Capstone Project

Aug 29, 2023
Team Name: Fish Duo Cali
Team Logo:

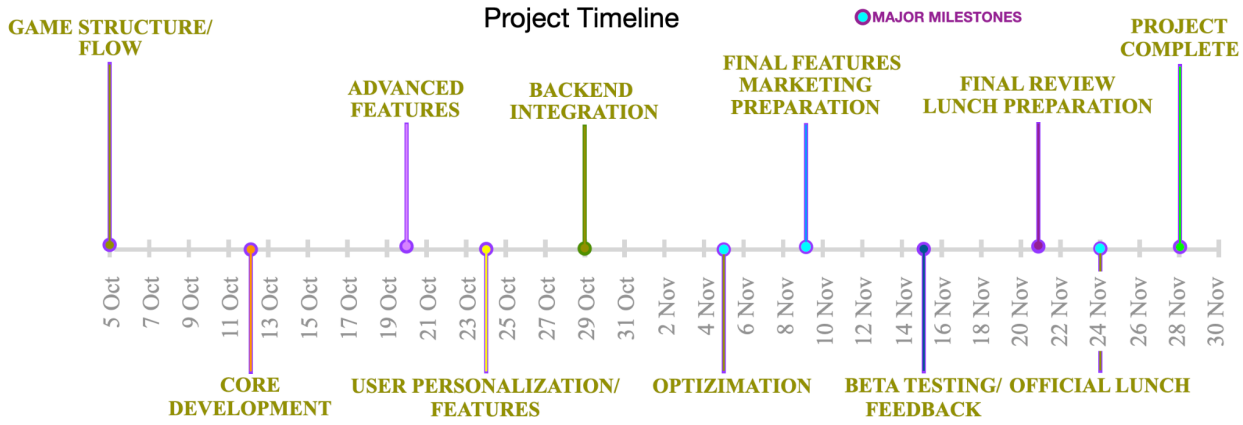
Students: Crystal Diaz , Laurand Osmeni

Game App Name: Oceanic Escape
Game Logo: In Progress….

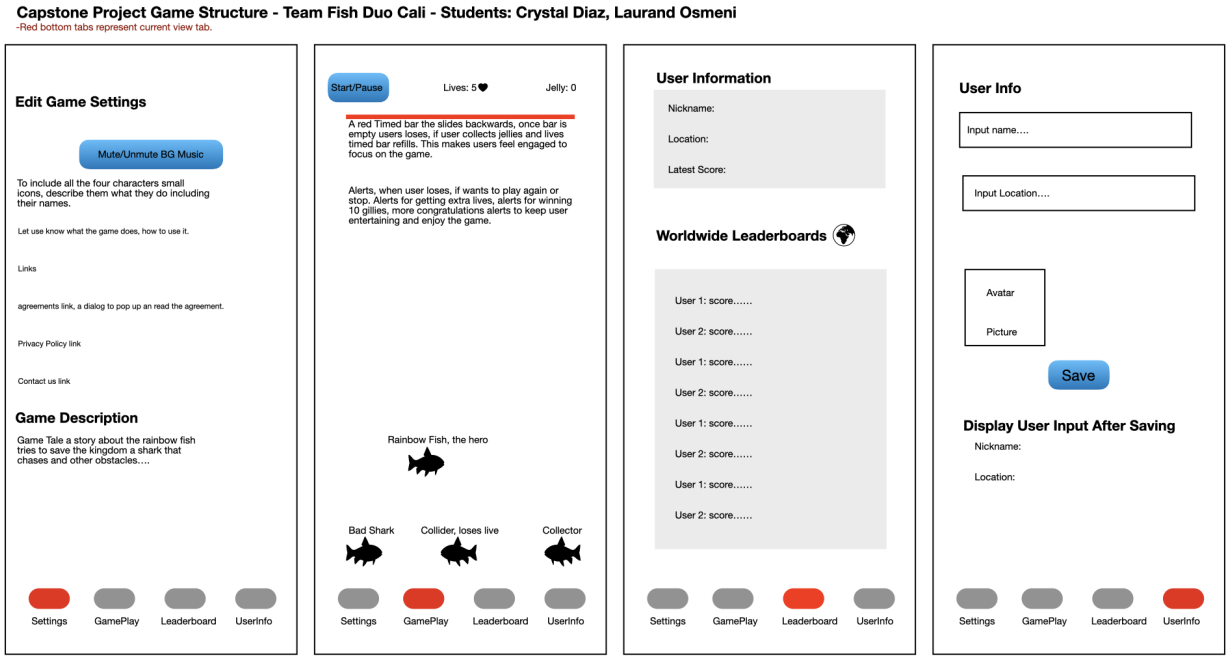
Communication: Slack, Google Document, Zoom to review final project.

Github Collaboration: https://github.com/Laurand/capstone_project

Timeline



Game Structure



Game Overview

A mobile game built using React Native, JSX Elements and Syntax, a popular framework for building mobile applications using TypeScript and React. Game designed for mobile devices iOS/Android. Game with obstacles, avoidance and coin collection game.
Will include four tabs. 1 Gameplay, 2 Settings, 3 User Information, 4 Leaderboards.

Front-end

User interface, game logic, and animations. Rendering game's characters, handling user input to move the character, collision detection, updating the score and lives.

Back-end

The game will use AsyncStorage to store the user's jellies, it involves data storage and retrieval. After testing using local storage, the back-end will involve server side operations, database interactions, that will update worldwide users and their leaderboard scores.

Gameplay View

The user will control Fish 1 and avoid Fish 4 and Fish 2 while collecting Fish 3 which is the jellies.
The player can move the Fish 1 left or right and can make it jump.
If the Fish 1 collides with a Fish 2 or Fish 4, the player loses a life and a bad sound is played.
If Fish 1 lands on top of a Fish 2 or Fish 4, the bad Fish is destroyed, the player earns a coin, and the Fish animation is restarted.
If Fish 1 collides with Fish 3, a good sound is played, and the player is rewarded with a coin.

Sounds

Three sounds, two playing character sounds and 1 background game music, one for good events collecting jellies or destroying Fishes and one for bad events colliding with Fishes.

Timer and Lives

The game includes a timer, and when it reaches zero, the game is paused, and the player is asked if they want to restart or end the game.
If the player collects a multiple of 10 jellies, the game is paused, and the player is awarded an extra life.

Game Over

When the player loses all their lives, a game over message is displayed, and the player is asked if they want to restart or stop the game.

Collections and User Data

The game keeps track of the jellies collected by the player.
The user's jellies are stored in async storage, so they persist even if the app is closed and reopened.
The user's nickname is displayed on the top during the game (Hello, nickname).
The game can be paused, resumed, or started by pressing a button.

Animations

Fish 1 moves from left to right.
The Fish 4 moves from right to left with a delay before it starts moving.
The Fish 3 moves from right to left with a delay before it starts moving and a delay after it reaches the end.
The Fish 1 can be moved left or right by dragging it and can be made to jump by dragging it up.

UserInfo View

UserInfo component allows the user to update their user information and avatar, and saves the updated information to the UserContext and AsyncStorage.

State Variables

It uses the useState hook to create state variables for nickname, location, and avatar. These are initialized with the current user's information retrieved from the UserContext.

Retrieving User Avatar

The useEffect hook is used to retrieve the user's avatar from AsyncStorage, a local storage solution for React Native when the component is first mounted.

Updating User Information

The saveUserInfo function updates the user information in the UserContext and AsyncStorage. It creates an updatedUser object with the current values of nickname, location, and avatar, then

updates the UserContext by calling setUser, updatedUser, and saves the updatedUser object and avatar to AsyncStorage.

Selecting an Avatar

The handleImageSelection function displays an action sheet with options to select an avatar from the library, take a new photo, or cancel. The launchPicker function is used to launch the image picker with the selected mode, library or camera and handle the response. If the user selects an image, the avatar state variable is updated with the selected image's URI.

Rendering

The component renders a form with TextInput fields for the nickname, age, and location, a button to select an avatar, and a button to save the updated user information. It also displays the current user information retrieved from the UserContext.

Leaderboard View

Displays a list of users and their scores in a leaderboard format.

Starting by importing necessary modules and components from react, react-native, @react-native-async-storage/async-storage, @react-navigation/native and a custom UserContext.

Interface Definition

It defines an interface UserScore which is a type for an object containing user information including name, stars, location, and avatar.

Component Definition

The Leaderboard functional component is then defined. It uses the useUser hook to get the current user's information, and useState to initialize two state variables: stars and leaderboard. The stars state is initially set to null and will later be updated with the current user's stars from the local storage.
The leaderboard state is initialized with a default list of users and their corresponding stars, locations, and avatars.

Fetching Stars

When the component is focused, when the user navigates to the screen where the component is rendered, it fetches the current user's stars from the local storage using AsyncStorage and sets it to the stars state.

Updating Leaderboard

When the star's state changes, the leaderboard is updated by adding the current user's information, if it's not already in the leaderboard or updating the current user's stars and location. The leaderboard is then sorted in descending order of stars and saved back to the local storage.

Rendering
The component then renders information about the current user including their nickname, location, and latest score.
The worldwide leaderboard which includes the avatar, name, location, and stars of each user in the leaderboard state.

Styling
Applies some styling to the component using the StyleSheet create method from react-native.

GameInfo Settings View
The GameInfo component can be used in a React Native application to allow the user to edit game settings related to background music and sound effects.

Importing necessary modules and components from react, react-native, and react-native-sound.

The GameInfo functional component is defined. It uses useState to initialize several state variables, as
bgSound, goodSound, badSound. These to be initialized to null and will later be updated with Sound objects for the background music, good sound effect, and bad sound effect, while playing the game, isBGSoundMuted, isGoodSoundMuted, isBadSoundMuted. These boolean variables are initialized to false and will be used to control whether the corresponding sounds are muted or not.

Loading Sounds
In the first useEffect hook, three Sound objects are created for the background music, good sound effect, and bad sound effect. The Sound objects are created using the Sound constructor from react-native-sound, which takes the file name of the sound, the sound type, in the case, Sound.MAIN_BUNDLE, and a callback function that is called when the sound is loaded. The background music is set to loop indefinitely, s.setNumberOfLoops(-1) and starts playing immediately (s.play()). The goodSound and badSound objects are simply created and stored in the state. The return function of the useEffect hook stops and releases all the sounds when the component is unmounted.

Pausing/Playing Background Music
In the second useEffect hook, the background music is paused or played based on the isBGSoundMuted state.
Playing Sound Effects: The playGoodSound and playBadSound functions play the good and bad sound effects, respectively, if they are not muted.

Rendering
The component then renders with a title Edit Game Settings.
Three switch components that allow the user to mute or unmute the background music, good sound, and bad sound.

Two buttons that allow the user to play the good and bad sound effects.

Getting started on Mac with React Native Framework app for iOS and Android
To get started with the React Native app, we installed dependencies that are offered to get started from the official website https://reactnative.dev/docs/environment-setup

In the Terminal type the line.
```
brew install node
```

Then command this line.
```
brew install watchman
```

In the Xcode installed Command Line Tools
Give the name of the app (name can be changed at anytime)
```
npm react-native@latest init CapstoneProject
```

Install and start npm
```
npm i
npm start
```
To run the app either run through the SDK or Terminal
```
npm run ios
npm run android
```

The app now is running by default successfully but with a simple hello world view.
React Native by default suggests using TypeScript file names.
To add more tabs in the app, in the terminal type the following line
```
npm install @react-navigation/native @react-navigation/bottom-tabs
```

In terminal this line as well
```
npm install @types/react-navigation
```

Then for iOS app only install pods
```
cd ios && pod install && cd
```

Customizing the appearance
https://reactnavigation.org/docs/tab-based-navigation/#customizing-the-appearance
Finally creating the tabs successfully but they come simple with text and without icons, so we need to install tabs icons using…
```
npm install react-native-vector-icons
```

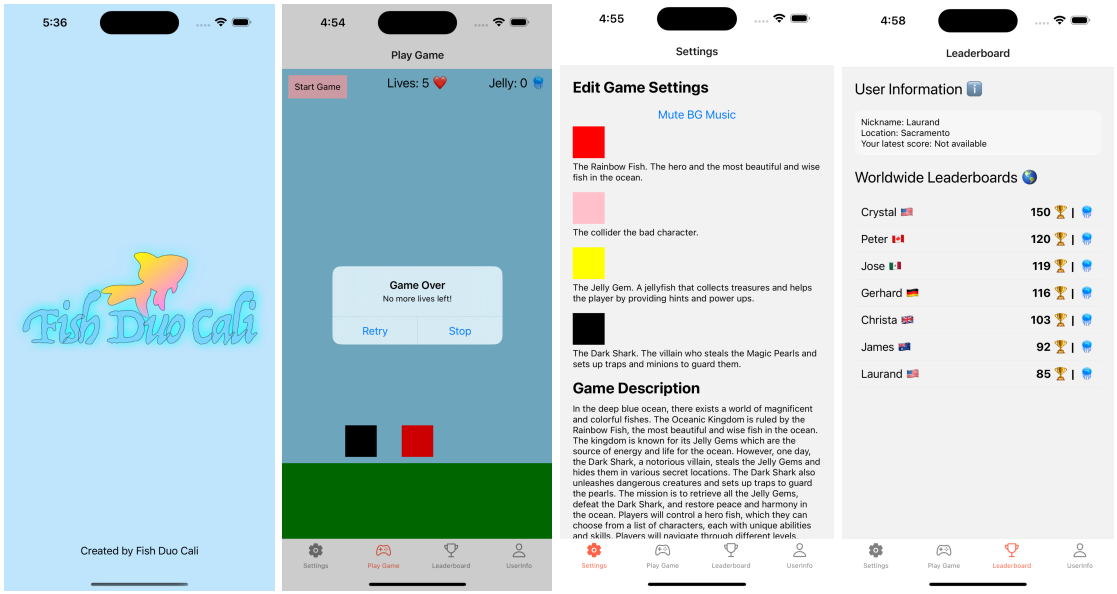
Installing async modules, this will help to local store the Leaderboard. For the worldwide board we need to create database servers. One way is to use Heroku and APIs, SQL databases to store users and new ones. (First we test the functionalities using local storage)
```
npm install @react-native-async-storage/async-storage
```

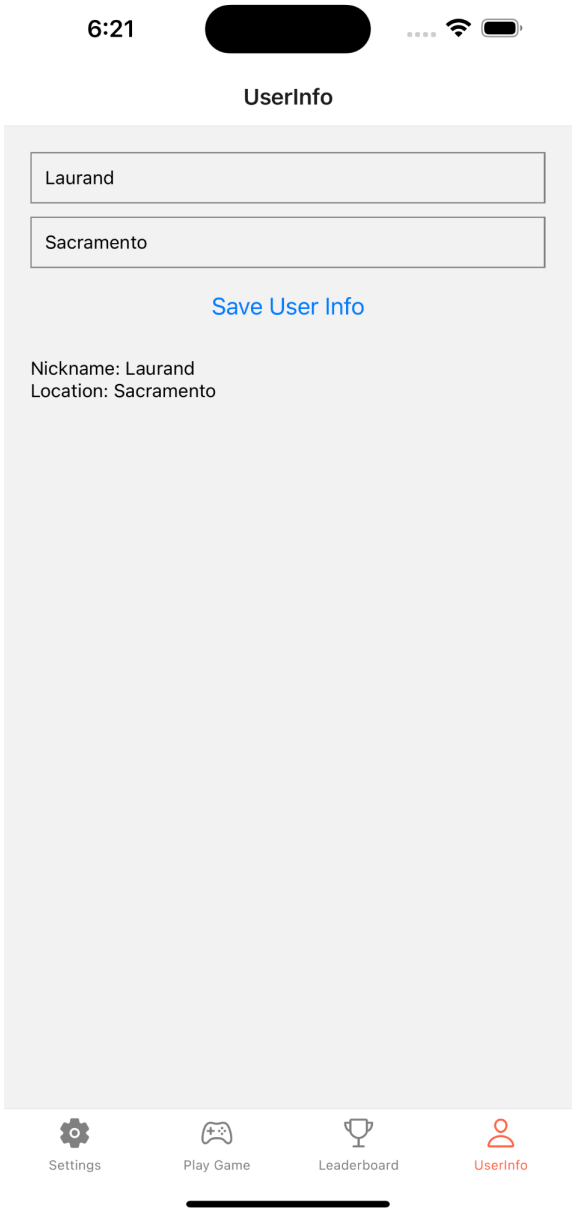
About the project

We are creating a mobile game using React Native, a framework built on React Facebook's JavaScript library for designing user interfaces. React targets browsers and React Native supports mobile platforms, making it a preferred choice for mobile app development. Our team uses React Native JSX Elements and Syntax providing necessary support to enhance and maintain our project.

With static typing, interface declarations, and other advanced features, TypeScript makes it easy to identify errors, code quality, and improves the overall development process. React Native is behind popular apps like Meta's applications Meta’s apps Facebook, Oculus, Messenger Desktop, Microsoft's Office and Skype, Shopify, WIX, and more apps as Walmart, NFL, Tesla, and Discord. More apps examples at https://reactnative.dev/showcase

Game Progress…




6:21
UserInfo
Laurand
Sacramento
Save User Info
Nickname: Laurand
Location: Sacramento
Settings
Play Game
Leaderboard
UserInfo