Annexure 1

Project Report

on

Title of the Project

Submitted

In Partial Fulfillment of

MASTER OF COMPUTER APPLICATIONS (MCA)

Submitted by:

Himanshu

23/SCA/BCA/020

Under the Supervision of:

(Dr. Sachin Sharma, Professor)

School of Computer Applications

Manav Rachna International Institute of Research and Studies

(DEEMED TO BE UNIVERSITY)

Sector-43, Aravalli Hills

Faridabad – 121001

June 2025

Annexure 2

Declaration

I do hereby declare that this project work entitled “App Development ” submitted by me for the partial fulfillment of the requirement for the award of **BACHELOR OF COMPUTER APPLICATIONS** is a record of my own work. The report embodies the finding based on my study and observation and has not been submitted earlier for the award of any degree or diploma to any Institute or University.

SIGNATURE

Name: Himanshu

Roll No:23/SCA/BCA/020

Date: 18-7-2025

Annexure 3

Certificate from the Guide

This is to certify that the project report entitled “App Development” submitted in partial fulfillment of the degree of **BACHELOR OF COMPUTER APPLICATIONS** to Manav Rachna International Institute of Research and Studies, Faridabad is carried out by Mr. Prashant (Roll No), 23/SCA/BCA/020 under my guidance.

Signature of the Guide

Name: Dr. Sachin Sharma

Date:18-7-2025

Head of Department

Name: Dr. Suhail Javed Quarishi

Date: 18-7-2025

ACKNOWLEDGEMENT

I gratefully acknowledge for the assistance, cooperation, guidance and clarification provided by Mr.Piyush Nanwani during the development of App Development. My extreme gratitude to **Dr. Raj Kumar, Associate Professor & TPO** who guided us throughout the project. Without his willing disposition, spirit accommodation, frankness, timely clarification and above all faith in us, this project could not have been completed in due time. His readiness to discuss all important matters at work deserves special attention of.

I would like to extend my sincere gratitude to **Prof. (Dr.) Suhail Javed Quraishi – HOD, Prof. (Dr.) Rashmi Agrawal – Associate Dean and Prof. (Dr.) Brijesh Kumar – Dean** for their valuable teachings and advice. I want to thank all the department faculty members for their cooperation and support. I want to thank non-teaching staff of the department for their cooperation and support.

This opportunity is a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, to attain desired career objectives. I hope to continue cooperation with all of you in the future.

Introduction

I am a student of BACHELOR OF COMPUTER APPLICATIONS. I have completed 2nd year. My extreme gratitude to Dr. Raj Kumar who guided us throughout the project. My internship is all about App Development from LaunchED Global. This is about 8 weeks internship in which I have study related to design and develop a complete application.

About the Organization

LauchED Global is a renowned EdTech company that focuses on project-based learning through online internships and hands-on mentorship. It provides practical knowledge in full stack web development, data analytics, AI/ML, and other trending domains. Their platform empowers students to work on real-world problems and get industry-ready skills.

a. **Internship Objective**

The main objective of this internship was to give me **practical exposure** to the real-world software development cycle, including designing, building, and deploying a working application. I was expected to:

- Learn and implement **React Native** for cross-platform mobile development.
- Use **Firebase Authentication** and **Realtime Database** for user and data management.
- Build a functional, user-friendly mobile app from scratch.
- Gain insights into UI/UX principles and design consistency.
- Understand project documentation and source code organization.

a. **Internship Duration**

The internship was of **two months**, during which I dedicated time daily to work on this project. It consiste of:

- Week 1–2: Learning tools & setting up the project environment.
- Week 3–6: Application development (screens, authentication, Firebase integration).
- Week 7–8: Testing, refinement, and preparing documentation.

Under the guidance of my mentor and with regular evaluations, I was able to steadily improve and learn how to independently manage app features and bug fixes.

a. **Project Overview – Anime & TV Series Tracker**

The project is a solution where users can search, save, and track their progress of anime and TV series episodes. Key features include watchlist creation, rating, viewing detailed episode lists, and secure user authentication. The admin dashboard allows content management and user access control.

The **Anime & TV Series Tracker** is a mobile application designed for anime and TV series enthusiasts. It allows users to:

- Register or log in to the app using email and password.
- Browse **popular anime titles** using dummy data and real Firebase database entries.
- Add custom anime entries with metadata like title, image, genres, rank, rating, description, season, and author.
- View data in organized, scrollable sections.
- Navigate through the app using a **Bottom Tab Navigator** with screens like Home, Discover, Seasonal, Discussion, and MyList.

Key Technologies Used:

- **React Native** – For building a native mobile experience.
- **Firebase Auth** – For secure user login and registration.
- **Firebase Realtime Database** – To read/write anime entries.
- **AsyncStorage** – For storing login sessions.

- **React Navigation** – For managing screen transitions.

Technologies Used

This section outlines the various technologies, tools, and platforms utilized during the development of the **Anime & TV Series Tracker** project. Each component was selected for its ease of integration, community support, and relevance to mobile app development.

Frontend Technologies

The **frontend** is the part of the application that users interact with. For this project, the frontend was developed using the **React Native framework**, which enables cross-platform mobile app development using JavaScript.

a. React Native

- **Why Used**: React Native was chosen for its capability to build apps for both Android and iOS platforms using a single codebase.
- **Key Features Used**:
  - Components such as View, Text, TextInput, ScrollView, FlatList, Button, TouchableOpacity, etc.
  - Custom styles using StyleSheet API.
  - Navigation between screens using React Navigation.

b. React Navigation

- **Library Used**: @react-navigation/native and @react-navigation/native-stack
- **Purpose**: Used to implement the stack and bottom tab navigators.
- **Usage in Project**:
  - Navigating between Login, Register, and main tab screens.
  - Bottom Tab Navigator for Home, Discover, Discussion, Seasonal, and MyList sections.

c. Expo Icons

- **Library**: @expo/vector-icons/MaterialCommunityIcons
- **Purpose**: To render attractive icons on the bottom tab navigation bar.

Backend Technologies

The backend in this project was managed using **Firebase**, a cloud-based Backend-as-a-Service (BaaS) platform provided by Google.

a. Firebase Authentication

- **Purpose**: To handle user registration, login, and authentication sessions.
- **Firebase Methods Used**:
  - createUserWithEmailAndPassword()
  - signInWithEmailAndPassword()

- onAuthStateChanged() to track login state
- signOut() for logout functionality

b. Firebase Realtime Database

- **Purpose**: To store and retrieve anime data submitted by users.
- **Structure**: Data is stored in JSON format under the anime/ path.
- **Firebase Methods Used**:
- ref() and push() to write new entries.
- onValue() and get() to read data in real-time.

1. Database

The database used in the project was the **Firebase Realtime Database**.

Firebase Realtime Database

- **Type**: NoSQL, cloud-hosted JSON tree structure.
- **Benefits**:
- Live syncing of data across devices.
- Simple structure, suitable for small- to medium-sized mobile apps.
- **Example Data**:

json

CopyEdit

```
{
  "anime": {
    "unique_id_1": {
      "title": "Naruto",
      "rating": 9.2,
      "genre": ["Action", "Adventure"]
    },
    ...
  }
}
```

2. Tools & Platforms

To ensure smooth development and testing, several tools and platforms were used.

a. Visual Studio Code (VS Code)

- **Purpose**: Primary code editor.
- **Benefits**:
- Support for ESLint, Prettier, Git, and React Native extensions.

- Integrated terminal for running and debugging.

b. Expo Go / Android Emulator

- **Expo Go**: Used for quickly testing the app on mobile.
- **Android Studio Emulator**: Used to run virtual devices for testing Android builds.

c. Firebase Console

- **Usage**: Managing database rules, authentication, and monitoring data structure.

d. GitHub

- **Purpose**: Version control and code backup.

e. LaunchED Global Resources

- The organization provided mentorship, resources, and guidance throughout the project. My mentor, **Mr. Piyush Nanwani**, helped in the setup and debugging phases.

3. System Study

This section presents an overview of the **existing systems**, identifies their limitations, and introduces the **proposed solution**—the Anime & TV Series Tracker app—highlighting its benefits and improvements over previous methods.

3.1 Existing System and Limitations

a. Manual Tracking or Spreadsheets

Before using dedicated applications, many anime and TV series enthusiasts relied on:

- Manual note-taking or diaries
- Excel or Google Sheets to record watched episodes
- Browser bookmarks and YouTube watchlists

Limitations:

- **No Notifications or Recommendations**: Users cannot get suggestions for trending or upcoming anime.
- **No Visual Data**: Lists are plain text without any visuals or cover images.
- **No Sharing or Discussion**: Users cannot interact with communities or share lists.
- **Difficult to Update**: Updating an Excel file on mobile is time-consuming.
- **No Real-Time Access**: Data is not synced across devices automatically.
- **No Centralized Storage**: No cloud support, risking data loss if local storage fails.

b. Web-Based Anime Trackers (e.g., MyAnimeList)

Some users do use platforms like **MyAnimeList**, **AniList**, or **IMDb** for tracking anime or shows.

Limitations:

- **Complex UI**: These platforms can be overwhelming for beginners.
- **Too Many Features**: Many unnecessary features for someone just wanting a clean tracking experience.
- **Web-first Design**: These platforms often offer suboptimal mobile app experiences.
- **Less Customization**: Cannot add personal fields like favorite author or exact airing date format preferences.
- **Requires Internet at All Times**: Limited offline functionality.

3.2 Proposed System and Benefits

Project Name: Anime & TV Series Tracker

Platform: Android Mobile App (built using React Native & Firebase)

The proposed system is a mobile application that allows users to:

- Register/Login securely
- Add anime/show entries with details like title, genre, rating, image, etc.
- Store and retrieve data from Firebase Realtime Database
- View most popular anime in a visually appealing format
- Navigate easily using bottom tab navigation

Key Benefits of Proposed System

Feature	Benefit
Firebase Integration	Real-time data sync across devices
Authentication	Secure login and registration
Clean UI (React Native)	Easy to use, mobile-optimized interface
Customizable Data Entry	Users can add multiple images, genres, ratings, etc.
Bottom Tab Navigation	Smooth user experience and easy access to sections
Popular Anime Section	Pre-loaded trending anime list for quick access
Cloud Storage	Safe and reliable backend using Firebase
Offline Code Access	App logic works offline and syncs when connected

User Benefits:

- Seamless anime tracking experience without overwhelming UI
- Personalized entry creation (title, description, season, author)
- Quick discovery of new content via Discover section
- Flexibility to edit and submit new data anytime

4. Feasibility Study

A feasibility study is conducted to analyze the practicality of developing the *Anime & TV Series Tracker* application. It ensures the project is viable in terms of technology, cost, and user acceptance. This section discusses the three major aspects of feasibility:

4.1 Technical Feasibility

Technical feasibility evaluates whether the project can be developed using available hardware, software, and development tools.

Key Technologies Used:

- **Frontend:** React Native (cross-platform mobile development)
- **Backend:** Firebase Authentication and Realtime Database
- **Development Tools:** Visual Studio Code, Expo CLI
- **Version Control:** Git/GitHub

Hardware Requirements:

- Android smartphone for testing
- Windows PC with minimum 4GB RAM for development

Feasibility Points:

- React Native allows rapid development and testing on both Android and iOS.
- Firebase offers a serverless backend, removing the need to manage complex servers.
- No high-end hardware required—app runs smoothly on mid-range smartphones.
- Easy integration between front-end and Firebase database using provided SDKs.

Conclusion: The project is technically feasible with the current infrastructure and tools available during the internship.

4.2 Economic Feasibility

Economic feasibility involves evaluating the cost-effectiveness of the project—whether the benefits outweigh the expenses.

Cost Breakdown:

Resource	Cost
Firebase Free Tier	₹0
React Native (Open Source)	₹0
Development Laptop	Existing
Internet Connection	Already Available
Time Invested	2 Months (Internship)
IDE (VS Code)	Free

Savings & Benefits:

- No need for external hosting or paid databases
- No licensing costs—entirely open-source
- Future potential for monetization through ads or premium features

Conclusion: The project is **highly economical**, with **zero infrastructure cost**, making it financially viable, especially for students or startups.

4.3 Behavioral Feasibility

Behavioral feasibility focuses on how well the target users (anime lovers, casual viewers, and binge-watchers) will accept and adopt the new system.

Target Users:

- College students
- Anime fans
- Users who track or manage TV show progress

User Expectations:

- Simple and clean UI
- Personalized tracking
- Access to trending shows

- Quick login and security

Acceptance Factors:

- The app includes features most anime fans want: genre filters, ratings, popularity tags, etc.
- Easy login/registration through Firebase Auth
- Consistent and intuitive tab-based navigation
- Light-weight and responsive design for mobile use

5. Requirement Specification

The requirement specification phase defines what the system must do (functional) and how well it must perform those tasks (non-functional). These specifications guide the development and ensure the final system aligns with user expectations.

5.1 Functional Requirements

Functional requirements define the **core operations and features** of the system.

1. User Authentication

- Users should be able to **register** with email, password, and birthday.
- Users should be able to **log in** securely with their credentials.
- Firebase Authentication will handle all auth logic securely.

2. Home Screen Functionality

- Users can **submit anime data** (title, genre, rating, etc.) through a form.
- Form validations are performed before submission.
- Data is stored in Firebase Realtime Database.

3. Discover Screen

- Displays **popular anime** using both hardcoded data and Firebase-fetched entries.
- Uses FlatList for horizontal scrolling.
- Dynamically updates when new data is retrieved from Firebase.

4. Discussion Screen

- Currently serves as a placeholder for community interaction.
- Users may be redirected or navigated to other screens from here.

5. Seasonal Screen

- Placeholder for seasonal anime features in the future.
- Structured for easy extension and scalability.

6. My List Screen

- Placeholder for saved or tracked anime by the user.
- Will later support adding/removing personal favorites.

7. Logout Functionality

- Authenticated users can log out using a button.
- Upon logout, users are navigated back to the login screen.

5.2 Non-Functional Requirements

Non-functional requirements define the **quality attributes** of the system.

1. Usability

- The app uses a clean and user-friendly interface.
- Tab-based navigation helps users switch between major sections easily.
- Icons from MaterialCommunityIcons enhance understanding.

2. Performance

- Fast screen loading and smooth UI interactions due to React Native optimization.
- Data read/write from Firebase occurs in near real-time.

3. Security

- User credentials are secured using Firebase Authentication.
- Firebase provides built-in protection against unauthorized access and abuse.
- Input validations minimize bad data submission.

4. Availability

- Firebase ensures high availability of backend services.
- Offline capabilities can be considered in future upgrades.

5. Scalability

- App architecture is modular (separate screens and components).
- Firebase Realtime Database allows for scalable data storage and querying.
- Future features like profile, favorites, and notifications can be added easily.

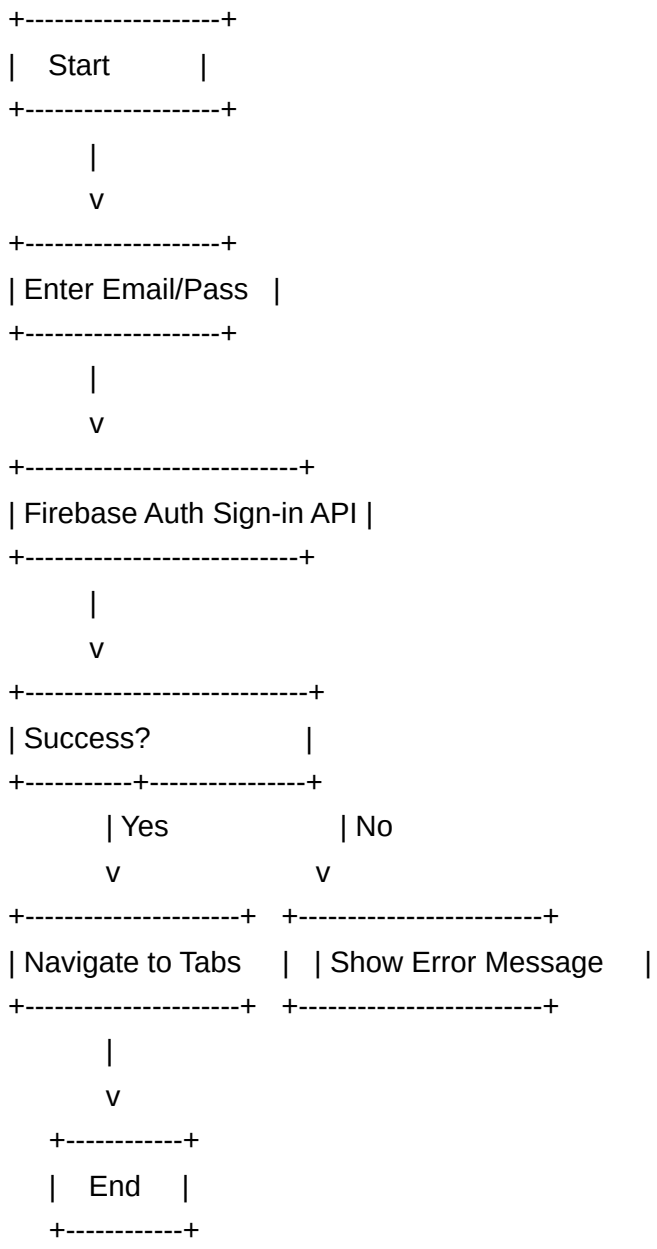
6. Compatibility

- Designed for Android devices using React Native.
- No platform-dependent code is used; can be extended to iOS with minimal changes.

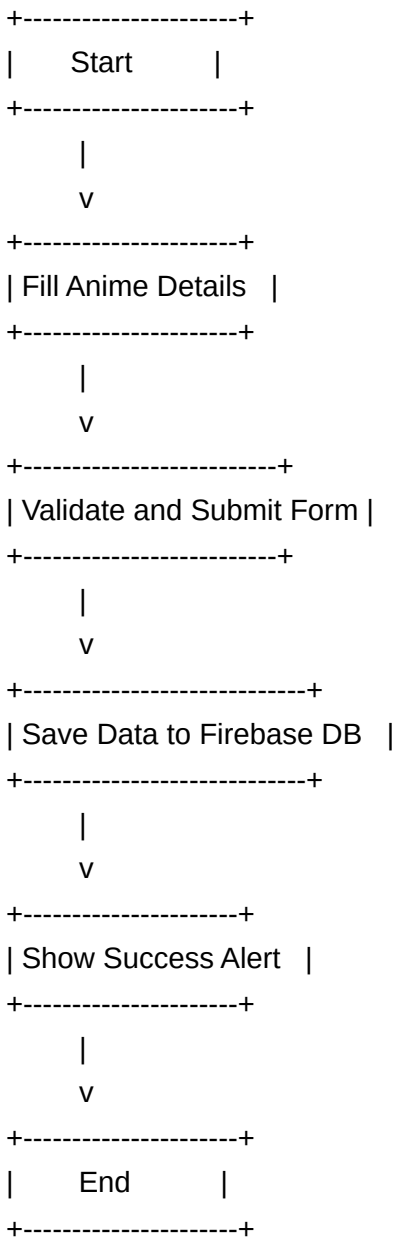
6. System Analysis

Flowcharts

1. User Authentication Flowchart

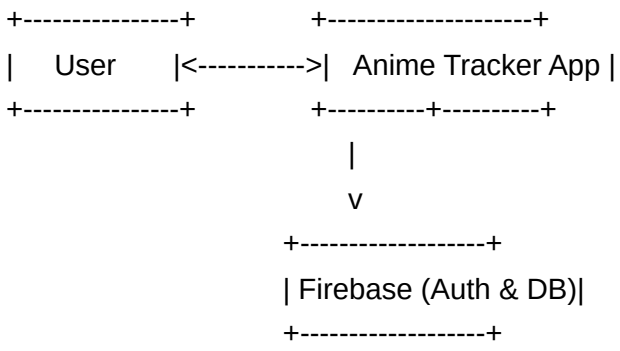


2. Anime Data Submission Flowchart

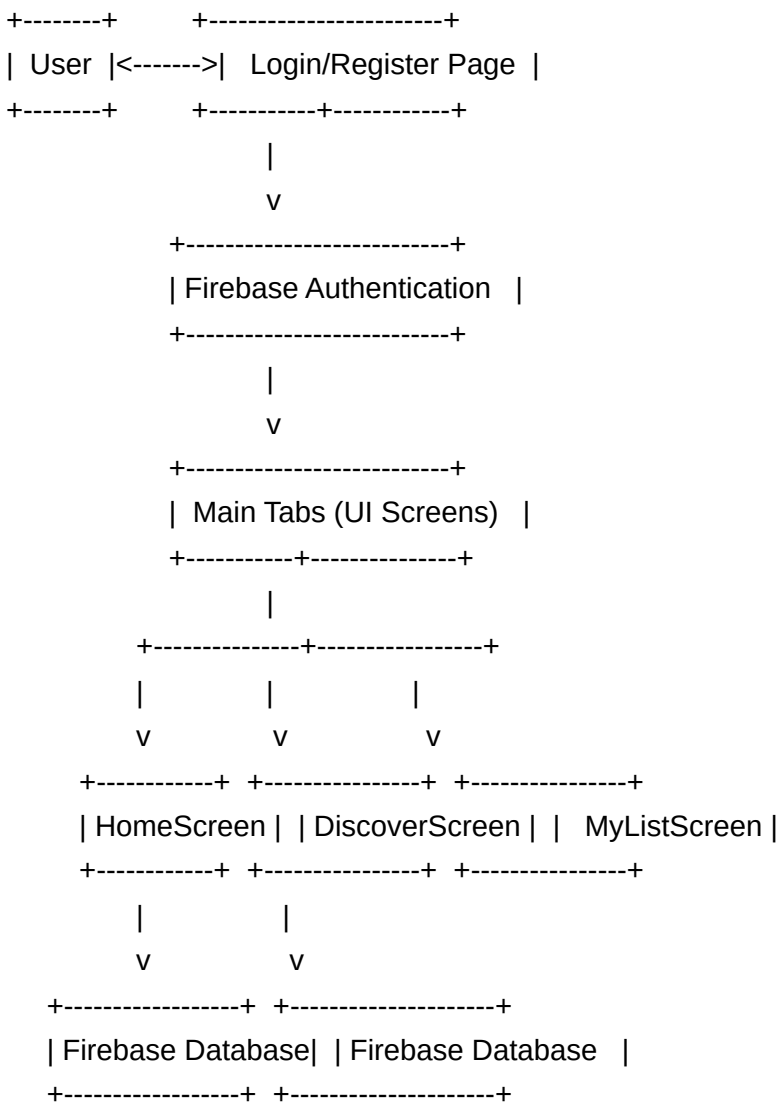


DFDs (Data Flow Diagrams)

Context Level DFD



Level 1 DFD



ER Diagram (Entity-Relationship Diagram)

Entities and Attributes:

1. **User**

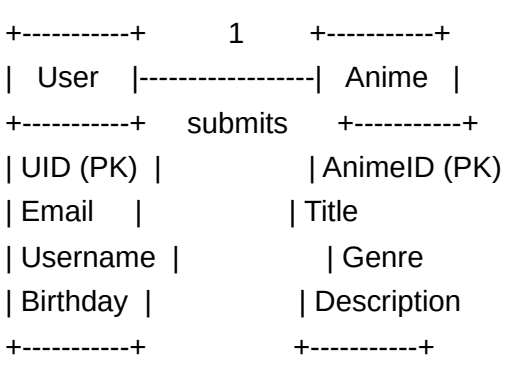
b. UID (Primary Key)
c. Email
d. Username
e. Birthday

6. **Anime**

g. AnimeID (Primary Key)
h. Title
i. Images (List)
j. Description
k. Genre
l. Rank
m. Rating
n. Popularity
o. Aired Date
p. Season
q. Author

Relationships:

- One User can submit multiple Anime entries.
- Each Anime entry is linked with one User (creator).



This system analysis helps visualize the flow of information and the relationship between different components of the Anime & TV Series Tracker app.

7. System Design

7.1 Architecture Overview

The Anime & TV Series Tracker app follows a **client-server architecture** using **React Native** as the frontend and **Firebase** as the backend (including authentication and Realtime Database).

- **Frontend (Mobile App):** Developed using React Native and JavaScript, the app runs on Android devices and handles user interactions, authentication, and screen navigation.
- **Backend:** Firebase Authentication is used for user sign-up/login, and Firebase Realtime Database stores user-generated content such as anime details.
- **Data Flow:** The app fetches anime lists and user data from Firebase and sends data (like new anime entries) back to the database.

This design allows for fast development, real-time updates, and scalability for future features like user ratings, comments, or bookmarking.

7.2 UI/UX Design

The UI is designed with a clean and simple layout using React Native components. The goal is to offer a user-friendly and smooth experience.

Key UI Features:

- **Login & Register Screens:** Basic input fields for email, password, username, and birthday with form validation.
- **Tab Navigation:**
  - **Home**: Used for adding anime data.
  - **Discover**: Displays lists of popular anime using FlatLists.
  - **Seasonal**: Placeholder screen for future seasonal anime display.
  - **MyList**: Placeholder for future saved list functionality.
  - **Discussion**: Placeholder for future discussion forums.
- **Buttons:** Used for form submission and navigation.
- **Alerts & Logs:** Provide user feedback and debugging output.

The UI is responsive and works across different Android devices. No third-party UI libraries were used except core React Native and Expo components.

7.3 Database Design

The Firebase Realtime Database follows a simple tree structure. Example schema:

yaml

```
anime/
  unique_id_1/
    title: "Jujutsu Kaisen"
    images: [URL1, URL2]
    rank: 25
    popularity: 90
    genre: ["Action", "Thriller"]
    description: "Anime about curses."
    season: "Fall 2020"
    rating: 8.9
    aired_on: "2020-10-03"
    author: "Gege Akutami"

  unique_id_2/
    ...
```

Each anime is stored under a unique key and contains multiple attributes such as title, image URLs, rating, genre, etc.

Benefits of This Design:

- Real-time updates.

- Easy read/write operations

8. Frontend Development

8.1 Component Structure

The frontend is developed using **React Native** with a modular, reusable component structure to organize screens and functionality. The directory structure is roughly as follows:

```
/App.js
/screens/
  ├── LoginScreen.js
  ├── RegisterScreen.js
  └── tabs/
      ├── HomeScreen.js
      ├── DiscussionScreen.js
      ├── DiscoverScreen.js
      ├── SeasonalScreen.js
      └── MyListScreen.js
/firebaseConfig.js
```

Component Overview:

- LoginScreen.js – Handles user authentication using Firebase Auth.
- RegisterScreen.js – Allows new users to register with email, username, password, and birthday.
- HomeScreen.js – Main input form to add anime data, including title, genres, ratings, author, etc.
- DiscoverScreen.js – Fetches and displays lists of popular anime, both dummy data and from Firebase.
- DiscussionScreen.js, SeasonalScreen.js, MyListScreen.js – Placeholder screens with basic navigation setup.

8.2 Routing

The app uses **React Navigation** for screen management:

- **Stack Navigator** (createNativeStackNavigator) is used to switch between the login and register screens.
- **Bottom Tab Navigator** (createBottomTabNavigator) provides tabbed navigation between the main sections of the app post-login.

Routing is handled centrally in App.js:

- If a user is logged in (onAuthStateChanged from Firebase), they are directed to the **TabsNavigator**.
- If not logged in, they are taken to the **AuthStack** (Login/Register).

javascript

{isUserLoggedIn ? <TabsNavigator /> : <AuthStack />}

8.3 Forms & Validations

The app includes two primary forms:

Register Form:

- Fields: email, username, password, birthday.
- Validation:
  - All fields must be filled.
  - Email must be in correct format.
  - Password field is secured with secureTextEntry.
- Validation is handled using simple if conditions before calling createUserWithEmailAndPassword().

Login Form:

- Fields: email, password.
- Firebase signInWithEmailAndPassword() is called only after basic validation checks.

Anime Submission Form (HomeScreen):

- Fields: title, images, rank, popularity, genres, description, season, rating, aired_on, author.
- Form is validated to ensure important fields (e.g., title, rank, rating) are not empty or invalid.
- Errors are caught using try-catch, and success is shown via alert().

9. Backend Development

The backend logic for the Anime & TV Series Tracker application is implemented using **Firebase**, a Backend-as-a-Service (BaaS) platform provided by Google. It provides several essential services like **Authentication**, **Realtime Database**, and **Hosting**, all integrated into the application without the need to build a separate custom server or REST API.

9.1 REST API

This project does not utilize a traditional REST API (like Node.js/Express or Django-based APIs). Instead, it uses **Firebase Realtime Database** which exposes a **REST-like interface** for reading/writing data in JSON format. Firebase manages:

- API calls internally through its SDK
- Secure communication via tokens
- Direct object-based interactions instead of URL-based endpoints

Each read/write operation is performed using Firebase's SDK functions, which internally hit secured REST endpoints.

Example (using Firebase SDK):

js

Copy code

```
const animeRef = ref(database, "anime/");
const newAnimeRef = push(animeRef);
await set(newAnimeRef, animeData);
```

9.2 Implementation

Firebase services used in this project include:

Firebase Realtime Database

- Data is stored in a **NoSQL JSON tree structure**
- Anime details are submitted via HomeScreen and stored using:

js

Copy code

```
const animeRef = ref(database, "anime/");
const newAnimeRef = push(animeRef);
```

await set(newAnimeRef, data);

- Data retrieval (in DiscoverScreen) uses onValue() for real-time updates.

Firebase Authentication

- Email and Password-based authentication is implemented
- Users can register and log in with Firebase's createUserWithEmailAndPassword() and signInWithEmailAndPassword()

Firebase provides secure user token generation, session management, and user identification without the need for custom backend code.

9.3 User Authentication

Authentication is implemented through Firebase's built-in services:

- Users register with:

js

Copy code

await createUserWithEmailAndPassword(auth, email, password);

- Login is handled using:

js

Copy code

await signInWithEmailAndPassword(auth, email, password);

- onAuthStateChanged() is used in App.js to observe the login status and toggle between **AuthStack** and **TabsNavigator**

Firebase stores user sessions in memory or persistent storage (when configured), eliminating the need to build login/session APIs manually.

9.4 Project Logic

The business logic includes:

- **Data Collection**: Via HomeScreen, users input anime details (title, images, ratings, etc.)
- **Data Storage**: Submitted data is structured and stored in the Realtime Database under anime/
- **Data Display**: Fetched in DiscoverScreen via onValue() and converted into a list format for rendering

- **Authentication Check**: Before allowing access to app features, Firebase Auth verifies the user session
- **Session Persistence**: Managed by Firebase Auth, allowing users to remain logged in across sessions

The project logic is designed to keep frontend and backend loosely coupled while leveraging Firebase's capabilities to manage users and data securely.

10. Integration & Deployment

Successful software development not only requires proper planning and development but also efficient integration and deployment. In this project, the **frontend (React Native)** and the **backend (Firebase)** are tightly integrated using Firebase's SDK, making deployment smoother and more manageable.

10.1 Connecting Frontend & Backend

In the Anime & TV Series Tracker app, the frontend is built using **React Native** and connected to Firebase using its **JavaScript SDK**.

Firebase Initialization

The Firebase services (Authentication and Realtime Database) are initialized in the firebaseConfig.js file:

js

Copy code

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getDatabase } from "firebase/database";

const firebaseConfig = {
  apiKey: "...",
  authDomain: "...",
  projectId: "...",
  storageBucket: "...",
  messagingSenderId: "...",
  appId: "..."
};

const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const database = getDatabase(app);

export { auth, database };
```

This configuration is imported into various components to authenticate users and store/retrieve anime data.

Firebase SDK Usage in Frontend

- **RegisterScreen.js** uses createUserWithEmailAndPassword() to create a new user.
- **LoginScreen.js** uses signInWithEmailAndPassword() for user login.
- **HomeScreen.js** writes anime details to the database using push() and set().
- **DiscoverScreen.js** reads data using onValue() from the Realtime Database.

By using the SDK methods, the frontend directly communicates with Firebase without needing a separate server or REST API.

10.2 Deployment on Vercel/Render

Since this is a **mobile application built using React Native**, it's not deployed on **Vercel** or **Render** (both of which are generally used for web apps or Node.js backend APIs).

Instead, the deployment of this project involves:

Firebase Hosting (for backend services)

- Firebase Authentication and Realtime Database are cloud-hosted.

- Firebase handles the backend infrastructure.

Running the App (Frontend Deployment)

- The React Native app is developed using **Expo CLI** or **React Native CLI**.
- The app is built and tested using:
  - **Android Emulator**
  - **Expo Go App** for Android
- Final deployment can be done to:
  - **Google Play Store** (for Android)
  - **Apple App Store** (for iOS, if applicable)

To build the APK or AAB file:

bash

Copy code

```
npx expo build:android
```

Or using EAS Build:

bash

Copy code

```
eas build --platform android
```

Summary of Deployment

Layer	Platform Used	Purpose
Frontend	React Native (Expo)	User interface and interaction
Backend	Firebase	Auth and Realtime Database
Testing Env.	Emulator / Expo Go	Mobile testing and debugging
Deployment	APK via Expo / EAS	Distribute app to users

11. System Testing

System testing is a critical phase in the software development lifecycle. It ensures that all components of the application work together as expected and that the software performs its intended functions correctly and reliably.

In the Anime & TV Series Tracker app, multiple layers of testing were conducted to validate both functional and non-functional aspects of the system.

11.1 Testing Strategy

The testing strategy for this project includes the following types:

Type	Description
Unit Testing	Testing individual components like forms, input fields, and buttons.
Integration Testing	Ensuring proper communication between Firebase and the frontend components.
Manual Testing	Verifying workflows like user login, data submission, and data retrieval.
UI Testing	Checking responsiveness, layout alignment, and user experience consistency.
Boundary Testing	Validating input fields with edge values (e.g., very long titles or empty inputs).

Testing was performed during development using **React Native Debugger**, **console logs**, and test devices/emulators.

11.2 Sample Test Cases and Results

Below are some representative test cases used during the testing phase:

Test Case 1: User Registration with Valid Input

Description	Input
Email	testuser@gmail.com
Username	testuser
Password	Test@1234
Birthday	2000-01-01
Expected Result	User is registered and redirected or confirmation is logged.
Actual Result	Passed. Firebase account created successfully.

Test Case 2: User Login with Invalid Credentials

Email	wronguser@gmail.com
Password	wrongpass
Expected Result	Show error / authentication fails.
Actual Result	Passed. Firebase returned an authentication error.

Test Case 3: Submitting Anime with Missing Fields

Input Fields	Title: "Naruto", Rank: "", Description: "", etc.
Expected Result	Show alert to fill all fields.
Actual Result	Passed. Alert displayed, form not submitted.

Test Case 4: Retrieving Anime Data from Firebase

Action	Click "Get Anime List" button
Expected Result	Anime list loads from Firebase into Discover tab.
Actual Result	Passed. Data loaded and displayed successfully.

Test Case 5: Logout Functionality

Action	Tap "Logout" button
Expected Result	User is signed out and redirected to Login screen.
Actual Result	Passed. Auth state changed, login screen shown.

11.3 Tools Used for Testing

Tool/Platform	Purpose
React Native Debugger	State monitoring, console logging
Firebase Console	Monitor auth/database entries
Expo / Emulator	Run tests on real and virtual devices
Manual Testing Checklist	Ensuring feature-by-feature testing

12. Project Screenshots

Screenshots are an essential part of the documentation as they visually represent the application's features, interface, and usability. Below are categorized screenshots with descriptions from the Anime & TV Series Tracker app.

12.1 Login & Register Screens

Login Screen

- **Purpose**: Allows users to sign in using their email and password.
- **Features**:
  - Input validation
  - Firebase Authentication integration
  - Error handling for incorrect credentials

Insert Screenshot of LoginScreen.js here

Register Screen

- **Purpose**: Enables new users to create an account.
- **Fields**: Email, Username, Password, and Birthday
- **Features**:
  - Firebase user creation
  - Input field validations
  - Alerts for missing data

Insert Screenshot of RegisterScreen.js here

12.2 Home, Anime & TV Series Submission Screens

Home Screen (Admin Panel)

- **Purpose**: Allows admin or users to submit new anime entries.
- **Fields**:
  - Title, Rank, Popularity, Description, Genres, Images, Season, Rating, Author
- **Features**:
  - Firebase Realtime Database integration
  - Genre multi-select
  - Validation and confirmation alerts

Insert Screenshot of HomeScreen.js (form with fields)

Discover Screen

- **Purpose**: Displays popular anime cards using dummy and Firebase data.
- **Features**:
  - Horizontal FlatLists with anime images and titles
  - Real-time fetching from Firebase
  - Multiple sections

Insert Screenshot of DiscoverScreen.js with horizontal scroll

12.3 Other Functional Screens

Discussion, My List, and Seasonal Tabs

- **Purpose**: Placeholder for future discussion, user lists, and seasonal anime.
- **Current Functionality**:
  - Basic navigation and routing using React Navigation
  - Placeholder text for future expansion

📸 *Insert Screenshot of DiscussionScreen.js / MyListScreen.js / SeasonalScreen.js*

12.4 Admin Controls (From Home Tab)

- **Functionality**:
  - Acts as a backend form for adding anime entries
  - No separate dashboard UI, integrated with Home tab
  - Firebase push operation on submit

13. Challenges Faced & Solutions

During the development of the **Anime & TV Series Tracker** app as part of the internship at **LaunchED Global**, several technical and project management challenges were encountered. These hurdles played a crucial role in shaping the learning experience and improving problem-solving capabilities.

13.1 Technical Challenges

Firebase Authentication Configuration

- **Challenge**: Setting up Firebase Authentication and integrating it properly with React Native.
- **Problem**: Warning about missing AsyncStorage for auth state persistence in React Native.
- **Solution**: Installed @react-native-async-storage/async-storage and used initializeAuth with getReactNativePersistence() to persist the login session.

Data Persistence in Realtime Database

- **Challenge**: Storing complex structured data like anime with multiple fields and arrays (e.g., genres, image URLs).
- **Problem**: Misstructured or missing values due to improper Firebase path or data validation.
- **Solution**: Used push() to generate unique IDs and validated all required fields before data submission.

FlatList Rendering Issues

- **Challenge**: Displaying dynamic and static lists using FlatList and updating them in real-time.
- **Problem**: Overlapping or non-responsive lists due to multiple simultaneous FlatLists.
- **Solution**: Used horizontal FlatLists with separate section titles, and extraData to re-render when needed.

Warnings During Development

- **Problem**: React Native runtime warning:
  "Text strings must be rendered within a <Text> component"
- **Cause**: A string or JSX expression placed directly inside a View without being wrapped in a Text component.
- **Solution**: Ensured all text content is wrapped in <Text> components.

Navigation Errors

- **Challenge**: Implementing conditional routing between login and authenticated states using React Navigation.
- **Problem**: Navigation to the wrong screen or blank screen when auth state changed.
- **Solution**: Managed authentication status using Firebase's onAuthStateChanged and toggled between AuthStack and TabsNavigator.

13.2 Project Management Challenges

Time Constraints

- **Challenge**: Completing the entire project within the 2-month internship period.
- **Solution**: Created a weekly development plan focusing on one major feature or screen at a time.

Learning Curve

- **Challenge**: Being a beginner in React Native, Firebase, and project structuring.
- **Solution**: Followed official documentation, tutorials, and guidance from mentor **Piyush Nanwani sir** at LaunchED Global.

Debugging & Testing

- **Challenge**: Identifying bugs in data flow between components and Firebase.
- **Solution**: Used console logs extensively and performed manual testing of all form submissions and navigations.

Lack of Real Device Testing

- **Challenge**: Testing only on emulator caused some UI inconsistencies on real devices.
- **Solution**: Deployed APK to test on physical Android phones before final submission.

14. Conclusion & Learning Outcomes

14.1 Conclusion

The **Anime & TV Series Tracker** app, developed during a two-month internship at **LaunchED Global** under the mentorship of **Piyush Nanwani sir**, serves as a full-fledged mobile application for managing anime-related content. The project successfully integrates user authentication, content submission, real-time data fetching, and intuitive navigation using **React Native** and **Firebase**.

The project aimed to bridge the gap between theoretical learning and real-world application by allowing a beginner-level developer to work with modern development tools and technologies. Through careful planning, continuous learning, and iterative development, all major goals of the application were met.

The app allows users to:

- Register and log in using Firebase Authentication.
- Add detailed anime information such as title, images, genres, ratings, author, etc.

- View dummy as well as dynamically fetched anime lists using FlatList.
- Navigate between multiple screens including Home, Discover, Discussion, Seasonal, and My List.

Overall, the project is a demonstration of how a student with limited experience can build a production-ready mobile app by leveraging open-source tools, cloud services, and guidance from a professional mentor.

14.2 Learning Outcomes

Technical Skills Gained

- Practical experience with **React Native** (JSX, components, state management).
- Integration of **Firebase Authentication** for secure login/logout.
- Use of **Firebase Realtime Database** to store and retrieve structured data.
- Implementing **React Navigation** with both Stack and Bottom Tab Navigators.
- Designing responsive forms, validations, and UI components.
- Handling **warnings**, **runtime errors**, and **debugging** in a real application.

Project Management Skills

- Structured planning and weekly goal setting.
- Prioritization of features due to time limitations.
- Iterative development, testing, and refining of features.
- Handling challenges independently with occasional mentorship support.

Professional Growth

- Real-world exposure to how frontend and backend connect via cloud services.
- Improved logical thinking and problem-solving ability.
- Understanding of development lifecycle from design to deployment.
- Building confidence in full-stack mobile app development.

15. Bibliography & References

This section lists all the major resources, documentation, and tools referred to and utilized during the development of the **Anime & TV Series Tracker** project.

15.1 Official Documentation

1. **React Native Documentation**
   https://reactnative.dev/docs
   Used for understanding component structure, navigation, styling, and integration of native APIs.
2. **Firebase Documentation**
   https://firebase.google.com/docs
   Referred for implementing Firebase Authentication, Realtime Database, and SDK initialization.
3. **React Navigation Docs**
   https://reactnavigation.org/docs
   Helpful in configuring stack and tab navigation for managing screen transitions.
4. **Expo Documentation**
   https://docs.expo.dev
   Used for managing the React Native environment during development and testing.
5. **@react-native-async-storage/async-storage**
   https://github.com/react-native-async-storage/async-storage
   Used for persistent login sessions and local data storage.

15.2 Learning Platforms & Articles

1. **Stack Overflow**
   https://stackoverflow.com
   Used frequently for solving specific technical issues and warnings.
2. **GeeksforGeeks – React Native & Firebase Integration**
   https://www.geeksforgeeks.org
   Referenced for understanding Firebase setup and integration examples.
3. **Medium Articles on Firebase Auth & React Native**
   Articles authored by developers explaining Firebase authentication and cloud data sync in React Native apps.

15.3 Tools Used

1. **Visual Studio Code** – Code editor used for writing and organizing code.
2. **Firebase Console** – For managing the backend (database, authentication).
3. **Expo Go App** – For testing the application during development on mobile devices.