ONLINE TURF BOOKING SYSTEM

A PROJECT REPORT

Submitted by

MOHAMUDHA SHIFA THAHSINA J
(Reg. No: 24MCR066)

SUJITH CHINNUSAMY S
(Reg. No: 24MCR110)

VARSHINI T
(Reg. No: 24MCR122)

in partial fulfilment of the requirements

for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DECEMBER 2024

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DECEMBER 2024

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"ONLINE TURF BOOKING SYSTEM"** is the bonafide record of project work done by **MOHAMUDHA SHIFA THAHSINA J (24MCR066), SUJITH CHINNUSAMY S (24MCR110)** and **VARSHINI T (24MCR122)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications of Anna University, Chennai during the year 2024-2025.

SUPERVISOR	**HEAD OF THE DEPARTMENT**
Date:	**(Signature)**

Submitted for end semester project viva voce examination held on ______________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

We affirm that the project entitled **“ONLINE TURF BOOKING SYSTEM”** being submitted in partial fulfilment of the requirements for the award of Masterof Computer Applications is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidates.

MOHAMUDHA SHIFA THAHSINA J
(Reg. No: 24MCR066)

SUJITH CHINNUSAMY S
(Reg. No: 24MCR110)

VARSHINI T
(Reg. No:24MCR122)

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: **Name and Signature of the Supervisor**

ABSTRACT

The **Online Turf Booking System** is an advanced web application designed to simplify the process of booking and managing turf facilities. Built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), this platform provides a seamless solution for both users and administrators. It addresses the growing demand for efficient turf management by offering an intuitive, user-friendly interface that enhances the overall booking experience. Users can easily browse available turfs, check real-time slot availability, make secure payments, and receive instant booking confirmations. For administrators, the system offers tools to manage bookings, monitor usage statistics, and address customer queries efficiently.

This platform eliminates the inefficiencies of manual turf management systems by automating key processes. Users can access detailed information about turf facilities, including pricing, availability and amenities, enabling informed decision-making. Meanwhile, administrators benefit from a centralized dashboard that provides insights into booking trends, allowing them to optimize resource allocation and ensure fair usage. Real-time notifications and updates keep both users and administrators informed, reducing the likelihood of errors or miscommunication.

The application's responsive design ensures it performs seamlessly across all devices, including desktops, tablets, and smartphones. This feature, combined with its secure payment integration through Razorpay, enhances trust and convenience for users. The scalable architecture of the MERN stack allows for future enhancements, such as integrating mobile apps, advanced analytics, and AI-driven recommendations. These features make the system a forward-thinking solution for turf facilities aiming to stay ahead in a competitive market.

By promoting transparency, efficiency, and accessibility, the Online Turf Booking System fosters a vibrant sports culture within the community. The platform encourages the optimal use of turf resources, reducing downtime and maximizing user satisfaction. With its robust features and potential for expansion, this system not only simplifies the booking process but also sets a new standard for turf management in sports facilities.

ACKNOWLEDGEMENT

We respect and thank our correspondent **THIRU.A.K.ILANGO, BCom., MBA., LLB.** and our Principal **Dr. V. BALUSAMY BE (Hons)., MTech, PhD.,** Kongu Engineering College, Perundurai for providing us with the facilities offered

We convey our gratitude and heartfelt thanks to our Head of the Department **Dr.A.TAMILARASI MSc., MPhil., PhD., MTech.,** Department of Computer Applications, Kongu Engineering College, for her perfect guidance and support that made this work to be completed successfully.

We also wish to express my gratitude and sincere thanks to our project coordinator **Ms. S. HEMALATHA MCA., Assistant Professor (Sr.G)**, Department of Computer Applications, Kongu engineering College, who has motivated us in all aspects for completing the project in the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide **Dr. T. KAVITHA MCA., MPhil., PhD., Assistant Professor (Sr.G)**, Department of Computer Applications, Kongu Engineering College for giving her valuable guidance and suggestions which helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping overwhelm in all proceedings. We bow our heart with heartfelt thanks to all those who thought us their warm services to succeed and achieve our work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBERIVATION	EXPANSION
API	Application Programming Interface
CSS	Cascading Style Sheets
HTTPS	Hypertext Transfer Protocol Secure
JS	Java Script
JWT	JSON Web Token
OTP	One Time Password
REST	Representational State Transfer
SPA	Single-Page Application
UI	User Interface Design
UPI	Unified Payments Interface
VGA	Video Graphics Array

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

1.1.1 Overview

ONLINE TURF BOOKING SYSTEM is a digital platform designed to streamline the booking, scheduling, and management of turf facilities, such as sports fields or courts, typically used for football, cricket, tennis, or other outdoor sports. It simplifies the process for both users and administrators by providing an accessible, user-friendly interface for managing turf bookings, payments, and availability.

1.1.2 Background

Every organization needs cloud storage of product information, a complete profile that contains the information about the turf maintained by an online turf booking. Online Turf Booking System is a web based application that delivers data management capabilities. This system has been developed with MERN stack.

1.1.3 Problem Description

The problem with traditional turf booking systems lies in their inefficiency, lack of transparency, and increased chances of human error. Typically, turf bookings are managed manually through phone calls, in-person visits, or physical registers, which results in time-consuming processes and limited access to real-time availability. This often leads to double bookings or scheduling conflicts, frustrating customers and causing operational disruptions.

Moreover, manual systems make it difficult to maintain accurate records of payments, turf conditions, or customer preferences, further complicating resource management. Customers also face inconvenience, as payments are typically cash-based or offline, and booking details may not be clearly communicated upfront. Without an integrated system, turf managers struggle to optimize the use of available slots, handle maintenance scheduling, or analyze booking patterns, leading to underutilized facilities and lost revenue opportunities. Additionally, the absence of an easy-to-use platform for cancellations and rescheduling further diminishes user satisfaction. In essence, these traditional methods hinder operational efficiency, create confusion, and impact the overall customer experience.

1.2 EXISTING SYSTEM

The existing system in online turf booking system generally refers to the software platforms currently in use by sports facilities to streamline the process of booking, scheduling, and managing turf (or sports field) reservations. These systems are designed to automate and optimize the process for both users and facility managers.

1.3 DRAWBACKS OF EXISTING SYSTEM

While existing turf booking systems offer many benefits, they also have several drawbacks that can limit their effectiveness for both users and facility managers. Some of the key drawbacks of these systems include:

- Limited Customization
- User Interface (UI) Complexity
- Scalability Issues
- Poor Integration with Other Systems
- Inflexible Payment Systems
- Inadequate Support for Complex Reservations
- Limited Reporting and Analytics
- Maintenance Scheduling Challenges
- Lack of Customer Support and Training
- Inability to Handle Cancellations/Reschedules Efficiently

1.4 PROPOSED SYSTEM

This system aims to provide a user-friendly platform for both customers and turf administrators, ensuring an efficient, hassle-free experience. With the increasing popularity of sports activities and recreational games, the demand for a robust and convenient booking system has grown significantly. This proposed system addresses the challenges of manual booking processes, such as scheduling conflicts, time-consuming communication, and inefficient management of resources.

For customers, the system offers a seamless online interface to check turf availability, book slots, make payments, and manage reservations. Users can browse through different turf facilities, view slot availability in real-time, and select a slot that fits their schedule. To enhance user experience, features such as customer profiles, automated reminders, and payment gateways are integrated, ensuring a smooth booking process. The system also includes options for group bookings, cancellations, and rescheduling, making it highly flexible and user-centric.

1.5 ADVANTAGES OF PROPOSED SYSTEM

- User friendliness is provided in the application with various controls.
- The system makes the overall project management much easier and flexible.
- It can be accessed over the internet.
- Enhancing the growth of turf booking and management.
- Customer can pay using different mode of payment through UPI.

1.6 ANTICIPATION OF CONCLUSIONS

Once the system is fully implemented and operational. It is anticipated that the introduction of this system will bring significant improvements to both the operational side of turf management and the user experience. For turf operators, the system promises to streamline administrative tasks, reducing the manual effort involved in scheduling, payments, and customer management. Automation will allow for real-time updates, accurate bookings, and fewer errors, making it easier to manage high volumes of users and turf usage.

From a customer perspective, the anticipation is that the system will provide a seamless and more convenient booking process. Users will likely appreciate the ability to reserve turfs quickly via an intuitive mobile app or website, with features such as real-time availability, instant booking confirmations, and multiple payment options. Personalized experiences, like loyalty rewards or tailored booking suggestions, can enhance customer retention and satisfaction. Additionally, with easy access to booking history and notifications about upcoming sessions, users will feel more engaged and connected to the facility.

SUMMARY

This chapter describes about the existing system and its drawbacks and proposed system and its advantages. The purpose of this project was to develop a web application for online turf booking. This project helped us in gaining valuable Information and practical knowledge on several topics like designing web pages using , usage of responsive templates, management of databases using mongodb. The system analysis and software requirements and hardware requirements of the project has been explained in Chapter 2.

CHAPTER 2

SYSTEM ANALYSIS

2.1 IDENTIFICATION OF NEED

Online Turf Booking System is developed to provide the solution of online booking of the available turf that would help the enthusiastic to book the turf. Through this the administrator who is one of the users of this webpage will manage all the activities of users such as search user, Add turf, Manage turf, View bookings. This online platform of Turf booking will help the users to book their choice of available slots online. This system will replace the manual process of going to the location to book with an automated online process. It has the benefits effective booking corridor or to hold their accessible ground with holding up through a webpage and enhance the popularity among their intended customers coupled with speedy and direct service availabilities. This webpage not only helps the players but also the owners who want to expand their business through online medium.

2.2 FEASIBILITY STUDY

A feasibility study evaluates whether the online turf booking system is worth implementing by assessing its viability, impact, and resource utilization. The proposed system addresses the inefficiencies of manual booking processes, such as errors, double bookings, and limited accessibility. It provides an automated platform where users can view turfs, check availability, select time slots, and complete bookings online. Features like secure payment integration, automated notifications, and an admin dashboard for managing bookings and turf data enhance usability and operational efficiency. This system aligns with modern user expectations, offering convenience and reliability.

Although the initial investment in development and hosting is significant, the system reduces manual costs over time and boosts revenue by improving operations and customer satisfaction. Its ability to streamline processes and meet user needs makes it a viable solution, promising long-term benefits aligned with organizational goals.

2.2.1 Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost/benefit analysis; the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs. If benefits outweigh costs, a decision is taken to design and implement the system.

2.2.2 Operational Feasibility

It is mainly related to human organizational and political aspects. The points to beconsidered were:

- What changes would be brought with the system?
- What organizational structures would be disturbed?
- What new skills would be required? Does the existing staff members have these skills? Ifnot, could they be trained in due course of time?

2.2.3 Technical Feasibility

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system includes:

- The facility to produce outputs in the given time.
- Response time under certain conditions.
- Ability to produce a certain volume of transaction at a particular speed.
- Facility to communicate data to distant location.

A software requirements specification is a complete description of the behavior of the software to be developed. It contains the functional, non-functional and user interface requirements of the proposed system.

Functional Requirements

- ➢ Insert records: This action is done to add new records into fields.
- ➢ Update records: This event is to modify or update the information on each process.
- ➢ Delete records: This action is to remove records from the system whenever they are no longer needed.
- ➢ Search for records: Whenever the admin wants to search for a record, this action is performed.
    - The validation of data entered should be done.
    - Specific condition has to be met.

System Feature 1

Authentication

- User Registration and Login : Accounts via email, phone, or third-party platforms like Google/Facebook.
- Secure Password Management : Strong password policies with recovery/reset options.
- Multi-Factor Authentication : Two-factor authentication (e.g., OTP).
- Session Management : Auto-logout for inactive users.
- Role-Based Access Control : Different access levels for users and administrators.
- Encrypted Data Transmission : Secure protocols (HTTPS) and encrypted password storage.
- Login Activity Monitoring : Tracks recent login activity to detect unauthorized access.

2.3 SOFTWARE REQUIREMENT SPECIFICATIONS

2.3.1 Software Requirements

Operating System	:	Windows/Linux/Mac
Frontend	:	React JS, Tailwind CSS, DaisyUI, Redux
Backend	:	Node JS, Express JS
Database	:	MongoDB
Payments	:	Razorpay
Image Hosting	:	Cloudinary

2.3.2 Hardware Requirements

Processor	:	Intel Core i5
RAM	:	16.0 GB (15.7 GB usable)
Monitor	:	15’’VGA Monitor
Hard Disk	:	250GB
Keyboard	:	ACCUTYPE Keyboard Optical Mouse

2.3 SOFTWARE DESCRIPTION

Frontend

The frontend of the online turf booking system is designed to provide an intuitive and responsive userinterface for customers and administrators. It is built using the following technologies:

- **React.js**
  - ➢ Used to build a dynamic and interactive single-page application (SPA).
  - ➢ Enables fast and seamless navigation through components like the homepage, searchfilters, booking forms, and user dashboards.

- **Tailwind CSS**
  - ➢ Utility-first CSS framework that ensures responsive design with minimal custom styling.
  - ➢ Enhances the speed of UI development while maintaining a modern look and feel.

- **DaisyUI**
  - ➢ A UI component library built on Tailwind CSS that provides pre-designed componentssuch as buttons, cards, modals, and form controls.
  - ➢ Used to maintain consistent styling and reduce UI development time.

- **Redux**
  - ➢ A state management library for handling application-wide states like user authentication,turf availability, and booking status.
  - ➢ Provides predictable state management and ensures a smooth user experience.

Backend

The backend forms the core logic of the application, responsible for managing business operations,data flow, and communication with the database. It is implemented using:

- **Node.js:**
  - ➢ A server-side JavaScript runtime that enables building a scalable and efficient backend.
  - ➢ Handles asynchronous operations for better performance.

- **Express.js:**
  - ➢ A lightweight framework for building RESTful APIs.
  - ➢ Manages routing, middleware, and HTTP request/response handling.
  - ➢ Includes features like input validation and error handling.

- **Backend Features:**
  - User authentication using JWT (JSON Web Tokens) for secure login.
  - Role-based access control for customers and admins.
  - Business logic for handling turf availability, bookings, and cancellations
  - Integration with third-party APIs for notifications or payments.

Database

The database layer stores and organizes all application data securely. It is implemented using:

- **MongoDB:**
  - A NoSQL database for handling flexible, schema-less data.
  - Collections are used for storing:
  - Users: Profile information, credentials, and roles (admin or customer).
  - Turfs: Turf details like name, location, price, and availability slots.
  - Bookings: User bookings with associated time slots, payment details, and status.

- **Database Features:**
  - Indexing: Optimizes search queries for faster data retrieval (e.g., finding turfs by locationor date).
  - Scalability: Easily scales with growing data volume and user base.
  - Data Relationships: Stores relationships between users, turfs, and bookings using referencesor embedded documents.

SUMMARY

The Online Turf Booking System is a user-friendly platform built using the MERN stack, combiningReact.js, Tailwind CSS, DaisyUI, and Redux for a responsive frontend, Node.js and Express.js for arobust backend, and MongoDB for scalable data storage. The frontend enables seamless browsing, booking, and user management, while the backend handles secure authentication, role-based access control, and real-time business logic through RESTful APIs. MongoDB efficiently manages user, turf, and booking data, ensuring fast retrieval and scalability. This architecture delivers a smooth, efficient, and secure experience for users and administrators. The system design, modules and data models for this projectare described in Chapter3.

CHAPTER 3

SYSTEM DESIGN

INTRODUCTION

The system design for an online turf booking system defines the structure and processes to streamline booking operations. It includes components for user registration and login, viewing turf availability, selecting slots, and completing payments. The admin module allows for managing turfs, monitoring bookings, and handling user interactions. This design ensures a seamless and efficient experience for users while providing robust control and management capabilities for administrators.

3.1 INPUT DESIGN

The input design of the online turf booking system using the MERN stack ensures smooth interaction between users and the system, focusing on simplicity, efficiency, and error prevention. It incorporates features like dropdown menus, date pickers, and real-time validation for inputs such as registration details, turf search, and payment information. The design is responsive across all devices, offering a consistent user experience. Pre-filled fields, tooltips, and real-time error messages guide users through the process, minimizing mistakes. Security is prioritized with encrypted data transmission, secure payment gateways, and careful handling of sensitive information, while adhering to minimal data collection for privacy. The objective is to create user-friendly, error-free data entry screens that facilitate accurate information input and efficient data management.

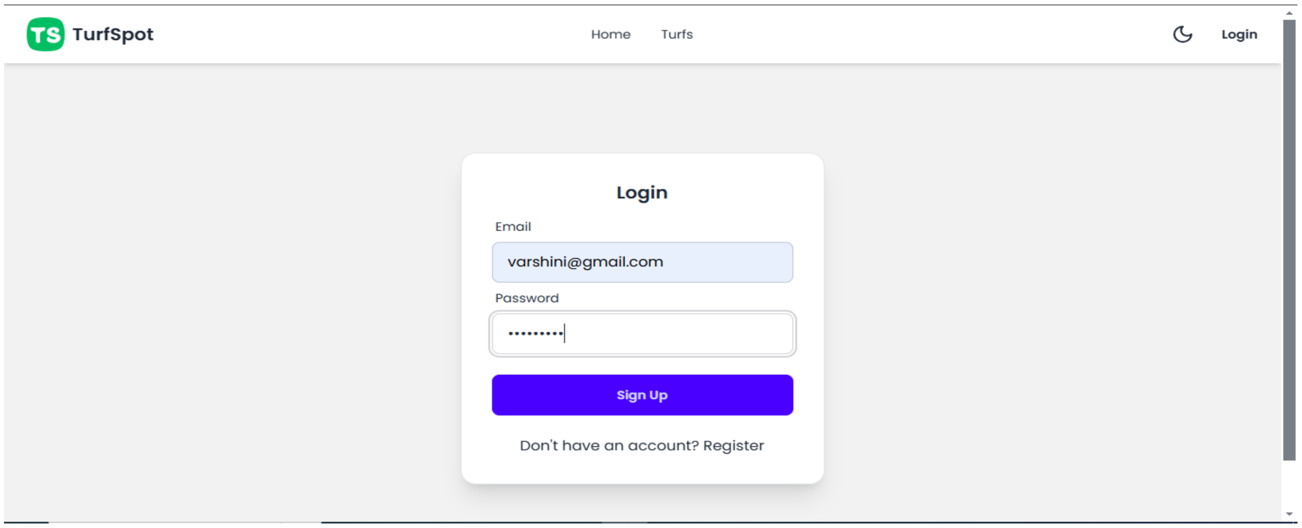


Figure 3.1 Login Input

The Figure 3.1 shows the login page image features clearly labeled fields for the username and password, with secure password masking and real-time validation for accurate data entry. It also includes options for error messages and password recovery to enhance user experience.

3.2 OUTPUT DESIGN

Output design in an online turf booking system is crucial for effectively communicating processed information to users and other systems, ensuring that the output meets user requirements and is presented clearly. It involves determining how information should be displayed for immediate use and in hard copy formats, while focusing on user-friendly, efficient, and relevant outputs that aid decision-making. The design process includes identifying necessary outputs, selecting appropriate methods for presenting information, and creating documents, reports, or other formats to convey past activities, current status, future projections, important events, or confirmations. A well-designed output system ensures that users can easily access and understand critical information, triggering actions or alerting them to key events, opportunities, or issues.

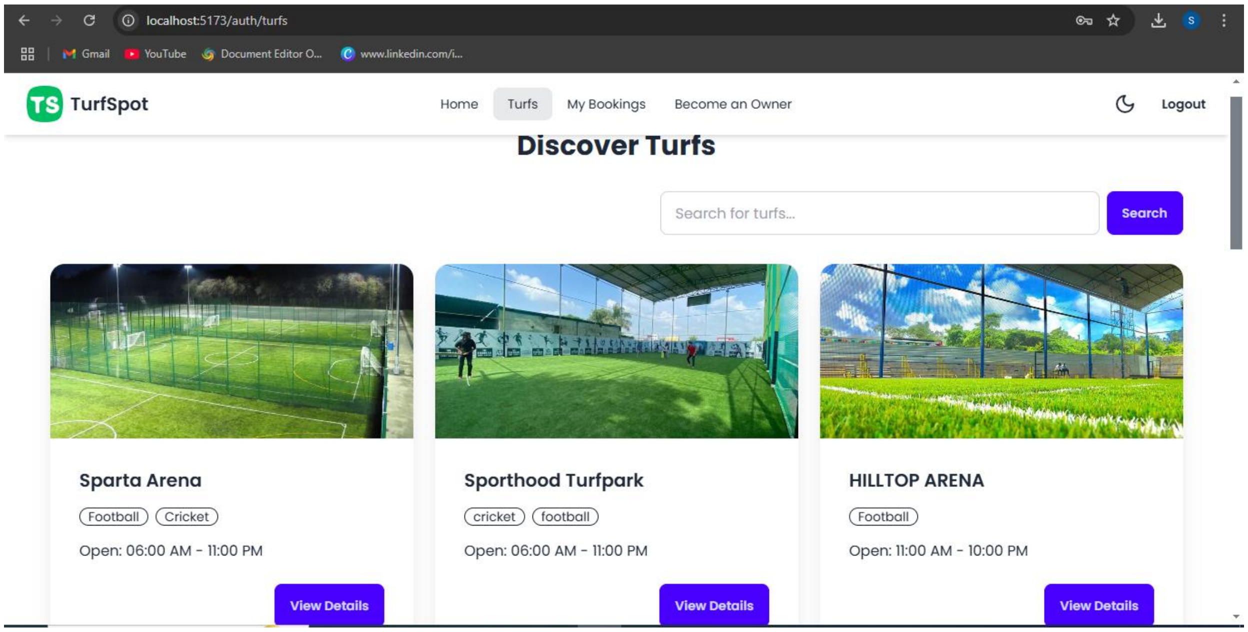


Figure 3.2 Home Page

The Figure 3.2 demonstrates a homepage for TurfSpot, where users can discover turfs with details such as name, sports available, opening hours, and an option to view more details. A search bar is provided for filtering turfs, alongside navigation links for bookings and becoming an owner.

3.3 MODULE DESIGN

The Online Turf Booking System is a comprehensive platform that streamlines the process of booking sports turfs. It includes three key modules: the User Module, where users can browse turfs, book time slots, rate turfs, and even apply to become owners; the Owner Module, where turf owners can manage their turfs, view booking statistics, and handle user reviews; and the Admin Module, which allows administrators to oversee user and owner management, turf listings, and transaction data. Built using the MERN stack, this system ensures a seamless and scalable experience for all stakeholders.

ADMIN MODULE

The Admin Module is the platform's management center, allowing administrators to oversee all activities, from user and turf management to transaction monitoring. This module includes:

1. **Owner Requests**
   - Admins can approve or reject requests from users who want to become turf owners.
   - When an approval occurs, the user will receive an email with a registration link to complete their owner profile and access the Owner Module.
   - Rejected users will receive a polite rejection email with possible reasons for the decision.

2. **User and Owner Management**
   - Admins can view all registered users and owners, including their profiles, booking histories, and any flagged activity.
   - Admins can suspend or ban users or owners who violate the platform’s terms of service.
   - Admins also have the ability to merge duplicate accounts or fix data inconsistencies.

3. **Turf Management**
   - Admins can view all turfs listed on the platform by the owners.
   - Admins can edit turf details if necessary, to ensure correct information is displayed to users.
   - Admins have the authority to remove turfs that do not meet the platform's quality standards or violate any rules.

4. **Transaction Overview**
   - Admins have access to a detailed view of all transactions made on the platform, including bookings, payments, and refunds.
   - The system will provide monthly transaction graphs showing overall revenue, popular turfs, and payment methods.
   - Admins can also access a breakdown of pending, completed, and canceled transactions, enabling them to quickly resolve issues like payment disputes or booking conflicts.

USER MODULE:

The User Module is designed to facilitate the entire customer journey, from browsing available turfs to booking time slots and leaving feedback. The key features of this module are as follows:

1. **Browse Turfs**
   - Users can view a comprehensive list of available turfs in their area or chosen location.
   - Each turf listing will display essential details, including turf name, location, surface type (grass, synthetic, etc.), available facilities (changing rooms, parking, etc.), and pricing.
   - Users can filter turfs based on factors like surface type, location, price range, and available facilities for an efficient browsing experience.
   - An interactive map view may also be available to help users quickly locate turfs within their desired area.

2. **Slot Booking**
   - Users can select a time slot for booking a turf based on availability, displayed in a calendar format.
   - Each available slot will display the pricing, duration, and any discounts (if applicable).
   - Users can proceed to make the booking by using the Razorpay payment gateway for secure and smooth transaction processing.

- After completing the booking and payment, users will receive an instant confirmation email with:
    - A detailed booking summary (including turf name, start time, end time, price, etc.)
    - A unique QR code containing all the booking details that can be scanned at the turf for quick check-in.

3. **Rate Turfs**
    - After the completion of a booking, users are encouraged to rate the turf based on their experience.
    - The rating system could be on a 1-5 star scale, with an optional text review.
    - Reviews and ratings will help other users make informed decisions about the quality of turfs before booking.

4. **Become Owner**
    - Users interested in becoming turf owners can apply by filling out an application form.
    - The application will ask for details such as the turf location, surface type, facilities, and ownership information.
    - Once the application is submitted, admins will review the details and either approve or reject the request.
    - If approved, the user will receive an email with a registration link, allowing them to access the Owner Module.

OWNER MODULE:

The Owner Module allows turf owners to manage their turfs, view bookings, and interact with users who have rated or reviewed their services. The core features of this module include:

1. **Turf Management:**
   - Owners can add new turfs to the platform by entering key information, such as turf name, location, surface type, available time slots, pricing, and facilities.
   - Existing turfs can be edited or updated to reflect any changes in pricing, facilities, or availability.
   - Owners can delete or archive turfs that are no longer available for booking.

2. **Dashboard:**
   - The dashboard provides owners with an overview of their transactions, bookings, and earnings over various time periods (daily, weekly, monthly).
   - Graphs and charts will provide insights into booking trends, most popular time slots, and revenue statistics.
   - Owners can view a breakdown of bookings by user, including details like the number of sessions, turf name, payment status, and total earnings.

3. **Review Management:**
   - Owners can view and manage reviews left by users who have booked their turfs.
   - Owners can respond to reviews, thank users for positive feedback, or address any complaints to improve their services.
   - Negative reviews can be flagged for investigation by admins if they violate platform policies.

USE CASE DIAGRAM

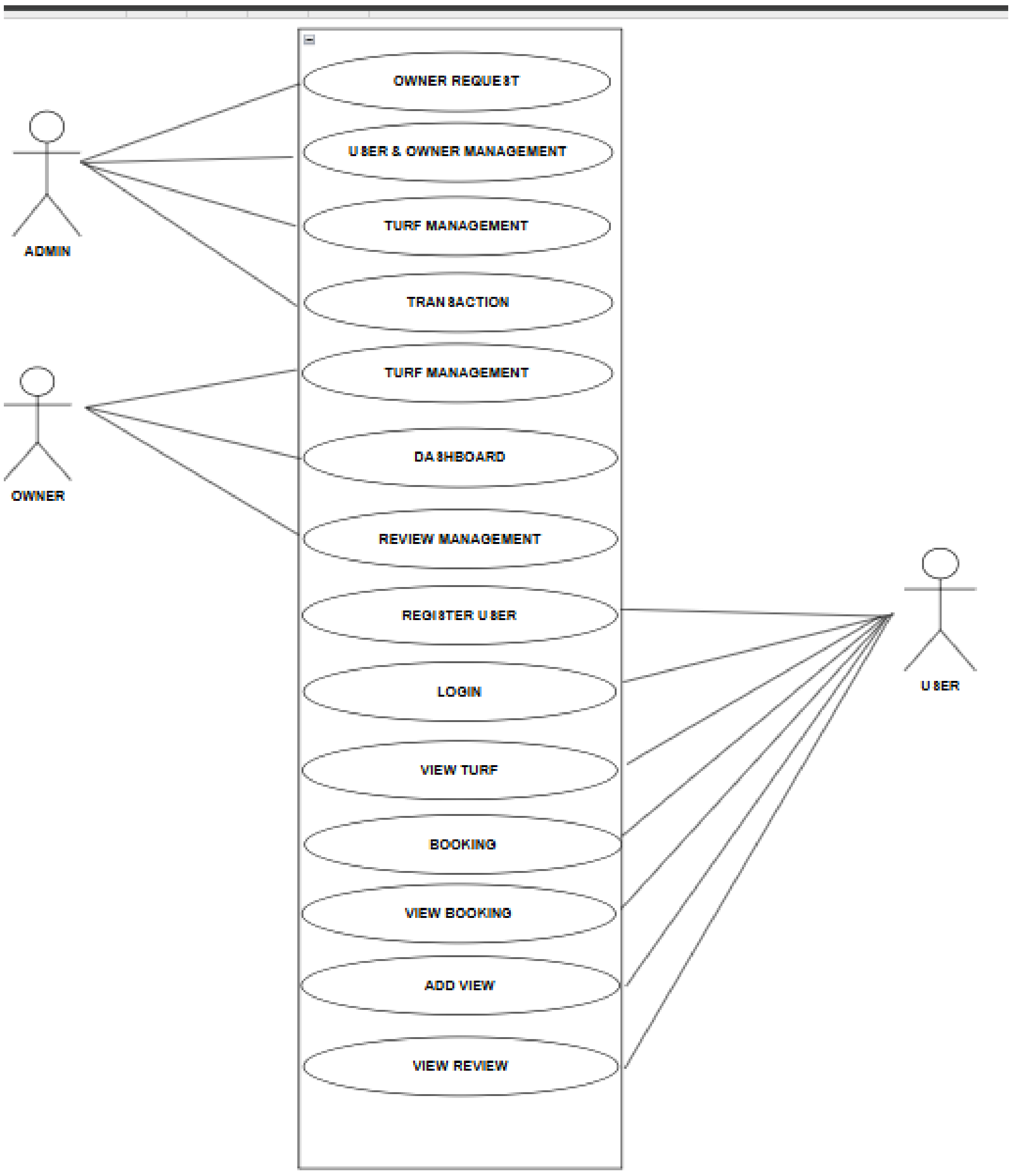


Figure 3.3 Use Case Diagram

The Figure 3.3 is a use case representation of a turf booking system, illustrating roles such as Admin, Owner, and User interacting with functionalities like turf management, bookings, reviews, and user registration. It highlights specific responsibilities and access levels for each role.

3.4 DATA MODELS

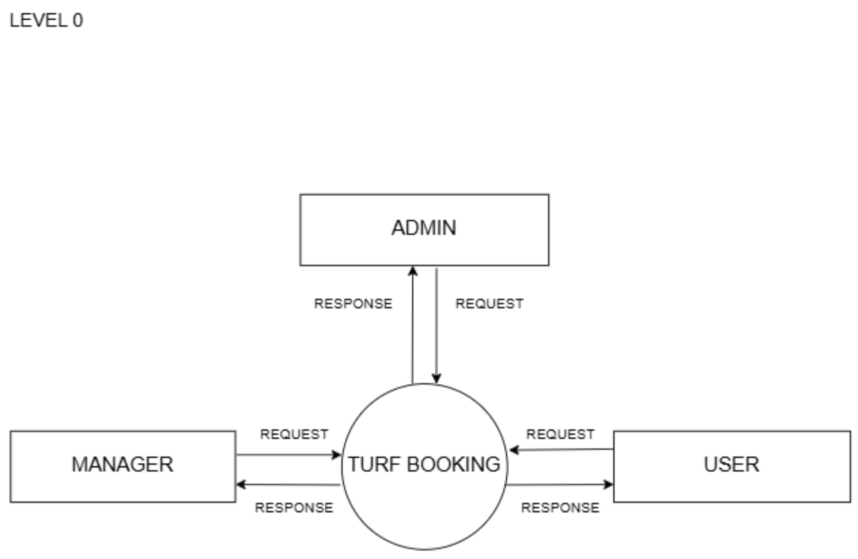


Figure 3.4 Level 0 Data Flow Diagram

The Figure 3.4 depicts a Level 0 Data Flow Diagram (DFD) for the system "TurfSpot." It illustrates interactions between three main actors—User, Owner, and Admin—and the central TurfSpot process. Data flows in and out of the TurfSpot system, connecting with the Database, showcasing how information is exchanged and stored among different system components.

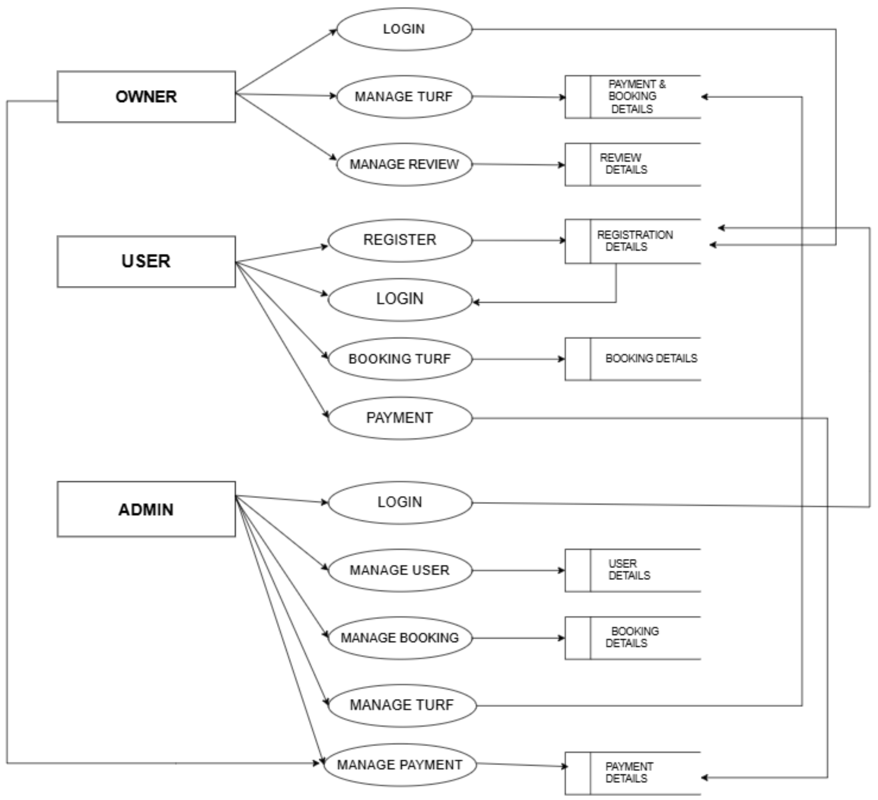


Figure 3.5 Level 1 Data Flow Diagram

The Figure 3.5 shows the Level 1 DFD for **User** in the system "TurfSpot" shows user interactions: Register, Login, View Turf, Booking, View Bookings, Add Review, and View Review. It connects users to components like Turfs, Timeslot, Bookings, and Reviews to manage turf booking and reviews.

SUMMARY

This Chapter describes the Input design, Output design, Module Description, such as Admin module and User module and data model. In the Admin module, the process is add products and delivery. User module, add details and login. The Data flow diagram, class diagram. database design, data integrity process also discussed in this chapter. The system implementation for this project are described in Chapter 4.

CHAPTER 4

IMPLEMENTATION

4.1 SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and giving confidence in the new system for the users, that it will work efficiently and effectively, It involves careful planning, investing of the current system, and its constraints on implementation, design of methods to achieve the changeover methods The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out in these plans; discussion has been made regarding the equipment, resources, and how to test activities.

The coding step translates a detailed design representation into a programming language realization. Programming languages are vehicles for communication between humans and computers programming language characteristics and coding styles can profoundly affect software quality and maintainability. The coding is done with the following characteristics in

- Ease of design to code translation.
- Code efficiency.
- Memory efficiency
- Maintainability

The user should be very careful while implementing a project to ensure what they have planned is properly implemented. The user should not change the purpose of project while implementing The user should not go in a roundabout way to achieve a solution; it should be direct, crisp and clear and up to the point.

4.2 STANDARDIZATION OF CODING

ReactJS follows few rates and maintains its style of coding. Making me of indentation for indicating the start and end of control structures along with a clear specification of where the code is between them. The consistency in the naming convention of the variables is followed throughout the code. Also, the data should be described in the code Comment lines are used in this code it helps the developers to understand the code. The use of comment lines in this code aids in the understanding of the code by the developers. This code has been written to improve readability without affecting the code's essential functioning. This code is done to enhance the readability of the code without modifying the basic functionality of the code.

4.3 ERROR HANDLING

An exception can be defined as an abnormal condition in a program resulting in the disruption in the flow of the program. Whenever an exception occurs, the program halts the execution, and thus the further code is not executed. Therefore, an exception is the error which ReactJS is unable to tackle with, ReactJS provides us with the way to handle the Exception so that the other part of the code can be executed without any disruption. ReactJS gives us a technique to handle the Exception so that the rest of the code may continue to run without interruption.

SUMMARY

In this chapter, Systems implementation is the process of defining how the information system should be built ensuring that the information system is operational and used, ensuring that the information system meets quality standard. we had gained knowledge on system implementation, and how to write a standardized code with command lines that a developer can understand and code readability and whenever exception occurs error can be handled. Testing and results for the project is described in chapter 5.

CHAPTER 5

TESTING AND RESULTS

5.1 SYSTEM TESTING

System testing is a crucial phase in software quality assurance, ensuring that all specifications, design choices, and code meet the expected requirements. This phase involves testing the system as a whole using various test data to verify its correctness, performance, and functionality. Preparing appropriate test data is essential to simulate real-world scenarios accurately. Once the test data is ready, the system is thoroughly tested to identify and resolve any issues. After the source code and related data structures are completed, the entire project undergoes comprehensive testing and validation to uncover and address any errors. Developers typically handle unit testing for individual modules and often conduct integration testing to build the system structure while ensuring all parts work together seamlessly.

The project has undergone the following testing procedures to ensure correctness and reliability:

- Unit Testing
- Integration Testing
- Validation Testing

5.2 UNIT TESTING

In unit testing, each individual module or component of the system is tested in isolation. In a large system like the Online Turf Booking System, many modules exist at different levels. The testing process begins from the smallest components, following a bottom-up approach, where each module is tested one at a time.

A short test program is executed for each module, supplying the necessary data to verify its functionality independently. This approach helps in identifying issues early and ensures each module performs its specific task correctly.

5.3 INTEGRATION TESTING

Integration testing focuses on combining the system modules and testing their interactions to identify errors related to interfaces. In the Online Turf Booking System, integration testing is conducted once all individual modules are developed and integrated into a complete system. The goal is to ensure that each module interacts correctly with others according to the system design. Errors are uncovered by validating the communication between components such as frontend components, backend services, and the database, ensuring a cohesive and functional system.

5.4 VALIDATION TESTING

Validation testing ensures that the system meets the requirements and expectations set by the customer. In simple terms, it verifies whether the system's performance and functionality align with the specified requirements. In this project, validation testing is carried out on critical modules, such as the User Authentication Module**.** It includes testing input fields like email ID with both valid and invalid data to ensure proper handling of user input and adherence to the required specifications. Any discrepancies or deviations are documented, and a list of deficiencies is created, ensuring issues are addressed before deployment. The validation testing process confirms that the system operates correctly and delivers a seamless user experience.

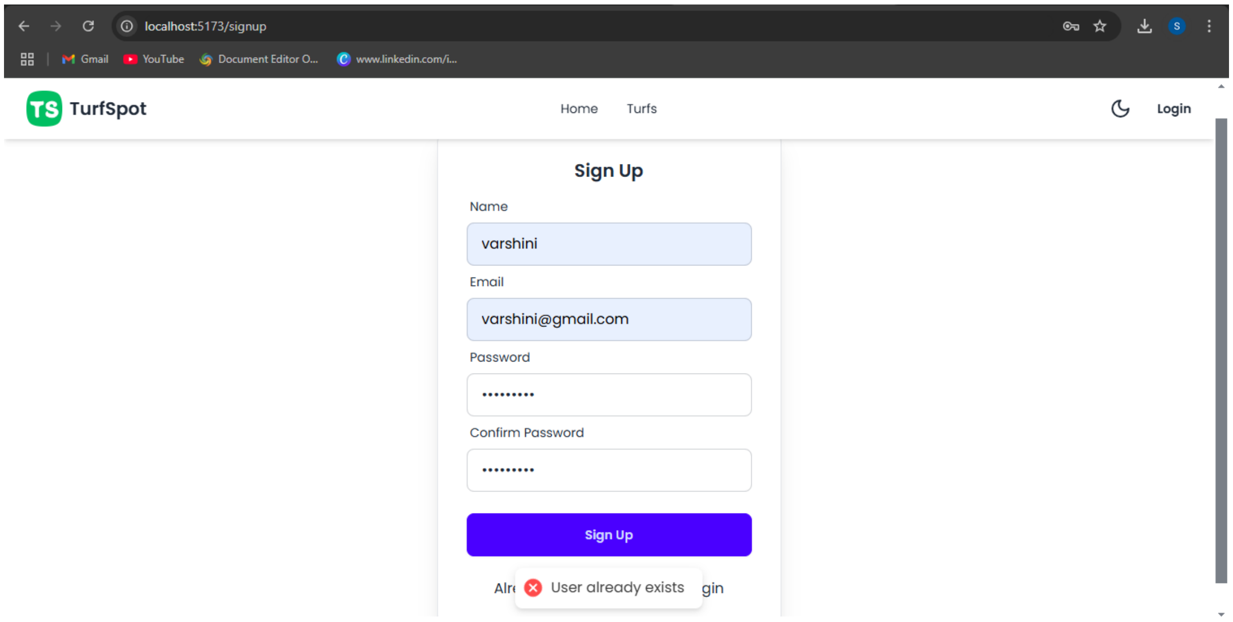


Figure 5.1 Registration Page

In Figure 5.1, registration page which gives the error message that user already exist in the database. So, the user need to give new username or new email id to create account.

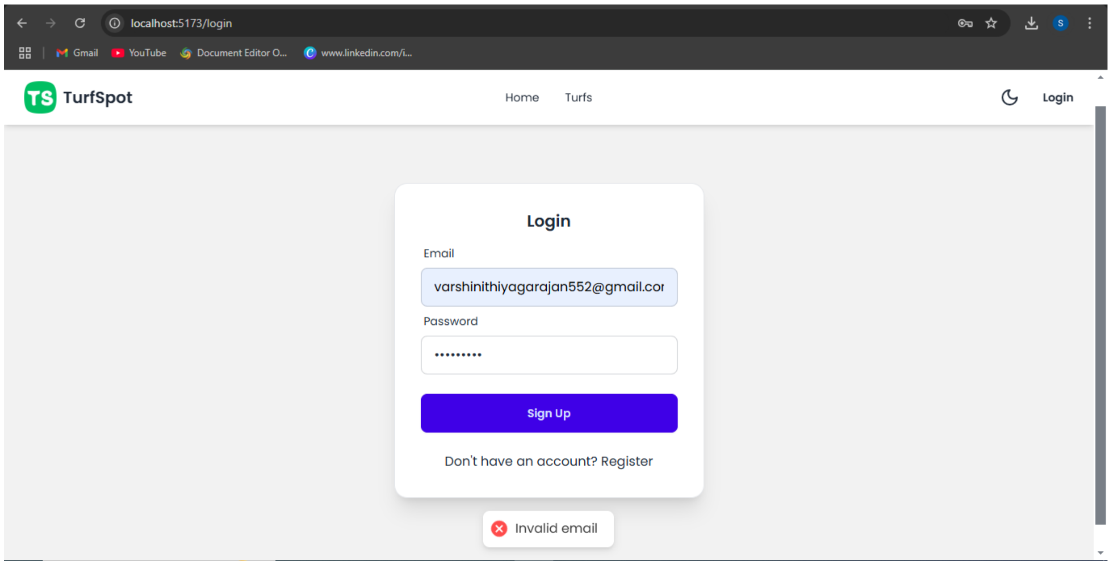


Figure 5.2 Login Page with invalid email

In Figure 5.2, when the user try's to login with the email id which not used in creating accont, it will show the error message as Email not found. User can login using the valid email.

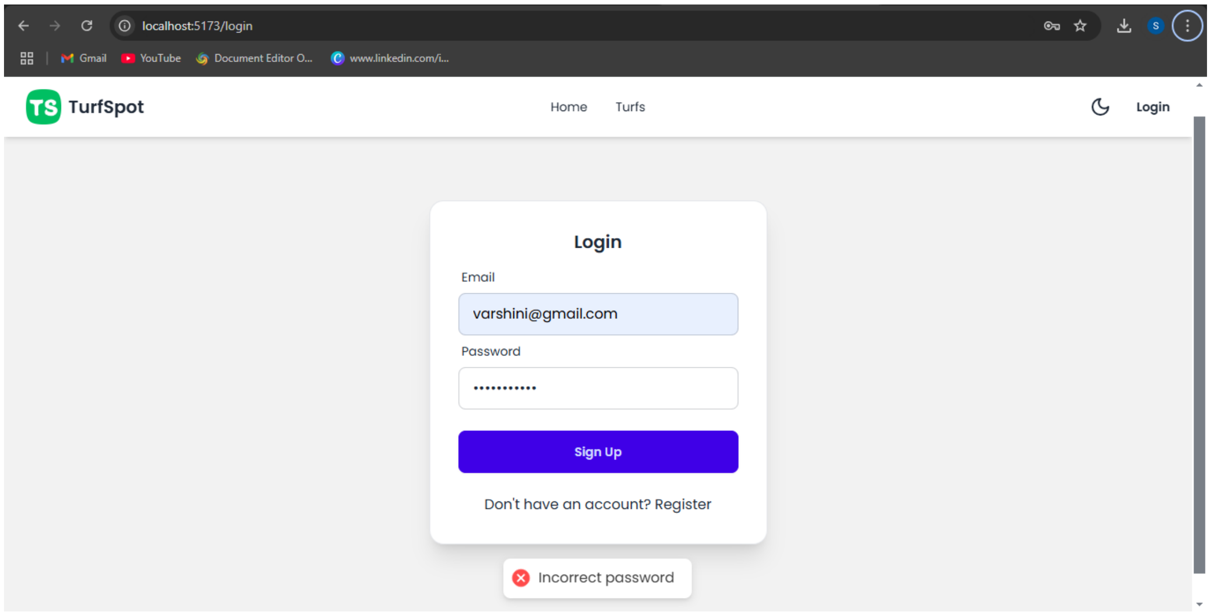


Figure 5.3 Login Page with Invalid password

In Figure 5.3, when the user try's to login with the wrong password, it will show the error message as Wrong password. User can login using the valid password.

SUMMARY

In this chapter, it shows the system testing and its results. System testing is the testing of a system as a whole. End to end testing is performed to verify that all the scenarios are working as expected. In result, The challenges experienced have been thoroughly outlined in this chapter, and all of those challenges have been overcome by achieving this online shoppingfor flowers. The conclusion and future work for the project is described in Chapter 6.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The Online Turf Booking System has been designed and developed according to the specified requirements, ensuring a robust and user-friendly platform. The system features a clean, well-structured codebase that is easy to understand and maintain. It has been tested with various sample data, ensuring accurate and reliable results. The application meets all user needs, offering a seamless booking experience with quick access to available turfs, pricing information, and booking details. The platform is adaptable and can be integrated effortlessly with other systems and workflows, making it highly versatile.

The system is built with a focus on performance, scalability, and user experience. Its intuitive interface simplifies interactions, reducing the time and effort required for bookings. The efficient design ensures fast response times and a smooth user experience across all functionalities. Future enhancements, such as converting the system into a web application, would enable scalability for multiple branches and expand accessibility. These updates would further improve operational efficiency and ensure the system remains adaptable to growing business needs while maintaining a high level of user satisfaction.

6.1 FUTURE ENHANCEMENT

The Online Turf Booking System is designed to be scalable and adaptable, with the potential for future enhancements. It can accommodate new features and modules to improve user interaction and system efficiency.

Future updates could include:

- Booking Cancellation: Allow users to cancel bookings easily.
- Online Payment Integration: Facilitate secure online payments and booking confirmations.
- Advanced Search Filters: Provide more detailed search options for better turf discovery.
- Mobile Application Support: Ensure seamless access through dedicated mobile apps.

These enhancements will improve customer satisfaction, streamline operations, and ensure the system remains robust and adaptable to growing business needs.

APPENDICES

A. SAMPLE CODING

LOGIN PAGE CODE

```
import useLoginForm from "../../hooks/useLoginForm";
import FormField from "../../components/common/FormField";
import { Link } from "react-router-dom";
import Button from "../../components/common/Button";

const Login = () => {
  const { register, handleSubmit, errors, onSubmit, loading } = useLoginForm();

  return (
    <div className="flex items-center justify-center min-h-screen max-md:p-4 bg-base-200 p-
4">
      <div className="card w-full border md:w-96 bg-base-100 shadow-xl">
        <div className="card-body">
          <h2 className="card-title justify-center">Login</h2>
          <form onSubmit={handleSubmit(onSubmit)}>
            <FormField
              label="Email"
              name="email"
              type="email"
              register={register}
              error={errors.email}
            />
            <FormField
              label="Password"
              name="password"
              type="password"
              register={register}
              error={errors.password}
            />
            <div className="form-control mt-6">
              <Button type="submit" className="btn-primary" loading={loading}>
                Sign Up
              </Button>
            </div>
          </form>
          <div className="text-center mt-4">
            <Link to="/signup" className="link link-hover">
              Don&#39;t have an account? Register
            </Link>
          </div>
        </div>
      </div>
    </div>
  );
};

export default Login;
```

HOME PAGE CODE

```
import { Link } from "react-router-dom";
import Carousel from "../components/common/Carousel";
import Footer from "../components/layout/Footer";
import useTurfData from "../hooks/useTurfData";
import TurfCard from "../components/turf/TurfCard";
import TurfCardSkeleton from "../components/ui/TurfCardSkeleton";
import { useSelector } from "react-redux";
import banner1 from "/banner-1.png";
import banner2 from "/banner-2.jpeg";
import banner3 from "/banner-3.jpeg";
```

```
const Home = () => {
  const isLoggedIn = useSelector((state) => state.auth.isLoggedIn);
  const { turfs, loading } = useTurfData();
  const slides = [banner1, banner2, banner3];

  return (
    <div className="min-h-screen bg-base-100 text-base-content">
      <div className="hero min-h-[70vh] bg-base-200">
        <div className="hero-content flex-col lg:flex-row-reverse animate-slide-in-right">
          <div className="w-full lg:w-1/2">
            <Carousel slides={slides} />
          </div>
          <div className="w-full lg:w-1/2 animate-zoom-in">
            <h1 className="text-5xl font-bold">Welcome to TurfSpot</h1>
            <p className="py-6">
              Discover and book the best turf fields in your area. Whether
              you&#39;re planning a casual game or a tournament, TurfSpot has
              got you covered.
            </p>
            <Link
              to={isLoggedIn ? "/auth/turfs" : "/signup"}
              className="btn btn-accent"
            >
              Get Started
            </Link>
          </div>
        </div>
      </div>

      <div className="container mx-auto p-4 animate-slide-in-left">
        <h2 className="text-3xl font-bold mb-6">Featured Turfs</h2>
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
          {loading
            ? Array.from({ length: 3 }).map((_, index) => (
              <TurfCardSkeleton key={`skeleton-${index}`} />
            ))
          : turfs
              .slice(0, 3)
              .map((turf) => <TurfCard key={turf._id} turf={turf} />)}
        </div>

        <div className="text-center mt-8">
          <Link
            to={isLoggedIn ? "/auth/turfs" : "/turfs"}
            className="btn btn-primary"
          >
            View More Turfs
          </Link>
        </div>
      </div>

      <Footer />
    </div>
  );
};

export default Home;
```

App.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import { Provider } from "react-redux";
import { store, persistor } from "./redux/store";
import { PersistGate } from "redux-persist/integration/react";
import { RouterProvider } from "react-router-dom";
import router from "./router";
import { Toaster } from "react-hot-toast";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <Provider store={store}>
      <PersistGate loading={null} persistor={persistor}>
        <RouterProvider router={router} />
        <Toaster position="bottom-center" duration={500} />
      </PersistGate>
    </Provider>
  </React.StrictMode>
);
```

```
  import ErrorPg from './Component/OtherPages/ErrorPg/ErrorPg'
import Allproductpage from
'./Component/Allproductpage/Allproductpage';import GiftIdea from
"./Component/Home/GiftIdea/GiftIdea";
import GiftIdea2 from
"./Component/Home/GiftIdea2/GiftIdea2";const App = ()
=> {
return (
<>
<BrowserRouter>
<Navbar />
<Routes>
<Route exact path="/" element={<Home />} />
<Route exact path="/Login" element={<Login />} />
<Route exact path="/SignUp" element={<SignUp />} />
<Route exact path="/Admin" element={<Addproduct />} />
<Route exact path="/Error" element={<ErrorPg />} />
<Route exact path="/Contact" element={<Contact />} />
<Route exact path="/Team" element={<Team />} />
<Route exact path="/Profile" element={<UserProfile />} />
<Route exact path="/Giftidia" element={<GiftIdea />} />
<Route exact path="/Giftidia2" element={<GiftIdea2 />} />
        <Route exact path="/product-type/Flowers-in-a-box"
element={<Allproductpagetype={'Flowers-in-a-box'} />} />
```

```
          <Route exact path="/product-type/Luxury-Collection" element={<Allproductpage
type={'Luxury-Collection'} />} />
          <Route exact path="/product-type/Flowers-in-a-vase" element={<Allproductpage
type={'Flowers-in-a-vase'} />} />
          <Route exact path="/product-type/Plant-Gifts" element={<Allproductpage
type={'Plant-Gifts'} />} />
          <Route exact path="/product-type/Hand-tied-Bouquet" element={<Allproductpage
type={'Hand-tied-Bouquet'} />} />
        </Routes>
      </BrowserRouter>
    </>
  );
};

export default App;
```

AdminDashboard.jsx

```
import useDashboardData from "@hooks/admin/useDashboardData";
import StatCard from "./StatCard";
import BookingHistoryChart from "./BookingHistoryChart";
import AdminDashboardSkeleton from "./AdminDashboardSkeleton";
import {
  Users,
  Building,
  MapPin,
  CreditCard,
  UserPlus,
  UserX,
  TrendingUp,
} from "lucide-react";
import { useState } from "react";

const AdminDashboard = () => {
 const { data, loading, error } = useDashboardData();
  const [selectedTimeRange, setSelectedTimeRange] = useState("30");

 if (loading)  {
   return <AdminDashboardSkeleton />;
 }

 if (error) {
   return (
     <div className="flex justify-center items-center h-screen">
       <p>Error loading dashboard data. Please try again later.</p>
     </div>
   );
 }

 if (!data) {
   return null;
 }

   const totalRevenue = data.bookingHistory.reduce((sum, day) => {
     return sum + day.amount;
   }, 0);

  return (
    <div className="min-h-screen bg-base-200 p-4">
      <div className="max-w-7xl mx-auto">
        <h1 className="text-4xl font-bold mb-8 text-center lg:text-left">
          Admin Dashboard
        </h1>
```

```
<div className="grid gap-6 md:grid-cols-2 lg:grid-cols-4 mb-8">
  <StatCard
    title="Total Users"
    value={data.totalUsers}
    icon={Users}
    className="bg-base-100"
  />
  <StatCard
    title="Total Owners"
    value={data.totalOwners}
    icon={Building}
    className="bg-base-100"
  />
  <StatCard
    title="Total Turfs"
    value={data.totalTurfs}
    icon={MapPin}
    className="bg-base-100"
  />
  <StatCard
    title="Total Bookings"
    value={data.totalBookings}
    icon={CreditCard}
    className="bg-base-100"
  />
</div>

<div className="grid gap-6 md:grid-cols-2 lg:grid-cols-3 mb-8">
  <StatCard
    title="Pending Requests"
    value={data.pendingRequests}
    icon={UserPlus}
    className="bg-warning text-warning-content"
  />
  <StatCard
    title="Rejected Requests"
    value={data.rejectedRequests}
    icon={UserX}
    className="bg-error text-error-content"
  />
  <StatCard
    title="Total Revenue"
    value={totalRevenue}
    icon={TrendingUp}
    prefix="₹"
    className="bg-success text-success-content"
  />
</div>

<div className="card bg-base-100 shadow-xl mb-8">
  <div className="card-body max:md:p-0">
    <h2 className="card-title mb-4">Booking History</h2>
    <div className="flex justify-end mb-4">
      <select
        className="select select-bordered w-full max-w-xs"
        value={selectedTimeRange}
        onChange={(e) => setSelectedTimeRange(e.target.value)}
      >
        <option value="7">Last 7 days</option>
        <option value="30">Last 30 days</option>
        <option value="90">Last 90 days</option>
      </select>
    </div>
    <BookingHistoryChart
      data={data.bookingHistory.slice(-selectedTimeRange)}
    />
  </div>
```

```
        </div>
      </div>
    </div>
  );
};

export default AdminDashboard;
```

Reservation.jsx

```
import { format } from "date-fns";
import { getEndTime } from "../../utils/dateUtils";

const ReservationSummary = ({
  selectedDate,
  selectedStartTime,
  duration,
  pricePerHour,
}) => {

  return (
    <div className="mt-6 p-4 bg-base-200 rounded-lg">
      <h3 className="text-lg font-semibold mb-2">Your Reservation</h3>
      <p>Date: {format(selectedDate, "dd-MM-yyyy")}</p>
      <p>
        Time: {selectedStartTime} to {getEndTime(selectedStartTime, duration)}
      </p>
      <p>
        Duration: {duration} hour{duration > 1 ? "s" : ""}
      </p>
<p className="font-bold">Price: {pricePerHour * duration} INR</p>
    </div>
  );
};

export default ReservationSummary;
```

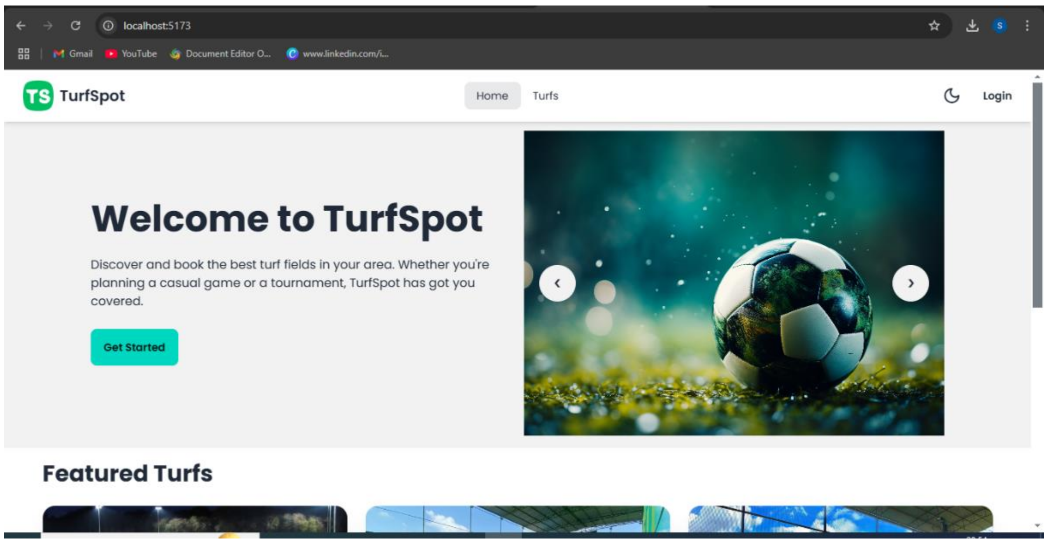
DateSelection.jsx

```
import DatePicker from "react-datepicker";
import "react-datepicker/dist/react-datepicker.css";
import { format, addDays, isSameDay } from "date-fns";

const DateSelection = ({ selectedDate, handleDateChange }) => {
  return (
    <div className="flex flex-col space-y-4 mb-6">
      <div className="w-full">
        <label className="label">
          <span className="label-text">Select Date</span>
        </label>
        <DatePicker
          selected={selectedDate}
          onChange={handleDateChange}
          dateFormat="dd-MM-yyyy"
          minDate={new Date()}
          className="input input-bordered w-full"
        />
      </div>
      <div className="flex flex-col sm:flex-row items-center space-y-2 sm:space-y-0 sm:space-x-
2">
<button
          className="btn btn-outline btn-sm w-full sm:w-auto"
          onClick={() => handleDateChange(addDays(selectedDate, -1))}
          disabled={isSameDay(selectedDate, new Date())}
        >
          PREV DATE
        </button>
        <div className="badge badge-primary text-lg p-4">
          {format(selectedDate, "dd-MM-yyyy")}
        </div>
        <button
          className="btn btn-outline btn-sm w-full sm:w-auto"
          onClick={() => handleDateChange(addDays(selectedDate, 1))}
        >
          NEXT DATE
        </button>
      </div>
    </div>
  );
};

export default DateSelection;
```

TimeSelection.jsx

```
import { parse, isAfter, addHours } from "date-fns";

const TimeSelection = ({
  availableTimes,
  selectedStartTime,
  handleTimeSelection,
  isTimeSlotBooked,
  timeSlots,
  duration,
}) => {
  // const isTimeSlotSelected = (time) => time === selectedStartTime;
  const isTimeSlotSelected = (time) => {
    if (!selectedStartTime || !duration) return false;
    const start = parse(selectedStartTime, "hh:mm a", new Date());
    const end = addHours(start, duration);
    const current = parse(time, "hh:mm a", new Date());
    return current >= start && current < end;
  };

  const isTimeSlotDisabled = (time) => {
    const closeTime = parse(timeSlots.closeTime, "hh:mm a", new Date());
    const currentTime = parse(time, "hh:mm a", new Date());
    return isAfter(currentTime, closeTime) || isTimeSlotBooked(time);
  };

  return (
    <div>
      <h3 className="text-lg font-semibold mb-4">Select Start Time</h3>
      <div className="grid grid-cols-3 sm:grid-cols-4 md:grid-cols-6 gap-2 sm:gap-4">
        {availableTimes.map((time) => (
          <button
            key={time}
            className={`btn btn-sm ${
              isTimeSlotSelected(time)
                ? "bg-blue-500 hover:bg-blue-600 text-white"
                : isTimeSlotDisabled(time)
                ? "btn-disabled"
                : "btn-ghost"
            }`}
            onClick={() => handleTimeSelection(time)}
            disabled={isTimeSlotDisabled(time)}
          >
            {time}
          </button>
        ))}
      </div>
    </div>
  );
};

export default TimeSelection;
```

B. SCREENSHOTS



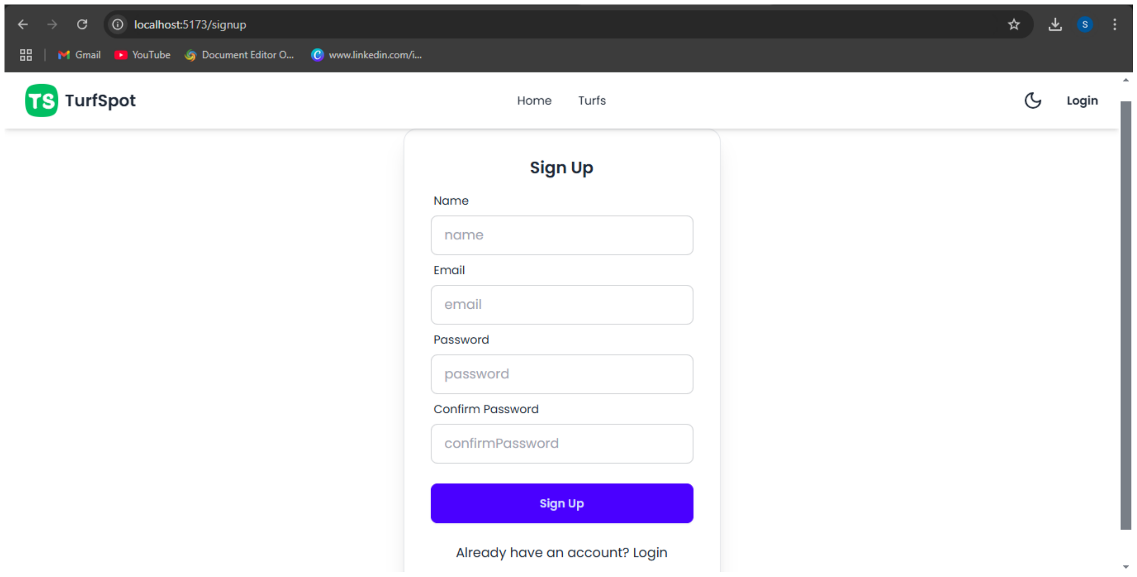
Welcome to TurfSpot

Discover and book the best turf fields in your area. Whether you're planning a casual game or a tournament, TurfSpot has got you covered.

Featured Turfs

Figure B.1 Home Page

In Figure B.1, the TurfSpot homepage includes a logo, a navigation bar with Home, Turfs, and Login, a welcome section highlighting the platform's purpose, and a "Featured Turfs" section displaying various available turfs.



Sign Up

Name: name

Email: email

Password: password

Confirm Password: confirmPassword

Sign Up

Already have an account? Login

Figure B.2 Sign Up Page

In Figure B.2, the Sign-Up page on TurfSpot allows new users to create an account by providing their personal details, such as name, email, and password. It features user-friendly input fields, error validation, and a "Sign Up" button. Additionally, there's a link redirecting existing users to the Login page for quick access.

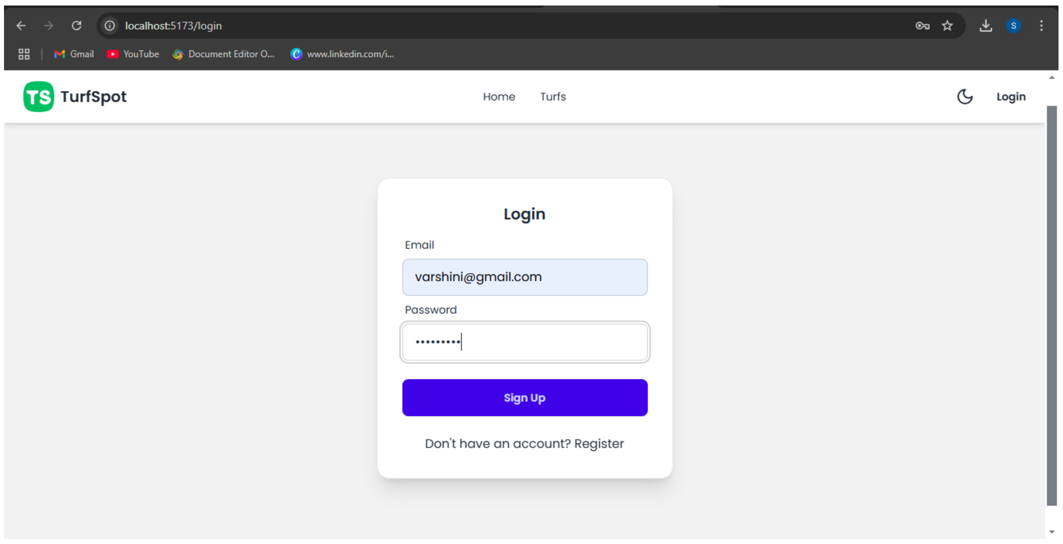
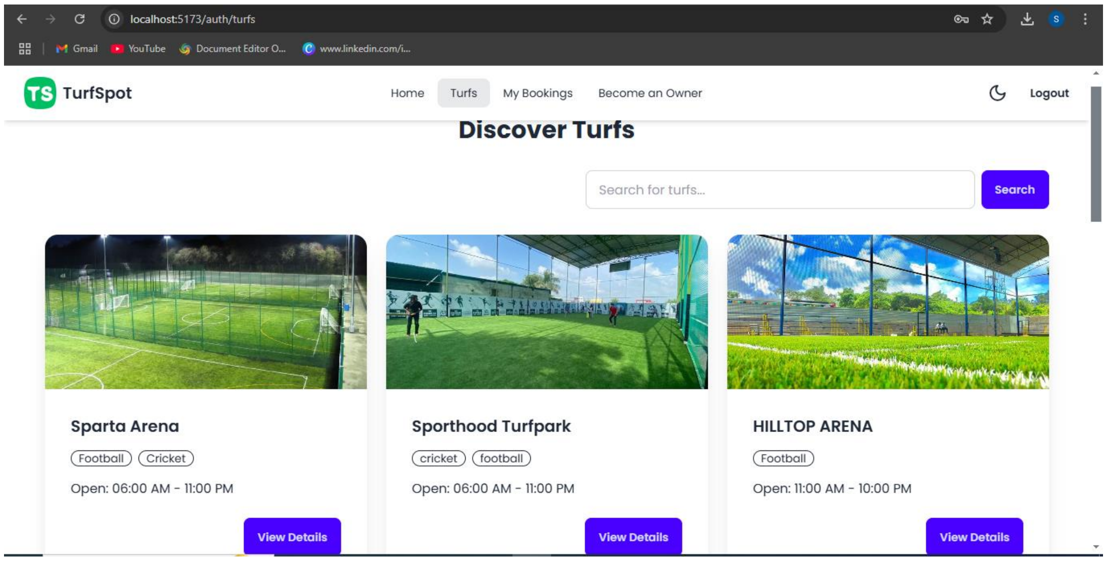


Figure B.3 Login Page

In Figure B.3, the Login page on TurfSpot allows existing users to access their accounts by entering their email and password. It includes a user-friendly interface with error validation for incorrect login details. There’s also a "Login" button and a link redirecting new users to the Sign-Up page to create an account.



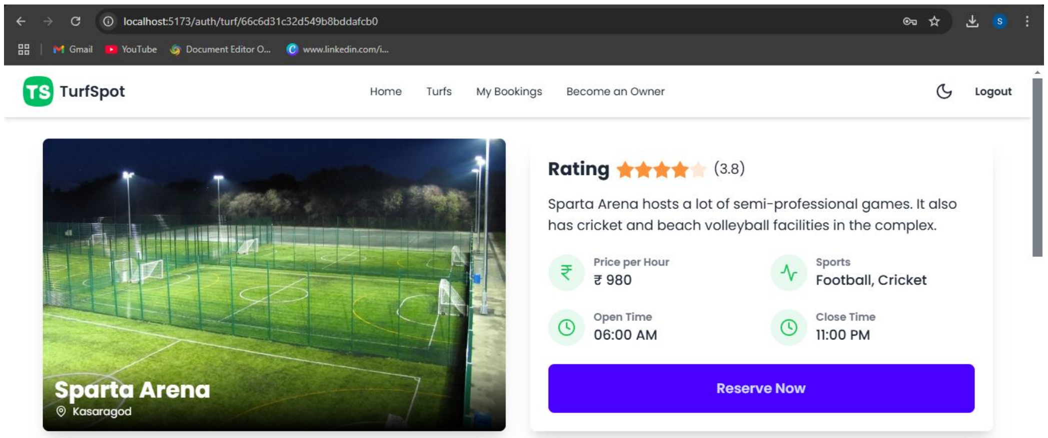
Sparta Arena
Football Cricket
Open: 06:00 AM - 11:00 PM

Sporthood Turfpark
cricket football
Open: 06:00 AM - 11:00 PM

HILLTOP ARENA
Football
Open: 11:00 AM - 10:00 PM

Figure B.4 List of Turfs

In Figure B.4, the List of Turfs page on TurfSpot displays a collection of available turf fields with details such as name, location, and availability. Each turf is presented in a card format with an image, brief description, and a button to view more details or proceed with booking. The page is designed for easy navigation and quick access to turf information.



Sparta Arena
Kasaragod

Rating (3.8)

Sparta Arena hosts a lot of semi-professional games. It also has cricket and beach volleyball facilities in the complex.

Price per Hour: ₹ 980
Sports: Football, Cricket
Open Time: 06:00 AM
Close Time: 11:00 PM

Figure B.5 Turf Details

In Figure B.5, the Turf Details page on TurfSpot provides detailed information about a selected turf, including its name, location, images, and pricing. Users can view reviews and proceed to book the turf. The layout is clean and user-friendly for a seamless experience.

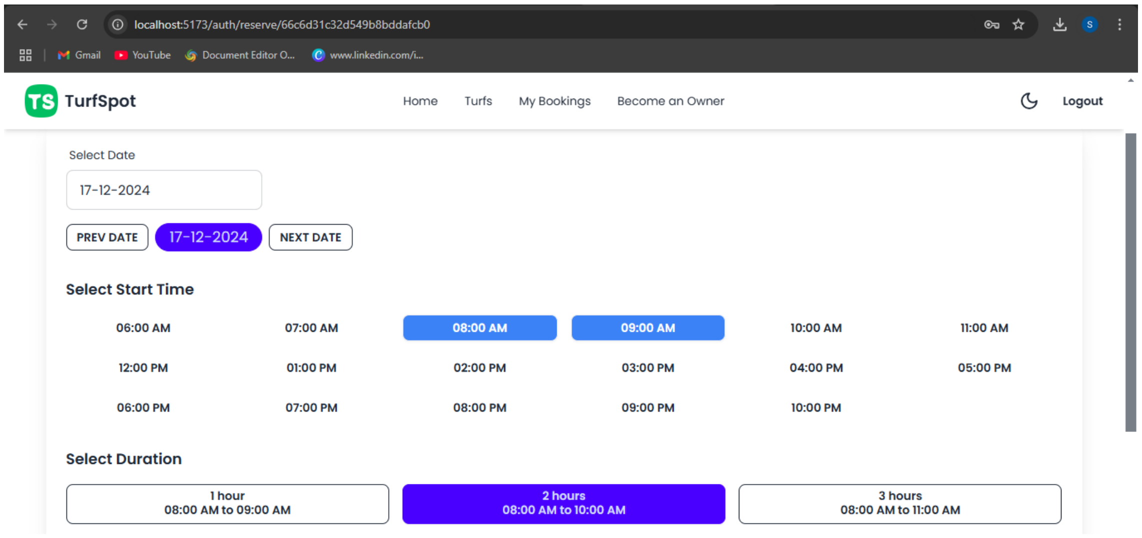


Figure B.6 Booking Turf

In Figure B.6, the Booking Turf page on TurfSpot allows users to select a date and choose an available time slot for their booking. The page features an intuitive interface with a calendar for date selection and a list of time slots to pick from. Once selected, users can proceed to confirm their booking.

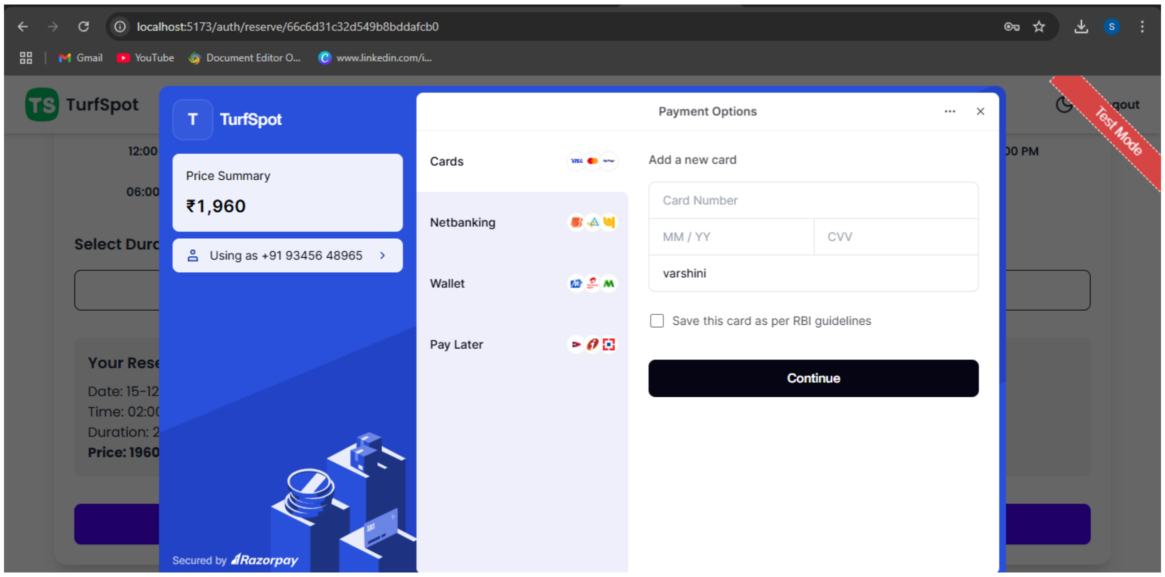
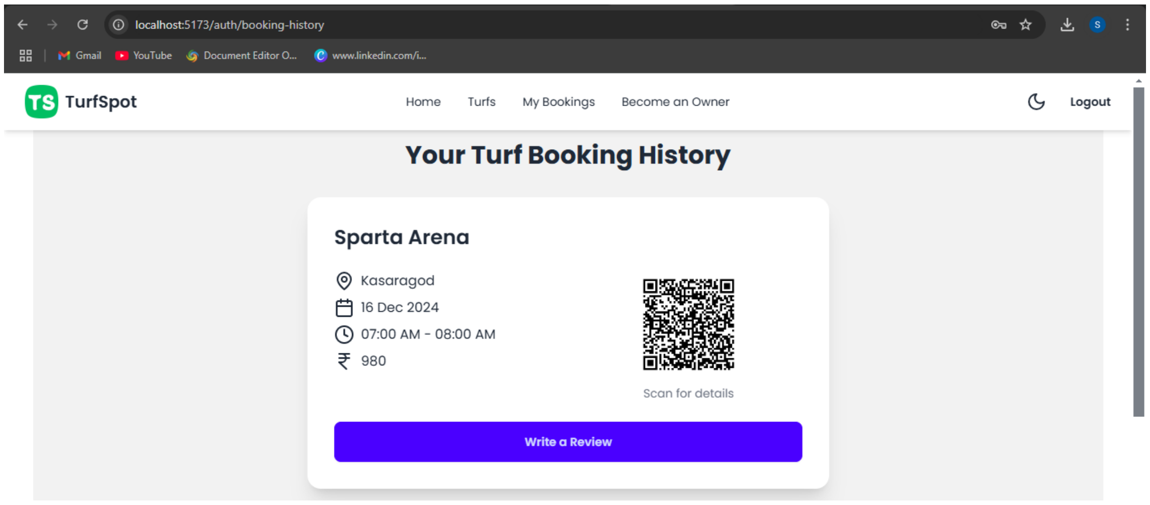


Figure B.7 Payment Page

The Figure B.7 demonstrates Payment Page on TurfSpot lets users securely complete bookings by selecting a payment method and reviewing the booking summary. A confirmation message is shown after successful payment.



Your Turf Booking History

Sparta Arena

Kasaragod

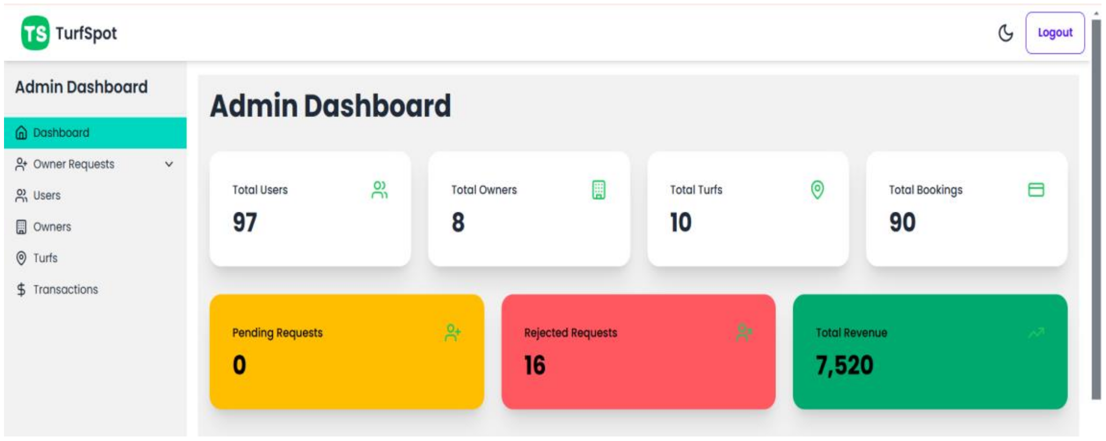
16 Dec 2024

07:00 AM - 08:00 AM

980

Figure B.8 Booking History

The Figure B.8 demonstrates Booking History page on TurfSpot displays a list of past bookings with details like turf name, date, time slot, and bill amount. Users can view and download their bills for reference.

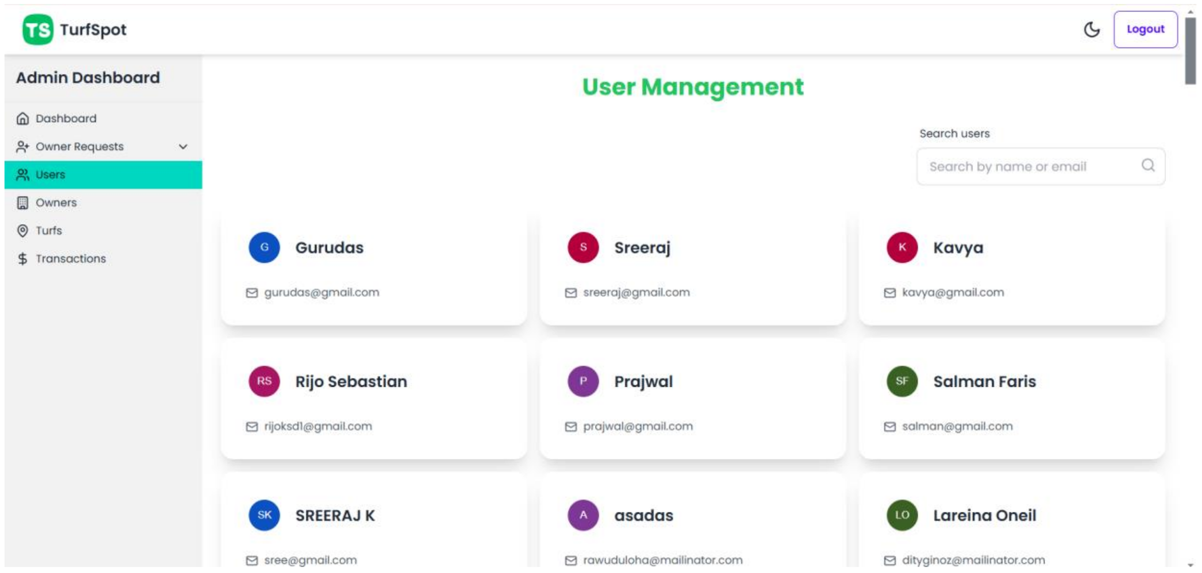


Admin Dashboard

Total Users 97	Total Owners 8	Total Turfs 10	Total Bookings 90
Pending Requests 0	Rejected Requests 16	Total Revenue 7,520	

Figure B.9 Admin Dashboard

The **Admin Dashboard** offers a comprehensive overview of platform activity, including user statistics, owner requests, booking history, and turf management. It provides visual analytics and tools for admins to efficiently monitor and manage users, bookings, and turfs.



User Management

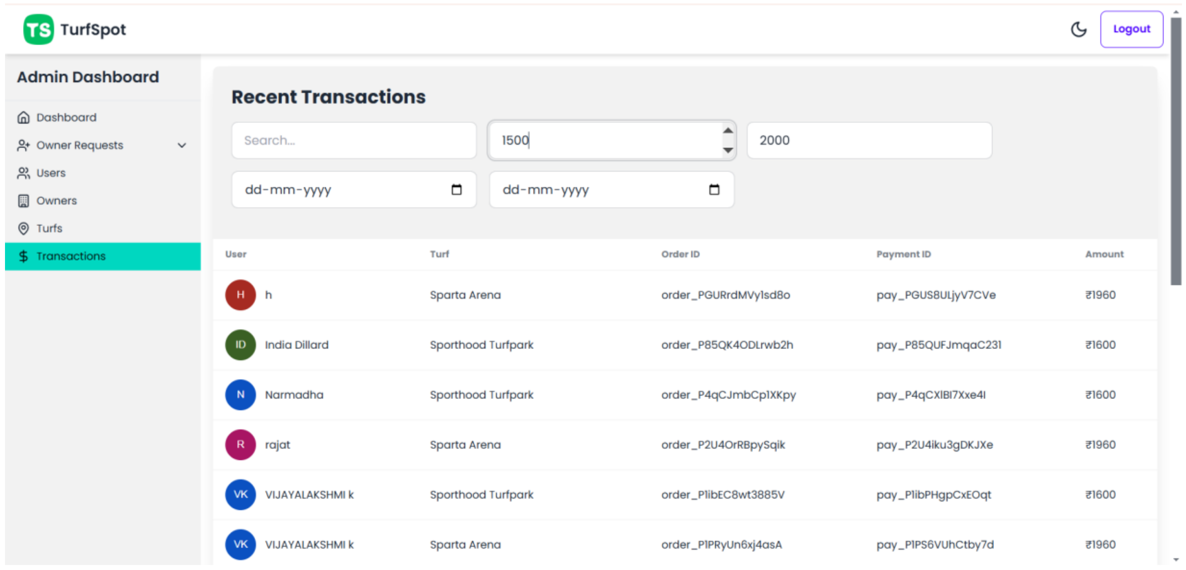
Search users

Search by name or email

Gurudas	gurudas@gmail.com
Sreeraj	sreeraj@gmail.com
Kavya	kavya@gmail.com
Rijo Sebastian	rijoksdl@gmail.com
Prajwal	prajwal@gmail.com
Salman Faris	salman@gmail.com
SREERAJ K	sree@gmail.com
asadas	rawuduloha@mailinator.com
Lareina Oneil	dityginoz@mailinator.com

Figure B.10 User Management

In Figure B.10, User Management allows admins to view and manage all registered users, including their profiles, activity, and status. Admins can track user engagement, suspend or ban accounts, and resolve any issues related to platform usage.



Recent Transactions

Search...

1500

2000

dd-mm-yyyy

dd-mm-yyyy

User	Turf	Order ID	Payment ID	Amount
h	Sparta Arena	order_PGURrdMVylsd8o	pay_PGUS8ULjyV7CVe	₹1960
India Dillard	Sporthood Turfpark	order_P85QK4ODLrwb2h	pay_P85QUFJmqaC231	₹1600
Narmadha	Sporthood Turfpark	order_P4qCJmbCp1XKpy	pay_P4qCXIBI7Xxe4I	₹1600
rajat	Sparta Arena	order_P2U4OrRBpySqik	pay_P2U4iku3gDKJXe	₹1960
VIJAYALAKSHMI k	Sporthood Turfpark	order_PlibEC8wt3885V	pay_PlibPHgpCxEOqt	₹1600
VIJAYALAKSHMI k	Sparta Arena	order_PlPRyUn6xj4asA	pay_PlPS6VUhCtby7d	₹1960

Figure B.11 Transaction History

In Figure B.11, Transaction History provides admins with a detailed overview of all platform transactions, including bookings, payments, and refunds. It allows for tracking of payment statuses, generating reports, and identifying revenue trends over time.

REFERENCES

[1]. Robin Wieruch (2017), The Road to Learn React: Your journey to master plain yet pragmatic React.js.

[2]. Mark Tielens Thomas. React in Action. Manning Publisher, 2018.

[3]. Azat Marden. React Quickly: Painless web apps with React, JSX, Redux, and GraphQL. Simon and Schuster published in 2017.

[4]. Konger, Snehasish (2022). "Advanced React Concepts that You need to Know". Scientyfic World.

[5]. Jonathan Wexler (2019). Get Programming with Node.js

[6]. https://www.geeksforgeeks.org/reactjs/

[7]. https://www.geeksforgeeks.org/expressjs/

[8]. https://www.geeksforgeeks.org/nodejs/

[9]. https://www.mongodb.com/resources/products/platform/mongodb-atlas-tutorial