

Mini-Project Report On

TurfSurf App(An Application for booking turfs)

*Submitted in partial fulfillment of the requirements for the
award of the degree of*

Bachelor of Technology

in

Computer Science & Engineering

By

John Sherry (U2003107)

Joyce Boban (U2003112)

Kristin Elizabeth Binu (U2003122)

Maria Sunil(U2003128)

Under the guidance of

Dr. Dhanya P.M



**Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039**

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039



CERTIFICATE

*This is to certify that the mini-project report entitled "**TurfSurf(Application for booking Turfs)**" is a bonafide work done by **Mr. John Sherry (U2003107)**, **Ms. Joyce Boban (U2003112)**, **Ms. Kristin Elizabeth Binu (U2003122)** **Ms. Maria Sunil (U2003128)**,, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.
Head of Department
Dept. of CSE
RSET

Mr. Uday Babu P.
Mini-Project Coordinator
Asst. Professor
Dept. of CSE
RSET

Dr. Dhanya P.M
Mini-Project Guide
Associate Professor
Dept. of CSE
RSET

ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, "Unify App".

We are highly indebted to our mini-project coordinators, **Mr. Uday Babu**, Assistant Professor, Department of Computer Science and Engineering, **Ms.**, Assistant Professor, Department of Computer Science and Engineering, and **Dr. Dhanya PM**, Associate Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Dr. Dhanya P.M**, for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

John Sherry

Joyce Boban

Kristin Elizabeth Binu

Maria Sunil

ABSTRACT

Online Turf booking is developed to provide the the solution of the online booking of the available turfs that would help the enthusiasts to book the turf. Through this the admin who is one of the modules of the application will manage all the activities of the users such as adding a turf ,managing a turf ,viewing the booking details. This online platform will help the users to book their choice of available slots .This system will replace the manual process of going to the location to book the with an automated online process.With the TurfSurf app, this process is made effortless with in-app booking and payment facilities. This avoids the hassle of calling up the owners and makes booking turfs simple and seamless.This application is not only helps the users but also helps the turf owners who would like to expand their business through an online platform.

Contents

Acknowledgements	ii
Abstract	iii
List of Figures	vi
1 Introduction	1
1.1 Background	1
1.1.1 Importance of Online Turf booking system	1
1.2 Existing System	1
1.3 Problem Statement	2
1.4 Objectives	2
1.5 Scope	3
2 Literature Review	4
3 System Analysis	5
3.1 Expected System Requirements	5
3.2 Feasibility Analysis	6
3.2.1 Technical Feasibility	6
3.2.2 Operational Feasibility	6
3.2.3 Economic Feasibility	7
3.3 Hardware Requirements	8
3.4 Software Requirements	8
3.4.1 Android Studio for flutter app development	8
3.4.2 Flutter SDK	9
3.4.3 Google Maps API	9
3.4.4 Dart 3.62.0	10

4	Methodology	11
4.1	Proposed Method	11
4.1.1	Modules of Turf Booking System :	11
5	System Design	14
5.1	Architecture Diagram	14
5.2	Sequence Diagram	15
5.3	User Diagram	16
5.4	Admin Diagram	17
5.5	Turf Owner Diagram	18
6	Testing	19
6.1	Unit Testing	19
6.2	Integration Testing	19
6.3	System Testing	19
6.4	Acceptance Testing	19
6.5	Cross Device Testing	20
6.6	Security Testing	20
6.7	User Interface Testing	20
7	Results	21
8	Risks and Challenges	24
9	Conclusion	25
	References	26
	Appendix A: Sample Code	27
	Appendix B: CO-PO and CO-PSO Mapping	75

List of Figures

5.1	Architecture Diagram	14
5.2	Flowchart for User	16
5.3	Flowchart for Admin	17
5.4	Flowchart for Turf Owner	18
7.1	Welcome page	21
7.2	District selection page	21
7.3	Turf view and slot selection	21
7.4	Booking confirmation message	21
7.5	Turf registration page	22
7.6	Turf updation page	22
7.7	User login page	22
7.8	Turf owner login page	22
7.9	Turf view on map	23
7.10	Payment page	23

Chapter 1

Introduction

1.1 Background

1.1.1 Importance of Online Turf booking system

Turf playground are used to play various sports like football, rugby, tennis, cricket, etc. People enjoy playing on the turf, it has vibrant environment and very safe to play. Many school teams and clubs prefer turf playground for practice and training purpose. Sometimes it becomes difficult to book turf playground because of the timing issues or the slot getting booked previously. This turf booking application is proposed for booking the turf in an easy and efficient way. It has three modules namely, User , Admin and Turf owner. Admin can login and can add turf locations, assign manager by creating login credentials for manager, add price details for the particular turf, manages turf and view the details of sports venues booking for all locations. Users can check the availability of the turf, select timings, fill personal details, can pay by providing bank details or card details and he/she can also see view previous turf booking history. Most of the Turf grounds presently run a manual reservation system and as customers are desirous to find a handy application that provides the services to avoid physical walking to the ground or contacting by call or reserving through a middle man. So thus an online turf booking system can be used to tackle the current situation by providing an effective booking system that can be directly accessible to every user.

1.2 Existing System

Currently, the type of system that is being used is a manual booking system .The problem faced by the turf owners is that the customers have to go to the counters to book a slot and will also have to pay cash while confirming the booking procedures. Since the data

is not computerized appropriately during manual booking there exists a huge chance of data loss and lack of securing the data .Manually booking the system can also be very time consuming and expensive as the extra transportation cost that has incurred to reach the turf. In order to make bookings convenient and hassle free TurfSurf application could be used which a more advance method which will help the users to book their desirable turf one click away in the comfort of their houses.

1.3 Problem Statement

Online Turf Booking is developed to provide the solution of online booking of the available turf that would help the Cricket and Football enthusiastic players to book the turf. Through, this application admin who is one of the users of this application will manage all the activities of Users such as search user, Add Turf, Manage Turf, and View Bookings.

The main Objective of Online Turf Booking is to manage the activity of Turf. This application can be used by the Cricket and Football enthusiastic players to book the available turf according to their need. This application not only helps the players but also the turf owners who want to expand their business through online medium.

This project provides a simple and beautiful interface for the admin as well as to the Players, or users. Just admin needs to add Turf and can view the booking History. All the information will be stored in the database and that will help to maintain all the information of Players or users.

1.4 Objectives

- To provide a bug-free Booking System to Players or users.
- To allow users to search and view the Information of Turf online using our application.
- Maintain the List of Turf and Booking History so it will be easy to access any time 24*7
- Easy to access the system anywhere and anytime.

1.5 Scope

1. Introduction

Briefly introduce the concept of an online turf booking system. Highlight the increasing popularity of sports and recreational activities, leading to a growing demand for turf facilities. Explain the significance of an online booking system to streamline the booking process.

2. Objectives of the Project

Define the objectives of developing an online turf booking system. Discuss the key goals, such as enhancing convenience, improving accessibility, and optimizing resource allocation.

3. System Features and Functionality

Provide an overview of the features and functionalities of the online turf booking system. Include aspects like user registration, login, and profile management. Describe the turf selection and availability checking process. Explain the booking and payment mechanisms. Discuss any additional features, such as reviews and ratings, notifications, and cancellation policies.

4. User Roles and Responsibilities

Define the various user roles within the system, such as customers, turf owners/operators, and administrators. Discuss the responsibilities and permissions associated with each role. Explain how the system ensures data security and privacy.

5. System Architecture and Technology Stack

Present an overview of the system architecture, including the front-end and back-end components. Discuss the technology stack employed in developing the online turf booking system, such as programming languages, frameworks, and databases. Highlight the scalability and robustness of the chosen architecture.

6. System Workflow

Illustrate the workflow of the online turf booking system, from user registration to completion of the booking process. Use flowcharts or diagrams to visually represent the steps involved. Explain any conditional or branching processes within the system.

Chapter 2

Literature Review

In this section, we will survey in brief on the usage of online booking systems which are used in public or private sports complex. In research we have found that there are no Turf grounds using such a system to manage and monitor their complex. Now a day's online business earns more capital then compared to offline businesses thus making them more potential to acquire the market. Keeping that in mind our system aims to provide a platform which provides a clean and detailed management to sports complex's in terms of bookings and cash flow making them more reliable and readily available in market. In countries like USA such systems are already available in market and hence creating a business opportunity in new dimensions. Their systems are quick and robust hence making a booking is so simple with online payments. When required to show their booking, mobile users would simply show the manager a virtual ticket on their smartphone. Other research also shows sports complex are less famous in their locality due to their absence on digital world, this makes their business less profitable and chances to a failure. Hence making business online is good solution to be famous in locality and getting more users making business good. Another drawback of not being an online business is getting less customers which eventually drops the growth rate or profit of your business. Hence making a choice of being an online business is always preferred. So far from the literature review, it is clear that an online booking/reservation system for an sports complex would be very much profitable and manageable.

Chapter 3

System Analysis

3.1 Expected System Requirements

The expected system requirements for a turf booking system application can vary depending on the specific features, scale, and technical architecture of the application. However, here are some general considerations for system requirements:

1. Server Requirements:

Sufficient processing power and memory to handle concurrent user requests. Adequate storage capacity to store turf information, user data, and booking details. A reliable and scalable database system to manage and retrieve data efficiently. Robust security measures to protect user information and prevent unauthorized access. Network Requirements:

A stable and high-speed internet connection to ensure seamless communication between clients and servers. Support for both wired and wireless connections to accommodate various user devices.

2. Client Device Requirements:

Compatibility with different operating systems (e.g., Windows, macOS, iOS, Android) to cater to a wide range of users. Responsive design and user interface optimization for different screen sizes, including desktops, laptops, tablets, and smartphones. Modern web browsers or mobile applications with support for the latest web technologies and frameworks.

3. Authentication and Authorization:

Secure user authentication mechanisms, such as username/password, social login (e.g., OAuth), or two-factor authentication. Role-based access control to define user roles (admin, manager, customer) and their respective permissions.

4. Booking and Scheduling:

User-friendly interface for customers to browse available turfs, view schedules, and

make bookings. Real-time availability updates and conflict resolution to prevent double bookings. Notifications and reminders for users regarding upcoming bookings, cancellations, or changes.

5. Payment Integration:

Integration with popular payment gateways to facilitate online payments securely.

3.2 Feasibility Analysis

3.2.1 Technical Feasibility

1. Infrastructure: Assess the availability and suitability of the necessary technical infrastructure, including servers, databases, and networking components. Consider factors such as scalability, reliability, and performance to ensure that the infrastructure can handle the expected user load and data storage requirements.

2. Technology Stack: Determine the appropriate technology stack for building the application. Consider factors such as programming languages, frameworks, libraries, and third-party integrations. Ensure that the chosen technologies align with the project's requirements and provide the necessary functionality and flexibility.

3. Integration: Evaluate the feasibility of integrating with external systems or services, such as payment gateways, mapping APIs, or notification services. Determine whether the required integrations are readily available, well-documented, and compatible with the chosen technology stack.

4. Data Management: Assess the feasibility of effectively managing and storing data related to turfs, bookings, user information, and other relevant entities. Consider factors such as database design, data security, backup mechanisms, and data retrieval performance.

5. User Experience: Determine the feasibility of delivering a seamless and intuitive user experience across different devices and platforms. Consider responsive web design or the development of native mobile applications to ensure accessibility and usability.

3.2.2 Operational Feasibility

1. Stakeholder Analysis: Identify and understand the needs, expectations, and concerns of key stakeholders, such as turf owners, customers, administrators, and staff members.

Consider their level of acceptance and willingness to adopt the new system.

2. **Business Processes:** Evaluate how the proposed system will fit into existing business processes and workflows. Identify any necessary changes or adaptations required to integrate the new system seamlessly. Consider the impact on staff roles, responsibilities, and training requirements.

3. **Resource Availability:** Assess the availability of resources required for operating the online turf booking system. This includes human resources, such as trained staff members to handle customer inquiries and manage bookings. Additionally, consider the availability of physical resources, such as turf availability, maintenance equipment, and other operational requirements.

4. **Training and Support:** Determine the training needs of staff members who will be using and managing the system. Evaluate the feasibility of providing comprehensive training and ongoing support to ensure a smooth transition and continued operational efficiency.

5. **Scalability and Growth:** Consider the system's ability to accommodate future growth and scale as the business expands or additional turfs are added to the system. Assess whether the proposed solution can handle increased user demand and booking volume without significant disruptions.

3.2.3 Economic Feasibility

1. **Cost Analysis:** Evaluate the costs associated with developing and implementing the online turf booking system. This includes expenses such as software development, hardware infrastructure, licensing fees, integration with payment gateways, and any necessary third-party services or APIs. Consider both the upfront costs and ongoing operational expenses.

2. **Return on Investment (ROI):** Estimate the potential financial benefits and return on investment that the online turf booking system can generate. Consider factors such as increased revenue through streamlined bookings, improved customer satisfaction leading to higher occupancy rates, reduced administrative costs, and potential cost savings through automation and process efficiencies.

3. **Revenue Generation:** Analyze how the online booking system can generate additional revenue streams. Consider opportunities for offering premium services, such as

equipment rentals, coaching sessions, or additional facilities. Assess the potential for partnerships or collaborations with other businesses to create value-added offerings.

4. **Market Analysis:** Evaluate the market demand and competitive landscape for turf bookings. Consider factors such as the target customer base, market size, growth potential, and any unique selling propositions that the online booking system can offer. Assess the potential market share and revenue capture based on customer demand and pricing strategies.

5. **Cost Reduction and Efficiency Gains:** Determine the potential cost savings and efficiency gains that the online turf booking system can bring. Assess how the system can streamline administrative processes, reduce manual errors, and optimize resource utilization. Consider potential labor cost savings and improved resource allocation as a result of automated booking management.

3.3 Hardware Requirements

The following are the system requirements to develop the Turf booking App.

- Processor: Intel Core i5
- Hard Disk: Minimum 100GB
- RAM: Minimum 8GB

3.4 Software Requirements

The following are the softwares used in the development of the app.

Operating System: Windows or Linux

3.4.1 Android Studio for flutter app development

Android Studio is a popular Integrated Development Environment (IDE) for developing Flutter apps, as it provides a wide range of tools and features that can help you build high-quality apps faster. Some of the key features of Android Studio for Flutter development include:

A rich set of tools for debugging, testing, and profiling your app. A powerful code editor with support for code completion, refactoring, and more A flexible build system with

support for building, testing, and deploying your app. Integration with popular version control systems like Git. A visual layout editor for building attractive user interfaces

3.4.2 Flutter SDK

The Flutter SDK (Software Development Kit) is a comprehensive and open-source framework developed by Google that allows developers to build high-performance, cross-platform mobile, web, and desktop applications. It provides a rich set of tools, widgets, and libraries to streamline the app development process and enables developers to write a single codebase that runs on both Android and iOS devices. Flutter's key feature is its fast and expressive UI, powered by the Dart programming language, which allows for hot reload functionality, enabling real-time updates to the app's interface during development. This significantly accelerates the iteration cycle and enhances developer productivity. With Flutter, developers can create visually appealing and native-like user interfaces that offer a consistent experience across different platforms. Additionally, Flutter's flexibility extends to web and desktop applications, enabling developers to target multiple platforms with minimal code changes. Its vibrant community, frequent updates, and growing ecosystem of plugins make it a popular choice for building modern, responsive, and high-performance applications across a wide range of platforms.

3.4.3 Google Maps API

The Google Maps API integration in FlutterFlow enables users to effortlessly incorporate powerful mapping and location-based functionalities into their Flutter apps without the need for extensive coding. Through a seamless process, developers can access various geographic data, such as interactive maps, geocoding services, directions, places, and Street View imagery. By obtaining an API key from the Google Cloud Console, users can integrate the google maps flutter package into their project and configure the initial map position as well as additional features. This integration empowers FlutterFlow users to create location-aware applications with visually appealing and highly functional maps, enhancing user experiences and opening up new possibilities for location-based services and applications.

3.4.4 Dart 3.62.0

Dart is a programming language developed by Google and commonly used for building web, mobile, and desktop applications. The version you mentioned, 3.62.0, does not correspond to any official releases of Dart as of my knowledge cutoff in September 2021. The latest stable release of Dart at that time was version 2.14.0.

Chapter 4

Methodology

4.1 Proposed Method

1. Requirements Gathering: Identify and document the specific requirements of the turf booking system by conducting interviews, surveys, and workshops with stakeholders. This includes understanding the needs of turf owners, customers, and administrators. 2. System Analysis and Design: Analyze the gathered requirements and create a system design that outlines the architecture, modules, and user interfaces of the booking system. This involves defining data models, workflows, and user interaction flows. Create wireframes or prototypes to visualize the system's interfaces and functionalities, allowing stakeholders to provide feedback and make necessary refinements. 3. Development: Implement the system based on the finalized design using appropriate programming languages, frameworks, and development methodologies. Adopt an iterative development approach, where the system is built incrementally, allowing for continuous testing and feedback. 4. Testing and Quality Assurance: Conduct rigorous testing to ensure the functionality, reliability, and performance of the system. This includes unit testing, integration testing, and user acceptance testing. Perform security testing to identify and address vulnerabilities and implement measures to protect user data and prevent unauthorized access. 5. Deployment and Training: Deploy the online turf booking system to the production environment, ensuring compatibility with the required hardware and software infrastructure. Provide comprehensive training to turf owners, administrators, and other relevant stakeholders on system usage, management, and support.

4.1.1 Modules of Turf Booking System :

- 1. User Module:

Allow users to create accounts and provide necessary details.

- Enable users to view and update their personal information, manage preferences, and view booking history.
- Provide a user-friendly interface for searching and booking turfs based on location, date, time, and other criteria.
- Integrate a secure payment gateway for users to make online payments.
- Allow users to view and manage their bookings, make modifications, and handle cancellations. Reviews and Ratings: Enable users to provide feedback and ratings for the turfs they have booked.

- 2. Admin Module:

Admin Dashboard: Provide a comprehensive dashboard for administrators to manage the system.

- User Management: Allow administrators to manage user accounts, handle registration approvals, and resolve user-related issues.
- Turf Management: Enable administrators to manage turf listings, including adding new turfs, updating details, and handling disputes.
- Booking Management: Provide the ability to view, modify, and cancel bookings, handle refunds, and resolve booking conflicts.

- 3. Turf Owner :

Enable turf owners to create and manage their turf listings, including adding details, uploading photos, and setting availability.

- Booking Management: Allow turf owners to view and manage bookings for their turfs, accept or reject booking requests, and handle availability conflicts.

- Pricing and Availability: Provide tools for turf owners to set pricing, define availability slots, and update them as needed.
- Communication: Enable turf owners to communicate with users regarding booking inquiries, special requests, or any other relevant information.

Chapter 5

System Design

Draw usecase diagrams, activity diagrams, sequence diagrams etc. Add a brief explanation for each UML diagram.

5.1 Architecture Diagram

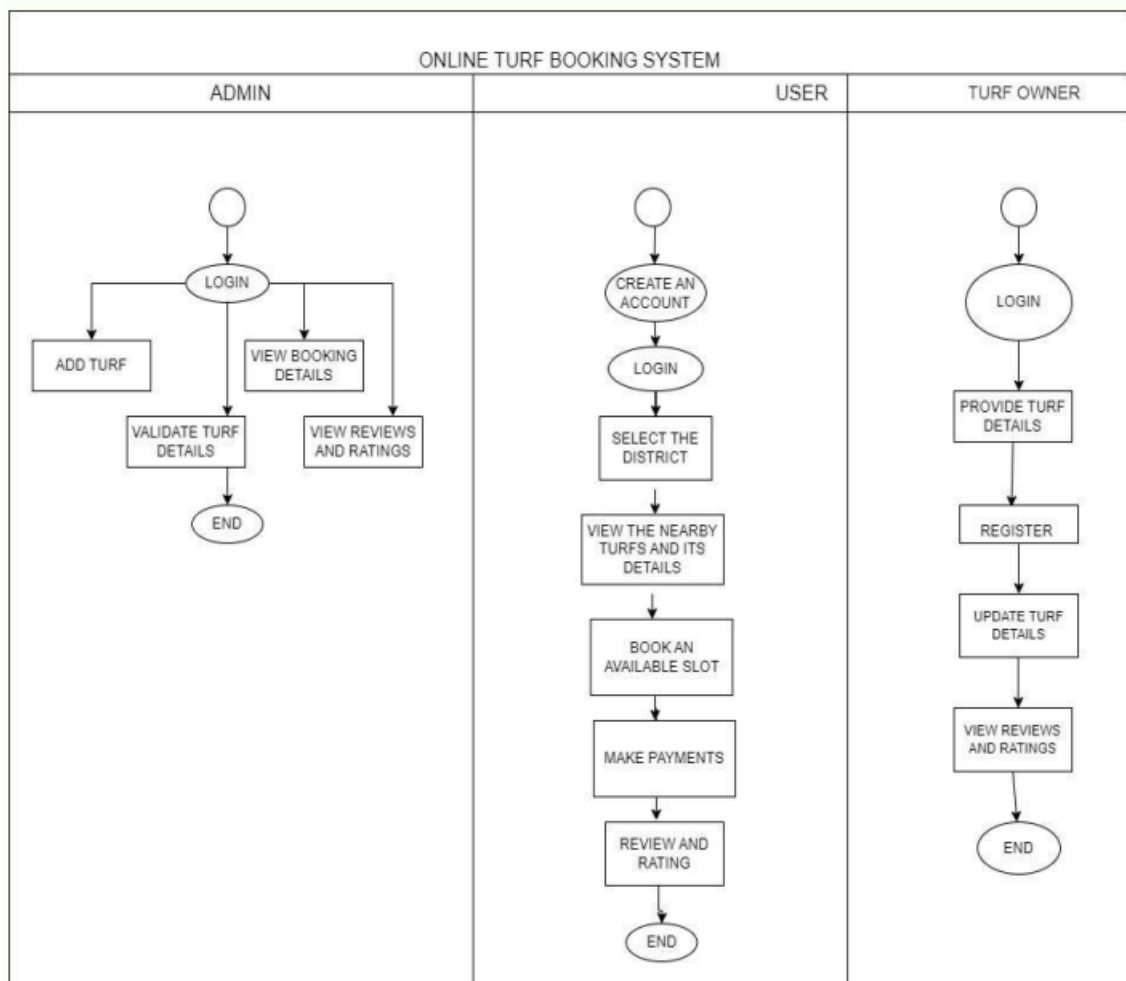
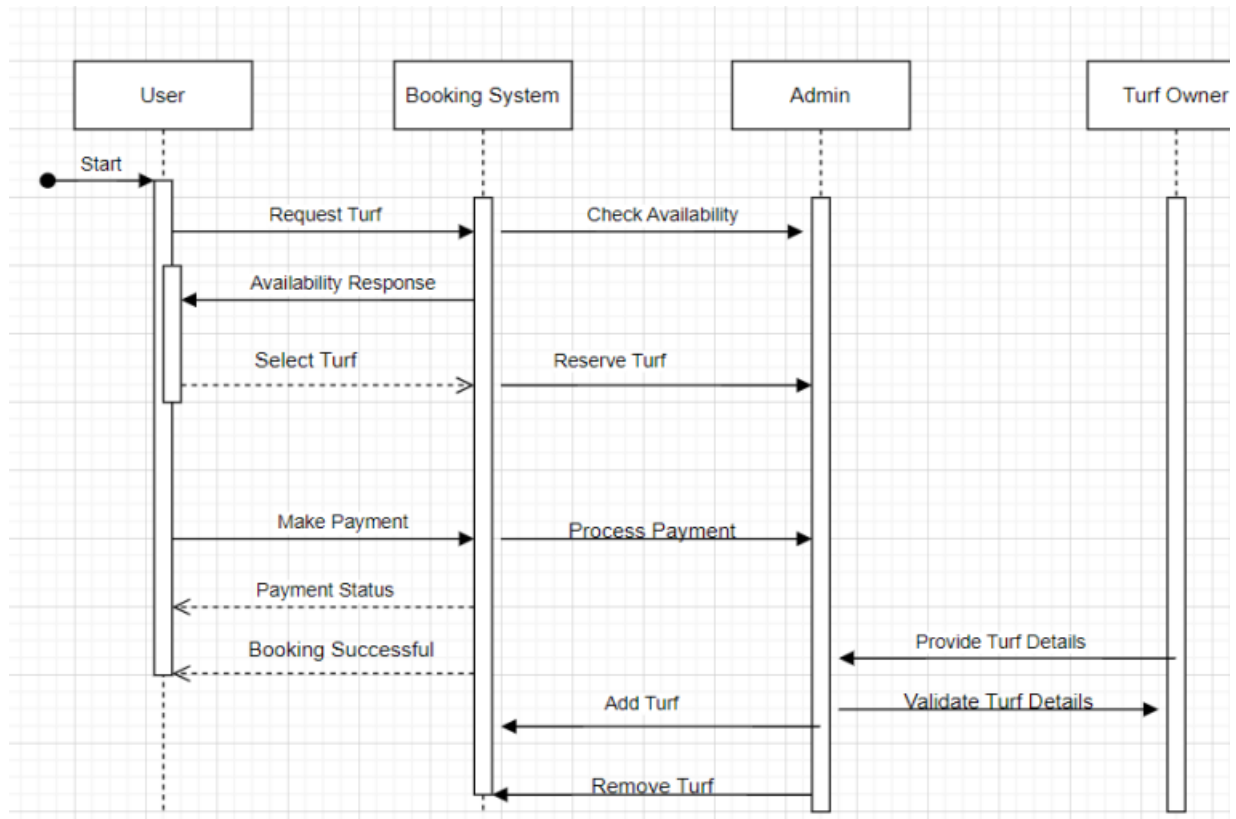


Figure 5.1: Architecture Diagram

5.2 Sequence Diagram



5.3 User Diagram

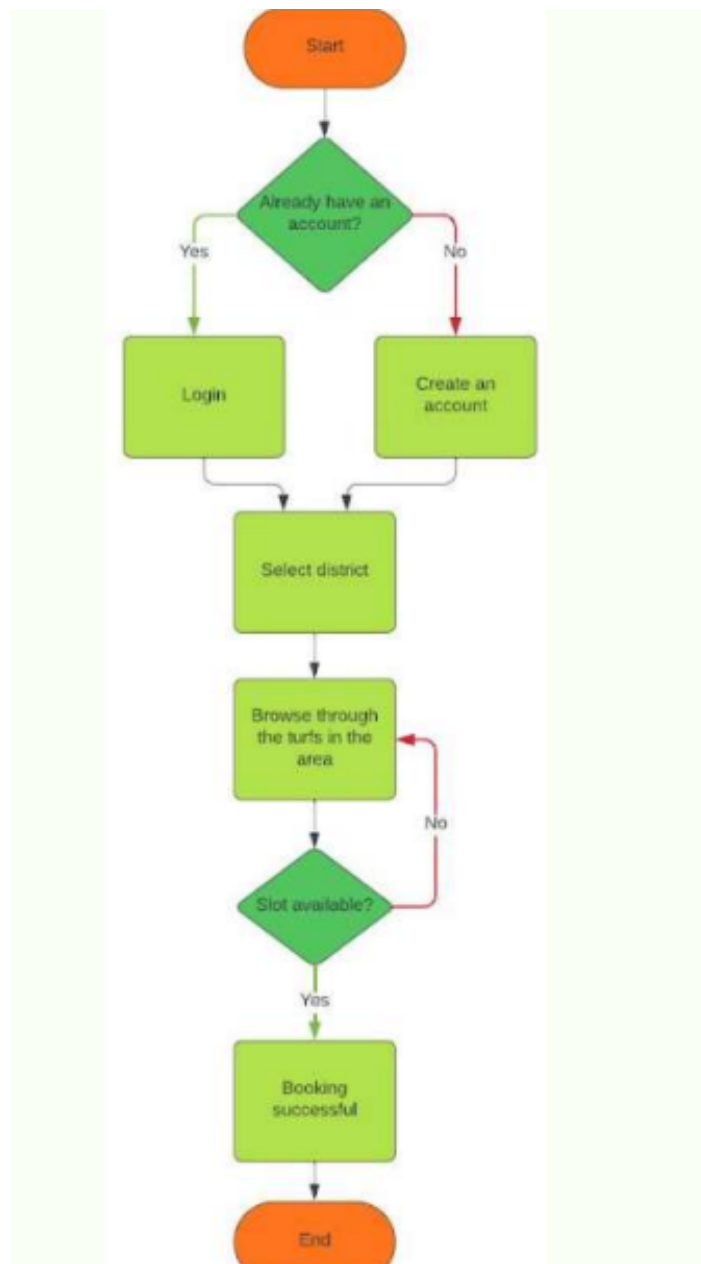


Figure 5.2: Flowchart for User

5.4 Admin Diagram

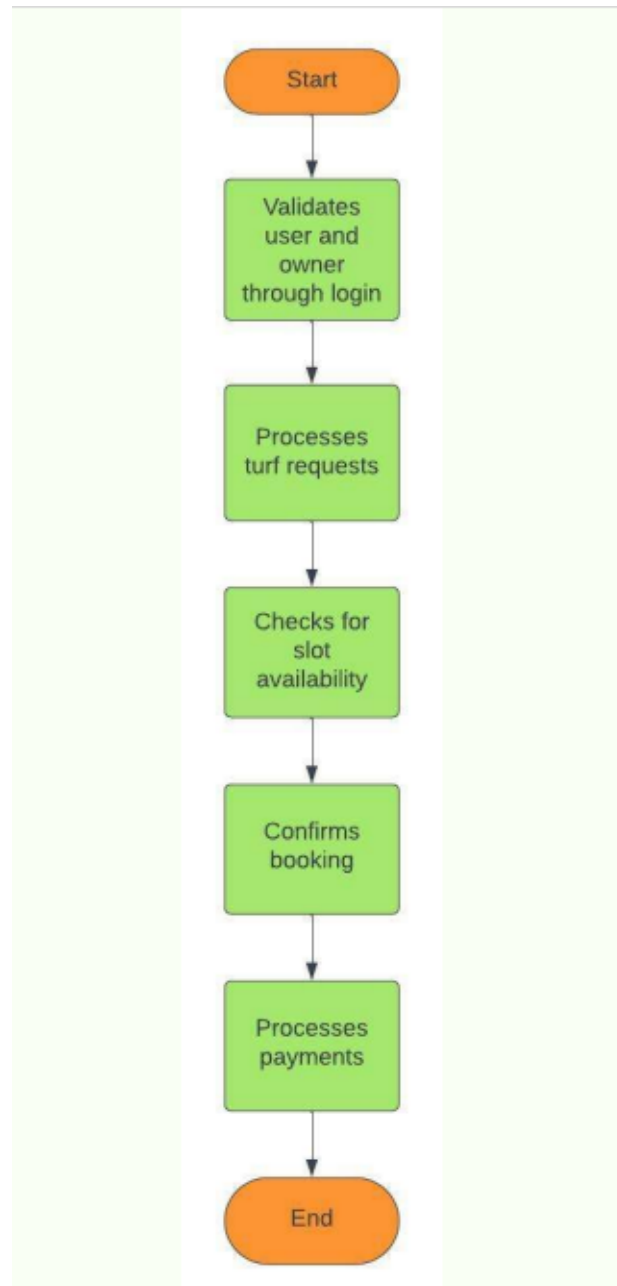


Figure 5.3: Flowchart for Admin

5.5 Turf Owner Diagram

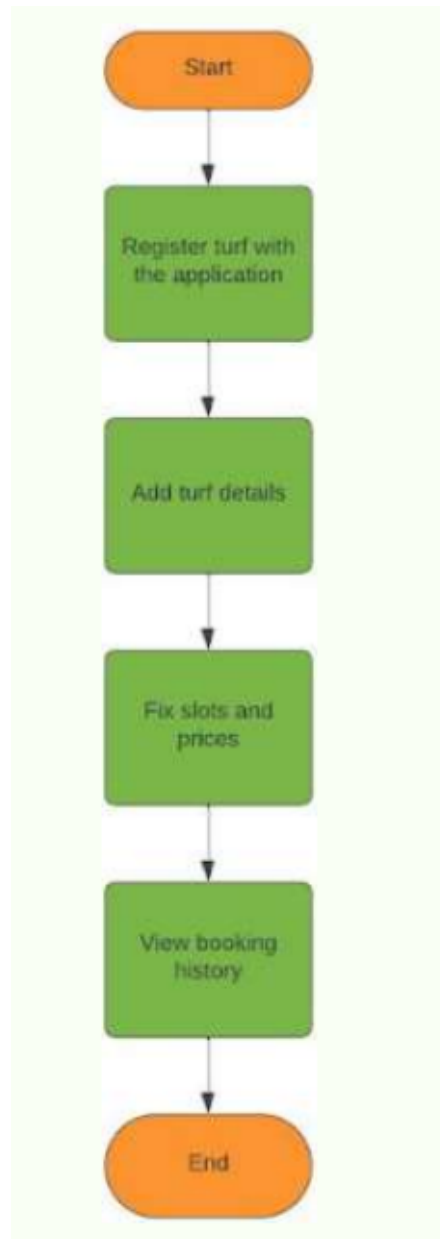


Figure 5.4: Flowchart for Turf Owner

Chapter 6

Testing

6.1 Unit Testing

Individual units of the turf booking app were tested in isolation. Each unit, such as a function or a module (Eg: login page for User or Turf Owner) , is tested to ensure it behaves as expected. In the context of the app, this might involve testing functions related to user authentication, turf availability, or booking confirmation.

6.2 Integration Testing

Focused on testing the interaction between different units or components of the app. The goal was to identify any issues that arise when these units are combined. For the Turf-Surf, this meant testing the integration between the user interface and the back-end server or testing how the booking module interacts with the payment gateway.

6.3 System Testing

Evaluated the entire system as a whole. Ensured that all components of the app function together as expected and meet the specified requirements. This involved testing various scenarios and use cases, including stress testing the app to assess its performance under high user loads. Cloud Firebase proved to be over-stressed when 100 user logins take place. Alternative to be adopted is to choose large databases like MySQL or Oracle.

6.4 Acceptance Testing

Acceptance testing is performed to determine whether the turf booking app meets the acceptance criteria set by the stakeholders. It typically involves end-to-end testing by real

users or stakeholders to validate that the app meets their expectations and requirements. The focus is on verifying that the app's functionalities align with the business needs.

6.5 Cross Device Testing

Cross device testing is crucial for a turf booking app as it needs to be accessible on various devices such as smartphones, tablets, and desktop computers. Testing was performed on different screen sizes to ensure consistent functionality and user experience across all devices.

6.6 Security Testing

Security testing is essential to identify and fix vulnerabilities that could compromise the confidentiality, integrity, and availability of the app and its users' data. This involved validating passwords of both user and turf-owner and ensuring that once logged out, there is no back button navigating to user profile.

6.7 User Interface Testing

Evaluated the app's visual elements and ensure that the user interface is user-friendly, intuitive, and visually appealing. It involves testing navigation, layout, responsiveness, and consistency of the app's interface.

Chapter 7

Results

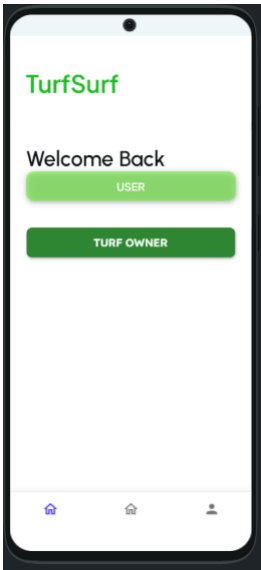


Figure 7.1: Welcome page

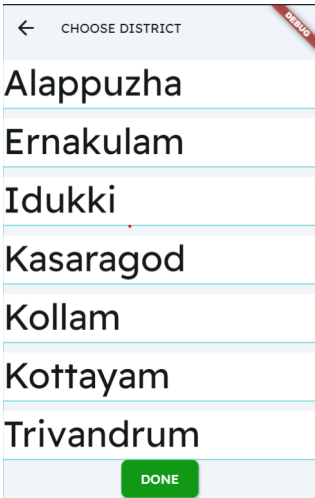


Figure 7.2: District selection page

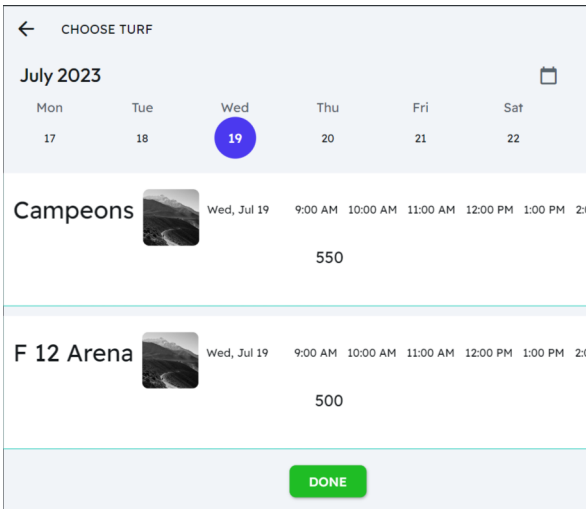


Figure 7.3: Turf view and slot selection

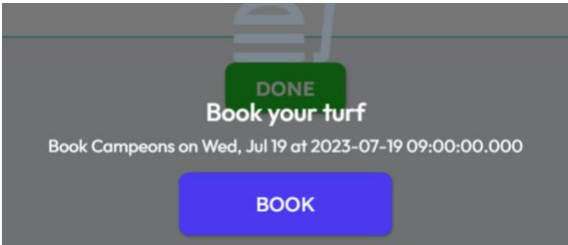
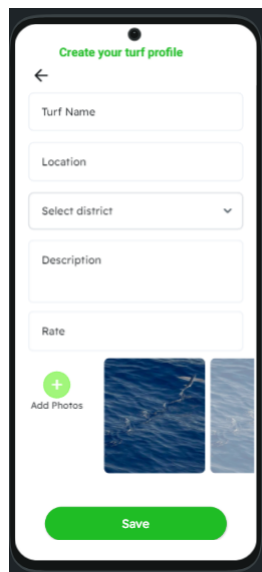
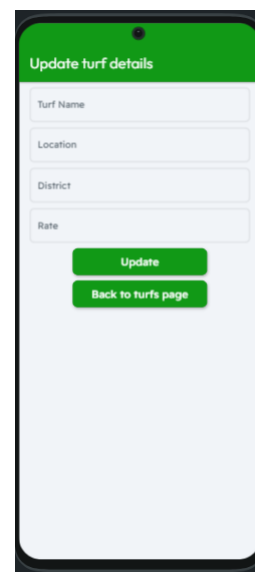


Figure 7.4: Booking confirmation message



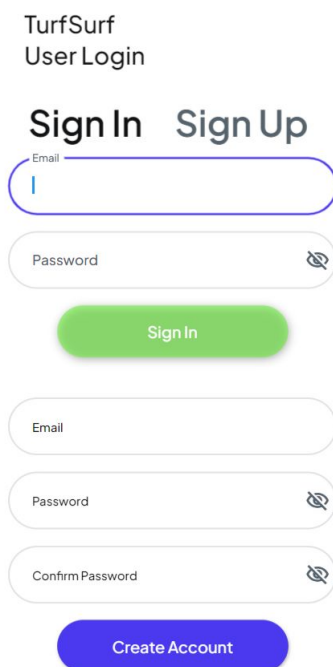
The image shows a mobile app screen titled "Create your turf profile". It features a back arrow at the top left. Below the title, there are input fields for "Turf Name", "Location", a "Select district" dropdown menu, "Description", and "Rate". At the bottom left, there is a green circular button with a plus sign labeled "Add Photos". To the right of this button are two placeholder images of a blue textured surface. A large green "Save" button is positioned at the bottom center of the screen.

Figure 7.5: Turf registration page



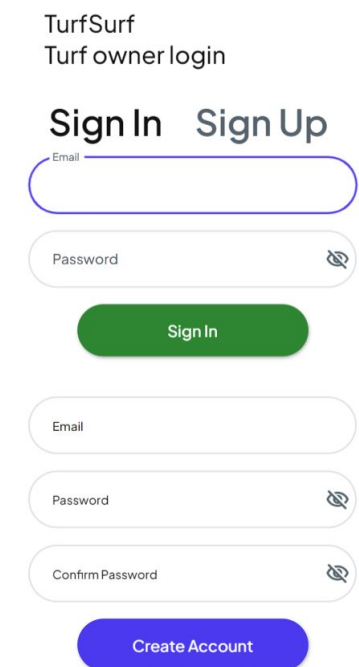
The image shows a mobile app screen titled "Update turf details". It has a green header bar with the title. Below the header, there are input fields for "Turf Name", "Location", "District", and "Rate". At the bottom, there are two green buttons: "Update" and "Back to turfs page".

Figure 7.6: Turf updation page



The image shows a web page titled "TurfSurf User Login". It has two tabs: "Sign In" (active) and "Sign Up". Under the "Sign In" tab, there are input fields for "Email" and "Password" with a toggle icon. Below these is a green "Sign In" button. Under the "Sign Up" tab, there are input fields for "Email", "Password", and "Confirm Password", each with a toggle icon. Below these is a blue "Create Account" button.

Figure 7.7: User login page



The image shows a web page titled "TurfSurf Turf owner login". It has two tabs: "Sign In" (active) and "Sign Up". Under the "Sign In" tab, there are input fields for "Email" and "Password" with a toggle icon. Below these is a green "Sign In" button. Under the "Sign Up" tab, there are input fields for "Email", "Password", and "Confirm Password", each with a toggle icon. Below these is a blue "Create Account" button.

Figure 7.8: Turf owner login page

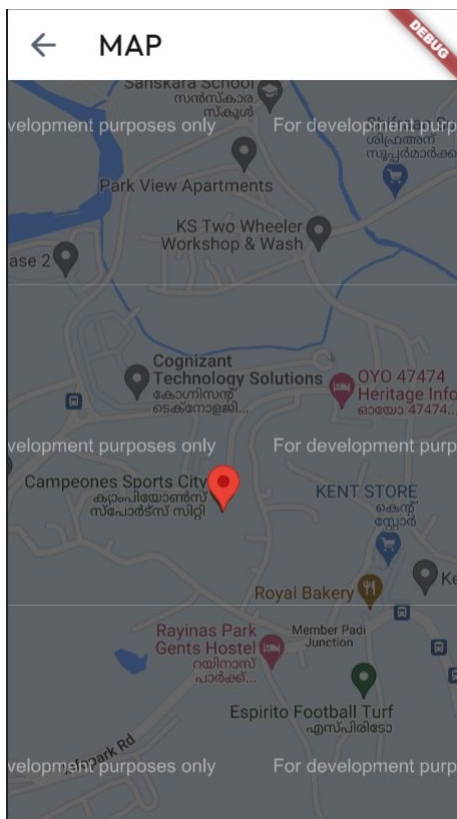


Figure 7.9: Turf view on map

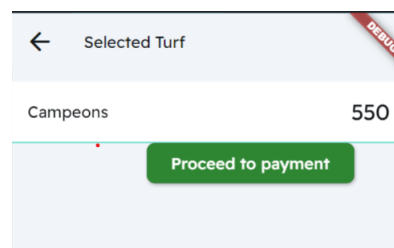


Figure 7.10: Payment page

Chapter 8

Risks and Challenges

1. Dependency on Internet Access: TurfSurf requires a stable internet connection to function. In areas with poor or unreliable signal, users may face difficulties accessing the system.
2. Limited Personal Interaction: With an online booking system, users may miss the personal interaction and direct communication they would have with facility or staff.
3. Users Who are Unfamiliar with Technology: These people may find it challenging to navigate and use an online booking system such as TurfSurf.

Chapter 9

Conclusion

Conclusion + Future scope

This project describes a turf booking system that progresses from the traditional method of turf booking as we know it into a hassle-free and intuitive android application by integrating features such as booking a particular turf of choice within a time slot, viewing of the turfs within a map and easy and straightforward payments made within the application via UPI.

Future Scope

We hope to extend the application by:

1. Adding different modes of payment such as by card, cash, paytm wallet, etc.
2. Adding tags to filter out turfs for different kinds of sports, for example, basketball courts, badminton courts, golf courses, etc.
3. Adding a feature to rent out a turf for an entire day or an extended period of time for tournaments and leagues.
4. Adding customer reviews and feedback taken from Google Maps API.
5. Adding a feature for Turf Owners to block or restrict certain users from accessing or booking their turfs
6. Adding a filter to search for turfs of different sizes, for example, 5v5 courts, 7v7 courts and 11v11 courts.

References

- [1] A. Juneja, R. Kumar and M. Gupta, "Digitization of Traditional Markets using DART based Cross Platform Development," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 2272-2276, doi: 10.1109/ICAC3N56670.2022.10074572.
- [2] J. Liu and J. Yu, "Research on Development of Android Applications," 2011 4th International Conference on Intelligent Networks and Intelligent Systems, Kuming, China, 2011, pp. 69-72, doi: 10.1109/ICINIS.2011.40.
- [3] Dhore B., Surabhi Thakar¹, Prajakta Kulkarni, Rasika Thorat, "Digital Table Booking and Food Ordering System Using Android Application" in International Journal of Emerging Engineering Research and Technology Volume 2, Issue 7, October 2014, PP 76-81.
- [4] K. Jiang and X. Gong, "The detail explanations on Google API development: the good combination of Google Maps and Google Earth" in , Beijing:Electronic Industry, China, pp. 150-163, 2008..
- [5] "Anonymous, "Google Maps. API", November 2010, [online] Available:.
- [6] Muhammad Helmi Bin Idris and Siti Zaiton Mohd Hashim, "UTM Sports App Mobile Application", UTM Computing Proceedings Innovation in Computing Technology and Applications, vol. 2, pp. 1-5, 2017.
- [7] A. M. Dean, "The DART real time control system," Third International Conference on Software Engineering for Real Time Systems, 1991., Cirencester, UK, 1991, pp. 86-92.
- [8] Bashir Omolaran Bello, Student-Teacher Online Booking Appointment System in Academic Institutions, November 2016.

- [9] V. Akshay, A. Kumar S., R. M. Alagappan and S. Gnanavel, "BOOKAZOR - an Online Appointment Booking System," 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Vellore, India, 2019, pp. 1-6, doi: 10.1109/ViTECoN.2019.8899460.

Appendix A: Sample Code

login.dart

```
import 'flutter_flow/flutter_flow_theme.dart';
import 'flutter_flow/flutter_flow_util.dart';
import 'flutter_flow/flutter_flow_widgets.dart';
import 'package:auto_size_text/auto_size_text.dart';
import 'package:flutter/material.dart';
import 'package:flutter/scheduler.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

import 'login_model.dart';
export 'login_model.dart';

class LoginWidget extends StatefulWidget {
  const LoginWidget({Key? key}) : super(key: key);

  @override
  _LoginWidgetState createState() => _LoginWidgetState();
}

class _LoginWidgetState extends State<LoginWidget> {
  late LoginModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => LoginModel());

    // On page load action.
    SchedulerBinding.instance.addPostFrameCallback((_) async {
      context.pushNamed('Districts');
    });
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    context.watch<FFAppState>();

    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_model.unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: Color(0xFFEEF6F9),
        body: SafeArea(
          top: true,
```

```

child: Row(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [
    Expanded(
      flex: 6,
      child: Container(
        width: 100,
        height: double.infinity,
        decoration: BoxDecoration(
          color: Colors.white,
        ),
        alignment: AlignmentDirectional(0, -1),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.max,
          children: [
            SingleChildScrollView(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.max,
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  if (responsiveVisibility(
                    context: context,
                    phone: false,
                    tablet: false,
                  ))
                    Container(
                      width: double.infinity,
                      height: 140,
                      decoration: BoxDecoration(
                        color: Colors.white,
                        borderRadius: BorderRadius.only(
                          bottomLeft: Radius.circular(16),
                          bottomRight: Radius.circular(16),
                          topLeft: Radius.circular(0),
                          topRight: Radius.circular(0),
                        ),
                      ),
                    ),
                  if (responsiveVisibility(
                    context: context,
                    tabletLandscape: false,
                    desktop: false,
                  ))
                    Container(
                      width: double.infinity,
                      height: 140,
                      decoration: BoxDecoration(
                        color: Colors.white,
                        borderRadius: BorderRadius.only(
                          bottomLeft: Radius.circular(16),
                          bottomRight: Radius.circular(16),
                          topLeft: Radius.circular(0),
                          topRight: Radius.circular(0),
                        ),
                      ),
                    ),
                ],
              ),
            ),
          ],
        ),
      ),
    ),
  ],
),

```

```

    ),
    alignment: AlignmentDirectional(-1, 0),
    child: Padding(
      padding: EdgeInsetsDirectional.fromSTEB(
        24, 0, 0, 0),
      child: Text(
        'TurfSurf',
        style: FlutterFlowTheme.of(context)
          .displaySmall
          .override(
            fontFamily: 'Plus Jakarta Sans',
            color: Color(0xFF1FBD25),
            fontSize: 36,
            fontWeight: FontWeight.w600,
          ),
      ),
    ),
  ),
),
Container(
  width: double.infinity,
  constraints: BoxConstraints(
    maxWidth: 430,
  ),
  decoration: BoxDecoration(
    color: Colors.white,
  ),
  child: Align(
    alignment: AlignmentDirectional(0, 0),
    child: Padding(
      padding: EdgeInsetsDirectional.fromSTEB(
        24, 24, 24, 24),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          Text(
            'Welcome Back',
            style: FlutterFlowTheme.of(context)
              .headlineLarge
              .override(
                fontFamily: 'Urbanist',
                color: Color(0xFF101213),
                fontSize: 32,
                fontWeight: FontWeight.w600,
              ),
          ),
          Padding(
            padding: EdgeInsetsDirectional.fromSTEB(
              0, 0, 0, 16),
            child: FFButtonWidget(
              onPressed: () async {
                context.pushNamed('userLogin');
              },
            ),
          ),
        ],
      ),
    ),
  ),
),

```

```
text: 'USER',
options: FFBUTTONOptions(
width: 340,
height: 44,
padding:
EdgeInsetsDirectional.fromSTEB(
0, 0, 0, 0),
iconPadding:
EdgeInsetsDirectional.fromSTEB(
0, 0, 0, 0),
color: Color(0x966BEF39),
textStyle:
FlutterFlowTheme.of(context)
.titleSmall
.override(
fontFamily:
'Plus Jakarta Sans',
color: Colors.white,
fontSize: 16,
fontWeight:
FontWeight.w500,
),
elevation: 3,
borderSide: BorderSide(
color: Colors.transparent,
width: 1,
),
borderRadius:
BorderRadius.circular(12),
hoverColor:
FlutterFlowTheme.of(context)
.secondary,
),
),
),
],
),
),
),
),
),
],
),
),
FFBUTTONWidget(
onPressed: () async {
context.pushNamed('ownerLogin');
},
text: 'TURF OWNER',
options: FFBUTTONOptions(
width: 310,
height: 44,
padding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
iconPadding:
```

```

        color: Color(0xFF2E8633),
        textStyle:
          FlutterFlowTheme.of(context).titleSmall.override(
            fontFamily: 'Readex Pro',
            color: Colors.white,
          ),
        elevation: 3,
        borderSide: BorderSide(
          color: Colors.transparent,
          width: 1,
        ),
        borderRadius: BorderRadius.circular(8),
      ),
    ),
    Padding(
      padding: EdgeInsetsDirectional.fromSTEB(0, 30, 0, 0),
      child: FFButtonWidget(
        onPressed: () async {
          context.pushNamed(
            'Maps',
            queryParameters: {
              'paymentsessionid': serializeParam(
                '0',
                ParamType.String,
              ),
              'amount': serializeParam(
                0.0,
                ParamType.double,
              ),
            }.withoutNulls,
          );
        },
        text: 'VIEW TURFS ON MAP',
        options: FFButtonOptions(
          height: 40,
          padding:
            EdgeInsetsDirectional.fromSTEB(24, 0, 24, 0),
          iconPadding:
            EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
          color: Color(0xFFA93132),
          textStyle: FlutterFlowTheme.of(context)
            .titleSmall
            .override(
              fontFamily: 'Readex Pro',
              color: Colors.white,
            ),
          elevation: 3,
          borderSide: BorderSide(
            color: Colors.transparent,
            width: 1,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
      ),
    ),
  ),
),

```



```

    ),
    ],
  ),
),
),
if (responsiveVisibility(
  context: context,
  phone: false,
  tablet: false,
))
Expanded(
  flex: 8,
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(16, 16, 16, 16),
    child: Container(
      width: 100,
      height: double.infinity,
      decoration: BoxDecoration(
        gradient: LinearGradient(
          colors: [Color(0xFF4B39EF), Color(0xFF39D2C0)],
          stops: [0, 1],
          begin: AlignmentDirectional(1, -1),
          end: AlignmentDirectional(-1, 1),
        ),
        borderRadius: BorderRadius.circular(16),
      ),
      child: Padding(
        padding: EdgeInsetsDirectional.fromSTEB(24, 24, 24, 24),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.max,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Container(
              width: double.infinity,
              constraints: BoxConstraints(
                maxWidth: 400,
              ),
              decoration: BoxDecoration(
                color: Colors.white,
                boxShadow: [
                  BoxShadow(
                    blurRadius: 3,
                    color: Color(0x2E000000),
                    offset: Offset(0, 2),
                  )
                ],
                borderRadius: BorderRadius.circular(8),
                border: Border.all(
                  color: Color(0xFFFF1F4F8),
                  width: 2,
                ),
              ),
            ),
            child: Padding(
              padding:

```

```

      EdgeInsetsDirectional.fromSTEB(4, 4, 4, 4),
child: Column(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [
    Padding(
      padding: EdgeInsetsDirectional.fromSTEB(
        12, 12, 12, 8),
child: Row(
  mainAxisAlignment: MainAxisAlignment.max,
  mainAxisAlignment:
    MainAxisAlignment.spaceBetween,
  crossAxisAlignment:
    CrossAxisAlignment.center,
  children: [
    Row(
      mainAxisAlignment: MainAxisAlignment.max,
      children: [
        Padding(
          padding: EdgeInsetsDirectional
            .fromSTEB(0, 0, 8, 0),
child: Container(
  width: 40,
  height: 40,
  decoration: BoxDecoration(
    color: Color(0xFFEEEEEE),
    shape: BoxShape.circle,
  ),
  child: Icon(
    Icons.person,
    color: Color(0xFF4B39EF),
    size: 24,
  ),
),
),
Text(
  'UserName',
  style:
    FlutterFlowTheme.of(context)
      .titleMedium
      .override(
        fontFamily:
          'Plus Jakarta Sans',
        color: Colors.white,
        fontSize: 18,
        fontWeight:
          FontWeight.w500,
      ),
),
],
),
Column(
  mainAxisAlignment: MainAxisAlignment.max,
  crossAxisAlignment:
    CrossAxisAlignment.end,

```

```
children: [
    Text(
      'Overall',
      style:
        FlutterFlowTheme.of(context)
          .bodySmall
          .override(
            fontFamily:
              'Plus Jakarta Sans',
            color:
              Color(0xFF101213),
            fontSize: 12,
            fontWeight:
              FontWeight.w500,
          ),
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.max,
      children: [
        Padding(
          padding:
            EdgeInsetsDirectional
              .fromSTEB(
                0, 0, 4, 0),
          child: Text(
            '5',
            style: FlutterFlowTheme
              .of(context)
              .headlineMedium
              .override(
                fontFamily:
                  'Plus Jakarta Sans',
                color: Color(
                  0xFF101213),
                fontSize: 24,
                fontWeight:
                  FontWeight.w500,
              ),
          ),
        ),
        Icon(
          Icons.star_rounded,
          color: Color(0xFF4B39EF),
          size: 20,
        ),
      ],
    ),
  ],
),
],
),
),
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(
```



```

import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:page_transition/page_transition.dart';
import 'package:provider/provider.dart';

import 'addturf_model.dart';
export 'addturf_model.dart';

class AddturfWidget extends StatefulWidget {
  const AddturfWidget({Key? key}) : super(key: key);

  @override
  _AddturfWidgetState createState() => _AddturfWidgetState();
}

class _AddturfWidgetState extends State<AddturfWidget> {
  late AddturfModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => AddturfModel());

    _model.turfNameController ??= TextEditingController();
    _model.locationController ??= TextEditingController();
    _model.myBioController ??= TextEditingController();
    _model.turfRateController ??= TextEditingController();
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    context.watch<FFAppState>();

    return Scaffold(
      key: scaffoldKey,
      backgroundColor: FlutterFlowTheme.of(context).primaryBtnText,
      appBar: PreferredSize(
        preferredSize: Size.fromHeight(60),
        child: AppBar(
          backgroundColor: FlutterFlowTheme.of(context).secondaryBackground,
          automaticallyImplyLeading: false,
          title: Padding(
            padding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 30),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.max,

```

```

mainAxisAlignment: MainAxisAlignment.end,
crossAxisAlignment: CrossAxisAlignment.start,
children: [
  Padding(
    padding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 8),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.max,
      children: [],
    ),
  ),
  Padding(
    padding: EdgeInsetsDirectional.fromSTEB(50, 0, 0, 0),
    child: Text(
      'Create your turf profile',
      style: FlutterFlowTheme.of(context).headlineMedium.override(
        fontFamily: 'Roboto',
        color: Color(0xFF1FBD25),
        fontSize: 18,
        fontWeight: FontWeight.bold,
      ),
    ),
  ),
  FlutterFlowIconButton(
    borderColor: Colors.transparent,
    borderRadius: 30,
    borderWidth: 1,
    buttonSize: 40,
    icon: Icon(
      Icons.arrow_back_rounded,
      color: FlutterFlowTheme.of(context).primaryText,
      size: 30,
    ),
    onPressed: () async {
      context.pushNamed('Login');
    },
  ),
],
),
),
actions: [],
centerTitle: true,
elevation: 0,
),
),
body: SafeArea(
  top: true,
  child: Column(
    mainAxisAlignment: MainAxisAlignment.max,
    children: [
      Padding(
        padding: EdgeInsetsDirectional.fromSTEB(20, 10, 20, 16),
        child: TextFormField(
          controller: _model.turfNameController,
          obscureText: false,

```

```

decoration: InputDecoration(
  labelText: 'Turf Name',
  labelStyle: FlutterFlowTheme.of(context).bodyLarge.override(
    fontFamily: 'Readex Pro',
    fontSize: 16,
  ),
  hintStyle: FlutterFlowTheme.of(context).bodySmall,
  enabledBorder: OutlineInputBorder(
    borderSide: BorderSide(
      color: FlutterFlowTheme.of(context).primaryBackground,
      width: 2,
    ),
    borderRadius: BorderRadius.circular(8),
  ),
  focusedBorder: OutlineInputBorder(
    borderSide: BorderSide(
      color: Color(0x00000000),
      width: 2,
    ),
    borderRadius: BorderRadius.circular(8),
  ),
  errorBorder: OutlineInputBorder(
    borderSide: BorderSide(
      color: Color(0x00000000),
      width: 2,
    ),
    borderRadius: BorderRadius.circular(8),
  ),
  focusedErrorBorder: OutlineInputBorder(
    borderSide: BorderSide(
      color: Color(0x00000000),
      width: 2,
    ),
    borderRadius: BorderRadius.circular(8),
  ),
  filled: true,
  fillColor: FlutterFlowTheme.of(context).secondaryBackground,
  contentPadding: EdgeInsetsDirectional.fromSTEB(20, 24, 0, 24),
),
style: FlutterFlowTheme.of(context).bodyMedium.override(
  fontFamily: 'Source Sans Pro',
),
maxLines: null,
validator:
  _model.turfNameControllerValidator.asValidator(context),
),
),
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(20, 0, 20, 16),
  child: TextFormField(
    controller: _model.locationController,
    obscureText: false,
    decoration: InputDecoration(
      labelText: 'Location',

```

```

labelStyle: FlutterFlowTheme.of(context).bodyLarge,
hintStyle: FlutterFlowTheme.of(context).bodySmall.override(
  fontFamily: 'Roboto',
),
enabledBorder: OutlineInputBorder(
  borderSide: BorderSide(
    color: FlutterFlowTheme.of(context).primaryBackground,
    width: 2,
  ),
  borderRadius: BorderRadius.circular(8),
),
focusedBorder: OutlineInputBorder(
  borderSide: BorderSide(
    color: Color(0x00000000),
    width: 2,
  ),
  borderRadius: BorderRadius.circular(8),
),
errorBorder: OutlineInputBorder(
  borderSide: BorderSide(
    color: Color(0x00000000),
    width: 2,
  ),
  borderRadius: BorderRadius.circular(8),
),
focusedErrorBorder: OutlineInputBorder(
  borderSide: BorderSide(
    color: Color(0x00000000),
    width: 2,
  ),
  borderRadius: BorderRadius.circular(8),
),
filled: true,
fillColor: FlutterFlowTheme.of(context).secondaryBackground,
contentPadding: EdgeInsetsDirectional.fromSTEB(20, 24, 0, 24),
),
style: FlutterFlowTheme.of(context).bodyMedium.override(
  fontFamily: 'Open Sans',
),
maxLines: null,
validator:
  _model.locationControllerValidator.asValidator(context),
),
),
Padding(
padding: EdgeInsetsDirectional.fromSTEB(20, 0, 20, 0),
child: FlutterFlowDropDown<String>(
  controller: _model.districtValueController ??=
    FormFieldController<String>(null),
  options: [
    'Alappuzha',
    'Ernakulam',
    'Idukki',
    'Kannur',

```



```

      'Kasargod',
      'Kollam',
      'Kottayam',
      'Kozhikode',
      'Malappuram',
      'Palakkad',
      'Pathanamthitta',
      'Thiruvananthapuram',
      'Thrissur',
      'Wayanad'
    ],
    onChanged: (val) => setState(() => _model.districtValue = val),
    width: double.infinity,
    height: 56,
    textStyle: FlutterFlowTheme.of(context).bodyLarge,
    hintText: 'Select district',
    icon: Icon(
      Icons.keyboard_arrow_down_rounded,
      color: FlutterFlowTheme.of(context).secondaryText,
      size: 15,
    ),
    fillColor: FlutterFlowTheme.of(context).secondaryBackground,
    elevation: 2,
    borderColor: FlutterFlowTheme.of(context).lineColor,
    borderWidth: 2,
    borderRadius: 8,
    margin: EdgeInsetsDirectional.fromSTEB(20, 4, 12, 4),
    hidesUnderline: true,
    isSearchable: false,
  ),
),
),
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(20, 15, 20, 12),
  child: TextFormField(
    controller: _model.myBioController,
    obscureText: false,
    decoration: InputDecoration(
      labelStyle: FlutterFlowTheme.of(context).bodySmall.override(
        fontFamily: 'Readex Pro',
        fontSize: 16,
      ),
    ),
    hintText: 'Description',
    hintStyle: FlutterFlowTheme.of(context).bodyLarge,
    enabledBorder: OutlineInputBorder(
      borderSide: BorderSide(
        color: FlutterFlowTheme.of(context).primaryBackground,
        width: 2,
      ),
      borderRadius: BorderRadius.circular(8),
    ),
    focusedBorder: OutlineInputBorder(
      borderSide: BorderSide(
        color: Color(0x00000000),
        width: 2,
      ),
    ),
  ),
),

```

```

    ),
    borderRadius: BorderRadius.circular(8),
  ),
  errorBorder: OutlineInputBorder(
    borderSide: BorderSide(
      color: Color(0x00000000),
      width: 2,
    ),
    borderRadius: BorderRadius.circular(8),
  ),
  focusedErrorBorder: OutlineInputBorder(
    borderSide: BorderSide(
      color: Color(0x00000000),
      width: 2,
    ),
    borderRadius: BorderRadius.circular(8),
  ),
  filled: true,
  fillColor: FlutterFlowTheme.of(context).secondaryBackground,
  contentPadding: EdgeInsetsDirectional.fromSTEB(20, 24, 0, 24),
),
style: FlutterFlowTheme.of(context).bodyMedium,
textAlign: TextAlign.start,
maxLines: 3,
validator: _model.myBioControllerValidator.asValidator(context),
),
),
Padding(
padding: EdgeInsetsDirectional.fromSTEB(20, 0, 20, 0),
child: TextFormField(
controller: _model.turfRateController,
obscureText: false,
decoration: InputDecoration(
labelText: 'Rate',
labelStyle: FlutterFlowTheme.of(context).bodySmall.override(
fontFamily: 'Readex Pro',
fontSize: 16,
),
hintStyle: FlutterFlowTheme.of(context).bodyLarge,
enabledBorder: OutlineInputBorder(
borderSide: BorderSide(
color: FlutterFlowTheme.of(context).primaryBackground,
width: 2,
),
borderRadius: BorderRadius.circular(8),
),
focusedBorder: OutlineInputBorder(
borderSide: BorderSide(
color: Color(0x00000000),
width: 2,
),
borderRadius: BorderRadius.circular(8),
),
errorBorder: OutlineInputBorder(

```

```

borderSide: BorderSide(
  color: Color(0x00000000),
  width: 2,
),
borderRadius: BorderRadius.circular(8),
),
focusedErrorBorder: OutlineInputBorder(
  borderSide: BorderSide(
    color: Color(0x00000000),
    width: 2,
  ),
  borderRadius: BorderRadius.circular(8),
),
filled: true,
fillColor: FlutterFlowTheme.of(context).secondaryBackground,
contentPadding: EdgeInsetsDirectional.fromSTEB(20, 24, 0, 24),
),
style: FlutterFlowTheme.of(context).bodyMedium.override(
  fontFamily: 'Readex Pro',
  fontSize: 16,
),
maxLines: null,
validator:
  _model.turfRateControllerValidator.asValidator(context),
),
),
Padding(
padding: EdgeInsetsDirectional.fromSTEB(0, 10, 0, 0),
child: Column(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [
    Row(
      mainAxisAlignment: MainAxisAlignment.max,
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        if (_model.uploadedFileUrls.length < 1)
          InkWell(
            splashColor: Colors.transparent,
            focusColor: Colors.transparent,
            hoverColor: Colors.transparent,
            highlightColor: Colors.transparent,
            onTap: () async {
              final selectedMedia = await selectMedia(
                mediaSource: MediaSource.photoGallery,
                multimage: true,
              );
              if (selectedMedia != null &&
                selectedMedia.every((m) => validateFileFormat(
                  m.storagePath, context))) {
                setState(() => _model.isDataUploading = true);
                var selectedUploadedFiles = <FFUploadedFile>[];

                var downloadUrls = <String>[];
                try {

```

```

showUploadMessage(
  context,
  'Uploading file...',
  showLoading: true,
);
selectedUploadedFiles = selectedMedia
  .map((m) => FFUploadedFile(
    name: m.storagePath.split('/').last,
    bytes: m.bytes,
    height: m.dimensions?.height,
    width: m.dimensions?.width,
    blurHash: m.blurHash,
  ))
  .toList();

downloadUrls = (await Future.wait(
  selectedMedia.map(
    (m) async => await uploadData(
      m.storagePath, m.bytes),
  ),
))
  .where((u) => u != null)
  .map((u) => u!)
  .toList();
} finally {
  ScaffoldMessenger.of(context)
    .hideCurrentSnackBar();
  _model.isDataUploading = false;
}
if (selectedUploadedFiles.length ==
  selectedMedia.length &&
  downloadUrls.length == selectedMedia.length) {
  setState(() {
    _model.uploadedLocalFiles =
      selectedUploadedFiles;
    _model.uploadedFileUrls = downloadUrls;
  });
  showUploadMessage(context, 'Success!');
} else {
  setState(() {});
  showUploadMessage(
    context, 'Failed to upload data');
  return;
}
},
child: Container(
  width: 125,
  height: 130,
  decoration: BoxDecoration(
    color: FlutterFlowTheme.of(context)
      .secondaryBackground,
  ),
  child: Padding(

```

```

padding:
  EdgeInsetsDirectional.fromSTEB(4, 0, 4, 0),
child: Column(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [
    Align(
      alignment: AlignmentDirectional(0, 0),
      child: Padding(
        padding: EdgeInsetsDirectional.fromSTEB(
          0, 20, 0, 0),
        child: FlutterFlowIconButton(
          borderColor: Color(0x9684F85A),
          borderRadius: 20,
          borderWidth: 1,
          buttonSize: 40,
          fillColor: Color(0xAD74F13D),
          icon: Icon(
            Icons.add,
            color: FlutterFlowTheme.of(context)
              .primaryBtnText,
            size: 24,
          ),
          onPressed: () {
            print('IconButton pressed ...');
          },
        ),
      ),
    ),
    Padding(
      padding: EdgeInsetsDirectional.fromSTEB(
        0, 2, 0, 0),
      child: Text(
        'Add Photos',
        style: FlutterFlowTheme.of(context)
          .bodyMedium,
      ),
    ),
  ],
),
),
),
),
),
),
),
),
if (_model.uploadedFileUrls.length >= 1)
Flexible(
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(4, 0, 4, 0),
    child: Builder(
      builder: (context) {
        final photoslist = _model.uploadedFileUrls
          .map((e) => e)
          .toList();
        return SingleChildScrollView(
          scrollDirection: Axis.horizontal,
          child: Row(

```

```
mainAxisSize: MainAxisSize.max,
children: List.generate(photoslist.length,
  (photoslistIndex) {
    final photoslistItem =
      photoslist[photoslistIndex];
    return Padding(
      padding: EdgeInsetsDirectional.fromSTEB(
        4, 0, 4, 0),
      child: InkWell(
        splashColor: Colors.transparent,
        focusColor: Colors.transparent,
        hoverColor: Colors.transparent,
        highlightColor: Colors.transparent,
        onTap: () async {
          await Navigator.push(
            context,
            PageTransition(
              type: PageTransitionType.fade,
              child:
                FlutterFlowExpandedImageView(
                  image: Image.network(
                    photoslistItem,
                    fit: BoxFit.contain,
                  ),
                  allowRotation: false,
                  tag: photoslistItem,
                  useHeroAnimation: true,
                ),
            ),
          );
        },
        child: Hero(
          tag: photoslistItem,
          transitionOnUserGestures: true,
          child: ClipRRect(
            borderRadius:
              BorderRadius.circular(8),
            child: Image.network(
              photoslistItem,
              width: 150,
              height: 130,
              fit: BoxFit.cover,
            ),
          ),
        ),
      ),
    );
  }
),
);
```

```

    ],
  ),
],
),
Align(
  alignment: AlignmentDirectional(0, 0.05),
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(10, 10, 10, 10),
    child: FFBUTTONWidget(
      onPressed: () async {
        context.goNamed('ownerView');
      },
      text: 'View Turfs',
      options: FFBUTTONOptions(
        width: 270,
        height: 30,
        padding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
        iconPadding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
        color: Color(0xFF2E8633),
        textStyle: FlutterFlowTheme.of(context)
          .titleMedium
          .override(
            fontFamily: 'Readex Pro',
            color: FlutterFlowTheme.of(context).primaryBtnText,
          ),
        elevation: 2,
        borderSide: BorderSide(
          color: Colors.transparent,
          width: 1,
        ),
        borderRadius: BorderRadius.circular(50),
      ),
    ),
  ),
),
Align(
  alignment: AlignmentDirectional(0, 0.05),
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(0, 10, 0, 0),
    child: FFBUTTONWidget(
      onPressed: () async {
        await TurfDetailsRecord.collection.doc().set({
          ...createTurfDetailsRecordData(
            turfName: _model.turfNameController.text,
            location: _model.locationController.text,
            district: _model.districtValue,
            description: _model.myBioController.text,
            rate: int.tryParse(_model.turfRateController.text),
          ),
          'turfImg': _model.uploadedFileUrls,
        });
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(

```

```

        content: Text(
          'Turf added successfully!',
          style: TextStyle(
            color: FlutterFlowTheme.of(context).primaryText,
          ),
        ),
        duration: Duration(milliseconds: 4000),
        backgroundColor: FlutterFlowTheme.of(context).secondary,
      ),
    );
  },
  text: 'Save',
  options: FFButtonOptions(
    width: 270,
    height: 30,
    padding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
    iconPadding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
    color: Color(0xFF1FBD25),
    textStyle: FlutterFlowTheme.of(context)
      .titleMedium
      .override(
        fontFamily: 'Readex Pro',
        color: FlutterFlowTheme.of(context).primaryBtnText,
      ),
    elevation: 2,
    borderSide: BorderSide(
      color: Colors.transparent,
      width: 1,
    ),
    borderRadius: BorderRadius.circular(50),
  ),
),
),
),
],
),
),
);
}
}

```

userView.dart

```

import '/backend/backend.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

import 'user_view_model.dart';
export 'user_view_model.dart';

```



```

class UserViewWidget extends StatefulWidget {
  const UserViewWidget({Key? key}) : super(key: key);

  @override
  _UserViewWidgetState createState() => _UserViewWidgetState();
}

class _UserViewWidgetState extends State<UserViewWidget> {
  late UserViewModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => UserViewModel());
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    context.watch<FFAppState>();

    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_model.unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).lineColor,
        appBar: AppBar(
          backgroundColor: Color(0xFF2FCD74),
          automaticallyImplyLeading: false,
          title: Text(
            'Turfs Available',
            style: FlutterFlowTheme.of(context).headlineMedium.override(
              fontFamily: 'Outfit',
              color: Colors.white,
              fontSize: 22,
            ),
          ),
          actions: [],
          centerTitle: false,
          elevation: 2,
        ),
        body: SafeArea(
          top: true,
          child: Row(
            mainAxisAlignment: MainAxisAlignment.max,
            children: [

```



```

        .bodyMedium
        .override(
          fontFamily: 'Readex Pro',
          fontSize: 20,
        ),
      ),
      Align(
        alignment: AlignmentDirectional(1, 0),
        child: Padding(
          padding:
            EdgeInsetsDirectional.fromSTEB(
              70, 20, 0, 0),
          child: FFButtonWidget(
            onPressed: () async {
              setState() {
                FFAppState().turfChosen =
                  FFAppState()
                    .turfChosen
                    .toList()
                    .cast<
                      DocumentReference>();
              });
            },
            text: 'View details',
            options: FFButtonOptions(
              width: 131,
              height: 42,
              padding: EdgeInsetsDirectional
                .fromSTEB(24, 0, 24, 0),
              iconPadding:
                EdgeInsetsDirectional
                  .fromSTEB(0, 0, 0, 0),
              color: Color(0xFF1F9955),
              textStyle:
                FlutterFlowTheme.of(context)
                  .titleSmall
                  .override(
                    fontFamily:
                      'Readex Pro',
                    color: Colors.white,
                  ),
              elevation: 3,
              borderSide: BorderSide(
                color: Colors.transparent,
                width: 1,
              ),
              borderRadius:
                BorderRadius.circular(8),
            ),
          ),
        ),
      ],
    ),
  ),

```

```
Row(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [
    InkWell(
      splashColor: Colors.transparent,
      focusColor: Colors.transparent,
      hoverColor: Colors.transparent,
      highlightColor: Colors.transparent,
      onTap: () async {
        context.pushNamed(
          'Maps',
          queryParameters: {
            'paymentsessionid':
              serializeParam(
                '9979',
                ParamType.String,
              ),
            'amount': serializeParam(
              9009.0,
              ParamType.double,
            ),
          }.withoutNulls,
          extra: <String, dynamic>{
            kTransitionInfoKey:
              TransitionInfo(
                hasTransition: true,
                transitionType:
                  PageTransitionType.fade,
                duration:
                  Duration(milliseconds: 0),
              ),
          },
        );
      },
    child: Text(
      'Location : ${listViewTurfDetailsRecord.location}',
      style: FlutterFlowTheme.of(context)
        .bodyMedium
        .override(
          fontFamily: 'Readex Pro',
          fontSize: 20,
        ),
    ),
  ],
);
```

```

        ],
      ),
    ),
  ],
),
),
),
);
}
}

```

chooseTurf.dart

```

import 'auth/firebase_auth/auth_util.dart';
import 'backend/backend.dart';
import 'components/book_action_widget.dart';
import 'flutter_flow/flutter_flow_calendar.dart';
import 'flutter_flow/flutter_flow_icon_button.dart';
import 'flutter_flow/flutter_flow_theme.dart';
import 'flutter_flow/flutter_flow_util.dart';
import 'flutter_flow/flutter_flow_widgets.dart';
import 'flutter_flow/custom_functions.dart' as functions;
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter/scheduler.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

import 'choose_turf_model.dart';
export 'choose_turf_model.dart';

class ChooseTurfWidget extends StatefulWidget {
  const ChooseTurfWidget({
    Key? key,
    this.districtRef1,
  }) : super(key: key);

  final DocumentReference? districtRef1;

  @override
  _ChooseTurfWidgetState createState() => _ChooseTurfWidgetState();
}

class _ChooseTurfWidgetState extends State<ChooseTurfWidget> {
  late ChooseTurfModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => ChooseTurfModel());

    // On page load action.
    SchedulerBinding.instance.addPostFrameCallback((_) async {

```

```

    setState(() {
      _model.startDate = getCurrentTimestamp;
    });
  });
}

@override
void dispose() {
  _model.dispose();

  super.dispose();
}

@override
Widget build(BuildContext context) {
  context.watch<FFAppState>();

  return GestureDetector(
    onTap: () => FocusScope.of(context).requestFocus(_model.unfocusNode),
    child: Scaffold(
      key: scaffoldKey,
      backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
      appBar: AppBar(
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        automaticallyImplyLeading: false,
        leading: FlutterFlowIconButton(
          borderColor: Colors.transparent,
          borderRadius: 30,
          borderWidth: 1,
          buttonSize: 60,
          icon: Icon(
            Icons.arrow_back_rounded,
            color: FlutterFlowTheme.of(context).primaryText,
            size: 30,
          ),
          onPressed: () async {
            context.goNamed('Login');
          },
        ),
        title: Text(
          'CHOOSE TURF',
          style: FlutterFlowTheme.of(context).bodyLarge,
        ),
        actions: [],
        centerTitle: false,
        elevation: 0,
      ),
      body: SafeArea(
        top: true,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.max,
          children: [
            SingleChildScrollView(
              child: Column(

```

```

mainAxisSize: MainAxisSize.max,
crossAxisAlignment: CrossAxisAlignment.start,
children: [
  FlutterFlowCalendar(
    color: FlutterFlowTheme.of(context).primary,
    iconColor: FlutterFlowTheme.of(context).secondaryText,
    weekFormat: true,
    weekStartsMonday: true,
    initialDate: _model.startDate,
    rowHeight: 64,
    onChange: (DateTimeRange? newSelectedDate) async {
      _model.calendarSelectedDay = newSelectedDate;
      setState(() {
        _model.startDate = _model.calendarSelectedDay?.start;
      });
      setState(() {});
    },
    titleStyle: FlutterFlowTheme.of(context).headlineSmall,
    dayOfWeekStyle: FlutterFlowTheme.of(context).labelLarge,
    dateStyle: FlutterFlowTheme.of(context).bodyMedium,
    selectedDateStyle:
      FlutterFlowTheme.of(context).titleSmall,
    inactiveDateStyle:
      FlutterFlowTheme.of(context).labelMedium,
  ),
  StreamBuilder<List<TurfDetailsRecord>>(
    stream: queryTurfDetailsRecord(),
    builder: (context, snapshot) {
      // Customize what your widget looks like when it's loading.
      if (!snapshot.hasData) {
        return Center(
          child: SizedBox(
            width: 50,
            height: 50,
            child: CircularProgressIndicator(
              valueColor: AlwaysStoppedAnimation<Color>(
                FlutterFlowTheme.of(context).primary,
              ),
            ),
          ),
        );
      }
    },
  ),
  List<TurfDetailsRecord> listViewTurfDetailsRecordList =
    snapshot.data!;
  return ListView.builder(
    padding: EdgeInsets.zero,
    primary: false,
    shrinkWrap: true,
    scrollDirection: Axis.vertical,
    itemCount: listViewTurfDetailsRecordList.length,
    itemBuilder: (context, listViewIndex) {
      final listViewTurfDetailsRecord =
        listViewTurfDetailsRecordList[listViewIndex];
      return Padding(

```

```

padding:
  EdgeInsetsDirectional.fromSTEB(0, 12, 0, 1),
child: StreamBuilder<List<BookingRecord>>(
  stream: queryBookingRecord(),
  builder: (context, snapshot) {
    // Customize what your widget looks like when it's loading.
    if (!snapshot.hasData) {
      return Center(
        child: SizedBox(
          width: 50,
          height: 50,
          child: CircularProgressIndicator(
            valueColor:
              AlwaysStoppedAnimation<Color>(
                FlutterFlowTheme.of(context)
                  .primary,
              ),
        ),
      ),
    );
  }
  List<BookingRecord>
    containerBookingRecordList =
      snapshot.data!;
  return InkWell(
    splashColor: Colors.transparent,
    focusColor: Colors.transparent,
    hoverColor: Colors.transparent,
    highlightColor: Colors.transparent,
    onTap: () async {
      context.pushNamed('TurfTransactions');
    },
    child: Container(
      width: double.infinity,
      height: 199,
      decoration: BoxDecoration(
        color: FlutterFlowTheme.of(context)
          .secondaryBackground,
        boxShadow: [
          BoxShadow(
            color: FlutterFlowTheme.of(context)
              .secondary,
            offset: Offset(0, 1),
          )
        ],
      ),
    child: SingleChildScrollView(
      scrollDirection: Axis.horizontal,
      child: Row(
        mainAxisAlignment: MainAxisAlignment.max,
        crossAxisAlignment:
          CrossAxisAlignment.center,
        children: [
          Align(

```



```

alignment:
  AlignmentDirectional(0, 0),
child: StreamBuilder<
  List<UsersRecord>>(
    stream: queryUsersRecord(
      queryBuilder: (usersRecord) =>
        usersRecord.where('email',
          isEqualTo:
            currentUserEmail),
      singleRecord: true,
    ),
    builder: (context, snapshot) {
      // Customize what your widget looks like when it's loading.
      if (!snapshot.hasData) {
        return Center(
          child: SizedBox(
            width: 50,
            height: 50,
            child:
              CircularProgressIndicator(
                valueColor:
                  AlwaysStoppedAnimation<
                    Color>(
                      FlutterFlowTheme.of(
                        context)
                        .primary,
                    ),
                ),
          ),
        );
      }
      List<UsersRecord>
        columnUsersRecordList =
          snapshot.data!;
      // Return an empty Container when the item does not exist.
      if (snapshot.data!.isEmpty) {
        return Container();
      }
      final columnUsersRecord =
        columnUsersRecordList
          .isEmpty
            ? columnUsersRecordList
              .first
            : null;
      return InkWell(
        splashColor:
          Colors.transparent,
        focusColor:
          Colors.transparent,
        hoverColor:
          Colors.transparent,
        highlightColor:
          Colors.transparent,
        onTap: () async {

```

```

context.pushNamed(
  'TurfTransactions');
},
child: Column(
  mainAxisAlignment:
    MainAxisAlignment.max,
  children: [
    Row(
      mainAxisAlignment:
        MainAxisAlignment.max,
      children: [
        Padding(
          padding:
            EdgeInsetsDirectional
              .fromSTEB(
                10,
                20,
                0,
                0),
          child: ClipRRect(
            borderRadius:
              BorderRadius
                .circular(
                  8),
            child: Image
              .network(
                'https://picsum.photos/seed/426/600',
                width: 84,
                height: 77,
                fit: BoxFit
                  .cover,
              ),
          ),
        ),
      ],
    Align(
      alignment:
        AlignmentDirectional(
          0, 6),
      child: Padding(
        padding:
          EdgeInsetsDirectional
            .fromSTEB(
              10,
              0,
              20,
              0),
        child:
          SelectionArea(
            child:
              Text(
                dateFormat(
                  'MMMEd',
                  functions
                    .getAvailSlots(

```

```
e.time).withoutNulls.toList(),
        _model.startDate!)
      .first),
    style: FlutterFlowTheme.of(
      context)
      .bodyMedium,
  )),
),
),
Padding(
  padding:
    EdgeInsetsDirectional
      .fromSTEB(
        12,
        0,
        10,
        0),
  child: Builder(
    builder:
      (context) {
        final availableSlots = functions
          .getAvailSlots(
            containerBookingRecordList
              .map((e) => e.time)
              .withoutNulls
              .toList(),
            _model.startDate!)
          .toList();
        return Wrap(
          spacing: 12,
          runSpacing:
            0,
          alignment:
            WrapAlignment
              .start,
          crossAxisAlignment:
            WrapCrossAlignment
              .start,
          direction: Axis
            .horizontal,
          runAlignment:
            WrapAlignment
              .start,
          verticalDirection:
            VerticalDirection
              .down,
          clipBehavior:
            Clip.none,
          children: List.generate(
            availableSlots
              .length,
            (availableSlotsIndex) {
              final availableSlotsItem =
```

```

        availableSlots[
          availableSlotsIndex];
return InkWell(
  splashColor:
    Colors.transparent,
  focusColor:
    Colors.transparent,
  hoverColor:
    Colors.transparent,
  highlightColor:
    Colors.transparent,
  onTap:
    () async {
      await showModalBottomSheet(
        isScrollControlled:
          true,
        backgroundColor:
          Colors.transparent,
        enableDrag:
          false,
        context:
          context,
        builder:
          (context) {
            return GestureDetector(
              onTap: () =>
                FocusScope.of(context).requestFocus(_model.unfocusNode),
              child: Padding(
                padding: MediaQuery.viewInsetsOf(context),
                child: BookActionWidget(
                  tname: listViewTurfDetailsRecord.turfName,
                  time: availableSlotsItem,
                  uid: columnUsersRecord!.email,
                ),
              ),
            );
          },
      ).then((value) =>
        setState(() {}));

      await TurfDetailsRecord
        .collection
        .doc()
        .set(createTurfDetailsRecordData(
          turfName: listViewTurfDetailsRecord.turfName,
        ));
    },
  child:
    Text(
      dateTimeFormat(
        'jm',
        availableSlotsItem),
      style:
        FlutterFlowTheme.of(context).bodyMedium,

```

```

        ),
      );
    }},
  );
},
),
),
],
),
Padding(
  padding:
    EdgeInsetsDirectional
      .fromSTEB(0,
        0, 0, 20),
  child: Row(
    mainAxisAlignment:
      MainAxisAlignment
        .max,
    children: [
      Padding(
        padding:
          EdgeInsetsDirectional
            .fromSTEB(
              0,
              2,
              250,
              10),
        child: Text(
          listViewTurfDetailsRecord
            .rate
              .toString(),
          style: FlutterFlowTheme.of(
            context)
            .bodyMedium
              .override(
                fontFamily:
                  'Readex Pro',
                fontSize:
                  20,
              ),
        ),
      ),
    ],
  ),
),
Padding(
  padding:
    EdgeInsetsDirectional
      .fromSTEB(1,
        0, 0, 0),
  child: Text(
    listViewTurfDetailsRecord
      .turfName,
    style: FlutterFlowTheme

```

```

        .of(context)
        .bodyMedium
        .override(
          fontFamily:
            'Readex Pro',
          fontSize: 30,
        ),
      ),
    ),
  ],
),
);
},
),
FlutterFlowIconButton(
  borderColor:
    FlutterFlowTheme.of(context)
      .primary,
  borderRadius: 20,
  borderWidth: 1,
  buttonSize: 40,
  fillColor:
    FlutterFlowTheme.of(context)
      .accent1,
  icon: Icon(
    Icons.add,
    color:
      FlutterFlowTheme.of(context)
        .primaryText,
    size: 24,
  ),
  onPressed: () async {
    setState(() {
      FFAppState().addToTurfChosen(
        listViewTurfDetailsRecord
          .reference);
    });
    ScaffoldMessenger.of(context)
      .showSnackBar(
        SnackBar(
          content: Text(
            'Turf added!',
            style: TextStyle(
              color:
                FlutterFlowTheme.of(
                  context)
                    .primaryText,
            ),
          ),
          duration: Duration(
            milliseconds: 4000),
          backgroundColor:
            FlutterFlowTheme.of(

```


maps.dart

```
import '/backend/backend.dart';
import '/flutter_flow/flutter_flow_google_map.dart';
import '/flutter_flow/flutter_flow_icon_button.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

import 'maps_model.dart';
export 'maps_model.dart';

class MapsWidget extends StatefulWidget {
  const MapsWidget({
    Key? key,
    required this.paymentsessionid,
    required this.amount,
  }) : super(key: key);

  final String? paymentsessionid;
  final double? amount;

  @override
  _MapsWidgetState createState() => _MapsWidgetState();
}

class _MapsWidgetState extends State<MapsWidget> {
  late MapsModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => MapsModel());
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    context.watch<FFAppState>();

    return Scaffold(
      key: scaffoldKey,
      resizeToAvoidBottomInset: false,
      backgroundColor: Color(0xFFEEF6F9),
    );
  }
}
```



```

appBar: AppBar(
  backgroundColor: FlutterFlowTheme.of(context).secondaryBackground,
  automaticallyImplyLeading: false,
  leading: FlutterFlowIconButton(
    borderColor: Colors.transparent,
    borderRadius: 30,
    borderWidth: 1,
    buttonSize: 60,
    icon: Icon(
      Icons.arrow_back_rounded,
      color: FlutterFlowTheme.of(context).secondaryText,
      size: 30,
    ),
    onPressed: () async {
      context.pop();
    },
  ),
  title: Text(
    'MAP',
    style: FlutterFlowTheme.of(context).headlineSmall,
  ),
  actions: [],
  centerTitle: false,
  elevation: 0,
),
body: Column(
  mainAxisAlignment: MainAxisAlignment.max,
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      child: StreamBuilder<List<TurfDetailsRecord>>(
        stream: queryTurfDetailsRecord(),
        builder: (context, snapshot) {
          // Customize what your widget looks like when it's loading.
          if (!snapshot.hasData) {
            return Center(
              child: SizedBox(
                width: 50,
                height: 50,
                child: CircularProgressIndicator(
                  valueColor: AlwaysStoppedAnimation<Color>(
                    FlutterFlowTheme.of(context).primary,
                  ),
                ),
              ),
            );
          }
        },
      ),
    ),
    List<TurfDetailsRecord> googleMapTurfDetailsRecordList =
      snapshot.data!;
    return FlutterFlowGoogleMap(
      controller: _model.googleMapsController,
      onCameraIdle: (latLng) => _model.googleMapsCenter = latLng,
      initialLocation: _model.googleMapsCenter ??=
        LatLng(9.9935, 76.3581),
    ),
  ],
),
)

```

```

markers: googleMapTurfDetailsRecordList
  .map(
    (googleMapTurfDetailsRecord) => FlutterFlowMarker(
      googleMapTurfDetailsRecord.reference.path,
      googleMapTurfDetailsRecord.geopoint!,
    ),
  )
  .toList(),
markerColor: GoogleMarkerColor.red,
mapType: MapType.normal,
style: GoogleMapStyle.standard,
initialZoom: 14,
allowInteraction: true,
allowZoom: true,
showZoomControls: true,
showLocation: true,
showCompass: false,
showMapToolbar: false,
showTraffic: false,
centerMapOnMarkerTap: true,
);
},
),
),
],
),
);
}
}

```

slotSelect.dart

```

import 'auth/firebase_auth/auth_util.dart';
import 'backend/backend.dart';
import 'flutter_flow/flutter_flow_theme.dart';
import 'flutter_flow/flutter_flow_util.dart';
import 'flutter_flow/flutter_flow_widgets.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

import 'update_slot_model.dart';
export 'update_slot_model.dart';

class UpdateSlotWidget extends StatefulWidget {
  const UpdateSlotWidget({
    Key? key,
    required this.turfRef,
  }) : super(key: key);

  final DocumentReference? turfRef;

  @override
  _UpdateSlotWidgetState createState() => _UpdateSlotWidgetState();

```

```
}
```

```
class _UpdateSlotWidgetState extends State<UpdateSlotWidget> {
  late UpdateSlotModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => UpdateSlotModel());

    _model.textController ??= TextEditingController();
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    context.watch<FFAppState>();

    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_model.unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: Color(0xFF119916),
          automaticallyImplyLeading: false,
          title: Text(
            'booked',
            style: FlutterFlowTheme.of(context).headlineMedium.override(
              fontFamily: 'Outfit',
              color: Colors.white,
              fontSize: 22,
            ),
          ),
          actions: [],
          centerTitle: false,
          elevation: 2,
        ),
        body: SafeArea(
          top: true,
          child: StreamBuilder<TurfDetailsRecord>(
            stream: TurfDetailsRecord.getDocument(widget.turfRef!),
            builder: (context, snapshot) {
              // Customize what your widget looks like when it's loading.
              if (!snapshot.hasData) {
                return Center(
```

```

child: SizedBox(
  width: 50,
  height: 50,
  child: CircularProgressIndicator(
    valueColor: AlwaysStoppedAnimation<Color>(
      FlutterFlowTheme.of(context).primary,
    ),
  ),
),
);
}
final columnTurfDetailsRecord = snapshot.data!;
return Column(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [
    Column(
      mainAxisAlignment: MainAxisAlignment.max,
      children: [
        Row(
          mainAxisAlignment: MainAxisAlignment.max,
          children: [
            Expanded(
              child: Padding(
                padding:
                  EdgeInsetsDirectional.fromSTEB(8, 0, 8, 0),
                child: TextFormField(
                  controller: _model.textController,
                  autofocus: true,
                  obscureText: false,
                  decoration: InputDecoration(
                    labelText: 'Enter slot',
                    labelStyle:
                      FlutterFlowTheme.of(context).labelMedium,
                    hintStyle:
                      FlutterFlowTheme.of(context).labelMedium,
                    enabledBorder: UnderlineInputBorder(
                      borderSide: BorderSide(
                        color: FlutterFlowTheme.of(context)
                          .alternate,
                        width: 2,
                      ),
                      borderRadius: BorderRadius.circular(8),
                    ),
                    focusedBorder: UnderlineInputBorder(
                      borderSide: BorderSide(
                        color:
                          FlutterFlowTheme.of(context).primary,
                        width: 2,
                      ),
                      borderRadius: BorderRadius.circular(8),
                    ),
                    errorBorder: UnderlineInputBorder(
                      borderSide: BorderSide(
                        color: FlutterFlowTheme.of(context).error,

```

```

        width: 2,
      ),
      borderRadius: BorderRadius.circular(8),
    ),
    focusedErrorBorder: UnderlineInputBorder(
      borderSide: BorderSide(
        color: FlutterFlowTheme.of(context).error,
        width: 2,
      ),
      borderRadius: BorderRadius.circular(8),
    ),
  ),
  style: FlutterFlowTheme.of(context).bodyMedium,
  validator: _model.textControllerValidator
    .asValidator(context),
),
),
),
],
),
Row(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [],
),
Row(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [],
),
Row(
  mainAxisAlignment: MainAxisAlignment.max,
  children: [],
),
],
),
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(0, 8, 0, 0),
  child: FFButtonWidget(
    onPressed: () async {
      await widget.turfRef!
        .update(createTurfDetailsRecordData(
          slot1: _model.textController.text,
        ));
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(
            'booked',
            style: TextStyle(
              color: FlutterFlowTheme.of(context).primaryText,
            ),
          ),
          duration: Duration(milliseconds: 4000),
          backgroundColor:
            FlutterFlowTheme.of(context).secondary,
        ),
      ),
    ),
  ),
),

```

```
;
},
text: 'Update',
options: FFBUTTON_OPTIONS(
width: 200,
height: 40,
padding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
iconPadding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
color: Color(0xFF119916),
TextStyle:
FlutterFlowTheme.of(context).titleSmall.override(
fontFamily: 'Readex Pro',
color: Colors.white,
),
elevation: 3,
borderSide: BorderSide(
color: Colors.transparent,
width: 1,
),
borderRadius: BorderRadius.circular(8),
),
),
),
),
),
),
);
```

```

    ),
  ),
);
}
}

```

pubspec.yaml

name: turf_booking
description: A new Flutter project.

environment:
 sdk: ">=3.0.0 <4.0.0"

dependencies:

```

flutter:
  sdk: flutter
auto_size_text: 3.0.0
cached_network_image: 3.2.1
cloud_firestore: 4.8.0
cloud_firestore_platform_interface: 5.15.0
cloud_firestore_web: 3.6.0
dropdown_button2: 2.0.0
equatable: 2.0.5
file_picker: 5.2.6
firebase_auth: 4.6.2
firebase_auth_platform_interface: 6.15.2
firebase_auth_web: 5.5.2
firebase_core: 2.13.1
firebase_core_platform_interface: 4.8.0
firebase_core_web: 2.5.0
flutter_animate: 4.1.1+1
flutter_cache_manager: 3.3.0
flutter_plugin_android_lifecycle: 2.0.9
flutter_rating_bar: 4.0.1
font_awesome_flutter: 10.1.0
from_css_color: 2.0.0
go_router: 7.1.1
google_fonts: 4.0.3
google_maps: 6.3.0
google_maps_flutter: 2.2.6
google_maps_flutter_platform_interface: 2.2.6
google_maps_flutter_web: 0.4.0+8
google_sign_in: 6.0.2
google_sign_in_android: 6.1.8
google_sign_in_ios: 5.6.1
google_sign_in_platform_interface: 2.4.0
google_sign_in_web: 0.11.0+2
image_picker: 0.8.5+3
image_picker_android: 0.8.5+3
image_picker_for_web: 2.1.10
image_picker_ios: 0.8.6+1
image_picker_platform_interface: 2.6.2
intl: 0.18.0

```

json_path: 0.4.1
mime_type: 1.0.0
page_transition: 2.0.4
path_provider: 2.0.14
path_provider_android: 2.0.25
path_provider_foundation: 2.2.2
path_provider_platform_interface: 2.0.6
plugin_platform_interface: 2.1.3
provider: 6.0.4
rxdart: 0.27.7
shared_preferences: 2.0.15
shared_preferences_android: 2.1.0
shared_preferences_ios: 2.1.1
shared_preferences_platform_interface: 2.2.0
shared_preferences_web: 2.1.0
sign_in_with_apple: 4.3.0
sign_in_with_apple_platform_interface: 1.0.0
sign_in_with_apple_web: 1.0.1
sqlite: 2.2.6
stream_transform: 2.1.0
table_calendar: 3.0.9
timeago: 3.2.2
url_launcher: 6.1.10
url_launcher_android: 6.0.27
url_launcher_ios: 6.1.4
url_launcher_platform_interface: 2.1.2
video_player: 2.6.0
video_player_android: 2.4.3
video_player_avfoundation: 2.4.2
video_player_platform_interface: 6.1.0
video_player_web: 2.0.15
webview_flutter: 4.0.5
webview_flutter_android: 3.3.1
webview_flutter_platform_interface: 2.0.2
webview_flutter_wkwebview: 3.1.1
webviewx_plus: 0.3.3

dev_dependencies:

flutter_test:
 sdk: flutter

flutter:
 uses-material-design: true
 assets:
 - assets/images/
 - assets/videos/
 - assets/audios/
 - assets/lottie_animations/
 - assets/rive_animations/
 - assets/pdfs/

Appendix B: CO-PO And CO-PSO Mapping

COURSE OUTCOMES:

After completion of the course the student will be able to

SL. NO	DESCRIPTION	Blooms' Taxonomy Level
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)	Level 3: Apply

CO-PO AND CO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	3	3	3		2	2	3	2	2	2	3	2	2	2
CO2	3	3	3	3	3	2		3	2	3	2	3	2	2	2
CO3	3	3	3	3	3	2	2	3	2	2	2	3			2
CO4	2	3	2	2	2			3	3	3	2	3	2	2	2
CO5	3	3	3	2	2	2	2	3	2		2	3	2	2	2

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

MAPPING	LOW/ MEDIUM/ HIGH	JUSTIFICATION
100003/CS6 22T.1-PO1	HIGH	Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS6 22T.1-PO2	HIGH	Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics.
100003/CS6 22T.1-PO3	HIGH	Design solutions for complex engineering problems by identifying technically and economically feasible problems.
100003/CS6 22T.1-PO4	HIGH	Identify technically and economically feasible problems by analysis and interpretation of data.
100003/CS6 22T.1-PO6	MEDIUM	Responsibilities relevant to the professional engineering practice by identifying the problem.
100003/CS6 22T.1-PO7	MEDIUM	Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions.
100003/CS6 22T.1-PO8	HIGH	Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems.
100003/CS6 22T.1-PO9	MEDIUM	Identify technically and economically feasible problems by working as a team.
100003/CS6 22T.1-PO10	MEDIUM	Communicate effectively with the engineering community by identifying technically and economically feasible problems.
100003/CS6 22T.1-PO11	MEDIUM	Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems.
100003/CS6 22T.1-PO12	HIGH	Identify technically and economically feasible problems for long term learning.
100003/CS6 22T.1-PSO1	MEDIUM	Ability to identify, analyze and design solutions to identify technically and economically feasible problems.
100003/CS6 22T.1-PSO2	MEDIUM	By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems.
100003/CS6 22T.1-PSO3	MEDIUM	Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems.
100003/CS6 22T.2-PO1	HIGH	Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals.

100003/CS6 22T.2-PO2	HIGH	Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes.
100003/CS6 22T.2-PO3	HIGH	Design solutions for complex engineering problems and design based on the relevant literature.
100003/CS6 22T.2-PO4	HIGH	Use research-based knowledge including design of experiments based on relevant literature.
100003/CS6 22T.2-PO5	HIGH	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools.
100003/CS6 22T.2-PO6	MEDIUM	Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature.
100003/CS6 22T.2-PO8	HIGH	Apply ethical principles and commit to professional ethics based on the relevant literature.
100003/CS6 22T.2-PO9	MEDIUM	Identify and survey the relevant literature as a team.
100003/CS6 22T.2-PO10	HIGH	Identify and survey the relevant literature for a good communication to the engineering fraternity.
100003/CS6 22T.2-PO11	MEDIUM	Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles.
100003/CS6 22T.2-PO12	HIGH	Identify and survey the relevant literature for independent and lifelong learning.
100003/CS6 22T.2-PSO1	MEDIUM	Design solutions for complex engineering problems by Identifying and survey the relevant literature.
100003/CS6 22T.2-PSO2	MEDIUM	Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices.
100003/CS6 22T.2-PSO3	MEDIUM	Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research.
100003/CS6 22T.3-PO1	HIGH	Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals.
100003/CS6 22T.3-PO2	HIGH	Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions.

100003/CS6 22T.3-PO3	HIGH	Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies.
100003/CS6 22T.3-PO4	HIGH	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS6 22T.3-PO5	HIGH	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
100003/CS6 22T.3-PO6	MEDIUM	Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues.
100003/CS6 22T.3-PO7	MEDIUM	Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS6 22T.3-PO8	HIGH	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics.
100003/CS6 22T.3-PO9	MEDIUM	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS6 22T.3-PO10	MEDIUM	Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies.
100003/CS6 22T.3-PO11	MEDIUM	Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies.
100003/CS6 22T.3-PO12	HIGH	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS6 22T.3-PSO3	MEDIUM	The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies.
100003/CS6 22T.4-PO1	MEDIUM	Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS6 22T.4-PO2	HIGH	Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation.

100003/CS6 22T.4-PO3	MEDIUM	Prepare Design solutions for complex engineering problems and create technical report and deliver presentation.
100003/CS6 22T.4-PO4	MEDIUM	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation.
100003/CS6 22T.4-PO5	MEDIUM	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation.
100003/CS6 22T.4-PO8	HIGH	Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
100003/CS6 22T.4-PO9	HIGH	Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS6 22T.4-PO10	HIGH	Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation.
100003/CS6 22T.4-PO11	MEDIUM	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation.
100003/CS6 22T.4-PO12	HIGH	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation.
100003/CS6 22T.4-PSO1	MEDIUM	Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas.
100003/CS6 22T.4-PSO2	MEDIUM	To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation.
100003/CS6 22T.4-PSO3	MEDIUM	To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation.
100003/CS6 22T.5-PO1	HIGH	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS6 22T.5-PO2	HIGH	Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project.

100003/CS6 22T.5-PO3	HIGH	Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs.
100003/CS6 22T.5-PO4	MEDIUM	Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS6 22T.5-PO5	MEDIUM	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO6	MEDIUM	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO7	MEDIUM	Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO8	HIGH	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO9	MEDIUM	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO11	MEDIUM	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO12	HIGH	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PSO1	MEDIUM	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project.

100003/CS6 22T.5-PSO2	MEDIUM	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PSO3	MEDIUM	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project.

