводиться оцінка можливостей розробки додатку, планування та розробка тестових сценаріїв для перевірки функціональності та стабільності додатку, а також тестування на різних пристроях та платформах.

Цей розділ є основою для подальшої реалізації та розробки мобільного додатку «Помічник ведучого гри "Мафія"». Він допомагає чітко визначити вимоги та цілі проекту, а також визначити необхідні методи та засоби для успішної реалізації задачі.

Бізнес-вимоги:

1. Розробка мобільного додатку «Помічник ведучого гри "Мафія"» для полегшення та автоматизації процесу проведення гри «Мафія».
2. Забезпечення зручного та ефективного інструменту для ведучих гри «Мафія», який дозволить контролювати хід гри та забезпечити її правильність.
3. Покращення якості та комфортності гри «Мафія» для учасників шляхом надання нових функцій та можливостей.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Вимоги користувачів:

1. Інтуїтивний та зрозумілий інтерфейс додатку, щоб ведучі гри «Мафія» могли легко орієнтуватися та використовувати всі його функції.
2. Можливість швидкого та точного розподілу ролей між учасниками гри.
3. Функціонал для керування грою, включаючи контроль ходу гри, призначення завдань та нагород, інтерактивну комунікацію з учасниками та інше.

Функціональні вимоги:

1. Розподіл ролей між учасниками гри «Мафія».
2. Ведення журналу гри з фіксацією ходу гри, результатів голосувань та розкриття ролей.
3. Забезпечення можливості створення та налаштування різних сценаріїв гри «Мафія».

Нефункціональні вимоги:

1. Висока продуктивність та швидкодія додатку, щоб забезпечити плавний та безперебійний хід гри.
2. Забезпечення безпеки даних та конфіденційності користувачів.
3. Кросплатформенність додатку, щоб забезпечити доступність для різних мобільних платформ [16].

Системні вимоги:

1. Мобільна платформа (IOS або Android) для розробки та запуску додатку [17].
2. Мова програмування та технології: JS, фреймворк React Native.

Бізнес-правила:

1. Дотримання правил та механіки гри «Мафія».
2. Підтримка можливості редагування та налаштування правил гри залежно від вимог користувачів.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

Антивимоги:

1. Відсутність зайвих складнощів та надмірної складності в інтерфейсі та функціоналі додатку.
2. Мінімальна кількість помилок та ситуацій непередбаченості під час використання додатку.

1.2. Аналіз аналогів програмного продукту

Під час проведення аналізу аналогів були виявлені наступні подібні додатки.

«Mafia Mystery» є мобільним додатком (рис. 1.1), який дозволяє користувачам грати у гру «Мафія» разом з друзями або іншими гравцями онлайн.

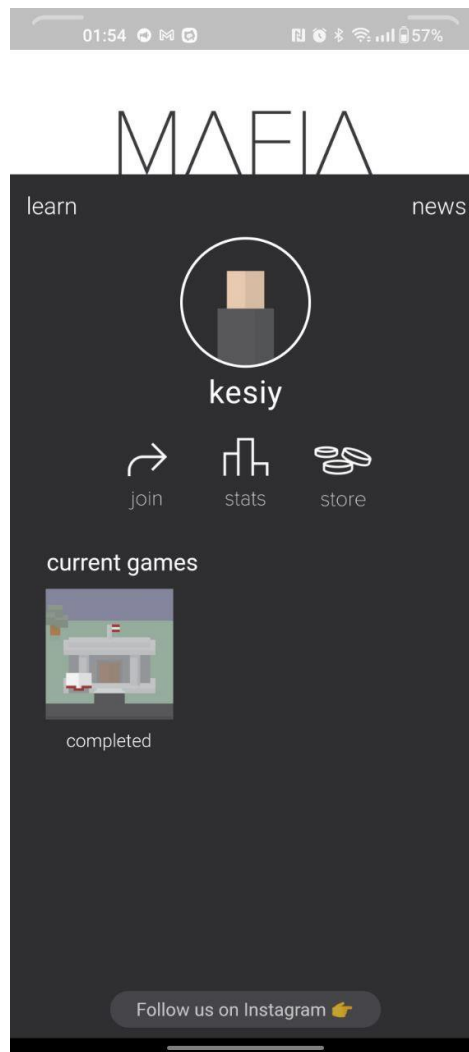


Рисунок 1.1 - Головне меню гри Mafia Mystery

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток надає функціональність для розподілу ролей, обміну повідомленнями (рис. 1.2), голосування та ведення гри.

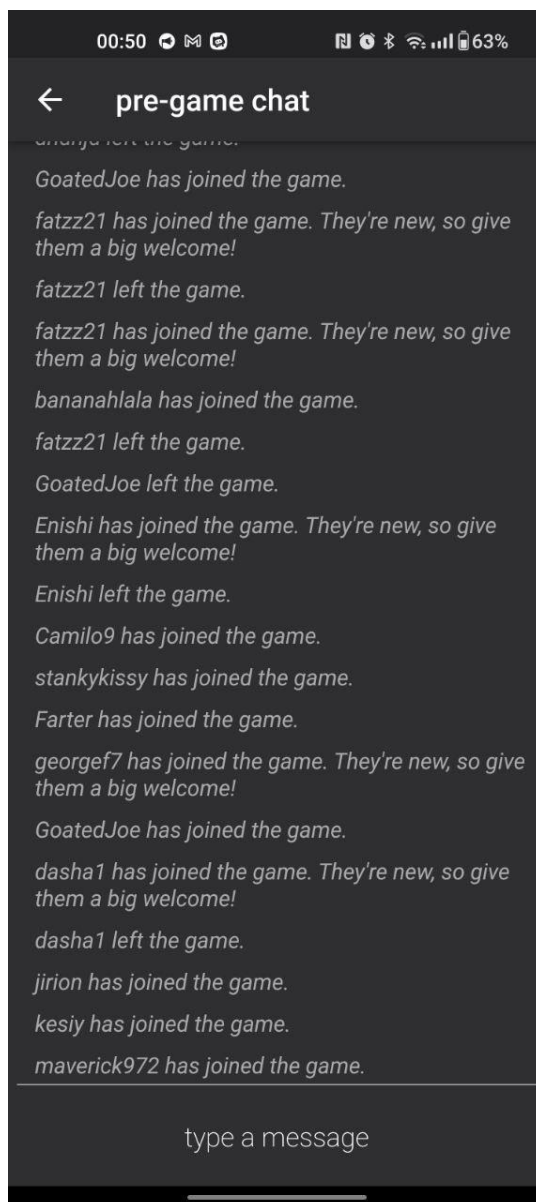


Рисунок 1.2 - Чат гри Mafia Mystery

Ви можете вибрати роль мафії, мирного мешканця або іншу роль і спробувати розкрити та знешкодити мафію (рис. 1.3), або ж допомогти мафії здобути перемогу. «Mafia Mystery» забезпечує затяжний та захоплюючий геймплей, де ви повинні використовувати стратегію, логіку та комунікацію, щоб досягти своїх цілей.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

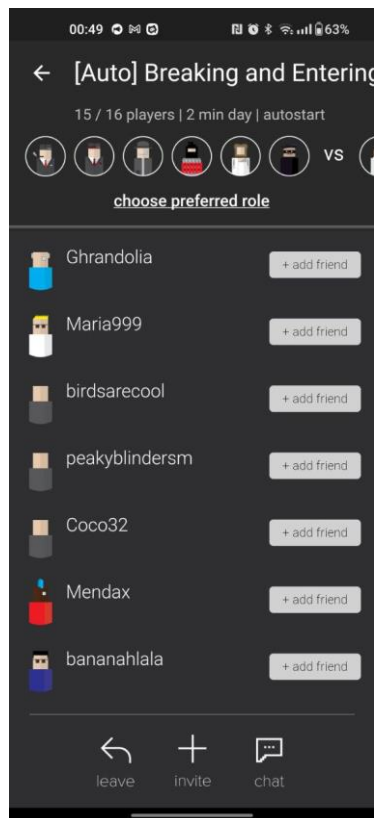


Рисунок 1.3 - Меню розподілу ролей Mafia Mystery

«Werewolf Online» є іншим захоплюючим додатком для гри у Мафію, з акцентом на рольову гру (рис. 1.4).



Рисунок 1.4 - Головне меню гри Werewolf Online

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

Ви можете приєднатися до онлайн-сесій з іншими гравцями і вибрати різні ролі, такі як: вовк-людодід, сирота або ворон, кожна з яких має свої унікальні здібності (рис. 1.5).

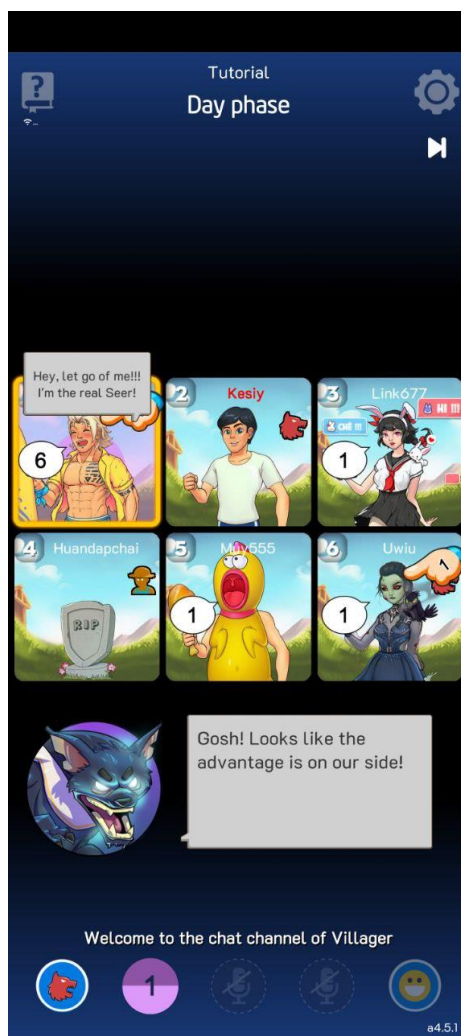


Рисунок 1.5 - Процес гри Werewolf Online

Додаток надає можливість взаємодії з іншими гравцями, обговорення, голосування та ведення гри. «Werewolf Online» пропонує захоплюючу гру, де ви повинні розкривати та виявляти ворожі елементи серед учасників і забезпечувати перемогу своїй команді (рис. 1.6).

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

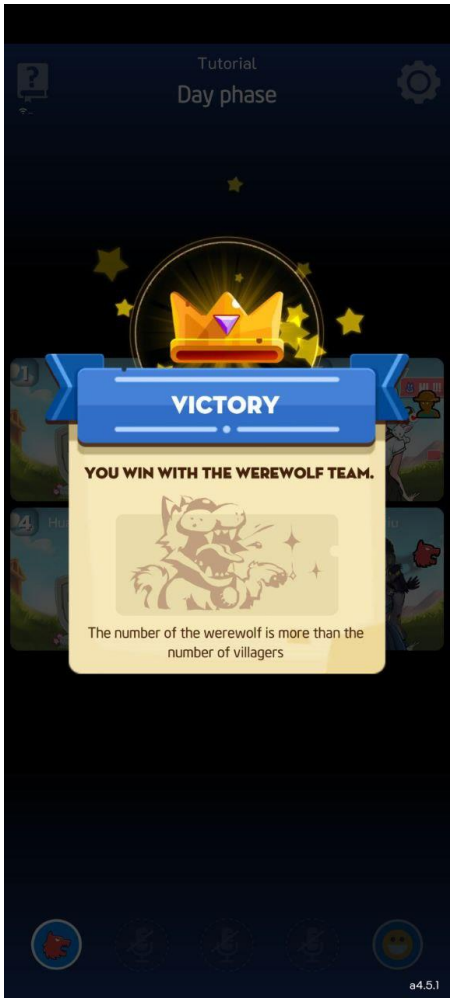


Рисунок 1.6 - Экран победы игры Werewolf Online

«Town of Salem» є ще одним додатком, що дозволяє гравцям насолоджуватися грою у Мафію, але з додатковими режимами та елементами стратегії (рис. 1.7).



Рис 1.7 Меню партії у грі Werewolf Online

Ви можете обирати різні ролі, такі як: детектив, вбивця, медична сестра та багато інших. Гра надає можливість спілкування з іншими гравцями, голосування, обговорення та ведення гри. «Town of Salem» пропонує вам виявляти зрадників серед містечка, здійснювати стратегічні ходи та забезпечувати перемогу своїй команді (рис. 1.8).



Рисунок 1.8 - Процес гри Werewolf Online

1.3. Інструментальні засоби реалізації програмного продукту

Основними мовами програмування для розробки мобільних додатків є Java для операційної системи Android та Swift для операційної системи IOS [16]. Java та Swift є популярними мовами програмування для розробки мобільних додатків. Вони надають доступ до нативних можливостей платформ [10], проте використання фреймворків може мати переваги з таких причин:

1. Кросплатформенність: Фреймворки дозволяють розробляти кросплатформенні додатки, які працюють на різних платформах, таких як Android та IOS. Це зменшує зусилля та час, необхідний для створення окремих версій додатків на кожную платформу.

2. Швидкість розробки: Фреймворки надають готові компоненти та інструменти, що прискорюють процес розробки [29]. Це дозволяє розробникам швидко створювати функціональний інтерфейс користувача та впроваджувати функціональність [23].

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

3. Навички розробника: Використання фреймворків, таких як RN, дозволяє використовувати наявні навички розробника веб-додатків або інших фреймворків. Це зменшує час необхідний для розробки.

Далі розглянемо деякі з найпопулярніших фреймворків для розробки мобільних додатків:

1. NativeScript: Це фреймворк для розробки мобільних додатків, який дозволяє використовувати нативний код для створення додатків для різних платформ [9]. NativeScript дозволяє розробникам використовувати мови програмування, такі як: JS, TypeScript або Angular, для створення кросплатформенних додатків [13].

2. Flutter: Розроблений компанією Google, Flutter є фреймворком, що дозволяє розробляти кросплатформенні мобільні додатки [7]. Він використовує одну спільну базу коду, яка може бути скомпільована для різних платформ, таких як Android та IOS [22]. Flutter використовує мову програмування Dart та має вбудовану бібліотеку віджетів для створення красивого інтерфейсу користувача.

3. Xamarin: Xamarin є фреймворком для розробки мобільних додатків, який використовує мову програмування C# та базується на платформі .NET [25]. За допомогою Xamarin можна розробляти кросплатформенні додатки, що працюють на платформах Android, IOS та Windows [30]. Він дозволяє використовувати спільний код та має доступ до нативних API для більшої функціональності.

Проте для реалізації програмного продукту «Помічник ведучого гри "Мафія"» було використано фреймворк RN і ось чому.

RN є відкритим фреймворком для розробки мобільних додатків, який дозволяє створювати додатки для платформ Android і IOS з використанням однієї спільної бази коду [15]. Використання RN у розробці програмного продукту дозволило зберегти час та ресурси, адже код може бути повторно використаний для обох платформ.

Однією з головних переваг RN є швидкість розробки та ефективне управління компонентами і станом додатку [18]. Використання JSX

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

(розширеного синтаксису JS) [1] дозволяє зручно описувати інтерфейс користувача та зв'язки між компонентами [4].

Додаток написаний на RN має переваги кросплатформенності, що означає, що один код може бути використаний для створення додатків для обох платформ Android і IOS. Це дозволяє зекономити час та зусилля на розробку та підтримку двох окремих додатків [3].

Завдяки великій спільноті розробників можна легко знайти додаткові компоненти, бібліотеки та плагіни для розширення функціональності додатку [28]. Це дозволяє швидко впроваджувати новий функціонал та вдосконалювати продукт з часом.

Використання RN разом з іншими інструментами та технологіями дозволяє реалізувати функціональність та інтерфейс додатку «Помічник ведучого гри "Мафія"» з високою продуктивністю та забезпечити його кросплатформенну сумісність.

Висновки до першого розділу

У цьому розділі було проведено аналіз проблематики та методів необхідних для розробки програмного продукту «Помічник ведучого гри "Мафія"».

У першому підрозділі «Технічне завдання на розробку системи» були визначені бізнес-вимоги, вимоги користувачів, функціональні вимоги, нефункціональні вимоги, системні вимоги, бізнес-правила, антивимоги, аналіз проекту та тестування системи. Це дало чітку картину того, які вимоги та задачі повинні бути вирішені при розробці програмного продукту.

У другому підрозділі «Аналіз аналогів програмного продукту» було проведено дослідження та огляд існуючих додатків, які виконують схожі функції. Цей аналіз дозволив виявити переваги та недоліки існуючих рішень та визначити потенційні можливості для покращення та унікальності майбутнього програмного продукту.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

У третьому підрозділі «Інструментальні засоби реалізації програмного продукту» були розглянуті різні технології та фреймворки для розробки мобільних додатків. Було обговорено переваги та недоліки використання мов програмування Java та Swift, а також були розглянуті інші можливі варіанти, але у кінцевому рахунку вибір був зроблений на користь фреймворку RN. Цей вибір був обґрунтований його кросплатформенністю, швидкістю розробки та наявністю великої спільноти розробників.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ДОДАТКУ

2.1. Визначення варіантів використання системи

Даний підрозділ розглядає основні функціональні можливості системи «Помічник ведучого гри "Мафія"» і описує різні сценарії використання, які задовольняють потреби користувачів. Детальний аналіз варіантів використання дозволяє краще розуміти основні функції системи та взаємодію з акторами [6]. Він включає в себе створення нової гри, додавання гравців до гри, розподіл ролей гравцям, проведення гри та керування налаштуваннями додатку. Аналіз варіантів використання є важливим кроком у розробці системи, що дозволяє визначити функціональні вимоги та гарантує відповідність потребам користувачів.

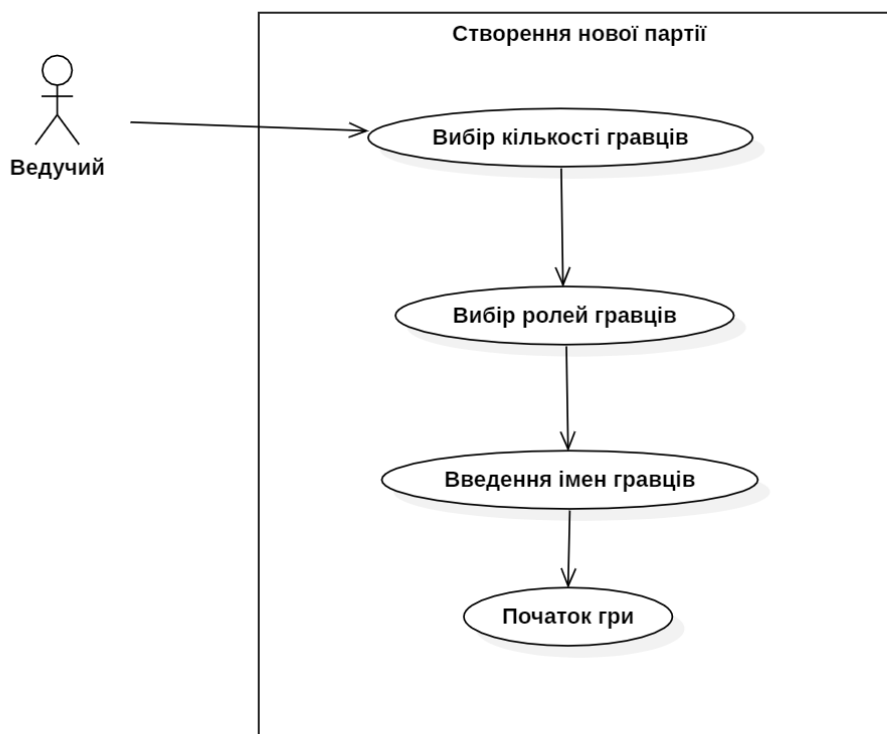


Рисунок 2.1 - Діаграма варіанту використання: Створення нової партії

Ведучий може створити нову партію Мафії. Він вибирає кількість гравців і ролі для гри. Система дозволяє користувачу встановити додаткові налаштування, такі як таймер або назва гри. Після завершення налаштувань гра готова до початку (рис. 2.1).

Ведучий керує партією. Він бачить поточний етап гри, запускає таймер для кожного етапу та контролює процес гри, такий як голосування або виконання ролей. Ведучий забезпечує правильний хід гри та виконання правил.

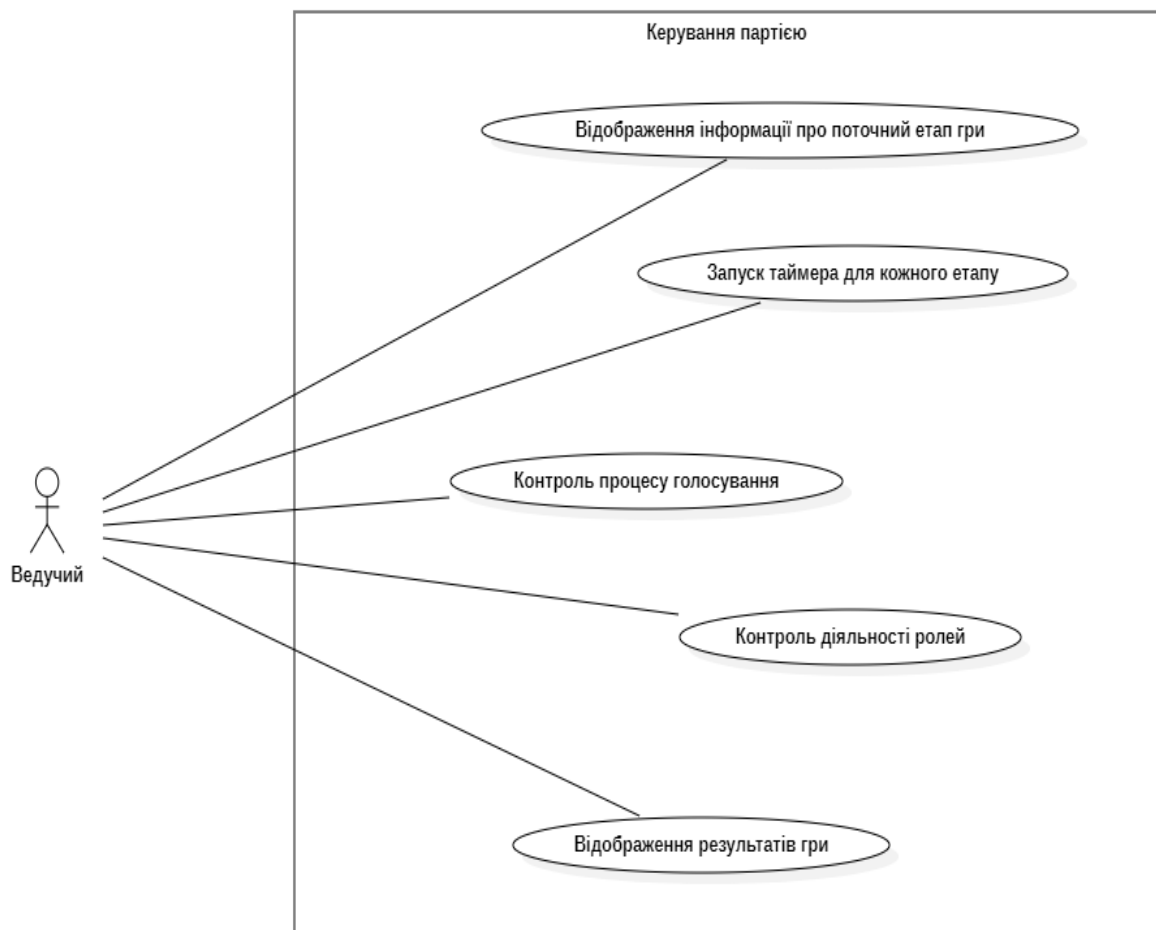


Рисунок 2.2 - Діаграма варіанту використання: Керування партією

Користувач може змінювати налаштування додатку, такі як звук, мова або вигляд. Він може зберегти змінені налаштування для подальшого використання.



Рисунок 2.3 - Діаграма варіанту використання: Керування іншими
можливостями додатку

Таблиця 2.1

Опис варіанту використання «Створення нової партії»

Назва варіанту використання	Створення нової партії
Опис варіанту використання	Створює нову партію
Актор	Ведучий
Передумови	Відсутні
Основні дії	Ввести кількість гравців
	Ввести імена гравців
	Ввести ролі гравців
Результат	Створена нова партія

Таблиця 2.2

Опис варіанту використання «Керування партією»

Назва варіанту використання	Керування партією
Опис варіанту використання	Процес керування партією
Актор	Ведучий
Передумови	Створена партія
Основні дії	Відображення інформації про поточний етап гри
	Запуск таймера для кожного етапу
	Контроль процесу голосування
	Контроль діяльності ролей

	Відображення результатів гри
Результат	Партія проведена

Таблиця 2.3

Опис варіанту використання «Керування іншими можливостями додатку»

Назва варіанту використання	Керування іншими можливостями додатку
Опис варіанту використання	Опис інших можливостей додатку
Актор	Користувач
Передумови	Відсутні
Основні дії	Перегляд екрану правил
	Перегляд екрану налаштувань
	Зміна мови додатку
	Увімкнення або вимкнення 0-го дня
	Зміна обмеження на мінімальну кількість гравців
	Зміна обмеження на максимальну кількість гравців
Результат	Партія проведена

2.2. Алгоритми роботи системи

У даному підрозділі описується логіка та послідовність операцій, які виконує система «Помічник ведучого гри "Мафія"» для забезпечення своєї функціональності. Алгоритми визначають, як система взаємодіє з користувачами та виконує різні операції для успішного проведення гри «Мафія».

Основні алгоритми роботи системи включають:

1. Створення нової гри: Система отримує від користувача параметри нової гри, такі як кількість гравців і налаштування гри. Вона генерує розподіл ролей і зберігає необхідну інформацію про гру.

2. Додавання гравців: Система дозволяє ведучому вибрати кількість гравців та ввести їм імена. Система перевіряє правильність введених даних і зберігає інформацію про гравців.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Розподіл ролей гравцям: Система дозволяє здійснити розподіл ролей між гравцями.

4. Проведення гри: Система дозволяє ведучому здійснювати різні дії під час гри, такі як голосування, виконання ролей. Вона забезпечує правильний хід гри відповідно до правил та зберігає поточний стан гри.

Діаграма активності є візуальним інструментом, який допомагає проілюструвати послідовність дій або активностей, які відбуваються в системі [6]. Нижче наведена діаграма активності для системи «Помічник ведучого гри "Мафія"», яка показує основні операції та взаємодію з користувачами під час проведення гри (рис. 2.4).

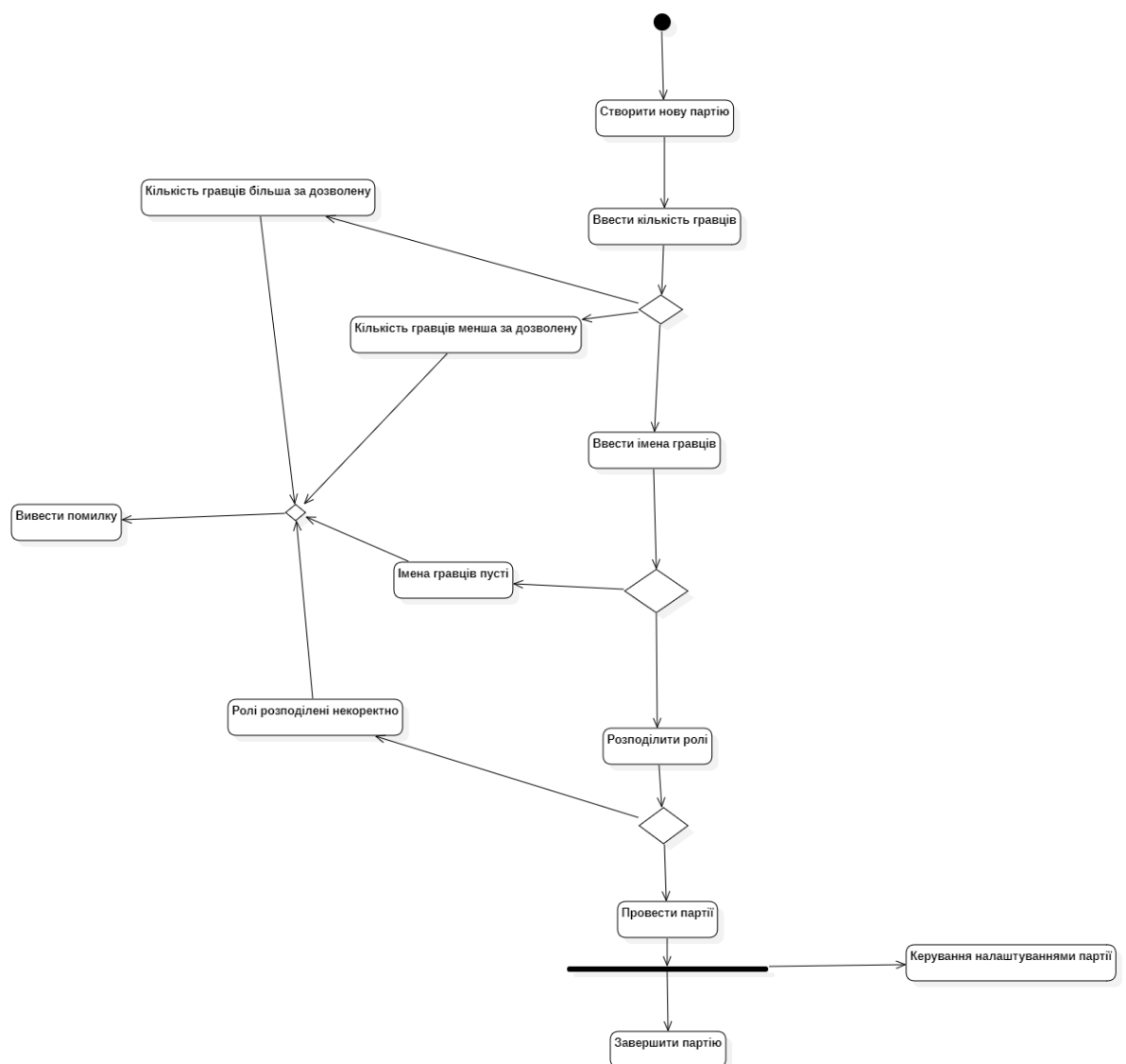


Рисунок 2.4 - Діаграма активності додатку

Спочатку виконується створення нової партії включаючи ініціалізацію змінних та налаштування параметрів гри.

Після створення партії користувачу пропонується ввести кількість гравців, які бажають взяти участь у грі. Це важливий крок, оскільки від кількості гравців залежить подальший розподіл ролей та проведення гри. Під час введення кількості гравців можуть бути встановлені обмеження або дозволені діапазони, які враховують можливості гри та бажання учасників.

Залежно від введеної кількості гравців виконується відповідний сценарій. Якщо кількість гравців перевищує дозволений ліміт, система повідомляє про помилку та вимагає ввести правильну кількість гравців. У разі, коли кількість гравців менша за дозволений ліміт, також виводиться помилка і користувачу пропонується внести корективи.

Якщо введена кількість гравців відповідає дозволеному ліміту, система просить користувача ввести імена гравців. Це дозволяє ідентифікувати учасників гри та враховувати їхні внески та досягнення під час проведення партії. В разі, якщо імена гравців не введені або введені значення є порожніми, система виводить відповідну помилку і вимагає вказати імена для всіх гравців.

Після успішного введення імен гравців, виконується сценарій розподілу ролей між гравцями. Цей етап може включати визначення ролей, обов'язків або позицій, які гравці займають у грі. У випадку, якщо розподіл ролей здійснений некоректно або не задовольняє правилам, система повідомляє про помилку та просить внести відповідні зміни.

Після успішного розподілу ролей виконується сценарій проведення партії. Гравці займають свої ролі і взаємодіють між собою згідно з правилами гри. Цей етап може включати ходи гравців, обмін інформацією, аналіз результатів та інші активності, які впливають на хід та висновки партії.

Завершення партії включає підведення підсумків та збереження даних. Під час підведення підсумків можуть бути розглянуті результати гри, а також визначено переможців.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

2.3. Структура компонентів системи

Даний підрозділ описує структуру розробленої системи та розташування різних компонентів і модулів. Структура системи складається з наступних складових частин [19]:

1. assets - у цій папці зберігаються картинки та іконки, необхідні для візуального відображення додатку.
2. components - ця папка містить загальні компоненти, які можуть використовуватися в різних частинах додатку. Сюди входять компоненти, такі як Button, Footer, Header, Input, Modal, ErrorMessage, які можуть бути використані в різних екранах або компонентах додатку.
3. i18n - у цій папці зберігаються налаштування для бібліотеки керування перекладами та файли з перекладами для двох мов: англійської та української. Для керування перекладами використовується бібліотека react-i18next, яка дозволяє легко локалізувати додаток для різних мов.
4. navigation - у цій папці зберігаються налаштування навігації між екранами додатку. Для реалізації навігації використовується бібліотека @react-navigation/native, яка надає потужні інструменти для створення навігаційних структур у додатку.
5. screens - у цій папці знаходяться екрани додатку. Кожен екран, такий як Game, GameOver, GameSetup, Menu, Rules, Settings, представлений окремим файлом, що містить компонент, який відповідає за відображення цього екрана.
6. store - цей модуль зберігає Redux store, який використовується для керування станом додатку. Redux - це бібліотека для управління станом додатку і дозволяє зручно спілкуватися між різними компонентами та модулями додатку [2].

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

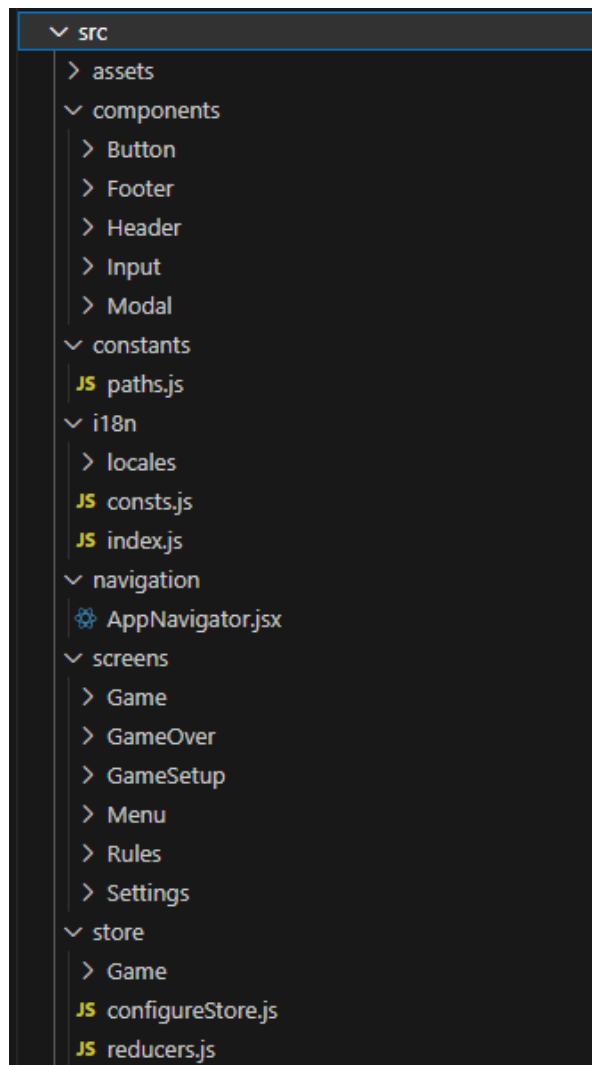


Рисунок 2.5 - Структура додатку

Використання такої структури допомагає зберігати код організованим, легко зрозумілим і підтримуваним [11]. Кожна частина системи має свою функціональність та відповідальність, що дозволяє розробнику швидко знаходити необхідний код та змінювати його.

2.4 Реалізація додатку

У цьому підрозділі буде розглянуто деталі реалізації додатку «Помічник ведучого гри "Мафія"».

Далі демонструється створення навігації між сторінками за допомогою бібліотеки react-navigation. Наведений нижче код представляє компонент

AppNavigator, який відповідає за налаштування навігації та визначає структуру додатку [12].

Спочатку були створені пусті компоненти в директорії screens. Це було зроблено з метою пізнішої реалізації функціональності для кожної сторінки додатку.

В файлі AppNavigator.js було використано ці пусті компоненти як екрани для навігації. Для цього було встановлено залежності @react-navigation/native та @react-navigation/native-stack, які дозволяють створювати навігаційну структуру в додатку [8].

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import MenuScreen from '../screens/Menu/Menu';
import GameSetupScreen from '../screens/GameSetup/GameSetup';
import GameScreen from '../screens/Game/Game';
import GameOverScreen from '../screens/GameOver/GameOver';
import RulesScreen from '../screens/Rules/Rules';
import SettingsScreen from '../screens/Settings/Settings';
import { PATHS } from '../constants/paths';
import Header from '../components/Header/Header';

const Stack = createNativeStackNavigator();

function AppNavigator() {
  return (
    <NavigationContainer>
      <Stack.Navigator screenOptions={{ headerShown: false }}>
        <Stack.Screen name={PATHS.MENU} component={MenuScreen} />
        <Stack.Screen
          options={{ header: Header }}
          name={PATHS.GAME_SETUP}
          component={GameSetupScreen}
        />
        <Stack.Screen name={PATHS.GAME} component={GameScreen} />
        <Stack.Screen name={PATHS.GAME_OVER} component={GameOverScreen} />
        <Stack.Screen name={PATHS.RULES} component={RulesScreen} />
        <Stack.Screen name={PATHS.SETTINGS} component={SettingsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default AppNavigator;
```

У компоненті Stack.Navigator встановлюються параметри за замовчуванням, такі як приховання заголовку сторінки (headerShown: false).

Далі для кожної сторінки використовуються компоненти Stack.Screen, які визначають ім'я сторінки та компонент, що його представляє. Наприклад, Stack.Screen name={PATHS.MENU} component={MenuScreen} визначає сторінку меню (MenuScreen), а name={PATHS.GAME_SETUP} встановлює шлях до сторінки налаштування гри.

Для деяких сторінок, таких як GameSetupScreen, використовується параметр options, де встановлюється власний заголовок (Header) для сторінки.

Таким чином, компонент AppNavigator встановлює структуру навігації між сторінками додатку і забезпечує відображення відповідних компонентів на основі поточного маршруту.

Далі розглянемо реалізацію головного меню.

```
import React from 'react';
import {
  View, StyleSheet, BackHandler, Image,
} from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { useTranslation } from 'react-i18next';
import Button from '../../components/Button/Button';
import { PATHS } from '../../constants/paths';
import MenuJpg from '../../assets/images/menu.jpg';

export default function Menu() {
  const { navigate } = useNavigation();
  const { t } = useTranslation();
  const styles = StyleSheet.create({
    container: {
      flex: 1,
      backgroundColor: '#424242',
      alignItems: 'center',
      paddingHorizontal: 20,
      paddingVertical: 50,
    },
    image: {
      width: 200,
      height: 200,
      borderRadius: 10,
    },
  });
  return (
```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<View style={styles.container}>
  <Image source={MenuJpg} style={styles.image} />
  <Button title={t('menuPage.startButton')} onPress={() =>
navigate(PATHS.GAME_SETUP)} />
  <Button title={t('menuPage.rulesButton')} />
  <Button title={t('menuPage.settingsButton')} />
  <Button title={t('menuPage.exitButton')} onPress={() =>
BackHandler.exitApp()} />
</View>
);
}

```

Компонент Menu реалізує екран меню додатку. У цьому компоненті використовуються різні імпорти з бібліотек react-native та @react-navigation/native для побудови інтерфейсу та навігації. Також використовується бібліотека react-i18next для міжнародного перекладу тексту [20].

У функціональному компоненті Menu використовуються хуки useNavigation та useTranslation для отримання навігаційного об'єкту та функціональності перекладу тексту відповідно.

У стилізації компонента використовуються об'єктні стилі, визначені за допомогою StyleSheet.create [14]. Встановлюються стилі для контейнера container, зображення image та інших елементів.

У методі render компонента розміщуються різні елементи, такі як зображення, кнопки та їх заголовки. Зображення отримується з файлу MenuJpg, який містить шлях до відповідного зображення. Кнопки мають текст, який отримується з перекладу за допомогою функції t.

Компонент Menu також відповідає за навігацію між екранами за допомогою функції navigate з використанням шляхів з константи PATHS. Наприклад, при натисканні на кнопку «Start» виконується навігація до екрану налаштування гри.

Кнопка «Exit» відповідає за вихід з додатку, при натисканні на неї викликається функція BackHandler.exitApp().

Далі розглянемо основний екран гри.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		31

```

import React, { useState, useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import {
  View, FlatList, StyleSheet, DrawerLayoutAndroid, Text, Image, TextInput,
} from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { useTranslation } from 'react-i18next';
import Button from '../components/Button/Button';
import { PATHS } from '../constants/paths';
import CountdownTimer from './components/CountdownTimer';
import { selectPlayers, selectVotedPlayers } from '../store/Game/selectors';
import {
  addPlayerOnVote, addVotes, removePlayer, resetVotedPlayers,
} from '../store/Game/actions';
import ButtonImage from '../components/ButtonImage/ButtonImage';
import playPng from '../assets/images/play.png';
import dayPng from '../assets/images/day.png';
import nightPng from '../assets/images/night.png';
import seerPng from '../assets/images/seer.png';
import masonPng from '../assets/images/mason.png';
import protectorPng from '../assets/images/protector.png';

export default function Game() {
  const dispatch = useDispatch();
  const { t } = useTranslation();
  const { navigate } = useNavigation();
  const players = useSelector(selectPlayers);
  const votedPlayers = useSelector(selectVotedPlayers);
  const [currentRound, setCurrentRound] = useState(0);
  const [currentTime, setCurrentTime] = useState('Night');
  const [isTimerPlaying, setIsTimerPlaying] = useState(false);
  const [currentPlayer, setCurrentPlayer] = useState(0);
  const [isVoting, setIsVoting] = useState(false);

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      backgroundColor: '#424242',
      justifyContent: 'center',
      paddingHorizontal: 20,
      color: '#fff',
    },
    input: {
      borderWidth: 1,
      borderColor: '#fff',
      borderRadius: 10,
      color: '#fff',
      paddingHorizontal: 10,
    },
  });
});

```

```

const onActionsButtonPress = (playerIndex) => {
  if (currentTime === 'Night') {
    dispatch(removePlayer(players[playerIndex].name));
  } else {
    dispatch(addPlayerOnVote(players[playerIndex]));
  }
};

const getPlayerRole = (role) => {
  if (role === 'masons') return masonPng;
  if (role === 'protector') return protectorPng;
  return seerPng;
};

const navigationView = (
  <View style={{
    flex: 1, backgroundColor: '#424242', alignItems: 'center', justifyContent:
'center',
  }}
  >
    <FlatList
      data={players}
      keyExtractor={({item, index}) => index.toString()}
      horizontal={false}
      renderItem={({ index }) => (
        <View style={{ paddingVertical: 5, flexDirection: 'column' }}>
          <View style={{ flexDirection: 'row', alignItems: 'center' }}>
            <Image
              source={getPlayerRole(players[index].role)}
              style={{ width: 20, height: 20, marginRight: 10 }}
            />
            <Text style={{ color: '#fff' }}>
              {players[index].name}
            </Text>
          </View>
          <Button
            title={currentTime === 'Night' ? 'Kill' : 'On vote'}
            onPress={() => onActionsButtonPress(index)}
          />
        </View>
      )}
    />
  </View>
);

const checkIsGameOver = () => {
  if (players.filter((player) => player.role === 'masons').length === 0) {
    return true;
  }
}

```

```

    if (players.filter((player) => player.role === 'masons').length >=
(players.filter((player) => player.role === 'protector').length +
players.filter((player) => player.role === 'seer').length)) {
        return true;
    }

    return true;
};
const onComplete = () => {
    setIsTimerPlaying(false);

    if (checkIsGameOver()) {
        navigate(PATHS.GAME_OVER);
    } else if (currentTime === 'Night') {
        setCurrentTime('Day');
        setCurrentRound(currentRound + 1);
    } else if (isVoting) {
        dispatch(
            removePlayer(
                votedPlayers.reduce(
                    (prev, current) => ((prev.votes > current.votes) ? prev : current),
                ).name,
            ),
        );
        dispatch(resetVotedPlayers());
        setCurrentTime('Night');
        setIsVoting(false);
    } else if (currentPlayer === players.length - 1) {
        if (votedPlayers.length > 0) {
            setIsVoting(true);
        } else {
            setCurrentTime('Night');
        }
    } else {
        setCurrentPlayer(currentPlayer + 1);
    }

    return [true, 5];
};

useEffect(() => {
    if (currentTime === 'Day' && votedPlayers.length > 0) {
        dispatch(
            removePlayer(
                votedPlayers.reduce(
                    (prev, current) => ((prev.value > current.value) ? prev : current),
                ).name,
            ),
        );
    }
}, []);

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const handleSubmitEditing = (e, name) => {
  console.log(e.nativeEvent.text);
  dispatch(addVotes(name, +e.nativeEvent.text));
};

return (
  <DrawerLayoutAndroid
    drawerWidth={200}
    drawerPosition="left"
    renderNavigationView={() => navigationView}
    style={styles.container}
  >
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Image
        source={currentTime === 'Day' ? dayPng : nightPng}
        style={{
          width: 50, height: 50, position: 'absolute', right: 10, top: 10,
        }}
      />
      <Text style={{
        margin: 10, fontSize: 15, textAlign: 'right', color: '#fff',
      }}
      >
        {t('gamePage.currentRound', { currentRound })}
      </Text>
      {currentTime === 'Day' && (
        <Text style={{
          margin: 10, fontSize: 15, textAlign: 'right', color: '#fff',
        }}
        >
          {isVoting ? t('gamePage.voting') : t('gamePage.talking', { playerName:
players[currentPlayer].name })}
        </Text>
      )}
      <CountdownTimer isTimerPlaying={isTimerPlaying} onComplete={onComplete} />
      <ButtonImage image={playPng} onPress={() =>
setIsTimerPlaying(!isTimerPlaying)} />
      {votedPlayers && votedPlayers.length > 0 && (
        <Text style={{ color: '#fff' }}>{t('gamePage.onVote')}</Text>)}
      <FlatList
        data={votedPlayers}
        keyExtractor={({item, index}) => index.toString()}
        horizontal={false}
        renderItem={({ index }) => (
          <View style={{
            paddingVertical: 5, flexDirection: 'row', alignItems: 'center',
width: 150, justifyContent: 'space-between', marginTop: 20,
          }}
          >
            <Text style={{ color: '#fff' }}>

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

```

        {votedPlayers[index].name}
      </Text>
      {isVoting && (
        <TextInput
          style={styles.input}
        />
      )}
    </View>
  )}
/>
</View>
<View style={{
  position: 'absolute', width: 5, height: 100, backgroundColor: '#607D8B',
top: '40%',
}}
/>
</DrawerLayoutAndroid>
);
}

```

У функціональному компоненті Game використовуються хуки useDispatch, useSelector та useTranslation для отримання доступу до диспетчера, стану та функціональності перекладу тексту відповідно. Також використовується хук useNavigation для отримання навігаційного об'єкту.

У стилізації компонента використовуються об'єктні стилі, визначені за допомогою StyleSheet.create. Встановлюються стилі для контейнера container, введення input та інших елементів.

У методі onActionsButtonPress оброблюється натискання кнопок на екрані гри в залежності від поточного часу.

Метод getPlayerRole повертає відповідне зображення для гравця в залежності від його ролі.

В компоненті DrawerLayoutAndroid використовується висувне меню, яке містить список гравців. Використовується компонент FlatList, щоб відобразити кожного гравця з кнопкою «Вбити» або «На голосування», залежно від поточного часу гри.

Метод checkIsGameOver перевіряє, чи гра завершена. Якщо не залишилося гравців з роллю «мафія» або кількість гравців з роллю «мафія» перевищує або

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

дорівнює сумі гравців з ролями «комісар» та «мирний житель», гра вважається завершеною.

Метод onComplete викликається при завершенні таймеру гри. В залежності від поточного часу гри та стану голосування, оновлюються різні значення: час гри, поточний раунд, поточний гравець, стан голосування. Якщо гра завершена, виконується навігація до екрану «Гра завершена». Якщо поточний час – «Ніч», час гри змінюється на «День» і збільшується поточний раунд. Якщо триває голосування, видаляється гравець з найбільшою кількістю голосів і скидаються дані голосування. Якщо поточний гравець - останній гравець у списку, перевіряється, чи є голоси, і якщо є, ввімкнеться голосування, в іншому випадку поточний час гри змінюється на «Ніч». В кінці методу повертаються значення для налаштування таймера.

У методі useEffect виконується перевірка, чи є голоси на день і, якщо є, видаляється гравець з найбільшою кількістю голосів.

Метод handleSubmitEditing оброблює введення тексту голосування гравця.

У методі render компонента розміщуються різні елементи, такі як зображення, текст, таймер, кнопка відтворення, список гравців на голосуванні. Зображення змінюється в залежності від поточного часу гри. Текст відображає поточний раунд та поточний стан гри. Компонент CountdownTimer відображає таймер гри. Кнопка відтворення міняє стан таймера при натисканні. Також відображається список гравців на голосуванні з можливістю вводу голосів.

У цілому, компонент Game відображає різні елементи гри та дозволяє зручно керувати станом гри.

Висновки до другого розділу

У другому розділі «Проектування та реалізація додатку» було проведено аналіз структури системи, розроблено навігацію між сторінками та реалізовано додаток.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

У підрозділі «Структура системи» була представлена загальна структура системи, що включає папки з різними компонентами, ресурсами та налаштуваннями для підтримки багатомовності.

У підрозділі «Реалізація додатку» було показано приклад реалізації навігації між сторінками за допомогою бібліотеки react-navigation. Було описано структуру компоненту AppNavigator, який встановлює маршрутизацію між різними екранами додатку, а також описано роботу екранів Menu та Game.

Завдяки правильно розробленій структурі системи та використанню відповідних інструментів, таких як навігаційна бібліотека react-navigation, було забезпечено зручну навігацію користувача між різними частинами додатку. Це сприяє покращенню користувацького досвіду і забезпечує легку навігацію між функціональними складовими додатку.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Архітектура додатку та розгортання додатку

Архітектура додатку грає ключову роль у забезпеченні ефективної та стабільної роботи системи. У цьому підрозділі буде розглянуто архітектурні принципи та розгортання додатку «Помічник ведучого гри "Мафія"».

Для розробки додатку «Помічник ведучого гри "Мафія"» була використана загальноприйнята структура проектів на базі RN. Ця структура включає наступні складові:

1. Компоненти (Components): Вони включають у себе повторно використовувані елементи інтерфейсу користувача, такі як кнопки, поля вводу, модальні вікна тощо. У додатку «Помічник ведучого гри "Мафія"» були створені загальні компоненти, такі як Button, Footer, Header, Input, Modal, ErrorMessage, які можуть бути використані на різних екранах додатку.
2. Екрани (Screens): Вони представляють окремі сторінки або екрани додатку. У нашому випадку були створені екрани, такі як MenuScreen, GameSetupScreen, GameScreen, GameOverScreen, RulesScreen, SettingsScreen, які відповідають різним функціональним частинам гри Мафія.
3. Навігація (Navigation): Для навігації між екранами додатку використовується бібліотека @react-navigation/native, яка забезпечує розташування екранів у формі стеку або табуляції, залежно від потреб додатку. У нашому випадку використовувалась навігація у формі стеку для переміщення між екранами.
4. Сховище даних (Store): Для збереження та управління станом додатку використовується бібліотека Redux. Redux дозволяє централізовано керувати даними, що спрощує їх передачу та зміну між компонентами.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

При розгортанні додатку «Помічник ведучого гри "Мафія"» використовується інструмент react-native-cli (рис.3.1). Цей інструмент дозволяє збудувати та розгорнути додаток на різних платформах, таких як Android та iOS.

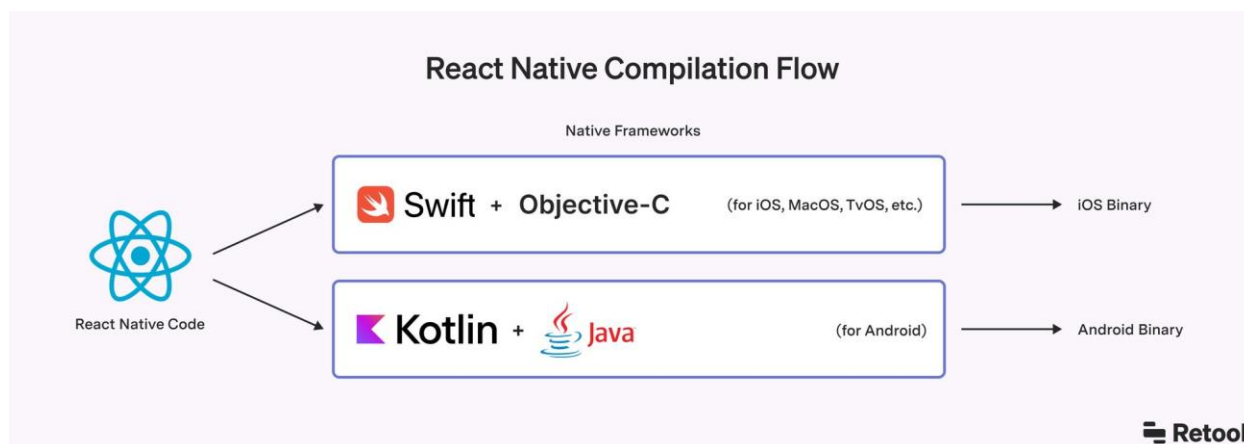


Рисунок 3.1 - Схема роботи react-native-cli

Процес розгортання включає наступні кроки:

1. **Компіляція:** За допомогою react-native-cli компілюється весь код та ресурси додатку.
2. **Упакування:** Компіляція генерує виконуваний файл додатку, який можна розповсюджувати.
3. **Розповсюдження:** Отриманий виконуваний файл можна розповсюджувати через відповідні магазини додатків, такі як Google Play для Android або App Store для iOS [21].

Розгортання додатку здійснюється шляхом підготовки інсталяційного пакету, який містить весь необхідний код та ресурси додатку [24]. Цей пакет можна розповсюджувати на різних пристроях для встановлення та використання додатку.

Використання стандартної структури проекту на базі RN та інструменту react-native-cli для розгортання додатку «Помічник ведучого гри "Мафія"» забезпечує зручність у розробці, стабільну архітектуру та простоту розгортання на різних платформах.

3.2. Інтерфейс та порядок роботи з системою

У цьому підрозділі буде розглянуто інтерфейс додатку «Помічник для ведучого гри "Мафія"» і описано порядок роботи з системою.

У рис. 3.2 показано головне меню додатку, де користувач може обрати потрібну опцію, таку як «Нова гра», «Налаштування», «Правила» або «Вийти». Це стартова точка для взаємодії з додатком.

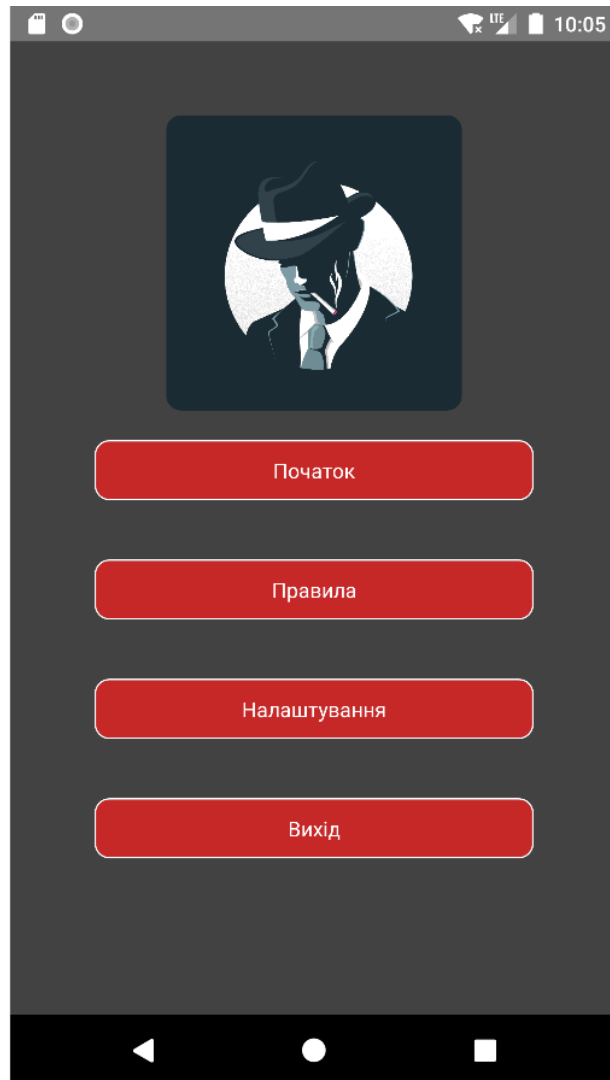


Рисунок 3.2 - Головний екран додатку

Після вибору опції «Нова гра», користувач потрапляє на екран налаштування нової гри. Тут він може ввести необхідні параметри, такі як кількість гравців (рис. 3.3), імена гравців (рис. 3.4) та розподіл ролей (рис. 3.5). Користувач може вибрати різні опції та налаштування, щоб створити гру за своїми уподобаннями.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.3 - Екран введення кількості гравців

Рисунок 3.4 - Екран введення імен гравців

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

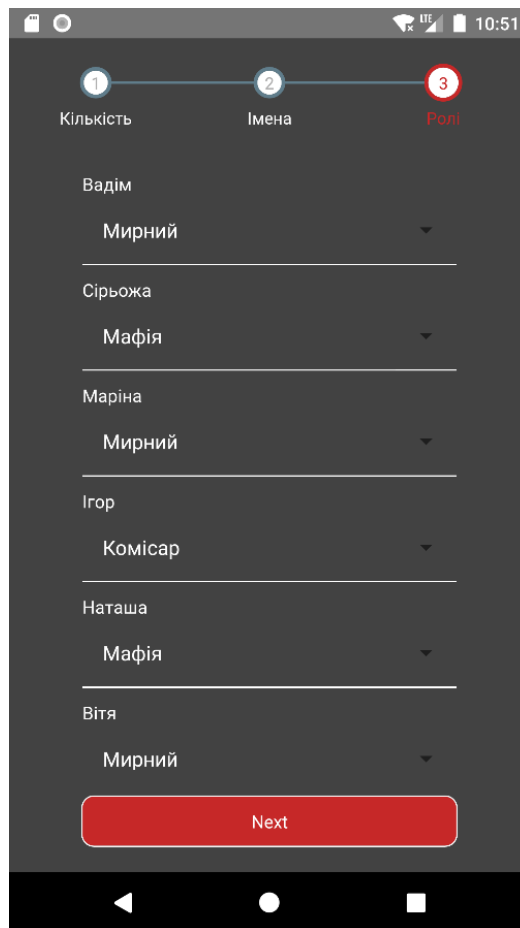


Рисунок 3.5 - Екран вибору ролей

На кожному кроці налаштування партії присутній прогрес кроків, який відображає прогрес налаштування партії. Прогрес кроків дозволяє користувачеві зорієнтуватись у процесі та відстежувати скільки етапів залишилось до завершення налаштування.

Один із важливих аспектів цього інтерфейсу - можливість повернутись до попереднього кроку та внести зміни. Користувач може змінити параметри, які вже були введені, натиснувши на попередній етап в прогресі кроків або використовуючи відповідну кнопку «Назад», на телефоні. Це дозволяє користувачу вносити корективи та змінювати свої вибори на будь-якому етапі процесу налаштування партії, забезпечуючи більшу гнучкість та зручність в роботі з додатком.

Після налаштування нової гри користувач потрапляє на екран гри (рис. 3.6). Тут він може бачити усю необхідну інформацію для проведення.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

Екран гри є одним з ключових екранів додатку і містить значну кількість інформації та функціональності, яка спрямована на підтримку та поліпшення гри «Мафія». Даний екран відображає загальну інформацію про поточну гру, таку як поточний раунд, таймер раунду, кнопка початку раунду та час доби. Розглянемо детальніше таймер.

Контроль таймера: ведучий може мати можливість контролювати таймер гри, запускати або зупиняти та відстежувати його прогрес. Таймер за замовчуванням починається з 60 секунд, цей параметр може бути зміненим у налаштуваннях. Таймер змінює свої кольори зі зменшенням часу. Це візуальне відображення допомагає ведучому зорієнтуватися в часових рамках та відстежувати, скільки часу залишилося для виконання певних дій. Початково таймер може мати зелений колір, що вказує на достатній часовий резерв. З часом, колір таймера поступово змінюється на жовтий та червоний, що вказує на наближення до завершення етапу гри. Цей візуальний ефект дозволяє ведучому бути в курсі часових обмежень.

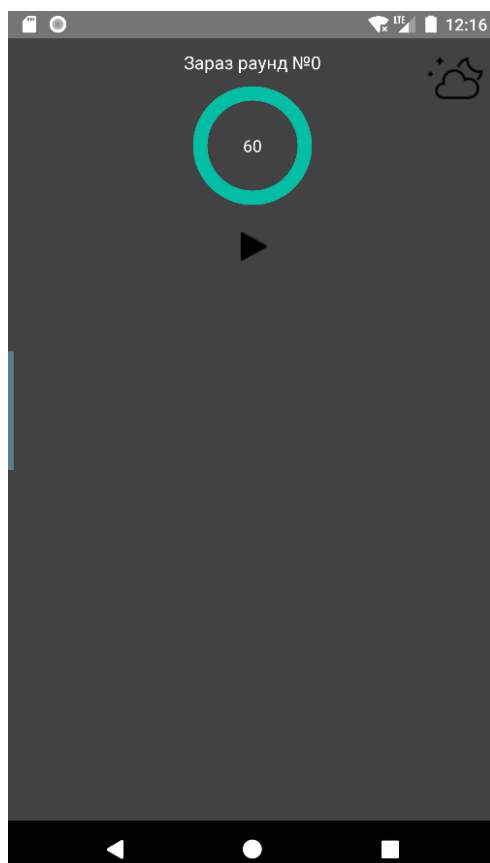


Рисунок 3.6 - Екран проведення партії

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

На екрані гри також можна побачити індикатор дня/ночі та поточного раунду. Ці елементи надають ведучому додаткову інформацію про хід гри.

Індикатор дня/ночі змінює свій стан залежно від фази гри. Він може мати два різних значки, що відповідають дню та ночі. Це допомагає ведучому визначити, коли настає день і коли настає ніч в межах гри.

Поточний раунд відображає інформацію про те, який раунд гри зараз триває. Це допомагає ведучому бути в курсі того, на якому етапі гра знаходяться.

Індикатор дня/ночі та поточний раунд є важливими елементами інтерфейсу, які допомагають ведучому керувати грою.

На цьому також присутнє бокове меню, яке можна викликати шляхом здійснення перетягування вправо (рис. 3.7). На наявність цього меню вказує спеціальна лінія з лівого боку екрана.

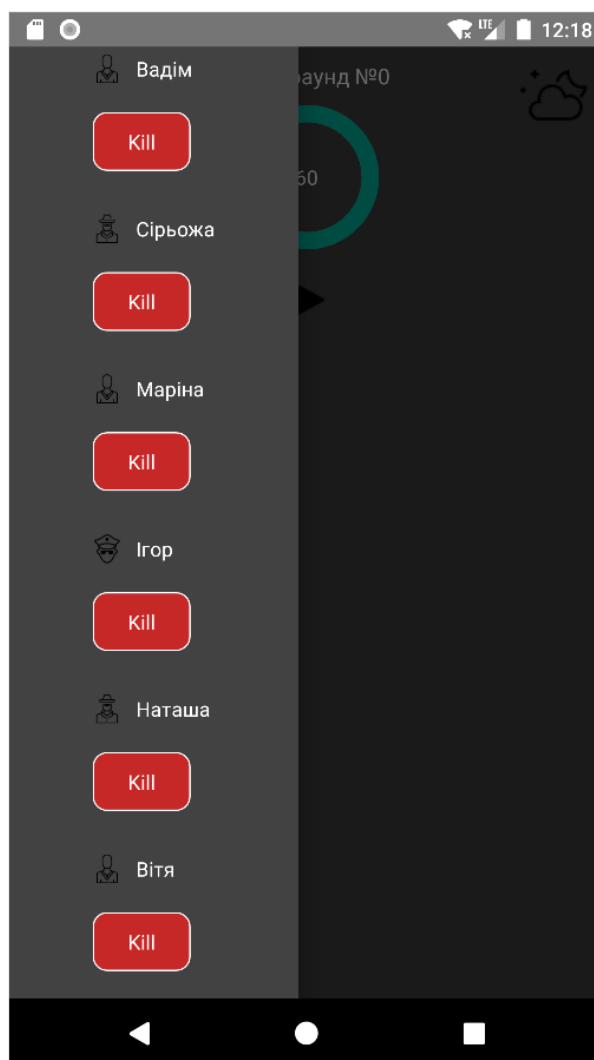


Рисунок 3.7 - Меню гравців

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

При відкритті бокового меню ведучий може побачити список учасників гри. Список гравців містить їх імена та ролі у вигляді відповідних значків. Це допомагає не забути, яка роль у кожного з гравців.

У боковому меню також може бути присутня кнопка для голосування або вбивства гравця. Ця кнопка дозволяє здійснювати дії, пов'язані з виключенням певного учасника гри або здійснення голосування за підозрюваного. Це є важливою частиною механіки гри і дозволяє зручно контролювати ігровий процес.

На етапі голосування в інтерфейсі додатку з'являється спеціальний список, розташований у нижній частині екрана (рис. 3.8).

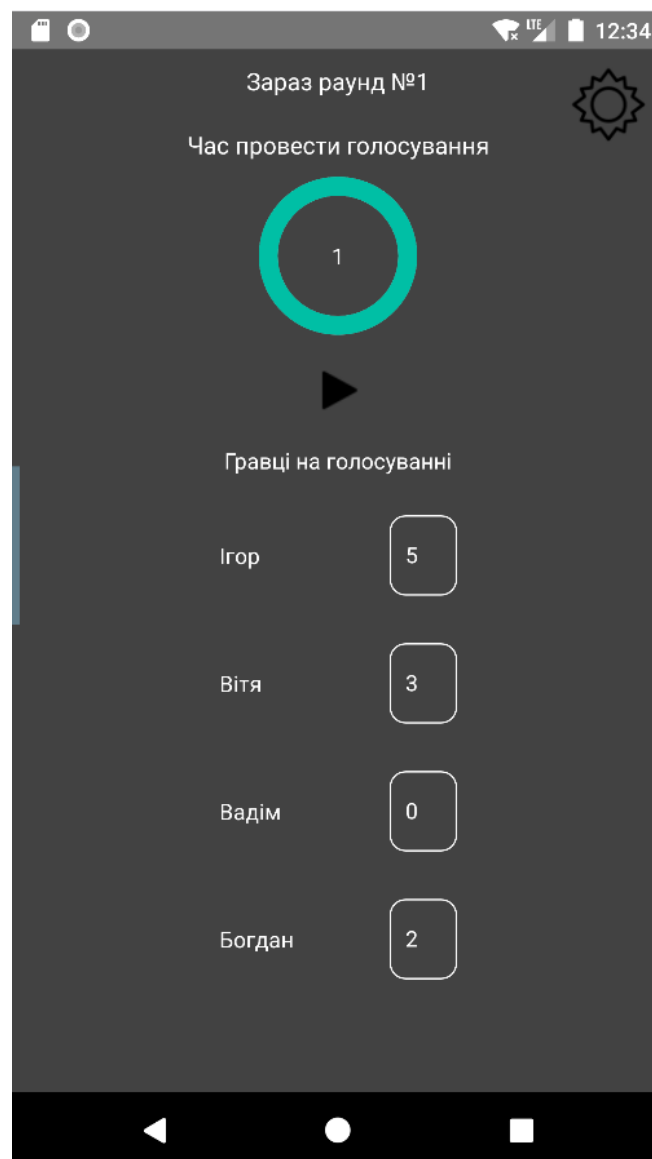


Рисунок 3.8 - Меню голосування

Цей список містить імена гравців, які були включені до голосування. Ведучий може легко переглянути цей список і відслідковувати, хто знаходиться на голосуванні.

Після того, як гравці висловили свою думку, ведучий переходить до фази голосування. На екрані з'являється ім'я кожного гравця, і ведучий проводить підрахунок голосів. За допомогою спеціального інтерфейсу, ведучий може ввести кількість голосів, отриманих кожним гравцем.

Також, в інтерфейсі додатку забезпечується автоматичне додавання голосів тих гравців, які не голосували, до останнього гравця в списку. Це сприяє повному урахуванню голосів і забезпечує, що нічий голос не залишиться непроголосованим.

Цей інтерфейсний підхід дозволяє ведучому зручно проводити голосування, відстежувати результати та забезпечувати справедливість у процесі прийняття рішень у грі.

Далі йде екран закінчення гри, який відображається після завершення останнього раунду гри (рис 3.9). На цьому екрані гравці отримують інформацію про те, яка фракція перемогла у грі.

Текст з'являється на екрані і він вказує на те, яка саме роль або фракція здобула перемогу. При цьому, поряд з текстом, відображається іконка, яка символізує цю роль або фракцію. Це додає візуальну підтримку і допомагає гравцям швидко розпізнати, хто саме переміг.

На екрані закінчення гри також присутня кнопка «Повернутись в меню». Ця кнопка дає можливість повернутись до головного меню додатку після закінчення гри. Вона надає зручну навігаційну опцію.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

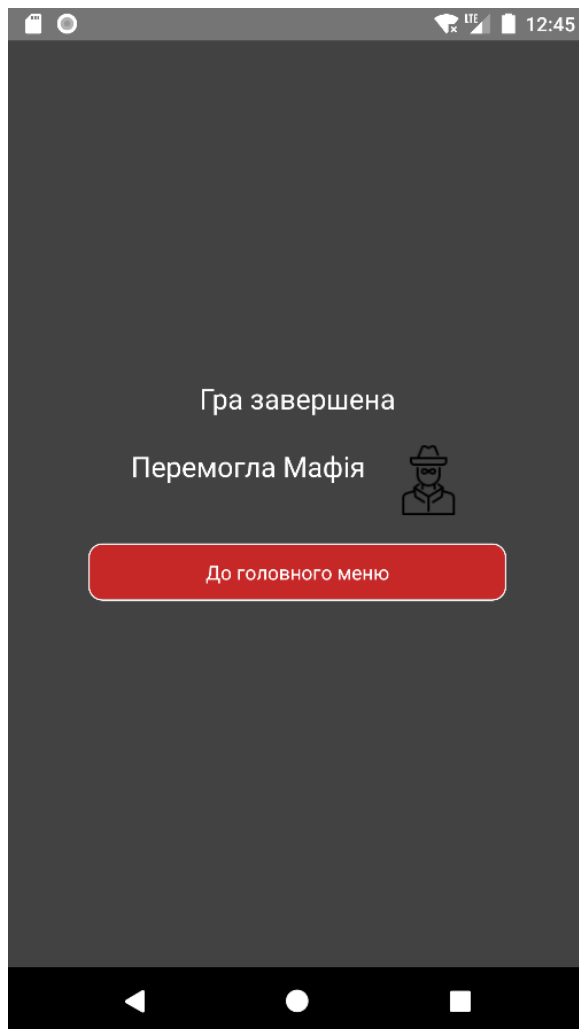


Рисунок 3.9 - Екран закінчення гри

Екран закінчення гри є важливою частиною інтерфейсу, оскільки він надає гравцям остаточну інформацію про результати гри і дозволяє їм продовжити взаємодію з додатком після завершення одного ігрового сеансу.

3.3. Тестування роботи додатку

Тестування є важливою частиною процесу розробки додатку. В проекті було вдосконалено якість програмного забезпечення, застосовуючи різні підходи до тестування.

Широко використовувалась бібліотека Jest для написання юніт-тестів. За допомогою Jest перевірялись окремі функції, компоненти та модулі на правильність їх роботи [26]. Це дозволило виявити й виправити помилки раніше та забезпечити стабільність додатку (рис. 3.10).

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Test Suites: 36 passed, 36 total
Tests:      1 todo, 260 passed, 261 total
Snapshots:  0 total
Time:       24.103 s
Ran all test suites.

Watch Usage: Press w to show more.

```

Рисунок 3.10 - Результат проходження юніт тестів

Кожне поле у додатку має валідацію, реалізовану за допомогою бібліотек Yup та react-hook-form [27]. Були написані тести, щоб переконатися, що валідація працює належним чином і вірно перевіряє введені дані. Це допомогло уникнути некоректного введення користувачів і забезпечити відповідну обробку помилок. Відповідні рисунки 3.11-3.14 показують приклад виведення різних помилок валідації.

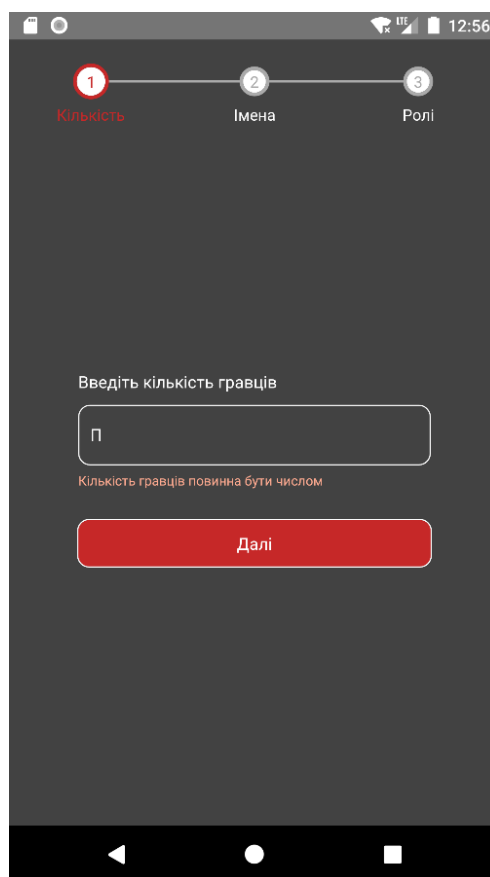


Рисунок 3.11 - Приклад роботи валідації номер 1

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.12 - Приклад роботи валідації номер 2

Рисунок 3.13 - Приклад роботи валідації номер 3

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

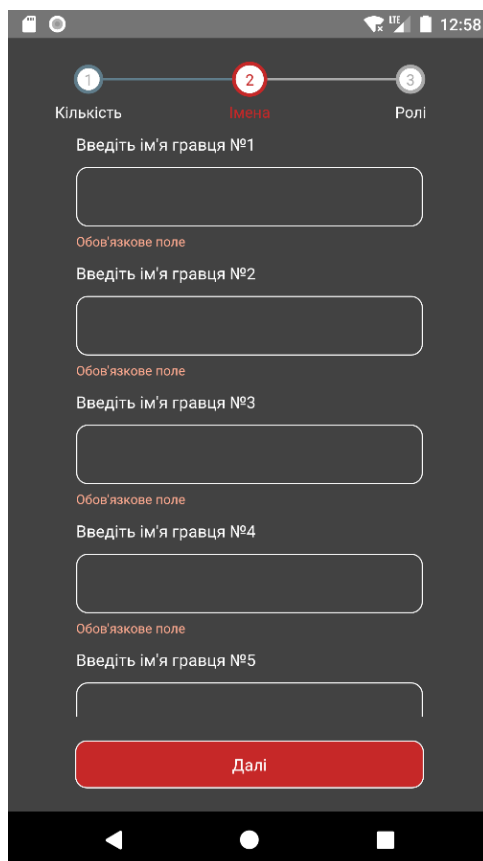


Рисунок 3.14 - Приклад роботи валідації номер 4

Також було проведено функціональне тестування для перевірки роботи основних функцій додатку. Це включало тестування навігації між сторінками, взаємодії з інтерактивними елементами, і т.д.

Загально кажучи, тестування додатку було комплексним процесом, що включав різні види тестів. Це дозволило нам забезпечити якість та надійність нашого додатку перед його розгортанням та використанням користувачами.

Висновки до третього розділу

У цьому розділі було розглянуто порядок роботи та тестування системи додатку. Починаючи з архітектури додатку та його розгортання, використовуємо загальноприйняту структуру реакт-проектів та react-native-cli. Це надає гнучкість та контроль над розробкою та розгортанням додатку.

У підрозділі «Інтерфейс та порядок роботи з системою» було описано інтерфейс додатку та його основні функціональні можливості. Ведучий гри має доступ до різних інструментів та інформації, яка допомагає йому вести гру та

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

контролювати її хід. Додаток надає зручний порядок роботи, дозволяючи вести гру крок за кроком, а також змінювати параметри та налаштування в будь-який момент.

У підрозділі «Тестування роботи додатку» було підкреслено важливість тестування для досягнення високої якості програмного забезпечення. Було використано бібліотеку Jest для написання юніт-тестів, що допомагають перевіряти правильність роботи окремих функцій та компонентів. Також було використано бібліотеку Yup для створення валідації полів у додатку.

В результаті система має структуровану архітектуру, зручний інтерфейс та пройшла ефективне тестування. Більша частину коду покрита юніт-тестами, що допомагає перевірити правильність роботи окремих компонентів. Валідація полів за допомогою Yup забезпечує коректність та надійність введених даних. Тестування взаємодії та безпеки допомагає забезпечити стабільність та захищеність системи.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В рамках дипломної роботи «Помічник ведучого гри "Мафія"», було успішно розроблено мобільний додаток, який має важливе значення для полегшення процесу проведення гри «Мафія». Протягом всієї роботи був проведений детальний аналіз проблематики гри та визначені вимоги до додатку. Було розглянуто різні аспекти реалізації та вибрані оптимальні методи та інструменти для розробки.

У розділі «Аналіз проблематики, методів та засобів вирішення задачі» були визначені вимоги до додатку і проведений докладний аналіз аналогів та технічних вимог до системи. Це дозволило визначити оптимальний напрямок роботи та забезпечити високу функціональність додатку.

У розділі «Проектування та реалізація додатку» була розроблена архітектура додатку та створені всі необхідні складові. Був розроблений інтерфейс користувача, логіка гри, система навігації та збереження даних. Кожен екран та його функціонал були детально розглянуті та впроваджені з використанням сучасних технологій.

У розділі «Порядок роботи та тестування роботи додатку» був описаний порядок роботи з додатком для ведучого гри «Мафія». Розглянуті кроки, що потрібно виконати для підготовки і проведення гри, включаючи створення нової гри, додавання гравців, розподіл ролей та керування ходом гри. Були надані детальні інструкції щодо кожного з цих кроків, що сприяє зручній та ефективній організації гри.

Також у цьому розділі було виконане тестування роботи додатку з використанням бібліотеки Jest. Тестування включало перевірку функціоналу, валідацію даних та забезпечення коректної роботи додатку в різних сценаріях використання. Були написані юніт-тести, що дозволяють перевірити правильність роботи окремих модулів та компонентів додатку.

В результаті роботи було створено ефективний та зручний у використанні додаток, який допомагає ведучому гри «Мафія» в організації та проведенні гри. Додаток має чітку структуру та забезпечує зручний інтерфейс для користувачів.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

Покриття коду тестами гарантує його стабільну та надійну роботу. Результати цієї роботи є вагомим внеском у сферу гри «Мафія» та можуть бути використані для поліпшення процесу гри та забезпечення задоволення для всіх учасників.

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Список використаних джерел

1. Сховище документації для веб-розробників по JS [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
2. Документація Redux [Електронний ресурс] – Режим доступу до ресурсу: <https://redux.js.org/>
3. Документація RN [Електронний ресурс] – Режим доступу до ресурсу: <https://reactnative.dev/>
4. Документація React [Електронний ресурс] – Режим доступу до ресурсу: <https://reactjs.org/>
5. Стаття про гру «Мафія» [Електронний ресурс] – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Mafia_\(party_game\)](https://en.wikipedia.org/wiki/Mafia_(party_game))
6. Документація Star UML [Електронний ресурс] – Режим доступу до ресурсу: <https://staruml.sourceforge.net/>
7. Eric Windmill Flutter in Action, 2020 – 368 с
8. Документація React Navigation [Електронний ресурс] – Режим доступу до ресурсу: <https://reactnavigation.org/>
9. Michael E. Branstein and Nicholas J. Branstein NativeScript in Action, 2016 – 350 с
10. Bonnie Eisenman Learning React Native: Building Native Mobile Apps with JavaScript, 2017. – 240 с
11. Блог розробників React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>
12. Блог розробників React Native [Електронний ресурс] – Режим доступу до ресурсу: <https://reactnative.dev/blog>
13. Robin Wieruch The Road to React: Your journey to master plain yet pragmatic React.js, 2018. – 292 с
14. Mark Tielens Thomas React in Action First Edition, 2018. – 360 с

		В.В. Галаган			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

15. Adam Boduch React and React Native: Build cross-platform JavaScript applications with native power for the web, desktop, and mobile, 4th Edition, 2022. – 600 с
16. Akshat Paul React Native for Mobile Development: Harness the Power of React Native to Create Stunning iOS and Android Applications, 2019. – 253 с
17. Emilio Rodriguez Martinez React: Cross-Platform Application Development with React Native: Build 4 real-world apps with React Native, 2018. – 182 с
18. Eric Masiello Mastering React Native, 2017. – 496 с
19. Курс для веб розробників на платформі Udemy [Електронний ресурс] – Режим доступу до ресурсу: <https://www.udemy.com/course/the-complete-web-development-bootcamp/>
20. Документація react-i18next [Електронний ресурс] – Режим доступу до ресурсу: <https://react.i18next.com/>
21. David Thomas The Pragmatic Programmer: Your Journey To Mastery, 20th Anniversary Edition, 2019. – 352 с
22. Stan Bershadskiy React Native Cookbook, 2016. – 536 с
23. Adam Boduch React and React Native: Complete guide to web and native mobile development with React, 2018. – 540 с
24. Theresa Neil Mobile Design Pattern Gallery: UI Patterns for Smartphone Apps, 2014 – 404 с
25. Jim Bennett Xamarin in Action, 2018 – 608 с
26. Документація jest [Електронний ресурс] – Режим доступу до ресурсу: <https://jestjs.io/docs/>
27. Документація react-hook-form [Електронний ресурс] – Режим доступу до ресурсу: <https://react-hook-form.com/get-started>
28. Greg Lim Beginning React with Hooks, 2020. – 122 с
29. Karl Wieggers Software Requirements (Developer Best Practices), 2013 – 672 с
30. Dan Hermes Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals, 2015 – 470 с

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент AppNavigator.jsx

```

import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import MenuScreen from '../screens/Menu/Menu';
import GameSetupScreen from '../screens/GameSetup/GameSetup';
import GameScreen from '../screens/Game/Game';
import GameOverScreen from '../screens/GameOver/GameOver';
import RulesScreen from '../screens/Rules/Rules';
import SettingsScreen from '../screens/Settings/Settings';
import { PATHS } from '../constants/paths';
import Header from '../components/Header/Header';

const Stack = createNativeStackNavigator();

function AppNavigator() {
  return (
    <NavigationContainer>
      <Stack.Navigator screenOptions={{ headerShown: false }}>
        <Stack.Screen name={PATHS.MENU} component={MenuScreen} />
        <Stack.Screen
          name={PATHS.GAME_SETUP}
          component={GameSetupScreen}
          options={{
            headerShown: true,
            header: () => <Header />,
          }}
        />
        <Stack.Screen name={PATHS.GAME} component={GameScreen} />
        <Stack.Screen name={PATHS.GAME_OVER} component={GameOverScreen} />
        <Stack.Screen name={PATHS.RULES} component={RulesScreen} />
        <Stack.Screen name={PATHS.SETTINGS} component={SettingsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default AppNavigator;

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент Menu.jsx

```

import React from 'react';
import {
  View, StyleSheet, BackHandler, Image,
} from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { useTranslation } from 'react-i18next';
import Button from '../../components/Button/Button';
import { PATHS } from '../../constants/paths';
import MenuJpg from '../../assets/images/menu.jpg';

export default function Menu() {
  const { navigate } = useNavigation();
  const { t } = useTranslation();
  const styles = StyleSheet.create({
    container: {
      flex: 1,
      backgroundColor: '#424242',
      alignItems: 'center',
      paddingHorizontal: 20,
      paddingVertical: 50,
    },
    image: {
      width: 200,
      height: 200,
      borderRadius: 10,
    },
  });

  return (
    <View style={styles.container}>
      <Image source={MenuJpg} style={styles.image} />
      <Button title={t('menuPage.startButton')} onPress={() =>
navigate(PATHS.GAME_SETUP)} />
      <Button title={t('menuPage.rulesButton')} />
      <Button title={t('menuPage.settingsButton')} />
      <Button title={t('menuPage.exitButton')} onPress={() =>
BackHandler.exitApp()} />
    </View>
  );
}

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент GameSetup.jsx

```

import React, { useState } from 'react';
import { GAME_SETUP_STEPS } from './consts';
import GameSetupPlayersCount from
'./subpages/GameSetupPlayersCount/GameSetupPlayersCount';
import GameSetupPlayersNames from
'./subpages/GameSetupPlayersName/GameSetupPlayersNames';
import GameSetupPlayersRoles from
'./subpages/GameSetupPlayersRoles/GameSetupPlayersRoles';

export default function GameSetup() {
  const [currentStep, setCurrentStep] = useState(GAME_SETUP_STEPS.PLAYERS_COUNT);

  switch (currentStep) {
    case GAME_SETUP_STEPS.PLAYERS_COUNT:
      return (
        <GameSetupPlayersCount
          nextStep={() => setCurrentStep(GAME_SETUP_STEPS.PLAYERS_NAMES)}
        />
      );
    case GAME_SETUP_STEPS.PLAYERS_NAMES:
      return (
        <GameSetupPlayersNames
          nextStep={() => setCurrentStep(GAME_SETUP_STEPS.PLAYERS_ROLES)}
        />
      );
    case GAME_SETUP_STEPS.PLAYERS_ROLES:
      return <GameSetupPlayersRoles />;
    default:
      return null;
  }
}

```

Компонент GameSetupPlayersCount.jsx

```

import React from 'react';
import { useDispatch } from 'react-redux';
import {
  FormProvider, useForm,
} from 'react-hook-form';
import { yupResolver } from '@hookform/resolvers/yup';
import {
  View, StyleSheet,
} from 'react-native';
import Button from '../../components/Button/Button';
import Input from '../../components/Input/Input';
import { setCountOfPlayers } from '../../store/Game/actions';
import { useGameSetupPlayersCountSchema } from './GameSetupPlayersCountConfig';
import { useTranslation } from 'react-i18next';

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		60

```

export default function GameSetupPlayersCount({ nextStep }) {
  const { t } = useTranslation();
  const dispatch = useDispatch();
  const schema = useGameSetupPlayersCOUNtSchema();

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      backgroundColor: '#424242',
      justifyContent: 'center',
      paddingHorizontal: 20,
      alignItems: 'center',
    },
  });

  const defaultValues = {
    playersCount: null,
  };

  const form = useForm({
    defaultValues,
    resolver: yupResolver(schema),
  });
  const { handleSubmit } = form;

  const goNext = handleSubmit((values) => {
    dispatch(setCountOfPlayers(values.playersCount));
    nextStep();
  });

  return (
    <FormProvider {...form}>
      <View style={styles.container}>
        <Input label={t('playersCountPage.label')} inputMode="numeric"
name="playersCount" />
        <Button title={t('common.next')} onPress={goNext} />
      </View>
    </FormProvider>
  );
}

```

Файл конфіг GameSetupPlayersCountConfid.js

```

import { useTranslation } from 'react-i18next';
import * as yup from 'yup';

export const useGameSetupPlayersCOUNtSchema = () => {
  const { t } = useTranslation();

  return yup

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    .object({
      playersCount: yup
        .number()
        .typeError(t('playersCountPage.errors.number'))
        .max(20, t('playersCountPage.errors.max'))
        .min(5, t('playersCountPage.errors.min'))
        .required(t('playersCountPage.errors.required')),
    })
    .required();
  });

```

Компонент GameSetupPlayersNames.jsx

```

import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import {
  View, FlatList, StyleSheet,
} from 'react-native';
import { yupResolver } from '@hookform/resolvers/yup';
import {
  FormProvider, useForm,
} from 'react-hook-form';
import Button from '../../components/Button/Button';
import Input from '../../components/Input/Input';
import { selectPlayers } from '../../store/Game/selectors';
import { changePlayers } from '../../store/Game/actions';
import { useGameSetupPlayersNameSchema } from './GameSetupPlayersNameConfig';
import { useTranslation } from 'react-i18next';

export default function GameSetupPlayersNames({ nextStep }) {
  const dispatch = useDispatch();
  const { t } = useTranslation();
  const players = useSelector(selectPlayers);
  const schema = useGameSetupPlayersNameSchema(players);

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      backgroundColor: '#424242',
      paddingHorizontal: 20,
      alignItems: 'center',
    },
  });

  const defaultValues = {
    playersCount: null,
  };

  const form = useForm({

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    defaultValues,
    resolver: yupResolver(schema),
  });
  const { handleSubmit } = form;

  const goNext = handleSubmit((values) => {
    const newPlayers = players.reduce((acc, player, index) => {
      acc[index].name = values[index + 1];
      return acc;
    }, players);
    dispatch(changePlayers(newPlayers));
    nextStep();
  });

  return (
    <FormProvider {...form}>
      <View style={styles.container}>
        <FlatList
          data={players}
          keyExtractor={({item, index}) => index.toString()}
          horizontal={false}
          style={{ width: '100%' }}
          renderItem={({ index }) => <Input label={t('playersNamePage.label', {
number: index + 1 })} name={`$${index + 1}`} />
          />
          <Button type="submit" title={t('common.next')} onPress={goNext} />
        </View>
      </FormProvider>
    );
  }

```

Файл конфіг GameSetupPlayersCountConfid.js

```

import { useTranslation } from 'react-i18next';
import * as yup from 'yup';

export const useGameSetupPlayersNameSchema = (players) => {
  const { t } = useTranslation();

  const schemaObject = players.reduce((acc, player, index) => {
    acc[index + 1] = yup
      .string()
      .required('Обов'язкове поле')
      .min(2, t('gameSetupPlayersName:min'))
      .max(20, t('gameSetupPlayersName:max'));
    return acc;
  }, {});

  return yup
    .object(schemaObject);
};

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

КОМПОНЕНТ GameSetupPlayersRoles.jsx

```
import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import {
  View, FlatList, StyleSheet, Text
} from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { Picker } from '@react-native-picker/picker';
import Button from '../../components/Button/Button';
import { PATHS } from '../../constants/paths';
import { selectPlayers } from '../../store/Game/selectors';
import { changePlayers } from '../../store/Game/actions';
import { useTranslation } from 'react-i18next';

export default function GameSetupPlayersRoles() {
  const dispatch = useDispatch();
  const { t } = useTranslation();
  const players = useSelector(selectPlayers);
  const [selectedValue, setSelectedValue] = useState(players);
  const { navigate } = useNavigation();

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      backgroundColor: '#424242',
      paddingHorizontal: 20,
      alignItems: 'center',
      width: '100%',
      paddingTop: 30,
    },
    text: {
      color: '#fff',
    },
    pickerItem: {
      color: '#fff',
      backgroundColor: '#424242',
      height: 100,
    },
    flatList: {
      width: '100%',
      paddingHorizontal: 38,
    },
  });

  const goNext = () => {
    dispatch(changePlayers(selectedValue));
    navigate(PATHS.GAME);
  };
}
```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return (
  <View style={styles.container}>
    <FlatList
      data={players}
      keyExtractor={({item, index}) => index.toString()}
      horizontal={false}
      style={styles.flatList}
      renderItem={({ index }) => (
        <View style={{ borderBottomWidth: 1, borderBottomColor: '#fff', width:
'100%', marginBottom: 10 }}>
          <Text style={styles.text}>{players[index].name}</Text>
          <Picker
            selectedValue={selectedValue[index].role}
            onChange={(itemValue) => {
              const newSelectedValue = [...selectedValue];
              newSelectedValue[index].role = itemValue;
              setSelectedValue(newSelectedValue);
            }}
            >
            <Picker.Item label={t('playersRolesPage.roles.seer')} value="seer"
style={styles.pickerItem} />
            <Picker.Item label={t('playersRolesPage.roles.protector')}
value="protector" style={styles.pickerItem} />
            <Picker.Item label={t('playersRolesPage.roles.mason')} value="masons"
style={styles.pickerItem} />
          </Picker>
        </View>
      )}
    />
    <Button title="Next" onPress={goNext} />
  </View>
);
}

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент Game.jsx

```

import React, { useState, useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import {
  View, FlatList, StyleSheet, DrawerLayoutAndroid, Text, Image, TextInput,
} from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { useTranslation } from 'react-i18next';
import Button from '../../components/Button/Button';
import { PATHS } from '../../constants/paths';
import CountdownTimer from './components/CountdownTimer';
import { selectPlayers, selectVotedPlayers } from '../../store/Game/selectors';
import {
  addPlayerOnVote, addVotes, removePlayer, resetVotedPlayers,
} from '../../store/Game/actions';
import ButtonImage from '../../components/ButtonImage/ButtonImage';
import playPng from '../../assets/images/play.png';
import dayPng from '../../assets/images/day.png';
import nightPng from '../../assets/images/night.png';
import seerPng from '../../assets/images/seer.png';
import masonPng from '../../assets/images/mason.png';
import protectorPng from '../../assets/images/protector.png';

export default function Game() {
  const dispatch = useDispatch();
  const { t } = useTranslation();
  const { navigate } = useNavigation();
  const players = useSelector(selectPlayers);
  const votedPlayers = useSelector(selectVotedPlayers);
  const [currentRound, setCurrentRound] = useState(0);
  const [currentTime, setCurrentTime] = useState('Night');
  const [isTimerPlaying, setIsTimerPlaying] = useState(false);
  const [currentPlayer, setCurrentPlayer] = useState(0);
  const [isVoting, setIsVoting] = useState(false);

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      backgroundColor: '#424242',
      justifyContent: 'center',
      paddingHorizontal: 20,
      color: '#fff',
    },
    input: {
      borderWidth: 1,
      borderColor: '#fff',
      borderRadius: 10,
      color: '#fff',
    },
  });

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        paddingHorizontal: 10,
      },
    });

const onActionsButtonPress = (playerIndex) => {
  if (currentTime === 'Night') {
    dispatch(removePlayer(players[playerIndex].name));
  } else {
    dispatch(addPlayerOnVote(players[playerIndex]));
  }
};

const getPlayerRole = (role) => {
  if (role === 'masons') return masonPng;
  if (role === 'protector') return protectorPng;
  return seerPng;
};

const navigationView = (
  <View style={{
    flex: 1, backgroundColor: '#424242', alignItems: 'center', justifyContent:
'center',
  }}
  >
    <FlatList
      data={players}
      keyExtractor={({item, index}) => index.toString()}
      horizontal={false}
      renderItem={({ index }) => (
        <View style={{ paddingVertical: 5, flexDirection: 'column' }}>
          <View style={{ flexDirection: 'row', alignItems: 'center' }}>
            <Image
              source={getPlayerRole(players[index].role)}
              style={{ width: 20, height: 20, marginRight: 10 }}
            />
            <Text style={{ color: '#fff' }}>
              {players[index].name}
            </Text>
          </View>
          <Button
            title={currentTime === 'Night' ? 'Kill' : 'On vote'}
            onPress={() => onActionsButtonPress(index)}
          />
        </View>
      )}
    />
  </View>
);

const checkIsGameOver = () => {
  if (players.filter((player) => player.role === 'masons').length === 0) {

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return true;
    }
    if (players.filter((player) => player.role === 'masons').length >=
(players.filter((player) => player.role === 'protector').length +
players.filter((player) => player.role === 'seer').length)) {
        return true;
    }

    return true;
};
const onComplete = () => {
    setIsTimerPlaying(false);

    if (checkIsGameOver()) {
        navigate(PATHS.GAME_OVER);
    } else if (currentTime === 'Night') {
        setCurrentTime('Day');
        setCurrentRound(currentRound + 1);
    } else if (isVoting) {
        dispatch(
            removePlayer(
                votedPlayers.reduce(
                    (prev, current) => ((prev.votes > current.votes) ? prev : current),
                ).name,
            ),
        );
        dispatch(resetVotedPlayers());
        setCurrentTime('Night');
        setIsVoting(false);
    } else if (currentPlayer === players.length - 1) {
        if (votedPlayers.length > 0) {
            setIsVoting(true);
        } else {
            setCurrentTime('Night');
        }
    } else {
        setCurrentPlayer(currentPlayer + 1);
    }

    return [true, 5];
};

useEffect(() => {
    if (currentTime === 'Day' && votedPlayers.length > 0) {
        dispatch(
            removePlayer(
                votedPlayers.reduce(
                    (prev, current) => ((prev.value > current.value) ? prev : current),
                ).name,
            ),
        );
    }
});

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
  }, []));

const handleSubmitEditing = (e, name) => {
  console.log(e.nativeEvent.text);
  dispatch(addVotes(name, +e.nativeEvent.text));
};

return (
  <DrawerLayoutAndroid
    drawerWidth={200}
    drawerPosition="left"
    renderNavigationView={() => navigationView}
    style={styles.container}
  >
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Image
        source={currentTime === 'Day' ? dayPng : nightPng}
        style={{
          width: 50, height: 50, position: 'absolute', right: 10, top: 10,
        }}
      />
      <Text style={{
        margin: 10, fontSize: 15, textAlign: 'right', color: '#fff',
      }}
      >
        {t('gamePage.currentRound', { currentRound })}
      </Text>
      {currentTime === 'Day' && (
        <Text style={{
          margin: 10, fontSize: 15, textAlign: 'right', color: '#fff',
        }}
        >
          {isVoting ? t('gamePage.voting') : t('gamePage.talking', { playerName:
players[currentPlayer].name })}
        </Text>
      )}
      <CountdownTimer isTimerPlaying={isTimerPlaying} onComplete={onComplete} />
      <ButtonImage image={playPng} onPress={() =>
setIsTimerPlaying(!isTimerPlaying)} />
      {votedPlayers && votedPlayers.length > 0 && (
        <Text style={{ color: '#fff' }}>{t('gamePage.onVote')}</Text>)}
      <FlatList
        data={votedPlayers}
        keyExtractor={({item, index}) => index.toString()}
        horizontal={false}
        renderItem={({ index }) => (
          <View style={{
            paddingVertical: 5, flexDirection: 'row', alignItems: 'center',
width: 150, justifyContent: 'space-between', marginTop: 20,
          }}

```

```

        >
        <Text style={{ color: '#fff' }}>
            {votedPlayers[index].name}
        </Text>
        {isVoting && (
            <TextInput
                style={styles.input}
            />
        )}
    </View>
  )}
/>
</View>
<View style={{
  position: 'absolute', width: 5, height: 100, backgroundColor: '#607D8B',
top: '40%',
}}
/>
</DrawerLayoutAndroid>
);
}

```

Компонент CountdownTimer.jsx

```

import React from 'react';
import { Text } from 'react-native';
import { CountdownCircleTimer } from 'react-native-countdown-circle-timer';

export default function CountdownTimer({ isTimerPlaying, onComplete }) {
  return (
    <CountdownCircleTimer
      key={isTimerPlaying}
      isPlaying={isTimerPlaying}
      duration={1}
      size={100}
      colors={['#00bfa5', '#ffea00', '#ff6d00', '#ff0000']}
      colorsTime={[60, 30, 15, 6]}
      onComplete={onComplete}
    >
      {(remainingTime) => <Text style={{ color: "#fff" }}>{remainingTime}</Text>}
    </CountdownCircleTimer>
  );
}

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент Button.jsx

```

import React from 'react';
import PropTypes from 'prop-types';
import {
  StyleSheet, Text, Pressable,
} from 'react-native';

export default function Button({ title, onPress }) {
  const styles = StyleSheet.create({
    button: {
      alignItems: 'center',
      paddingVertical: 10,
      backgroundColor: '#c62828',
      marginVertical: 20,
      width: '80%',
      borderRadius: 10,
      borderWidth: 1,
      borderColor: '#fff',
    },
    buttonText: {
      color: '#fff',
    },
  });

  return (
    <Pressable style={styles.button} onPress={onPress}>
      <Text style={styles.buttonText}>{title}</Text>
    </Pressable>
  );
}

Button.propTypes = {
  title: PropTypes.string.isRequired,
  onPress: PropTypes.func,
};

Button.defaultProps = {
  onPress: () => {},
};

```

Компонент ButtonImage.jsx

```

import React from 'react';
import PropTypes from 'prop-types';
import {
  StyleSheet, Text, Pressable, TouchableOpacity, Image,
} from 'react-native';

export default function ButtonImage({ image, onPress }) {

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const styles = StyleSheet.create({
  button: {
    marginVertical: 20,
  },
});

return (
  <TouchableOpacity onPress={onPress} style={styles.button}>
    <Image
      source={image}
      style={{ width: 30, height: 30 }}
    />
  </TouchableOpacity>
);
}

ButtonImage.propTypes = {
  title: PropTypes.string.isRequired,
  onPress: PropTypes.func,
};

ButtonImage.defaultProps = {
  onPress: () => {},
};

```

Компонент ErrorMessage.jsx

```

import React from 'react';
import PropTypes from 'prop-types';
import {
  StyleSheet, Text,
} from 'react-native';

export default function ErrorMessage({
  error,
}) {
  const styles = StyleSheet.create({
    errorMessage: {
      color: '#ffab91',
      width: '80%',
      fontSize: 11,
      marginTop: 5,
    },
  });

  return (
    <Text style={styles.errorMessage}>{error && error.message}</Text>
  );
}

ErrorMessage.propTypes = {

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```
label: PropTypes.string.isRequired,
};
```

Компонент Header.jsx

```
import React from 'react';
import {
  Image, StyleSheet, Text, View, ProgressBarAndroid,
} from 'react-native';
import StepIndicator from 'react-native-step-indicator';

function Header() {
  const styles = StyleSheet.create({
    container: {
      backgroundColor: '#424242',
      paddingTop: 20,
    },
  });

  const customStyles = {
    stepIndicatorSize: 25,
    currentStepIndicatorSize: 30,
    separatorStrokeWidth: 2,
    currentStepStrokeWidth: 3,
    stepStrokeCurrentColor: '#c62828',
    stepStrokeWidth: 3,
    stepStrokeFinishedColor: '#607D8B',
    stepStrokeUnFinishedColor: '#aaaaaa',
    separatorFinishedColor: '#607D8B',
    separatorUnFinishedColor: '#aaaaaa',
    stepIndicatorFinishedColor: '#ffffff',
    stepIndicatorUnFinishedColor: '#ffffff',
    stepIndicatorCurrentColor: '#ffffff',
    stepIndicatorLabelFontSize: 13,
    currentStepIndicatorLabelFontSize: 13,
    stepIndicatorLabelCurrentColor: '#c62828',
    stepIndicatorLabelFinishedColor: '#aaaaaa',
    stepIndicatorLabelUnFinishedColor: '#aaaaaa',
    labelColor: '#fff',
    labelSize: 13,
    currentStepLabelColor: '#c62828',
  };

  const labels = ['Кількість', 'Імена', 'Ролі'];

  return (
    <View style={styles.container}>
      <StepIndicator
        customStyles={customStyles}
        currentPosition={1}
        labels={labels}

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        stepCount={3}
      />
    </View>
  );
}

export default Header;

```

Компонент Input.jsx

```

import React from 'react';
import PropTypes from 'prop-types';
import {
  StyleSheet, Text, View, TextInput,
} from 'react-native';
import {
  Controller, FormProvider, useForm, useFormContext,
} from 'react-hook-form';
import ErrorMessage from '../ErrorMessage/ErrorMessage';

export default function Input({
  label, value, onChange, inputMode, defaultValue, onSubmitEditing, name,
}) {
  const { control, getValues } = useFormContext();
  const styles = StyleSheet.create({
    container: {
      paddingVertical: 5,
      width: '100%',
      paddingHorizontal: 38,
    },
    label: {
      marginBottom: 10,
      color: '#fff',
    },
    input: {
      borderWidth: 1,
      borderColor: '#fff',
      borderRadius: 10,
      color: '#fff',
      paddingHorizontal: 10,
    },
  });

  console.log(getValues('playersCount'));

  return (
    <View style={styles.container}>
      <Text style={styles.label}>{label}</Text>
      <Controller
        name={name}
        control={control}

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

render={({ field, fieldState: { error } }) => (
  <>
    <TextInput
      style={styles.input}
      onBlur={onSubmitEditing}
      onChangeText={field.onChange}
    />
    <ErrorMessage error={error} />
  </>
)}
/>
</View>
);
}

Input.propTypes = {
  label: PropTypes.string.isRequired,
};

```

		В.В. Галаган			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл paths.js

```
export const PATHS = {
  MENU: '/',
  GAME_SETUP: '/game-setup',
  GAME: '/game',
  GAME_OVER: '/game-over',
  RULES: '/rules',
  SETTINGS: '/settings',
};
```

Файл локалізації index.js

```
import i18next from 'i18next';
import { initReactI18next } from 'react-i18next';
import { LANGUAGES } from './consts';

import en from './locales/en.json';
import ua from './locales/ua.json';

const resources = {
  [LANGUAGES.en]: {
    translation: en,
  },
  [LANGUAGES.ua]: {
    translation: ua,
  },
};

i18next
  .use(initReactI18next)
  .init({
    resources,
    debug: true,
    fallbackLng: LANGUAGES.ua,
    react: {
      useSuspense: false,
    },
    interpolation: {
      escapeValue: false,
    },
    compatibilityJSON: 'v3',
  });
```