Project Report

on

Turf Management system

Software Requirements Specification

1. Introduction

1.1 Purpose

The purpose of this document is to provide a comprehensive overview of the requirements for the Turf Management System developed using Django framework. It outlines the functionalities, constraints, and expectations of the system.

1.2 Scope

The Turf Management System aims to provide a web-based platform for efficient management of turfs, including booking, scheduling, and administration. It will serve as a tool for both turf owners and customers.

1.3 Definitions, Acronyms, and Abbreviations

- TMS: Turf Management System
- Django: A high-level Python web framework

2. OverView

2.1 Product Perspective

The Turf Management System is a standalone web application that interacts with users through a web browser. It interfaces with a database to store information about turfs, bookings, customers, and owners.

2.2 User Classes and Characteristics

- Guest: A visitor who can view available turfs but cannot book or manage them.
- Customer: Registered users who can browse, book, and manage their bookings.
- Admin: Admin who owns the turf can manage his turf's listing and booking.

2.3 Operating Environment

The system will be accessible through modern web browsers on desktop and mobile devices. It will be developed using Django framework and utilize a relational database (e.g., PostgreSQL) for data storage.

3. Functional Requirements

3.1 User Authentication and Authorization

1. Users must be able to register and log in with valid credentials.
2. Different user roles (Guest, Customer, Turf Owner) will have distinct permissions.

3.2 Turf Listing and Viewing

1. Guests and registered users can view a list of available turfs.
2. Turf details (price, location, features) will be displayed for each listing.

3.3 Turf Booking

1. Customers can book available turfs for specific dates and times.
2. Turf owners can manage and approve bookings for their own turfs.

3.4 Booking Management

1. Customers can view their booked turfs and make cancellations within a specified timeframe.
2. Turf owners can manage booking requests, approve or reject them.

3.5 Admin Dashboard

1. An admin interface will allow system administrators to manage users, turfs, and bookings.
2. Admins can manage reported issues and resolve disputes.

4. Non-Functional Requirements

4.1 Performance

The system must handle a reasonable number of concurrent users without significant performance degradation.

4.2 Security

1. User passwords must be securely stored using hashing and salting techniques.
2. HTTPS encryption must be employed to ensure data transmission security.

4.3 Usability

The user interface should be intuitive and user-friendly for all user classes.

4.4 Availability

The system should aim for a high uptime percentage, with scheduled maintenance communicated to users in advance.

5. Constraints

- The system development will use Django framework and Python programming language.
- The system will require a compatible relational database management system (e.g., MYSQL,SQLite).