PROJECT REPORT

Project Title : MoodRythm – an weather based music player

Course No: CSE 3218

Submitted to:

Most. Kaniz Fatema Isha
Lecturer
Department of Computer Science and Engineering
Khulna University of Engineering & Technology (KUET)

Argha Chandra Dhar
Lecturer
Department of Computer Science and Engineering
Khulna University of Engineering & Technology (KUET)

Submitted by:

Mohammad Abtahe Alam
1907103
Soummo Bhattacharya
1907105
Md. Rakibul Hasan Adnan
1907106
Saugata Roy Arghya
1907116
Aciful Islam Khan Swopnile
1907119

Submission Date: 28 November 2023

1. Introduction:

1.1 Background Information:

MoodRythm is an iOS application designed to enhance the user's music listening experience by providing personalized music suggestions based on the current weather conditions. The app aims to create a seamless connection between the user's mood and the atmospheric ambiance, offering a unique and dynamic playlist tailored to the weather.

1.2 Objectives and Goals:

The primary objectives of the MoodRythm project are as follows:

- Develop an iOS app that integrates with weather data.
- Analyze weather conditions and correlate them with suitable music genres.
- Provide users with real-time music recommendations based on the weather.
- Enhance user engagement and satisfaction with a personalized music experience.

2. Methodology:

2.1 Detailed Description:

The project involves the following key components:

2.1.1 Firebase Authentication:

For first time user, first signup and data will be saved in firestore database and registration will be done by firebase authentication. Returning user can login using authenticated email and passwords.

For authentication, **LoginViewController** and **SignUpViewController** class is used to control the login and signup process.

- **ViewController Class:** ViewController that inherits from UIViewController. In the context of iOS development, a UIViewController manages the visual representation of a screen in an app.
- **ViewDidLoad Method:** The ViewDidLoad method is a lifecycle method in the view controller. It is called when the view controller's content view is loaded into memory. In this case, it calls the setUpElements method for additional setup.
- **signUpTapped Method:** The 'signUpTapped' method is triggered when a signup button is tapped. It calls 'validateFields' to check if the user-entered data is valid,it uses firebase authentication to create a new user and Firestore to store additional user data.
- **loginButtonTapped Method:** It validates the email and password text fields, and then uses Firebase authentication to sign in the user . The 'Auth.auth().signIn' method is called to authenticate the user with Firebase using the provided email and password.

2.1.2 Weather Integration:

- **Fetchdata Method:** After the authentication process, the user will go to the WeatherView page where an API is used to get the weather data from WeatherApi.com . For this, we have used a method called fetchData() .This method uses the 'URLSession' to make an asynchronous request to a weather API. It retrieves the JSON response, decodes it into a 'weatherdata' object . In this method, we collected the region,country,weather condition,current temp,current temp,current wind speed and current weather's image from the api and set these things in the UI.
- **RefreshData Method:** This method is triggered when 'Refresh Data Button' is tapped . It calls the 'fetchData' method to update the displayed weather information.
- **SuggestSongs Method:** This method is triggered when the 'Suggest Songs button' is tapped . It instantiates a 'HomeViewController', passes the weather data to it , and sets it as the root view controller. It uses the 'guard' statement to attempt to instantiate a 'HomeViewController' from the storyboard using the identifier "homeViewController" defined in 'Constants.Storyboard'. If successful, it assigns the 'fullWeatherData; to the 'receivedWeatherData' property of the 'homeViewController'.

On top of WeatherView page, there is a button named suggestSongs, which will take us to the PlayerView class, when clicked.

2.1.3 Music Recommendation:

After the suggest song button is pressed , a new view will be loaded, which will contain a music playlist based on the current weather data . User can play any of the music that is given in the playlist.

- **ConfigureSongs Methods:** configureSongs1(), configureSongs2(), configureSongs3() methods populate the ‘songs’ array based on different weather conditions(sunny, rainy and others). Each method adds ‘Song’ instances to the array.
- **Struct Songs:** A simple struct representing a song with various properties like song name, album name, artist name, image name and track name.
- **TableView Methods:**
    - **tableView(_:numberOfRowsInSection:) Method:** It specifies the number of rows in the table view, which is equal to the number of songs in the ‘songs’ array.
    - **tableView(_:cellForRowAt:) Method:** It is responsible for configuring and returning a cell for a given row in the table view. It dequeues a reusable cell with the identifier cell for the specified index path. It then retrieves the corresponding ‘Song’ object from the ‘songs’ array based on the index path. The cell’s properties (textLabel, detailTextLabel, imageView) are set based on the properties of the ‘Song’ object.

- **tableView(_:didSelectRowAt:) Method:** It is called when a row is selected in the table view. It first deselects the selected row with animation. It then creates an instance of 'PlayerViewController'('vc') from the storyboard with the identifier "player".The songs array and the selected position are set on the 'PlayerViewController'. Finally it presents the 'PlayerViewController' animatedly.

2.1.4 PlayerViewController: This class is used to control the view of the music suggestion page.

- **viewDidLayoutSubviews()**: It overrides the method called after the view's layout is updated. It also configures the UI elements if the holder (a **UIView**) has no subviews.
- **configure()**: This method sets up the player by initializing an **AVAudioPlayer** with the selected song. It also configures the UI elements such as album cover image, song name, album name, artist name labels, player controls (play/pause, next, previous), and volume slider.
- **didTapBackButtton()**: This method handles the tap action for the back button to play the previous song. It stops the current player, removes subviews, updates the position, and reconfigures the view for the new song.
- **didTapNextButtton()**: This method handles the tap action for the next button to play the next song. It also stops the current player, removes subviews, updates the position, and reconfigures the view for the new song.
- **didTapPlayPauseButttton()**: This method handles the play/pause functionality. It toggles between play and pause states for the audio

player. After that, it updates the play/pause button icon and adjusts the album image size accordingly.

- **didSliderSlider(_:)**: This method is used for handling changes in the volume slider value. It, also, adjusts the player's volume based on the slider value.
- **viewWillDisappear(_:)**: The method is used to overrides the method called when the view is about to disappear. It stops the audio player when the view disappears to ensure proper cleanup.

2.1.5 iOS Application:

- Implemented the app using Swift and Xcode.
- Created user-friendly interfaces for weather display and music recommendations.
- Integrated audio playback functionality for seamless music streaming.

3. Results:

Attached are several screenshots showcasing the MoodRythm app:

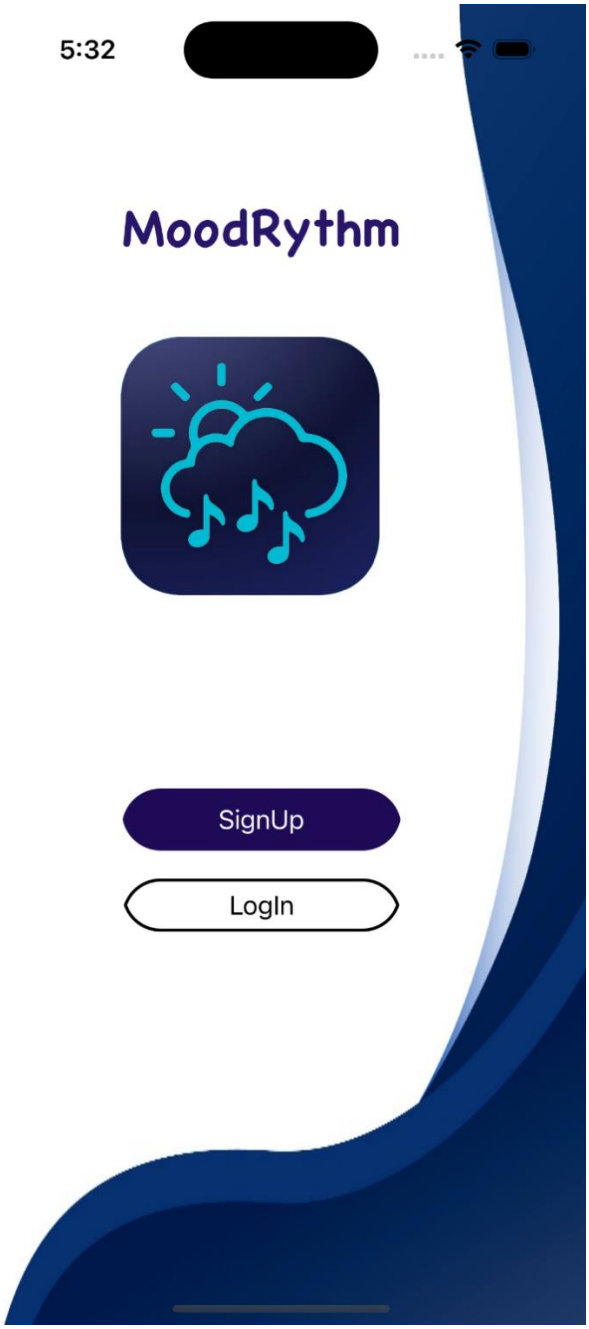

Figure 3.1: Start Screen

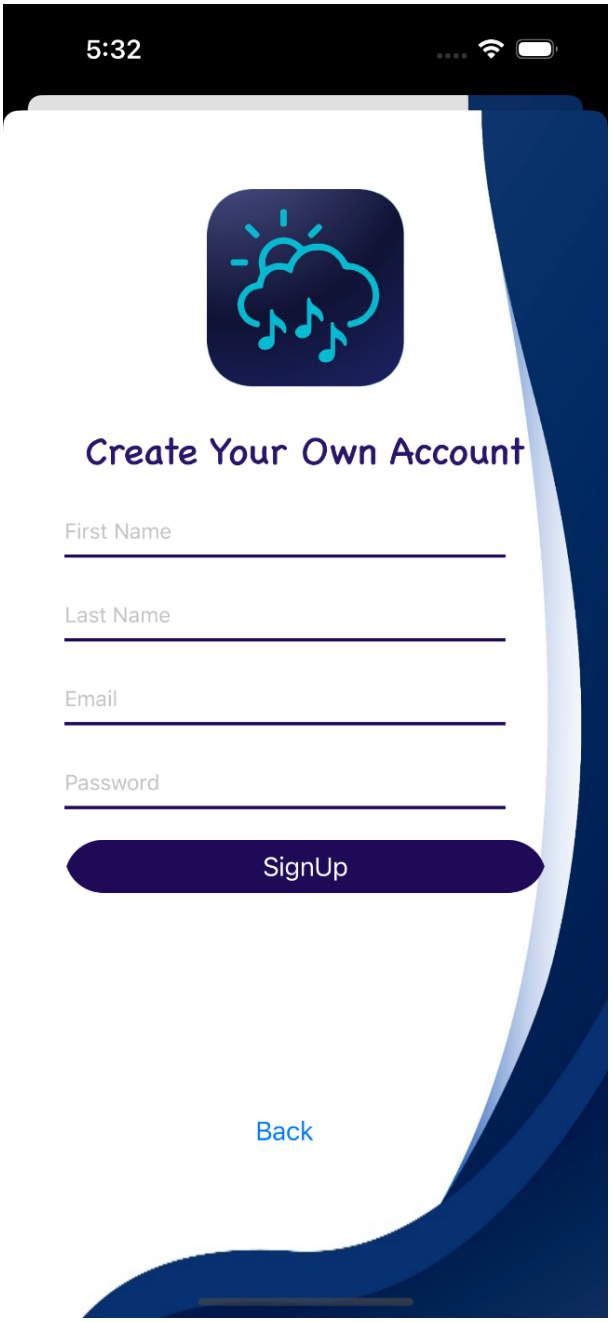

Figure 3.2: SignUp page

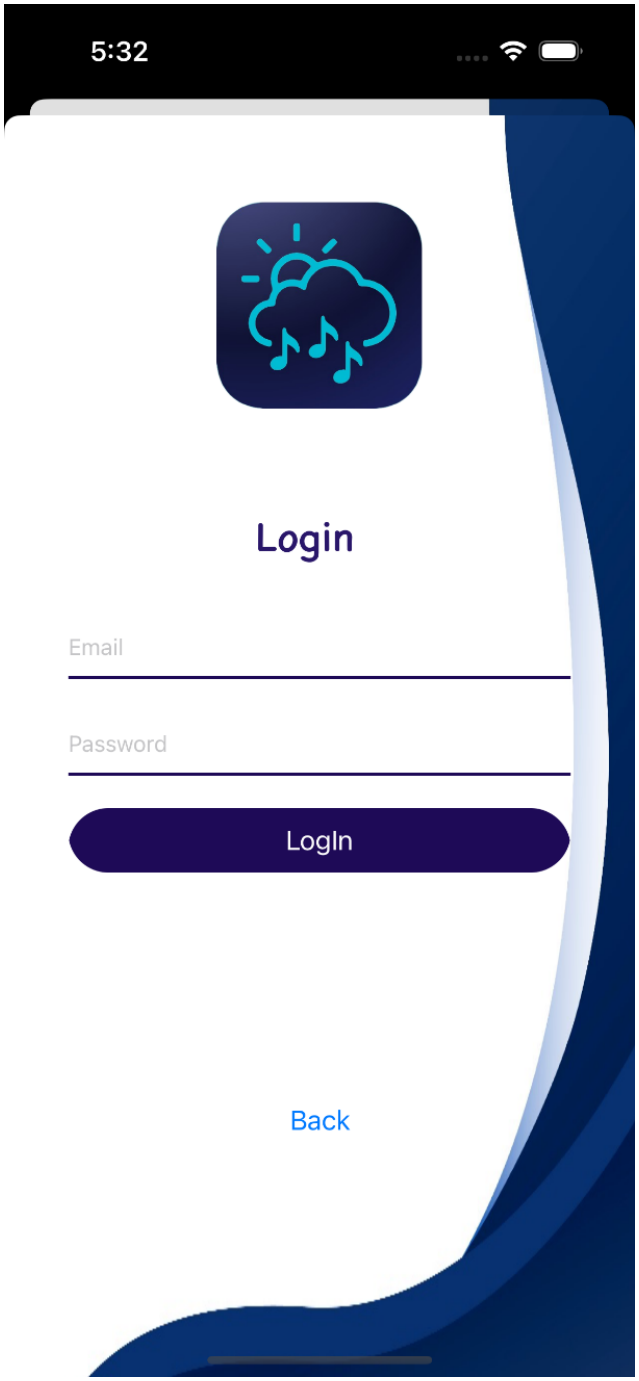

Figure 3.3: Login Page

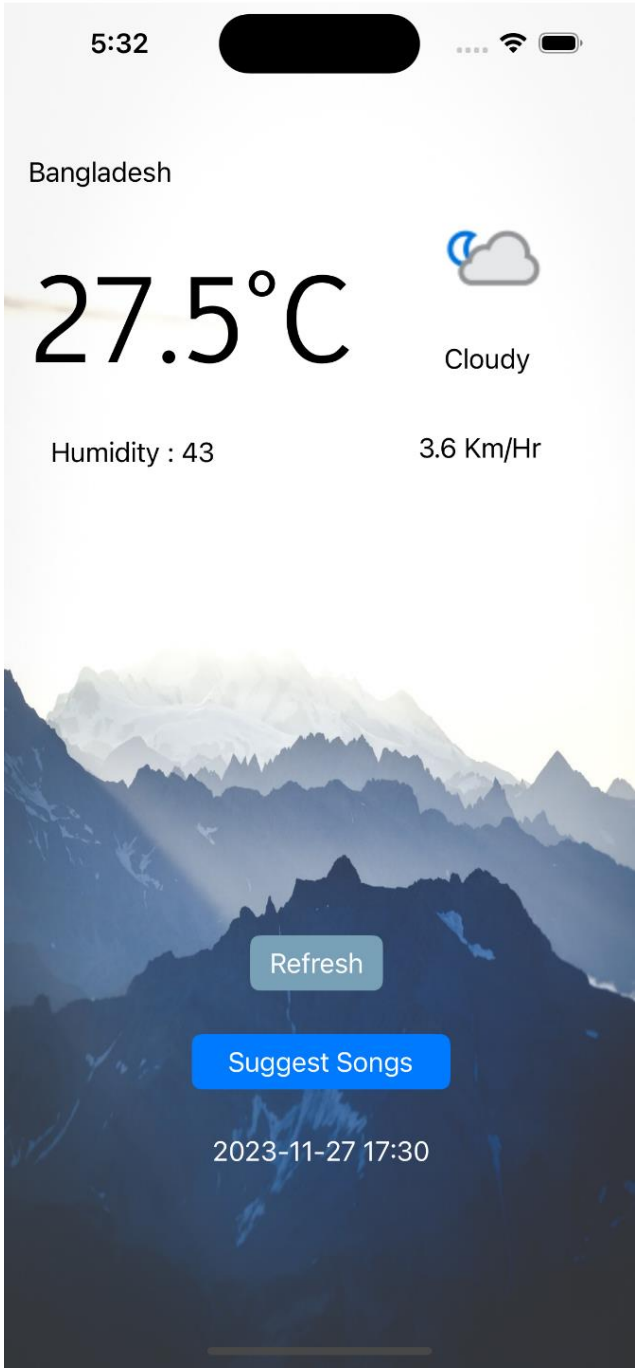

Figure 3.4: Home screen displaying current weather

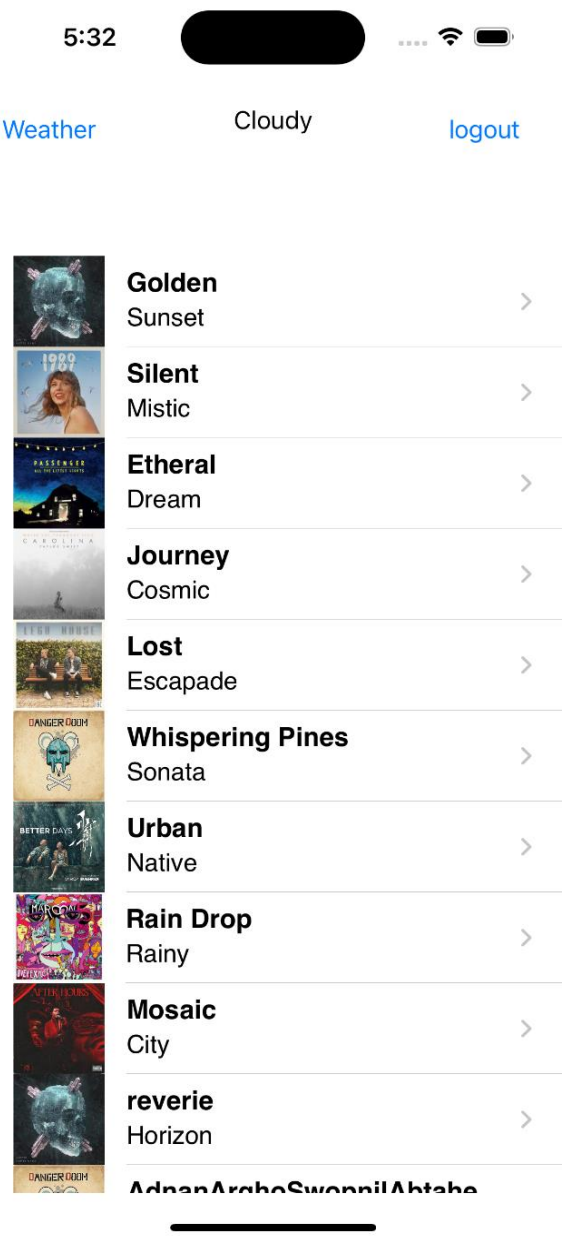

Figure 3.5: Music suggestion based on the weather.

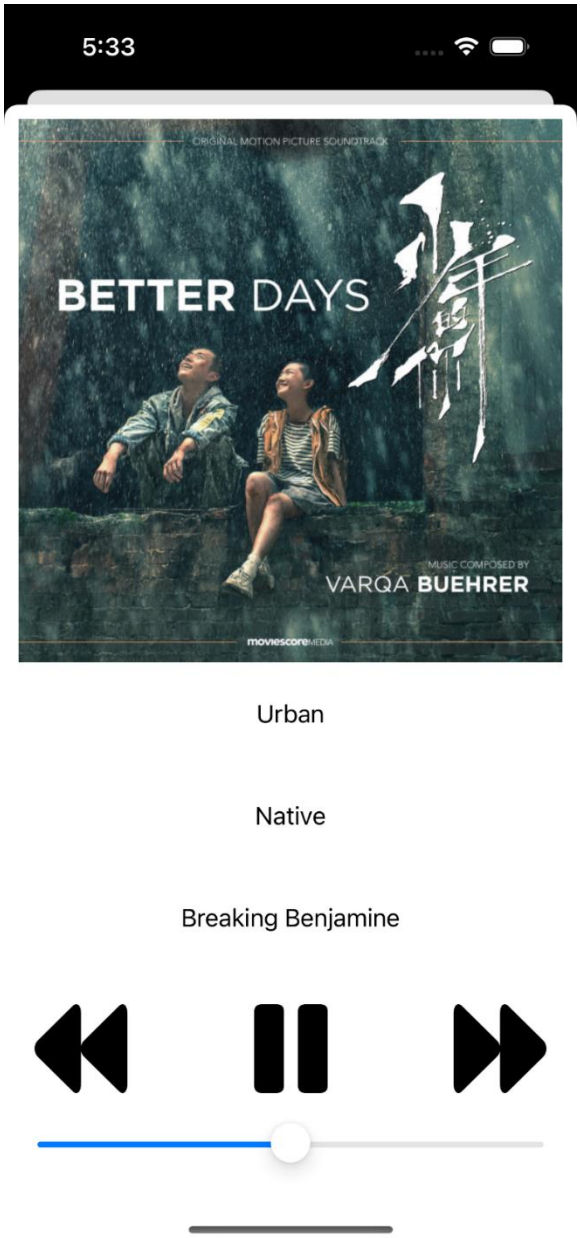

Figure 3.6: Music Player.

3.1 Functionalities:

3.1.1 Real-time Weather Display:

- Users can view current weather conditions, including temperature and weather description.

3.1.2 Music Recommendations:

- The app provides personalized music suggestions based on the analyzed weather data.

3.1.3 Playlist Creation:

- Users can create and save playlists based on their preferences and mood.

4. Discussion:

MoodRythms combines weather info with music suggestions, giving users a special and personal experience. The algorithm links weather and music vibes, making a lively playlist for users. The dynamic playlists adapt to the changing weather, offering users a refreshing and mood-appropriate selection of music. The easy-to-use interface keeps users interested. User engagement is elevated through a visually appealing interface, while ongoing refinement and attention to privacy underscore the app's commitment to continual improvement and user satisfaction.

5. Conclusion and Future Work:

The MoodRythm project has achieved its initial goals of providing personalized music recommendations based on weather conditions. Future work could include:

- Further refinement of the recommendation algorithm.
- Integration with additional external APIs for expanded data sources.
- Collaboration with music streaming services for a wider range of song options.
- Integration of Machine Learning algorithm which will track user's activities across device and give personalized music suggestions.

6. References:

1. Weather API Documentation: [https://www.weatherapi.com/docs/]
2. Swift Programming Language Documentation: [https://www.programiz.com/swift-programming]
3. Xcode Development Environment Documentation: [https://developer.apple.com/documentation/xcode]