ДОДАТОК А

ГРАФІЧНИЙ МАТЕРІАЛ КВАЛІФІКАЦІЙНОЇ РОБОТИ

На рисунку А.1 зображено DFD діаграму потоку даних додатку.

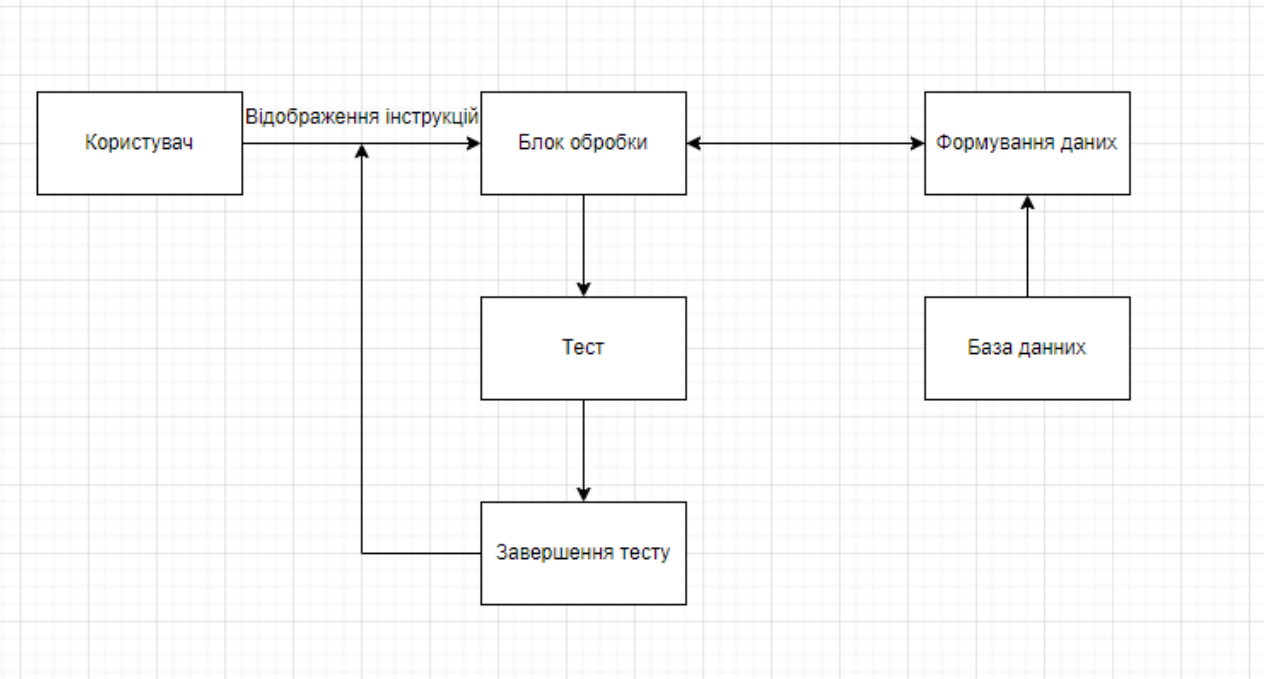


Рисунок А.1 - DFD діаграму потоку даних в додатку.

ДОДАТОК Б

ТЕКСТ ПРОГРАМИ

```
import React, { useState, useEffect } from 'react';
import { StyleSheet, Text, View, Image, TouchableOpacity, Animated, ScrollView }
from 'react-native';
import { Audio } from 'expo-av';

export default function App() {
  const [screen, setScreen] = useState('menu');
  const [countdown, setCountdown] = useState(10);
  const [phase, setPhase] = useState('start'); // start, exercise, rest,
finalRest, preExercise
  const [exerciseCount, setExerciseCount] = useState(0);
  const [fadeAnim] = useState(new Animated.Value(0)); // Initial opacity value for
the animated square
  const [squareFadeAnim1] = useState(new Animated.Value(0)); // Initial opacity
value for the first animated empty square
  const [squareFadeAnim2] = useState(new Animated.Value(0)); // Initial opacity
value for the second animated empty square
  const [progressAnim] = useState(new Animated.Value(1)); // Initial width value
for the progress bar

  const images = [
    require('./gifs/morg.gif'), // Перша вправа
    require('./gifs/krug.gif'), // Друга вправа
    require('./gifs/masage.gif'), // Третя вправа
    require('./gifs/masage2.gif'), // Четверта вправа
    require('./gifs/images.jpg') // П'ята вправа
  ];

  const backgroundColors = ['#F0FFF0', '#E6E6FA', '#FFFACD', '#FFE4E1',
'#E0FFFF'];

  useEffect(() => {
    let timer;
    let duration = phase === 'start' ? 10 : phase === 'exercise' ? 1 * 60 : 30;

    const playSound = async () => {
      const { sound } = await Audio.Sound.createAsync(
        require('./music/notification.mp3')
      );
      await sound.playAsync();
    };

    if (screen === 'exercise' && countdown > 0) {
      timer = setInterval(() => setCountdown(countdown - 1), 1000);
      Animated.timing(progressAnim, {
        toValue: countdown / duration,
        duration: 1000,
        useNativeDriver: false,
      }).start();
    } else if (screen === 'exercise' && countdown === 0) {
      playSound();
      if (phase === 'start') {
        setPhase('exercise');
        setCountdown(1 * 60);
```

```
        Animated.timing(fadeAnim, {
          toValue: 1,
          duration: 1000,
          useNativeDriver: true,
        }).start();
      } else if (phase === 'exercise') {
        setPhase('rest');
        setCountdown(30);
        fadeAnim.setValue(0); // Hide the square during rest phase
      } else if (phase === 'rest') {
        if (exerciseCount < 4) {
          setExerciseCount(exerciseCount + 1);
          setPhase('preExercise');
          squareFadeAnim1.setValue(0); // Reset the fade-in animation for the
first empty square
          squareFadeAnim2.setValue(0); // Reset the fade-in animation for the
second empty square
          Animated.timing(squareFadeAnim1, {
            toValue: 1,
            duration: 1000,
            useNativeDriver: true,
          }).start();
          Animated.timing(squareFadeAnim2, {
            toValue: 1,
            duration: 1000,
            useNativeDriver: true,
          }).start();
        } else {
          setPhase('finalRest');
          setCountdown(30);
        }
      } else if (phase === 'finalRest') {
        setScreen('congratulations');
      }
    }
    return () => clearInterval(timer);
  }, [screen, countdown, phase, exerciseCount]);

  const handleStart = () => {
    setScreen('exercise');
    setPhase('preExercise');
    Animated.timing(squareFadeAnim1, {
      toValue: 1,
      duration: 1000,
      useNativeDriver: true,
    }).start();
    Animated.timing(squareFadeAnim2, {
      toValue: 1,
      duration: 1000,
      useNativeDriver: true,
    }).start();
  };

  const handleContinue = () => {
```

```
    if (phase === 'preExercise') {
      setPhase('start');
      setCountdown(10);
      squareFadeAnim1.setValue(0);
      squareFadeAnim2.setValue(0);
    } else {
      setScreen('exercise');
      setPhase('exercise');
      setCountdown(1 * 60);
      setExerciseCount(0);
      fadeAnim.setValue(0);
      Animated.timing(fadeAnim, {
        toValue: 1,
        duration: 1000,
        useNativeDriver: true,
      }).start();
    }
  };

  const handleMenu = () => {
    setScreen('menu');
    setExerciseCount(0);
    fadeAnim.setValue(0);
    squareFadeAnim1.setValue(0);
    squareFadeAnim2.setValue(0);
  };

  const getPreExerciseContent = () => {
    if (exerciseCount === 0) {
      return (
        <>
          <Text style={styles.instruction}>Підготуйтеся до вправ</Text>
          <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim1
}]}>
            <Text style={styles.squareTextSmall}>Швидко моргайте протягом кількох
хвилин. Це допомагає зволожити очі та покращити кровообіг.</Text>
          </Animated.View>
          <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim2
}]}>
            <Text style={styles.squareText}>Моргання</Text>
          </Animated.View>
        </>
      );
    } else if (exerciseCount === 1) {
      return (
        <>
          <Text style={styles.instruction}>Підготуйтеся до вправ</Text>
          <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim1
}]}>
            <Text style={styles.squareTextSmall}>
              Виберіть об'єкт на відстані близько 30 см від очей.
              Сфокусуйтеся на ньому протягом 10-15 секунд.
              Потім переведіть погляд на віддалений об'єкт (близько 3 метрів) і
фокусуйтеся на ньому також 10-15 секунд.
```

```
            Повторіть кілька разів.
          </Text>
        </Animated.View>
        <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim2
}]}>
          <Text style={styles.squareText}>Кругові рухи очима</Text>
        </Animated.View>
      </>
    );
  } else if (exerciseCount === 2) {
    return (
      <>
        <Text style={styles.instruction}>Підготуйтеся до вправ</Text>
        <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim1
}]}>
          <Text style={styles.squareTextSmall}>Закрийте очі та акуратно
помасажуйте повіки круговими рухами кінчиками пальців.</Text>
        </Animated.View>
        <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim2
}]}>
          <Text style={styles.squareText}>Масаж повік</Text>
        </Animated.View>
      </>
    );
  } else if (exerciseCount === 3) {
    return (
      <>
        <Text style={styles.instruction}>Підготуйтеся до вправ</Text>
        <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim1
}]}>
          <Text style={styles.squareTextSmall}>Помістіть пальці на скроні та
робіть легкі натискання круговими рухами.</Text>
        </Animated.View>
        <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim2
}]}>
          <Text style={styles.squareText}>Масаж скронь</Text>
        </Animated.View>
      </>
    );
  } else if (exerciseCount === 4) {
    return (
      <>
        <Text style={styles.instruction}>Підготуйтеся до вправ</Text>
        <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim1
}]}>
          <Text style={styles.squareTextSmall}>Сядьте комфортно та закрийте очі.
Покладіть долоні на очі так, щоб вони не тиснули на повіки. Розслабтеся та
залишайтеся в цьому положенні, уявляючи собі темне та спокійне місце.</Text>
        </Animated.View>
        <Animated.View style={[styles.emptySquare, { opacity: squareFadeAnim2
}]}>
          <Text style={styles.squareText}>Пальминг</Text>
        </Animated.View>
      </>
```

```
      );
    }
    return null;
  };

  return (
    <View style={[styles.container, screen === 'exercise' && { backgroundColor:
backgroundColors[exerciseCount % 5] }]}>
      <View style={styles.topBar} />
      <ScrollView contentContainerStyle={styles.content}>
        {screen === 'menu' && (
          <>
            <Text style={styles.title}>Головне Меню</Text>
            <TouchableOpacity style={styles.startButton} onPress={handleStart}>
              <Text style={styles.startButtonText}>Вправи</Text>
            </TouchableOpacity>
          </>
        )}

        {screen === 'exercise' && (
          <>
            <TouchableOpacity style={styles.menuButton} onPress={handleMenu}>
              <Text style={styles.menuButtonText}>Вихід до головного меню</Text>
            </TouchableOpacity>
            {phase === 'preExercise' && (
              <>
                {getPreExerciseContent()}
                <TouchableOpacity style={styles.continueButton}
onPress={handleContinue}>
                  <Text style={styles.continueButtonText}>Продовжити</Text>
                </TouchableOpacity>
              </>
            )}
            {phase === 'start' && <Text style={styles.timer}>Початок через:
{countdown}</Text>}
            {phase === 'exercise' && (
              <>
                <Image source={ images[exerciseCount % 5] } style={styles.image}
/>
                <Text style={styles.timer}>Час на вправу: {countdown}</Text>
              </>
            )}
            {phase === 'rest' && (
              <>
                <Text style={styles.timer}>Молодець! Відпочинок:
{countdown}</Text>
              </>
            )}
            {phase === 'finalRest' && (
              <>
                <Text style={styles.timer}>Гарна робота! Відпочинок:
{countdown}</Text>
              </>
            )}
```

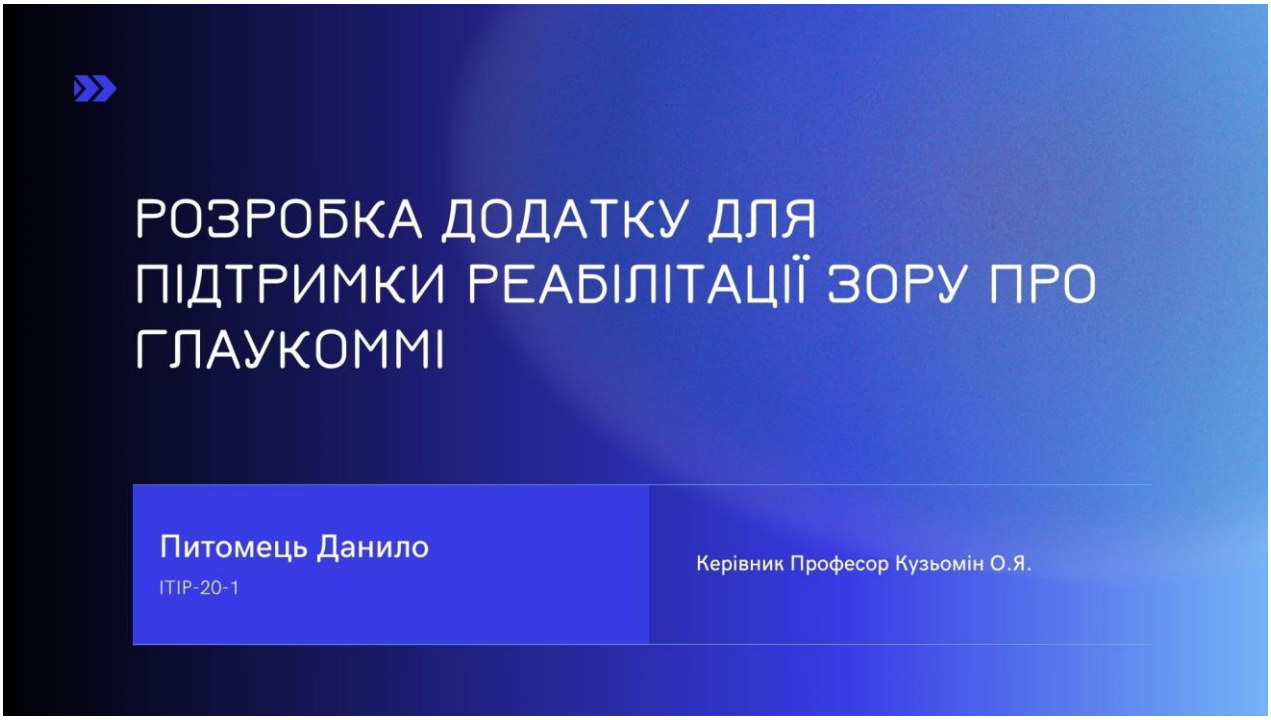
```
            {phase !== 'preExercise' && phase !== 'finalRest' && (
              <View style={styles.progressBarContainer}>
                <Animated.View style={[styles.progressBar, {
                  width: progressAnim.interpolate({
                    inputRange: [0, 1],
                    outputRange: ['0%', '100%'],
                  })
                }]} />
              </View>
            )}
          </>
        )}

        {screen === 'congratulations' && (
          <>
            <Text style={styles.congratulations}>Гарна робота!</Text>
            <TouchableOpacity style={styles.continueButton}
onPress={handleContinue}>
              <Text style={styles.continueButtonText}>Продовжити</Text>
            </TouchableOpacity>
            <TouchableOpacity style={styles.menuButton} onPress={handleMenu}>
              <Text style={styles.menuButtonText}>Вихід до головного меню</Text>
            </TouchableOpacity>
          </>
        )}
      </ScrollView>
      <View style={styles.bottomBar} />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
  },
  topBar: {
    height: 50,
    backgroundColor: '#4A90E2',
  },
  bottomBar: {
    height: 50,
    backgroundColor: '#4A90E2',
  },
  content: {
    flexGrow: 1,
    alignItems: 'center',
    justifyContent: 'center',
    padding: 20,
  },
  emptySquare: {
    width: 250,
    height: 100,
    backgroundColor: '#F3F3F3',
```

```
  margin: 10,
  justifyContent: 'center',
  alignItems: 'center',
  borderRadius: 10,
  padding: 10,
},
squareText: {
  fontSize: 18,
  fontWeight: 'bold',
  textAlign: 'center',
},
squareTextSmall: {
  fontSize: 14,
  textAlign: 'center',
},
title: {
  fontSize: 32,
  fontWeight: 'bold',
  color: '#333',
  marginBottom: 20,
},
instruction: {
  fontSize: 24,
  color: '#333',
  marginBottom: 20,
},
timer: {
  fontSize: 36,
  fontWeight: 'bold',
  color: '#4A90E2',
  marginBottom: 20,
},
image: {
  width: 150,
  height: 150,
  marginBottom: 20,
  borderRadius: 10,
},
menuButton: {
  position: 'absolute',
  top: 10,
  left: 10,
  backgroundColor: '#4A90E2',
  padding: 10,
  borderRadius: 5,
},
menuButtonText: {
  fontSize: 16,
  color: '#fff',
},
startButton: {
  backgroundColor: '#4A90E2',
  paddingVertical: 15,
  paddingHorizontal: 30,
```

```
    borderRadius: 5,
  },
  startButtonText: {
    fontSize: 18,
    color: '#fff',
  },
  continueButton: {
    backgroundColor: '#4A90E2',
    paddingVertical: 15,
    paddingHorizontal: 30,
    borderRadius: 5,
    marginTop: 20,
  },
  continueButtonText: {
    fontSize: 18,
    color: '#fff',
  },
  congratulations: {
    fontSize: 30,
    marginBottom: 20,
    color: 'green',
    textAlign: 'center',
  },
  progressBarContainer: {
    width: '80%',
    height: 20,
    backgroundColor: '#F3F3F3',
    borderRadius: 10,
    marginTop: 20,
  },
  progressBar: {
    height: '100%',
    backgroundColor: '#4A90E2',
    borderRadius: 10,
  },
});
```

Додаток В
(Рекомендований)

КОПІЇ ПРЕЗЕНТАЦІЇ



Вступ

Глаукома – це захворювання, яке пошкоджує зоровий нерв і спричиняє сліпоту. Понад 7600 мільйонів людей страждають на глаукому. Традиційні методи лікування лише уповільнюють прогресування. Що призвело до потреби підтримуючої терапії для відновлення зору.

Мобільні додатки пропонують нові можливості для реабілітації, включаючи візуальні вправи, відстеження прогресу та нагадування про ліки. Вони покращують зорові функції, незалежність і якість життя.

Програми візуальної реабілітації допомагають пацієнтам з глаукомою керувати станом і зберігати незалежність. Розробка таких додатків перспективна для майбутніх досліджень та інновацій.

Аналіз предметної області

1 Глаукома – це захворювання очей, що пошкоджує зоровий нерв і спричиняє сліпоту. Вона викликана підвищеним внутрішньоочним тиском, який прогресує і є незворотним. Глаукома вражає мільйони людей у світі. Традиційні методи лікування лише уповільнюють прогресування захворювання, але не відновлюють пошкоджений зір.

2

Рання діагностика глаукоми важлива, оскільки захворювання часто протікає безсимптомно до серйозного пошкодження зорового нерва. Регулярні огляди очей дозволяють виявити глаукому на ранніх стадіях і запобігти значній втраті зору. Раннє втручання та лікування, такі як медикаментозна терапія, лазерна терапія або хірургія, можуть уповільнити прогресування хвороби та зберегти залишковий зір пацієнта, підвищуючи якість його життя.

Аналіз існуючих аналогів

Glaucoma EyeCare

eyecareAmerica®
The Foundation of the American Academy of Ophthalmology

- Призначений для допомоги пацієнтам з глаукомою в контролі їх стану.
- Основні функції: нагадування про прийом ліків, моніторинг поля зору, освітній контент, керування призначеннями.
- Переваги: підвищує прихильність пацієнтів до прийому ліків, забезпечує доступ до освітніх ресурсів, допомагає відстежувати стан і готуватися до медичних консультацій

Аналіз існуючих аналогів

Vivid Vision

- Використовує віртуальну реальність для вправ, що покращують зорові функції.
- Особливості: вправи у віртуальній реальності, індивідуальні плани терапії, відстеження прогресу.
- Переваги: захоплююча та інтерактивна терапія зору, підвищення залученості пацієнтів, детальні звіти про прогрес

Вибір середовища розробки

JavaScript:

- Універсальна мова програмування, що використовується для веб-розробки.
- Підтримується всіма сучасними браузерами, що забезпечує широку доступність.
- Швидке виконання завдяки оптимізованим рушіям.

React Native:

- Дозволяє розробляти крос-платформні мобільні додатки для iOS та Android з використанням спільної кодової бази.
- Використання компонентів React забезпечує модульність і ефективність розробки.
- Гаряче перезавантаження прискорює процес розробки, дозволяючи бачити зміни майже миттєво.
- Міст для взаємодії між JavaScript і нативним кодом дозволяє використовувати специфічні для платформи можливості, такі як камера, GPS та сенсори

Постановка задачі
Основні цілі розробки додатку та очікуванні результати

Основні цілі розробки додатку – це створення інструменту для підтримки реабілітації зору у пацієнтів з глаукомою, забезпечення індивідуальної підтримки через візуальні вправи, відстеження прогресу, нагадування про прийом ліків .

Очікувані результати від розробки додатку включають покращення зорових функцій, підвищення незалежності пацієнтів, збільшення прихильності до реабілітаційних програм і загальне покращення якості життя користувачів .

Програмні та апаратні

Сумісність з операційними системами:

Додаток повинен бути сумісним з основними операційними системами, такими як iOS та Android, для задоволення потреб широкого кола мобільних користувачів. Він має бути розроблений з використанням надійних і масштабованих середовищ, таких як React Native, щоб забезпечити кросплатформенну сумісність і однаковий користувальницький інтерфейс на всіх пристроях .

Вимоги до апаратного забезпечення:

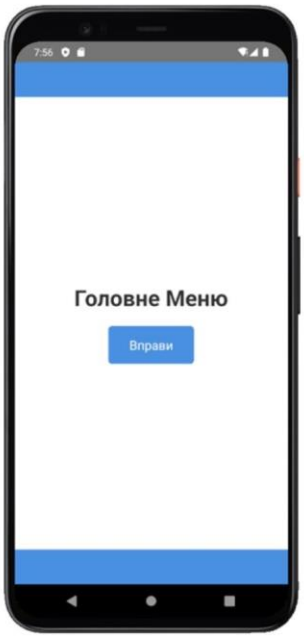
Для безперебійної роботи програми знадобиться смартфон або планшет з достатньою обчислювальною потужністю та пам'яттю. Пристрій повинен мати достатню пам'ять для обробки великих обсягів даних, що генеруються під час безперервного моніторингу та реабілітаційних вправ. Можливе використання спеціальних апаратних компонентів або адаптерів, таких як USB або Bluetooth, для зв'язку між пристроями і мобільними платформами .

Розробка інтерфейсу додатку

Принципи дизайну інтерфейсу:

Інтерфейс повинен бути доступним для людей із вадами зору, використовуючи великі шрифти, висококонтрастні кольорові схеми та голосову навігацію. Це допоможе користувачам зі слабким зором легко взаємодіяти з додатком .

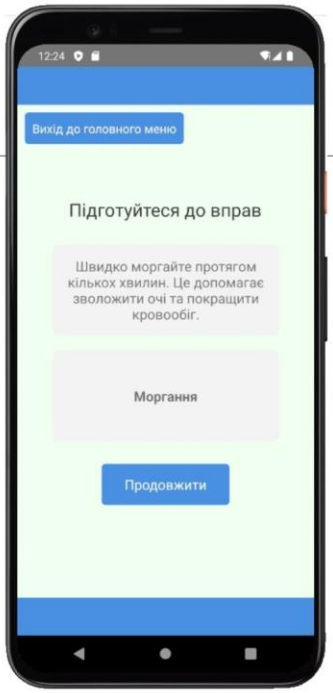
Інтерфейс додатку включає великі кнопки та чіткі інструкції для полегшення взаємодії. UX розроблений для заохочення регулярного використання додатку та містить елементи гейміфікації, такі як винагороди та відстеження прогресу, що підвищує залученість користувачів



Розробка функціональних вимог

Вправи для візуальної реабілітації:

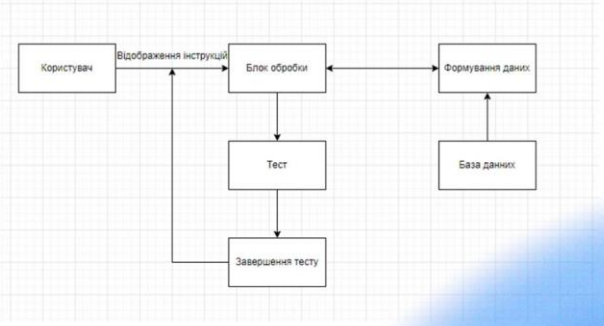
Вправи спрямовані на поліпшення залишкового зору пацієнтів. Вони мають бути інтерактивними і цікавими, фокусуючись на таких візуальних навичках, як периферичний зір, контрастна чутливість та швидкість візуальної обробки. Вправи адаптуються до різних рівнів порушення зору, що дозволяє користувачам поступово покращувати свої навички та відстежувати прогрес



Розробка моделі потоків даних

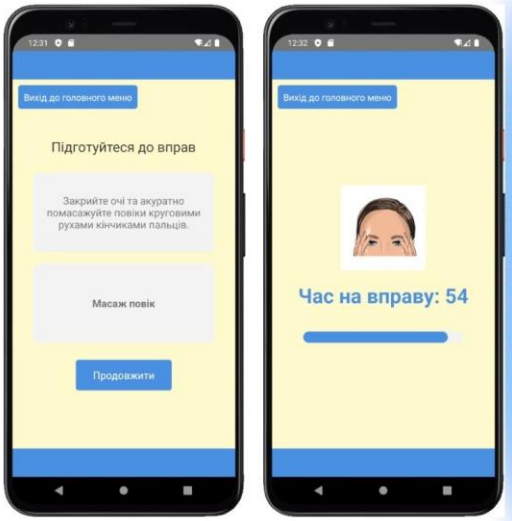
Концептуальна основа моделі:

Модель потоку даних програми складається з взаємопов'язаних процесів, які забезпечують підтримку реабілітації глаукоми. Включає взаємодію з користувачем, збір, обробку та зберігання даних.



Вправи та їх реалізація:

Вправи реалізовані через інтерфейс з анімацією, зворотним відліком та інструкціями. Кожна вправа має чіткий опис і керується таймером, який сигналізує про перехід між фазами. Інтерфейс також включає елементи гейміфікації, такі як прогрес-бар та звукові сигнали, що підвищують залученість користувачів та ефективність виконання вправ .



Висновки

Основні досягнення роботи

Розроблено інноваційний мобільний додаток для реабілітації зору у пацієнтів з глаукомою. Використання JavaScript та React Native забезпечує кросплатформенну сумісність і високу продуктивність. Інтерфейс адаптований для людей з вадами зору.

Перспективи розвитку

Вдосконалення додатку, впровадження нових функцій, інтеграція з іншими медичними пристроями та програмами. Проведення клінічних випробувань для підвищення ефективності реабілітації і покращення якості життя пацієнтів з глаукомою.

Додаток В

(Обов'язковий)

ВІДОМІСТЬ АТЕСТАЦІЙНОГО ПРОЕКТУ

№ п/п	Позначення	Найменування	Додаткові відомості
		Текстові документи	
		Пояснювальна записка	72 с.

Зм	Арк.	№ докум.	Подп.	Дата	ГЮІК.ХХХХХХ.000Д2		
Розроб.		Питомець			Розробка додатку для підтримки реабілітації зору при глаукомі Відомість кваліфікайійного проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Кузьомін				У	
Н. контр.		Бітченко				ХНУРЕ Кафедра РТІКС	
Затв.		Зарудний					