



slington college
(इस्लिंग्टन कलेज)

Module Code & Module Title

CS6P05NI Project

Final Report

Futsal Booking System

Year and Semester

2019-20 Year Long

Student Name: Aadil Ratna Kansakar

London Met ID: 17031232

College ID: NP01CP4A170140

External Supervisor:

Ishwor Shrestha

Internal Supervisor:

Subeksha Shrestha

Acknowledgement

I would like to thank my supervisors Mr. Ishwor Shrestha and Ms. Subeksha Shrestha and Mr Monil Adhikari. All of them have helped me throughout the duration of this module and give advice and share their expertise on the topic without which was very valuable for the project to be completed. Also, I would like to thank Islington College for providing this learning opportunity which has helped me learn a lot.

Abstract

Futsal grounds are still being booked through traditional means like phone calls. A website uses a booking system for a futsal ground. Customers and admins use the website to get information about booking to make it easier and faster. Laravel is used for development of the website. The development is being carried out using agile methodology. Comparisons can be made with other booking systems like QFX movie booking and Booking.com.

Table of Contents

1	Introduction	1
1.1	Project Description	1
1.2	Current Scenario	1
1.3	Problem Statement	1
1.4	Aims and Objectives	2
1.5	Report Structure.....	2
1.5.1	Background	2
1.5.2	Development	2
1.5.3	Testing and Analysis	3
1.5.4	Conclusion.....	3
2	Background	4
2.1	About the End Users	4
2.2	Project Elaboration.....	4
2.3	Function and Features	5
2.4	System Architecture	6
2.5	Tools Used.....	7
2.5.1	Programming Language	7
2.5.2	IDE	7
2.5.3	Database	7

2.5.4	Other.....	7
2.6	Similar System Comparison	8
2.6.1	QFX Cinemas	8
2.6.2	Booking.com.....	9
3	Development	10
3.1	Considered Methodology	10
3.2	Selected Methodology.....	10
3.3	Analysis of project	11
3.4	Wireframes.....	13
3.5	Initial Use-Case Diagram	25
3.5.1	Extended Use Case Diagram	26
3.6	Detailed Development.....	28
3.6.1	Database	28
3.6.2	Authentication Page	31
3.6.3	Home Page.....	41
3.6.4	Booking Page	43
3.6.5	Admin Dashboard.....	54
4	Testing and Analysis	73
4.1	Testing	73
4.1.1	Database Structure Test.....	73

4.1.2	Client Registration Test	74
4.1.3	Admin Login Test.....	78
4.1.4	Client Login Test.....	81
4.1.5	Add Admin Test	84
4.1.6	Delete Admin Test	88
4.1.7	Delete Client Test	90
4.1.8	Admin Delete Booking Test	92
4.1.9	Make Booking Test.....	93
4.1.10	Client Delete Booking Test	96
4.1.11	Change Password Test	98
4.2	Critical Analysis.....	102
5	Future Works.....	103
6	Conclusion	104
7	References.....	105
8	Appendix	107
8.1	Appendix-A: Survey	107
8.1.1	Survey Result	107
8.2	Appendix-B: Sample Codes	110
8.2.1	Sample Code of the UI	110
8.2.2	Sample Code of the back-end	114

8.3	Appendix-C: Designs	117
8.3.1	Gantt Chart.....	117
8.4	Appendix-D: Screenshots of the system	117

Table of Figures

Figure 1: QFX Cinemas.....	8
Figure 2: Booking.com	9
Figure 3: Wireframe Home	13
Figure 4: Wireframe Register	14
Figure 5: Wireframe Login.....	15
Figure 6: Wireframe Dashboard	16
Figure 7: Wireframe View Admins.....	17
Figure 8: Wireframe Add Admin	18
Figure 9: Wireframe View Clients.....	19
Figure 10: Wireframe Booking Details.....	20
Figure 11: Wireframe New Booking 1.....	21
Figure 12: Wireframe New Booking 2.....	22
Figure 13: Wireframe My Bookings	23
Figure 14: Wireframe Change Password	24
Figure 15: Initial Use Case Diagram.....	25
Figure 16: Final ER-Diagram.....	28
Figure 17: Migrating Database	29
Figure 18: Database migrated to localhost.....	29
Figure 19: Database Migration File	30
Figure 20: Authentication Use Case Diagram	31
Figure 21: Registration Collaboration Diagram	33
Figure 22: Login Collaboration Diagram.....	33

Figure 23: Registration Sequence Diagram	34
Figure 24: Login Sequence Diagram.....	35
Figure 25: Authentication Page Register Wireframe	36
Figure 26: Authentication Page Login Wireframe	37
Figure 27: Authentication Page Register UI	38
Figure 28: Authentication Page Login UI.....	39
Figure 29: Authentication RegistrationController	39
Figure 30: Authentication LoginController	40
Figure 31: Home Page Wireframe.....	41
Figure 32: Home Page UI.....	42
Figure 33: Booking Use Case Diagram	43
Figure 34: Booking Collaboration Diagram:.....	45
Figure 35: Booking Sequence Diagram.....	46
Figure 36: New Booking Wireframe	47
Figure 37: Booking Time Wireframe.....	48
Figure 38: My Bookings Wireframe	49
Figure 39: New Booking Page UI	50
Figure 40: Booking Time UI.....	51
Figure 41: My Bookings UI	51
Figure 42: Make a booking back-end	52
Figure 43: My Bookings back-end.....	53
Figure 44: Admin Dashboard Use Case Diagram	54
Figure 45: Admin Dashboard View Admin Collaboration Diagram	57

Figure 46: Admin Dashboard Add Admin Collaboration Diagram	58
Figure 47: Admin Dashboard Delete Admin Collaboration Diagram	58
Figure 48: Admin Dashboard View Admin Sequence Diagram	59
Figure 49: Admin Dashboard Add Admin Sequence Diagram	60
Figure 50: Admin Dashboard Delete Admin Sequence Diagram	61
Figure 51: Dashboard View Admin Wireframe	62
Figure 52: Dashboard Add Admin Wireframe.....	63
Figure 53: Dashboard View Clients Wireframe	64
Figure 54: Dashboard Booking Details Wireframe	65
Figure 55: Dashboard View Admin UI	66
Figure 56: Dashboard Add Admin UI.....	67
Figure 57: Dashboard View Clients UI	67
Figure 58: Dashboard Booking Details UI	68
Figure 59: Dashboard View Admin Back-end.....	69
Figure 60: Dashboard Add Admin back-end	70
Figure 61: Dashboard View Clients back-end	71
Figure 62: Dashboard Booking Details back-end	72
Figure 63: Database Structure Test	73
Figure 64: Client Registration Valid Data Test 1	74
Figure 65: Client Registration Valid Data Test 2	75
Figure 66: Client Registration Invalid Email Test.....	76
Figure 67: Client Registration Empty Fields Test	77
Figure 68: Admin Login Valid Data Test 1	78

Figure 69: Admin Login Valid Data Test 2	79
Figure 70: Admin Login Invalid Data Test	80
Figure 71: Client Login Valid Data Test 1	81
Figure 72: Client Login Valid Data Test 2	82
Figure 73: Client Login Invalid Data Test	83
Figure 74: Add Admin Valid Data Test 1	84
Figure 75: Add Admin Valid Data Test 2	85
Figure 76: Add Admin Invalid Data Test 1	86
Figure 77: Add Admin Invalid Data Test 2	87
Figure 78: Delete Admin Test 1	88
Figure 79: Delete Admin Test 2	89
Figure 80: Delete Client Test 1	90
Figure 81: Delete Client Test 2	91
Figure 82: Admin Delete Booking Test 1	92
Figure 83: Admin Delete Booking Test 2	92
Figure 84: Make Booking Test 1	93
Figure 85: Make Booking Test 2	94
Figure 86: Make Booking Test 3	94
Figure 87: Make Booking Invalid Phone Number Test	95
Figure 88: Delete Client Booking Test 1	96
Figure 89: Delete Client Booking Test 2	97
Figure 90: Change Password Test 1	98
Figure 91: Change Password Test 2	99

Figure 92: Change Password using incorrect current password Test	100
Figure 93: Change Password using unmatching password Test.....	101
Figure 94: Gantt Chart.....	117

Table of Tables

Table 1: Login/Register Extended Use Case	26
Table 2: Make Booking Extended Use Case	26
Table 3: Add Futsal Info Extended Use Case	27
Table 4: View Bookings Extended Use Case	27
Table 5: Authentication Extended Use Case Diagram	32
Table 6: Booking Extended Use Case Diagram	44
Table 7: Admin Dashboard Extended Use Case View Admin	56
Table 8: Admin Dashboard Extended Use Case Add Admin	56
Table 9: Admin Dashboard Extended Use Case Delete Admin	57
Table 10: Database Structure Test	73
Table 11: Client Registration Valid Data Test.....	74
Table 12: Client Registration Invalid Email Test	76
Table 13: Client Registration Empty Fields Test	77
Table 14: Admin Login Valid Data Test	78
Table 15: Admin Login Invalid Data Test.....	80
Table 16: Client Login Valid Data Test.....	81
Table 17: Client Login Invalid Data Test.....	83
Table 18: Add Admin Valid Data Test	84
Table 19: Add Admin Invalid Data Test	86
Table 20: Delete Admin Test.....	88
Table 21: Delete Client Test.....	90
Table 22: Admin Delete Booking Test.....	92
Table 23: Make Booking Test.....	93

Table 24: Make Booking Invalid Phone Number Test	95
Table 25: Delete Client Booking Test	96
Table 26: Change Password Test	98
Table 27: Change Password using incorrect current password Test	100
Table 28: Change Password using unmatching password Test	101

1 Introduction

1.1 Project Description

Futsal Booking System is a website which lets the user book a futsal ground in the Kathmandu valley. This website where users can register and login to make the booking process faster and easier.

1.2 Current Scenario

There are many futsal grounds in the Kathmandu valley. Futsal is mainly popular among the youths and matches are played on a regular basis. The futsal grounds are still being booked through phone calls or through face to face interaction. Futsal Booking Website allows the user to book a futsal ground through the website which makes it easier for the player as well as the admin who works at the ground.

1.3 Problem Statement

In Nepal, futsal is a very popular sport for playing with friends and family. This is an age where most of the services are online. So, a proper futsal booking system for players where a booking can take place is needed to serve the customers. The customers can also view available times to book along with the time and price during different times during the day.

1.4 Aims and Objectives

The project aims to develop a website which gives information about futsal grounds available for booking. The booking system aims to make it easier for futsal players to book for a ground of their choice.

Objectives

- Develop a website to list futsal grounds in Nepal.
- Customers can enter information about futsal grounds.
- Customers can book the futsal ground for their date and time.

1.5 Report Structure

This report gives information on the project being done.

1.5.1 Background

Background section shows how the project was conceived and how the research was done for the project to be carried out. All the physical and software tools needed for the project is listed here as well. Websites with booking systems which are similar to the projected system is compared along with screenshots. Features which may be used for futsal booking system are identified and explained.

1.5.2 Development

Development section shows the development done for the project so far. Here, the architecture of the system and the methodology used is explained in detail. Initial wireframes of the UI are also shown. The use case diagram shows who can use which

feature of the system. This is also explained in detail in the expanded use case diagram. The initial ERD made before normalisation to get a grasp of concept of the working of the system is presented. Each step of the normalisation process is shown as well as the final ERD conceived after normalisation is presented.

1.5.3 Testing and Analysis

Testing phase of the booking system where various tests are run on the website. All the necessary tests are done so that the final developed website can be released.

Analysis of report section analyses the development work done during the first phase. The work done is compared to the Gantt chart to see if the development process is being carried out on time or not. If not, a plan on how the development process can be brought on time is given.

1.5.4 Conclusion

This part of the report wraps up the entire project and gives a final thought about the effectiveness and usability of the booking system and the learning experience as a whole.

2 Background

2.1 About the End Users

Futsal booking system can be used by customers and admins.

Admins are the ones who use this website for displaying the information of their futsal grounds. They will get notifications about booking of their ground at a specific date and time.

Clients are the people who books the futsal ground with their name as ID. Clients can search, view and book a ground on a certain date and time if it is not booked by another customer.

2.2 Project Elaboration

Futsal booking system is a booking system. This system lets a user get information about various futsal grounds in the Kathmandu valley. The user can book a ground for a date and time. Events like tournaments, training is also listed. Ratings can also be given based on the service provided, behaviour of staff and the quality of play. This may affect other customers before booking the ground. Photos of the place is also shown in the website to let the customer see the ground before booking. Location of the place is also shown using maps to get the exact place.

2.3 Function and Features

The merchant needs to register in the website to make their ground available in the website to the players. They can provide information like opening time, closing time, price, photos and other necessary info. The client needs to be registered firstly to make any kind of booking. Once they are registered with all their information and logged in, they are redirected to the dashboard. Here, they can choose a date and time that is available to book. The available timing is listed in blue and booked timing are listed in red. After choosing an available timing, the bookings page is displayed where they can see all the bookings that they have made. The booking page lists out all the booking made by the player using the website for their statements.

2.4 System Architecture

For the back-end programming the website, Laravel framework is used. The Laravel works on the basis of the MVC system architecture.

MVC Framework

MVC stand for Model View Controller. It is a software architecture pattern which separates the functionality, logic and interface of an application.

The model is responsible for interacting with the database like MySQL. Model does the querying part like select, insert, update and delete. Controller can also request data through the model. (Patel, 2019)

The view takes care of the UI of the application. This is what the user sees and interacts with.

The controller performs the actions requested by the user, like visiting a page or submitting a form. It tells the model to fetch data from the database and the controller takes that data to load a view. (Majeed & Rauf, 2018)

Flutter serves as the UI with which the end user interacts. All the data is stored in the database. This data can be created, edited or deleted by the end user with the UI where the user can command.

2.5 Tools Used

2.5.1 Programming Language

The development of the website is to be done using the Laravel framework which uses PHP as the programming language. Laravel is one the widely used PHP frameworks which is used for building web applications. Laravel framework follows MVC structure which makes it easy to read. This framework provides built in features like authentication, mail, routing and so on. It can easily be customized to fit the needs of the project. (larashout, 2018)

2.5.2 IDE

The IDE to be used for the development of this website is Visual Studio Code. Also, this program can be used as a code editor for the Laravel framework that will be used in this project.

2.5.3 Database

MySQL is the database to be used. It is an Oracle backed open source relational database management system. (Rouse, 2018) This is where all the data about futsal, customer, admins and bookings is stored.

2.5.4 Other

- Balsamiq Mock-ups was used for designing the wireframes of the website.
- Lucidchart was used for making the ER-Diagrams, Use-Case diagram, Collaboration diagrams and Sequence diagrams.
- SQL Datamodeler was used to generate database statements.

2.6 Similar System Comparison

2.6.1 QFX Cinemas

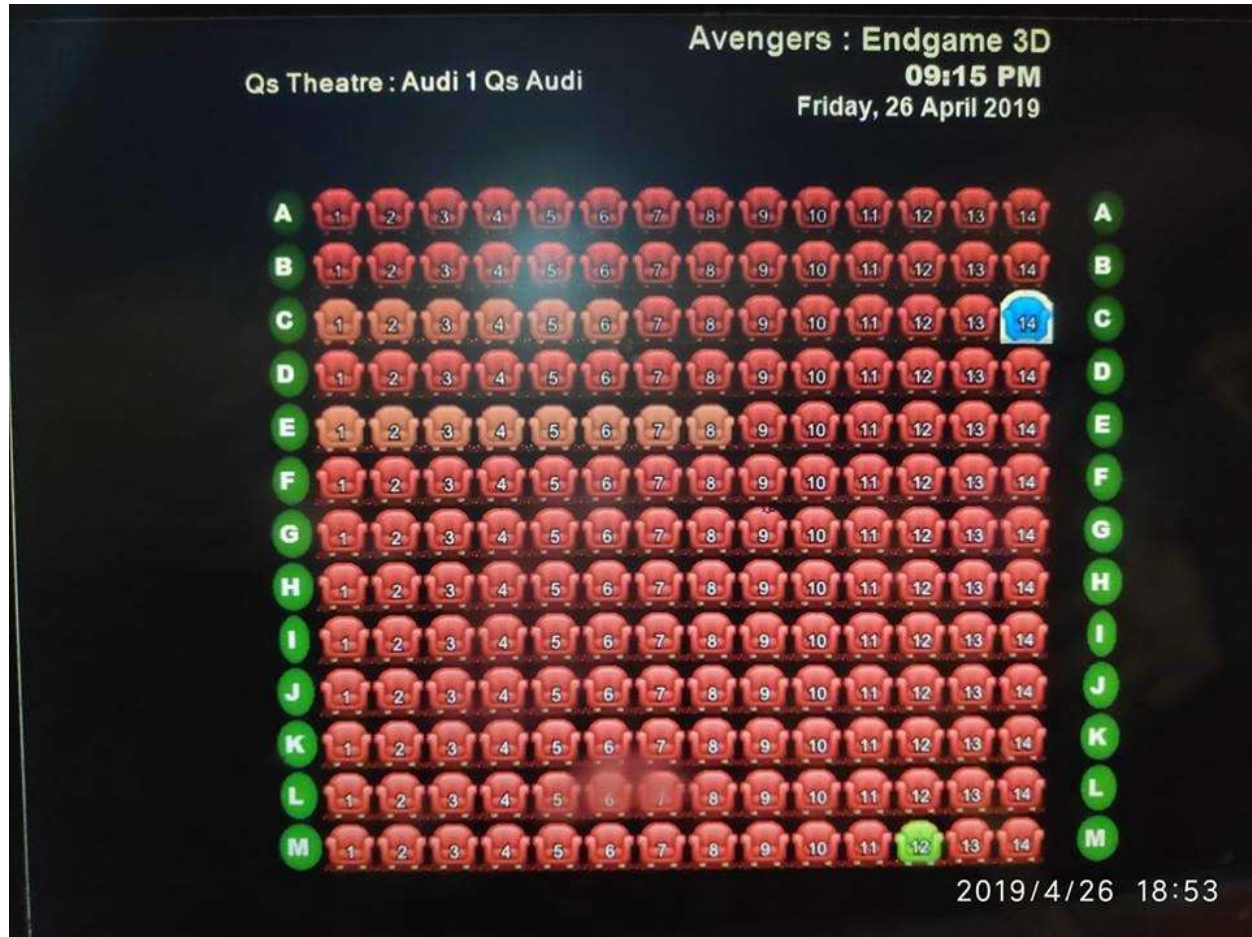


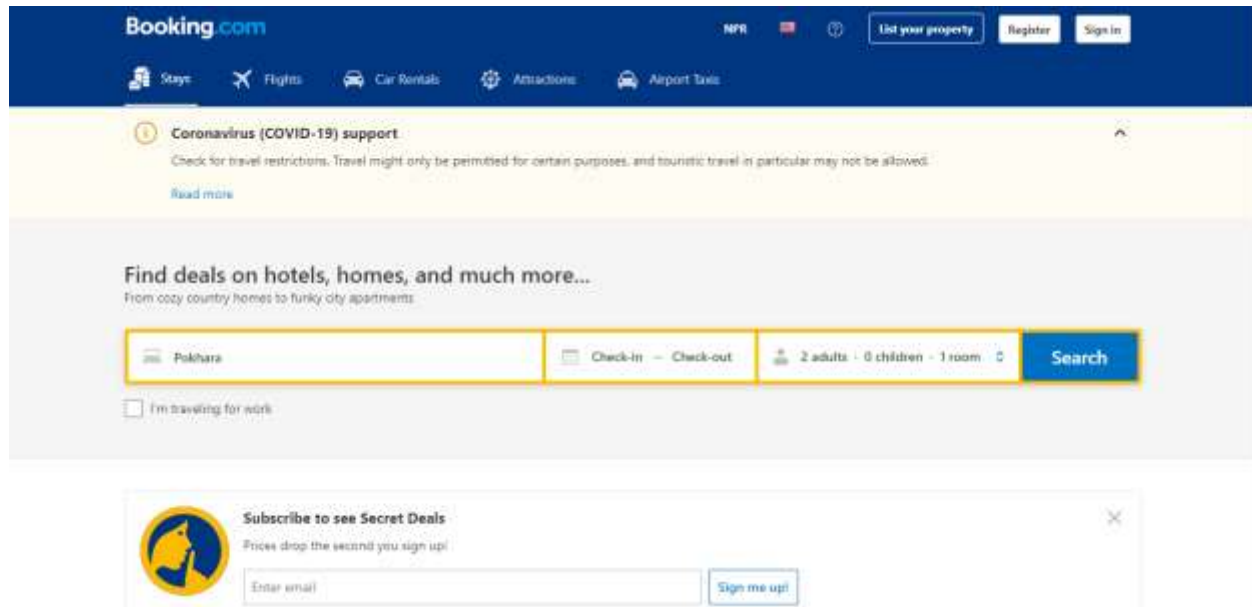
Figure 1: QFX Cinemas

QFX Cinema uses a booking system for the booking of movie tickets using their website as well as their mobile application. This site provides a list of all the movies shown during the day and lets the user choose a seat number.

This is a very useful booking system for people who don't want to stay in line.

This system has very good features like showing available and booked seats assigned with green and red colours.

2.6.2 Booking.com



The image is a screenshot of the Booking.com website. At the top, there is a dark blue navigation bar with the Booking.com logo on the left and links for 'List your property', 'Register', and 'Sign in' on the right. Below the navigation bar, there is a yellow banner with a coronavirus support message. The main section features a search bar with the text 'Find deals on hotels, homes, and much more...'. The search bar contains the following fields: a destination field with 'Pokhara', a check-in and check-out date field, a guest selection field with '2 adults', '0 children', and '1 room', and a blue 'Search' button. Below the search bar, there is a checkbox labeled 'I'm traveling for work'. At the bottom, there is a white box with a yellow circular icon and the text 'Subscribe to see Secret Deals'. It includes a text input field for an email address and a 'Sign me up!' button.

Figure 2: Booking.com

Booking.com is a hotel booking website, which is one of the most popular hotels booking websites in the world. This website can be used to search hotel and rooms available for booking.

This website has a feature where a user can check if the room is booked and if other rooms are available for booking.

3 Development

Initial research on the topic is done and open to change accordingly. Wireframes of all the pages of the website have been made to get a grasp of how the website will look. ERD was made to see all the entities needed for the website to function. Use Case diagram has also been completed to see who gets to use which functionality of the website. Normalisation was done to remove any anomalies in the system and the final ERD was prepared accordingly.

3.1 Considered Methodology

The following methodologies were considered for this project:

- Waterfall Methodology
- Kanban Methodology

3.2 Selected Methodology

After much thinking, the methodology used for this project was chosen as scrum.

Agile Methodology

Agile is a process that allows companies design and build the right product as per the demand of the client. This process helps the company to analyse and improve the product throughout its development. (Cprime, 2019)

Scrum

Scrum is an agile process that is mostly used while managing complex software and product development, using iterative and incremental practices. Scrum process allows

companies to adjust smoothly to rapidly changing requirements. This agile process provides better estimation taking less time creating them and also let us be in more control of the project schedule.

The agile process allows for the face to face interactions of customer, software development team, product owner which makes it easier for ideas to be circulated.

3.3 Analysis of project

The project title has been finalized as Futsal Booking System. The required research has been done and the software tools needed has been installed.

The wireframes to get an idea of the UI design of all the pages of the website has been made. All the entities needed for storing data in the database has been identified. The database has been normalised and the final ER-Diagram has been prepared. Use Case Diagram has also been made along with the Extended Use Case to know who can use which features of the website.

Survey was carried out as well and some responses were recorded. The project will be adjusted as per the responses received.

According to the proposed plan, the project is not moving on time. Since the learning time of Laravel took far longer than expected, the development of the project could not move forward. After spending time getting familiar with the Laravel framework, the development process can move forward with some changes to the proposed plan.

So, to bring the project on time, the length of testing phase is decreased to 14 days and the deployment phase is decreased to 4 days. In the 19 extra days, the project is to be brought on time. The project can then be carried out accordingly.

3.4 Wireframes



Figure 3: Wireframe Home

The image shows a wireframe for a registration form within a web browser window. The browser's title bar reads "Futsal Booking System". The address bar contains the URL "https://127.0.0.1:8000". The registration form is centered on a white background, flanked by dark grey vertical bars. The form is titled "Create an Account" and contains six input fields: "First Name", "Last Name", "Email", "Password", "Password Again", and a "Register" button. The browser window has standard navigation icons (back, forward, close, home) and a search icon.

Futsal Booking System

https://127.0.0.1:8000

Create an Account

First Name

Last Name

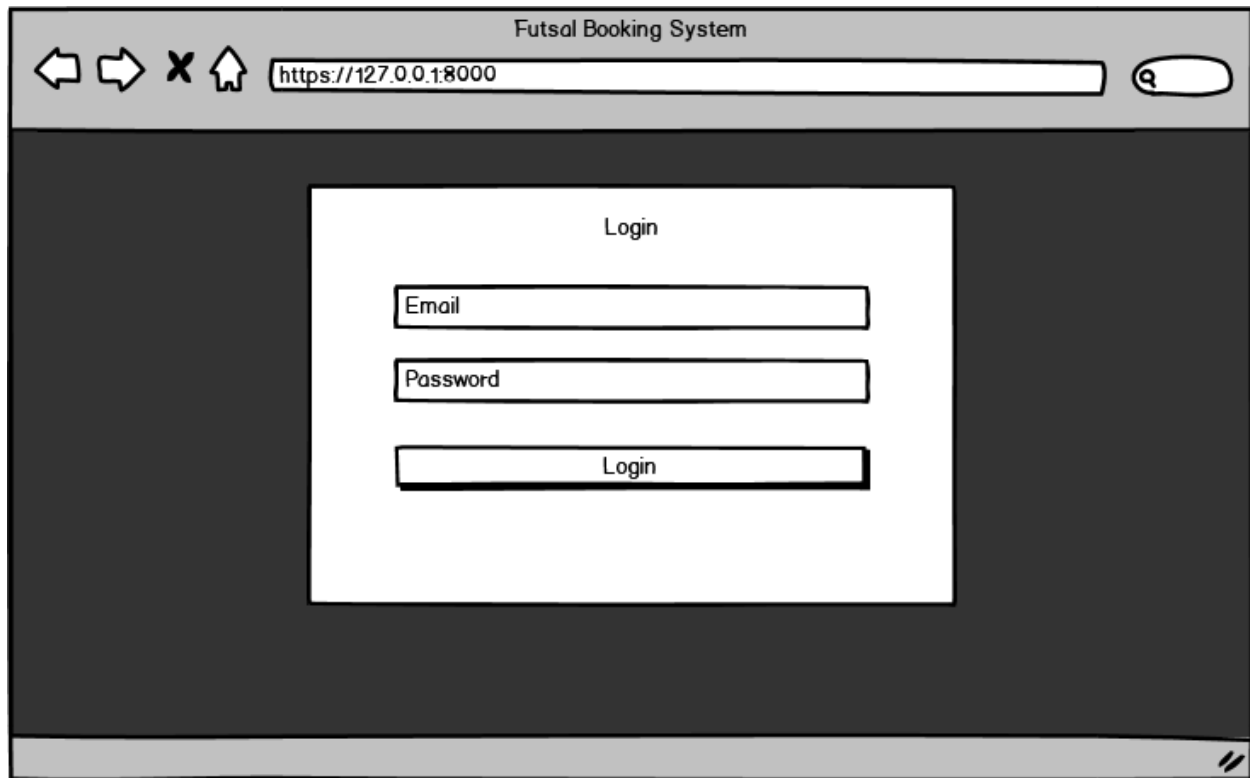
Email

Password

Password Again

Register

Figure 4: Wireframe Register



The image shows a wireframe of a login page within a web browser window. The browser's title bar reads "Futsal Booking System". The address bar contains the URL "https://127.0.0.1:8000". The main content area has a dark gray background. Centered on this background is a white rectangular box containing the login form. The form is titled "Login" at the top. Below the title are three input fields: "Email", "Password", and a "Login" button. The "Email" and "Password" fields are simple text boxes, while the "Login" button is a rectangular box with the text "Login" centered inside it.

Figure 5: Wireframe Login

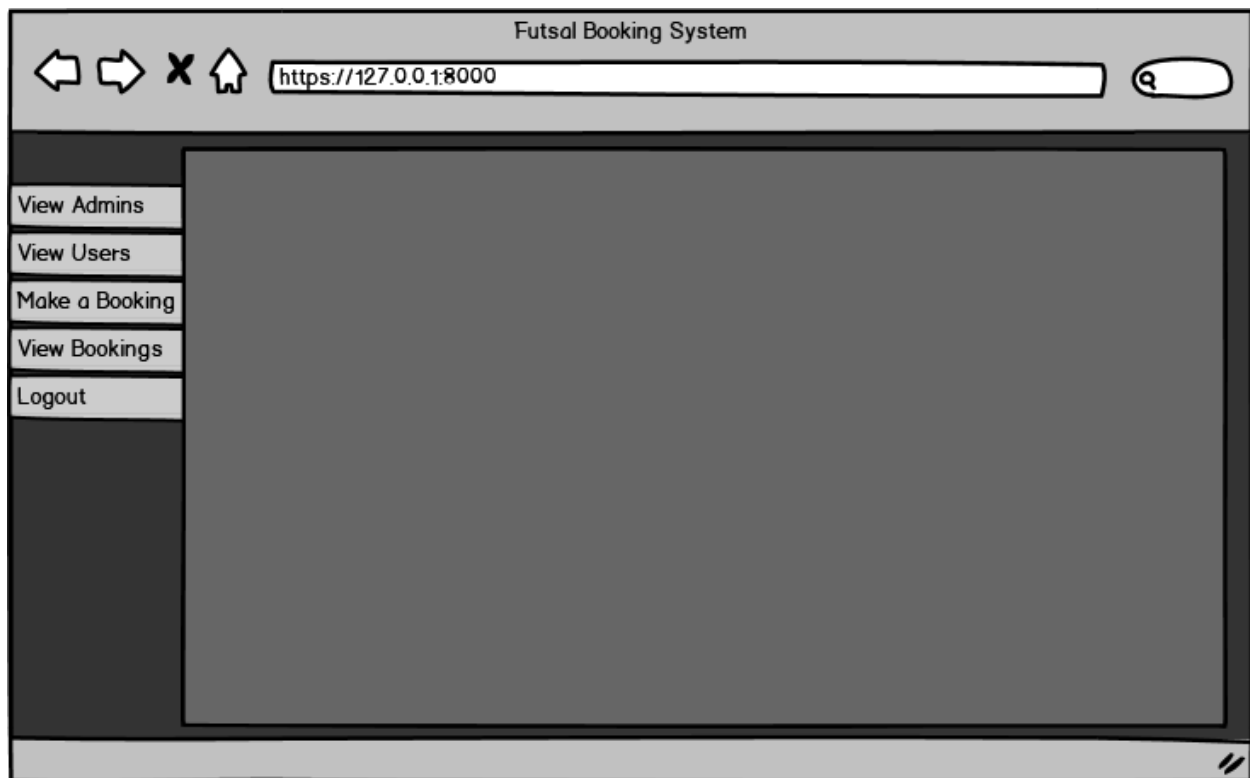


Figure 6: Wireframe Dashboard

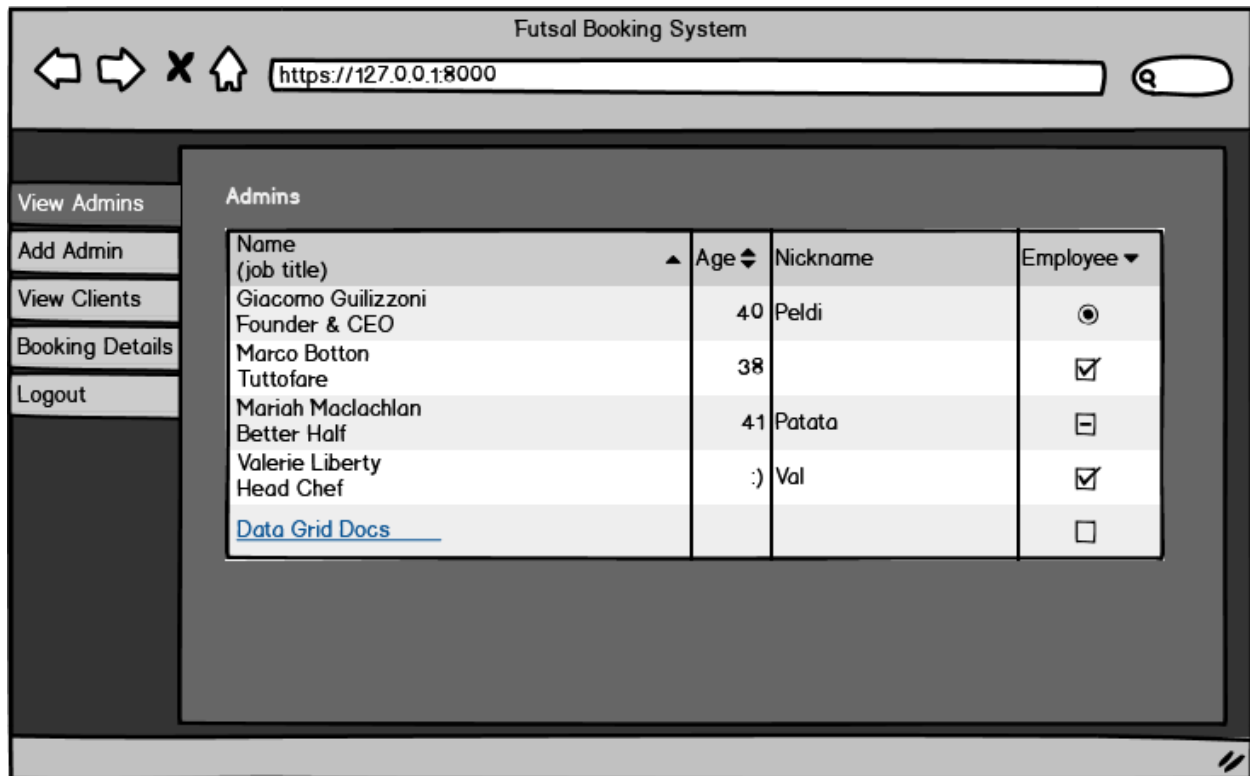


Figure 7: Wireframe View Admins

The image shows a wireframe of a web browser window titled "Futsal Booking System". The address bar displays "https://127.0.0.1:8000". On the left, a sidebar menu contains the following items: "View Admins", "Add Admin" (highlighted), "View Clients", "Booking Details", and "Logout". The main content area features a central white box titled "Add Admin" with the following form elements: "First Name", "Last Name", "Email", "Password", "Password Again", and an "Add" button.

Figure 8: Wireframe Add Admin

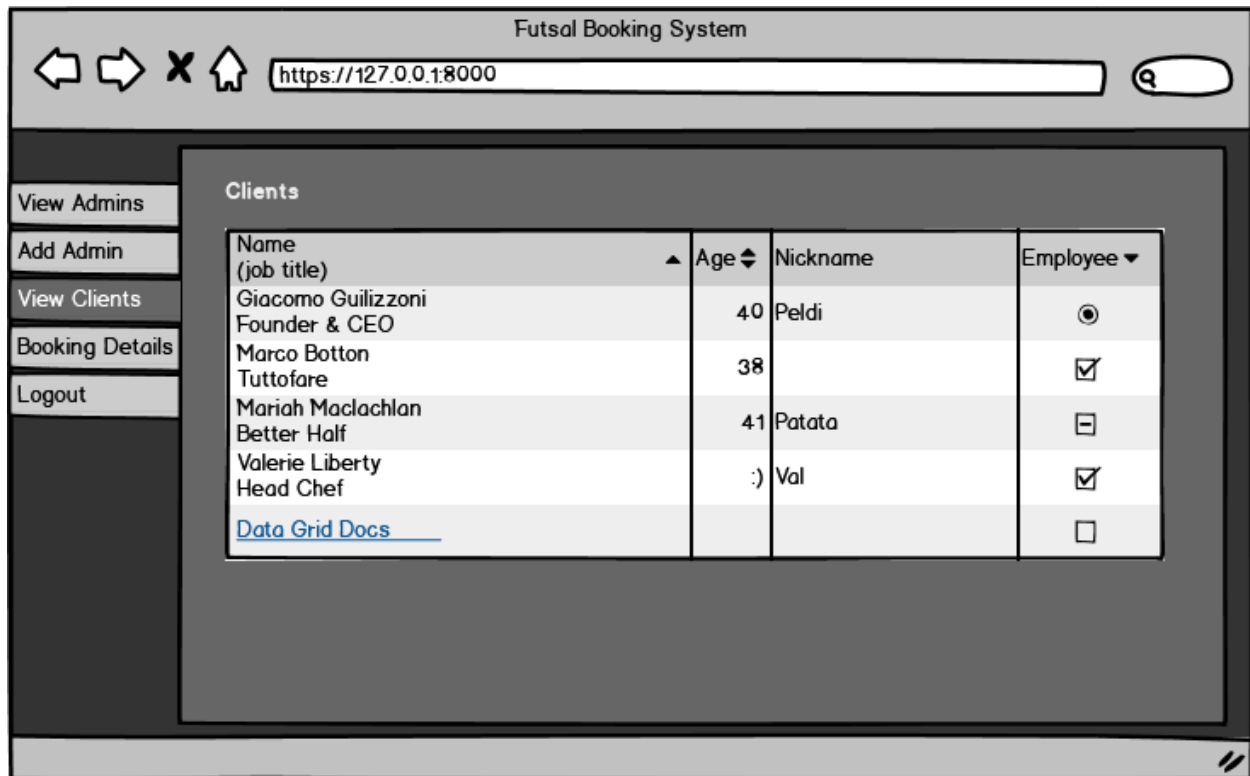


Figure 9: Wireframe View Clients

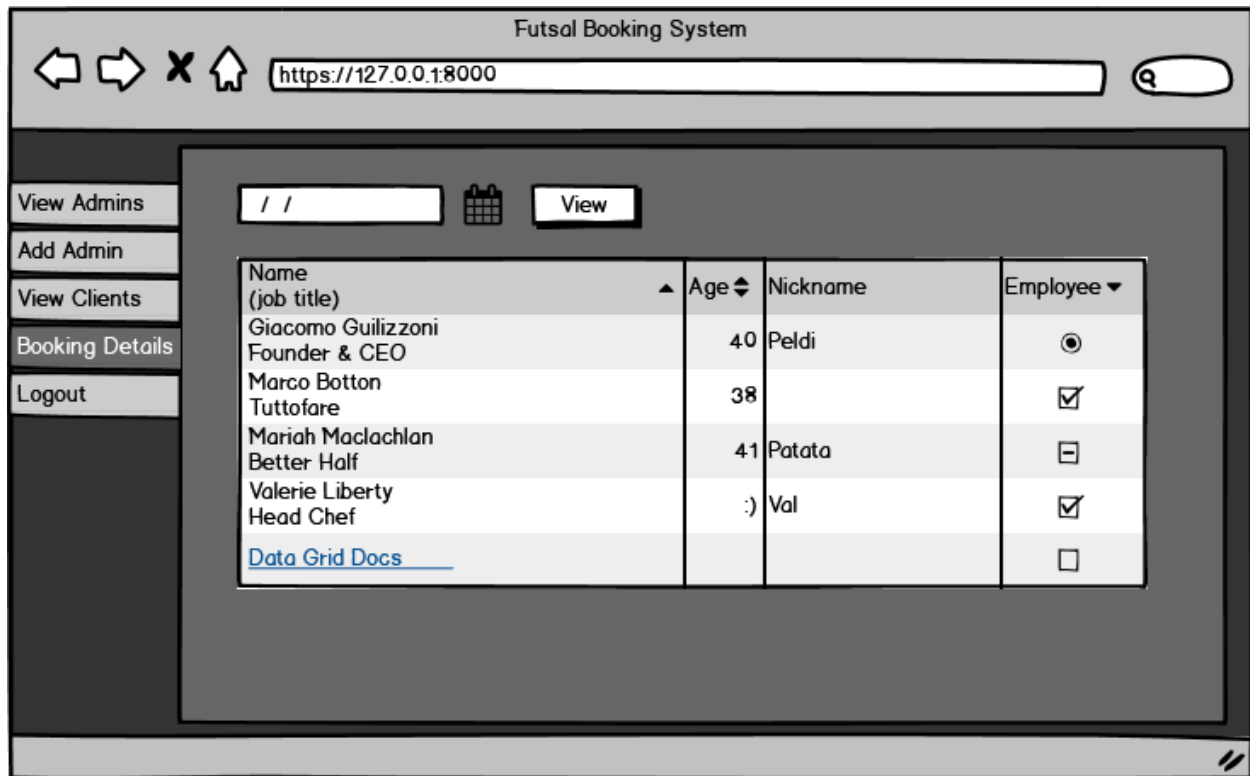


Figure 10: Wireframe Booking Details

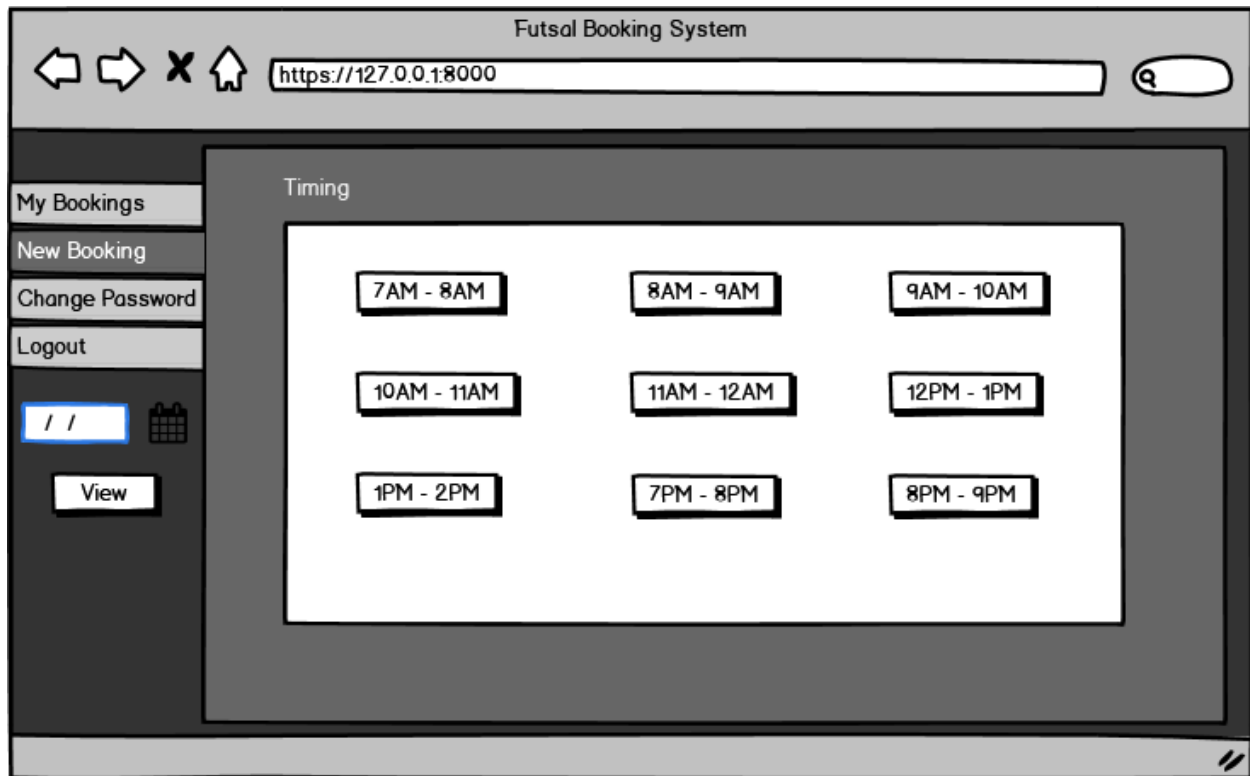


Figure 11: Wireframe New Booking 1

Futsal Booking System

https://127.0.0.1:8000

My Bookings

New Booking

Change Password

Logout

/ /

View

Booking

Date

Time

Phone

Price

Book

Figure 12: Wireframe New Booking 2

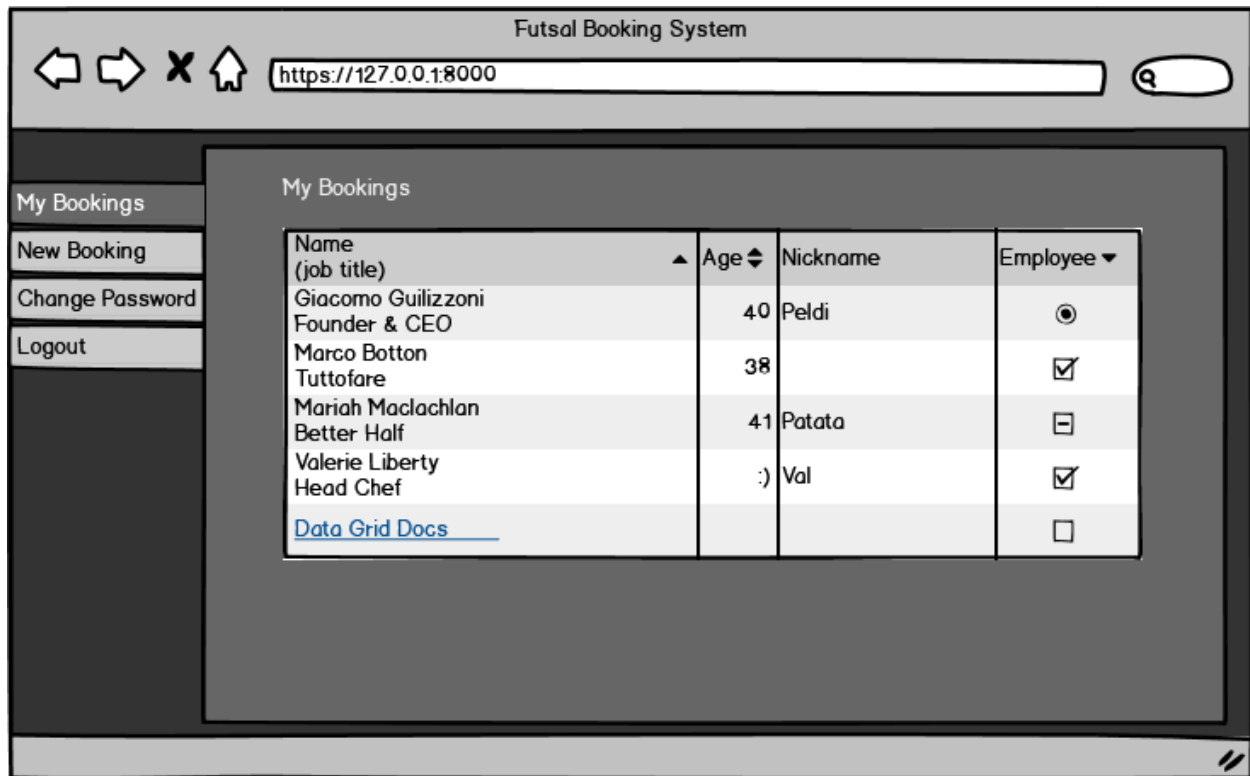


Figure 13: Wireframe My Bookings

The wireframe shows a web browser window titled "Futsal Booking System". The address bar contains "https://127.0.0.1:8000". On the left is a sidebar menu with four items: "My Bookings", "New Booking", "Change Password", and "Logout". The "Change Password" item is highlighted. The main content area displays a "Change Password" form with four input fields: "Old Password", "New Password", "Confirm Password", and a "Change" button.

Futsal Booking System

https://127.0.0.1:8000

My Bookings

New Booking

Change Password

Logout

Change Password

Old Password

New Password

Confirm Password

Change

Figure 14: Wireframe Change Password

3.5 Initial Use-Case Diagram

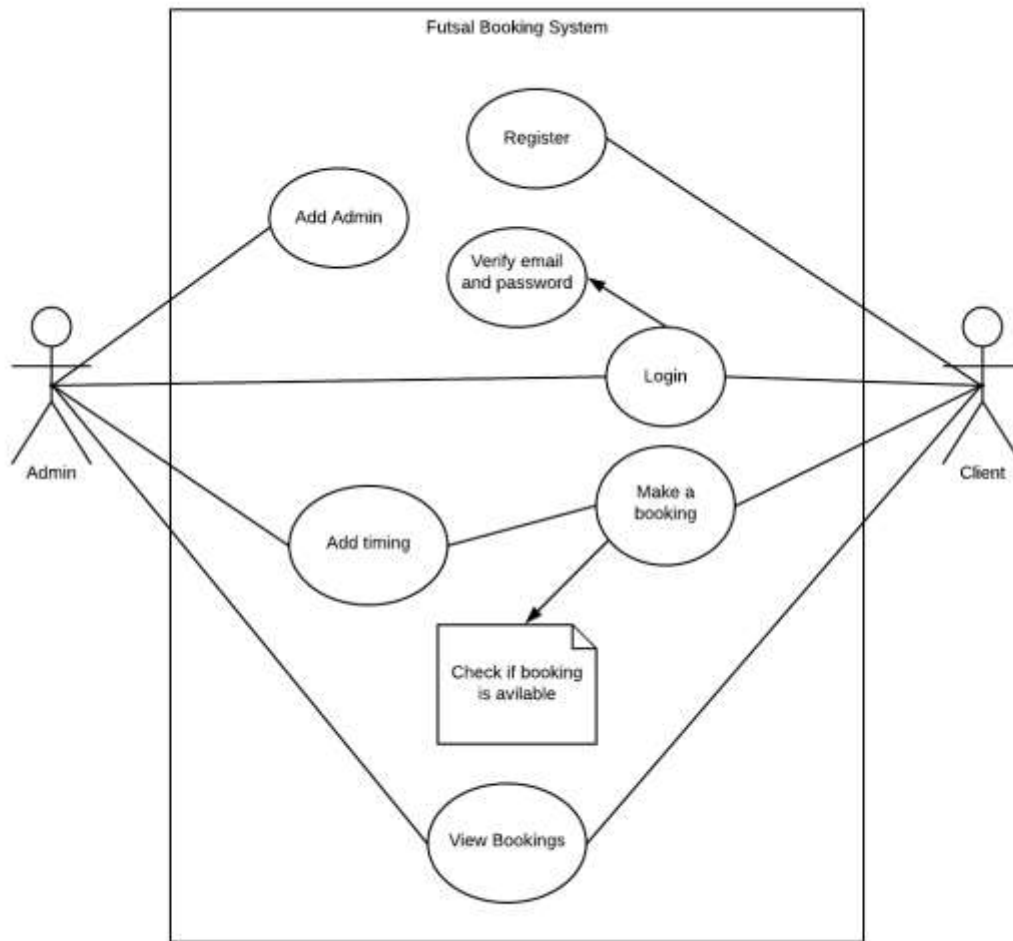


Figure 15: Initial Use Case Diagram

3.5.1 Extended Use Case Diagram

3.5.1.1 Login/Register

Customer/Admin	System
1. Enter user details.	
	2. Record details/ Verify username and password.
	3. Registration/ Login successful.
	4. Redirect to front page.

Table 1: Login/Register Extended Use Case

3.5.1.2 Make Booking

Customer	System
1. View futsal ground page.	
2. Press check availability button.	
	3. Bring up calendar page with available time.
4. Pick available time.	
	5. Redirect to payment page.

Table 2: Make Booking Extended Use Case

3.5.1.3 Add Futsal Info

Admin	System
1. Add/Update Futsal ground.	
	2. Bring up futsal ground form.
3. Enter updated information.	
	4. Update in database.

*Table 3: Add Futsal Info Extended Use Case***3.5.1.4 View Bookings**

Customer/Admin	System
1. View bookings page.	
	2. Retrieve information about customer booking.
	3. Show all bookings made by customer.

Table 4: View Bookings Extended Use Case

3.6 Detailed Development

3.6.1 Database

3.6.1.1 Design

For designing the booking system, Sentinel was used. Sentinel is a modern PHP framework which provides authorization and authentication package which features roles, permissions and other additional security features. (Cartalyst, 2018)

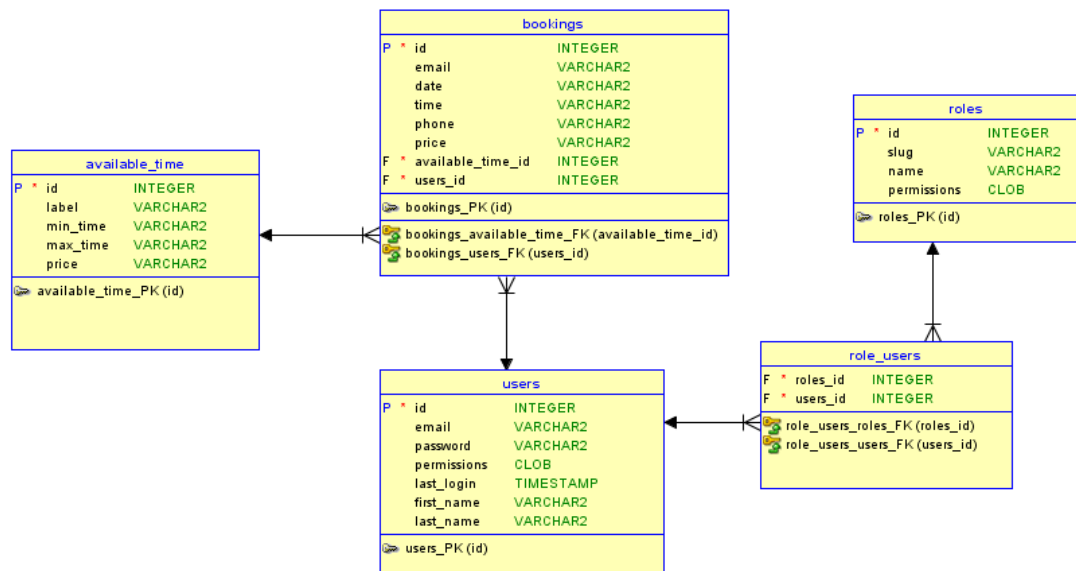


Figure 16: Final ER-Diagram

3.6.1.2 Development

For the development, migrations were created inside of the Laravel framework. Sentinel framework was also imported which made another migration containing all the important tables like roles, permissions and activation. Other migrations were also created which created tables like bookings and available times.

The migrations were carried out using the terminal in VSCode.

```
C:\xampp\htdocs\futsal_site>php artisan migrate:refresh
Rolling back: 2020_05_20_150443_create_available_times
Rolled back: 2020_05_20_150443_create_available_times
Rolling back: 2020_05_15_072639_create_bookings
Rolled back: 2020_05_15_072639_create_bookings
Rolling back: 2014_07_02_230147_migration_cartalyst_sentinel
Rolled back: 2014_07_02_230147_migration_cartalyst_sentinel
Migrating: 2014_07_02_230147_migration_cartalyst_sentinel
Migrated: 2014_07_02_230147_migration_cartalyst_sentinel
Migrating: 2020_05_15_072639_create_bookings
Migrated: 2020_05_15_072639_create_bookings
Migrating: 2020_05_20_150443_create_available_times
Migrated: 2020_05_20_150443_create_available_times
```

Figure 17: Migrating Database

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> activations	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> available_times	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> bookings	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> persistences	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> reminders	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> roles	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> role_users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> throttle	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
10 tables	Sum	5	InnoDB	utf8mb4_general_ci	224.0 KiB	0 B

Figure 18: Database migrated to localhost

3.6.1.3 Implementation

```
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class MigrationCartalystSentinel extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('activations', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('user_id')->unsigned();
            $table->string('code');
            $table->boolean('completed')->default(0);
            $table->timestamp('completed_at')->nullable();
            $table->timestamps();

            $table->engine = 'InnoDB';
        });
    }
}
```

Figure 19: Database Migration File

After the final database was created and migrated, two roles admin and client were inserted into the roles table. This is done so that the user registering a new account gets assigned the role of a client so, bookings are made available for them.

3.6.2 Authentication Page

3.6.2.1 Design

3.6.2.1.1 Use-Case

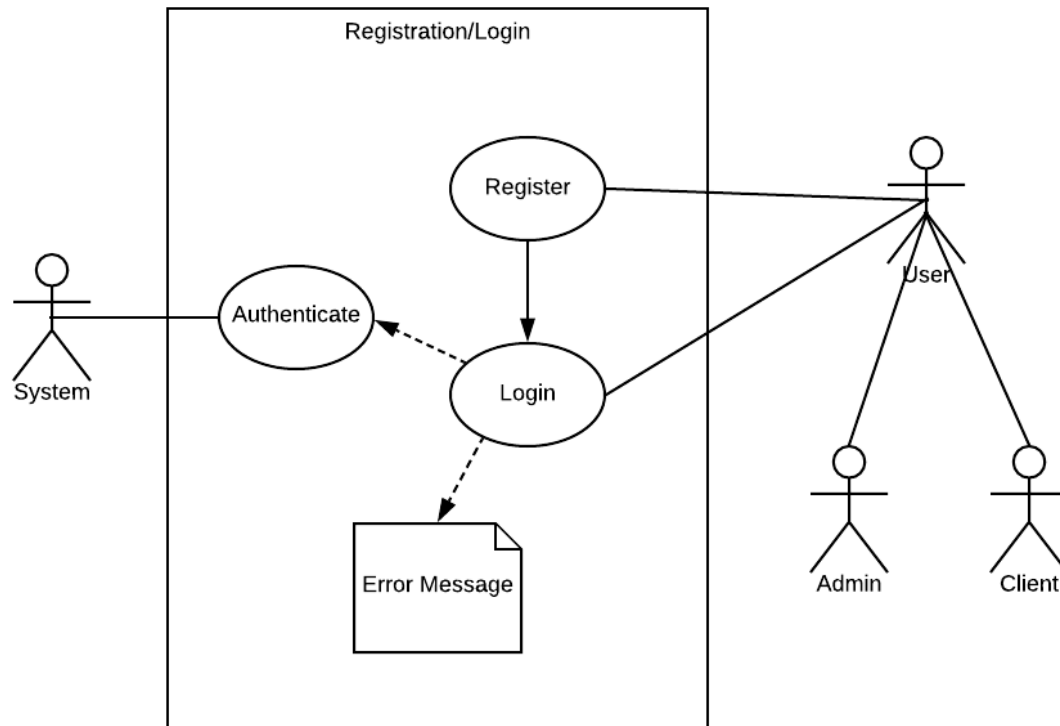


Figure 20: Authentication Use Case Diagram

3.6.2.1.2 High Level Description

- Register

Actor: User

Description: The user registers and account for the futsal booking system using the website.

- Login

Actor: User

Description: The user logs into the system after having a registered account. Both admin and client can login using the same method.

3.6.2.1.3 Extended Use-Case Diagram

User	System
1. Register	
	2. Check details
	3. Register and Activate user
4. Login	
	5. Authenticate user
	6. Redirect user to dashboard

Table 5: Authentication Extended Use Case Diagram

3.6.2.1.4 Collaboration Diagram

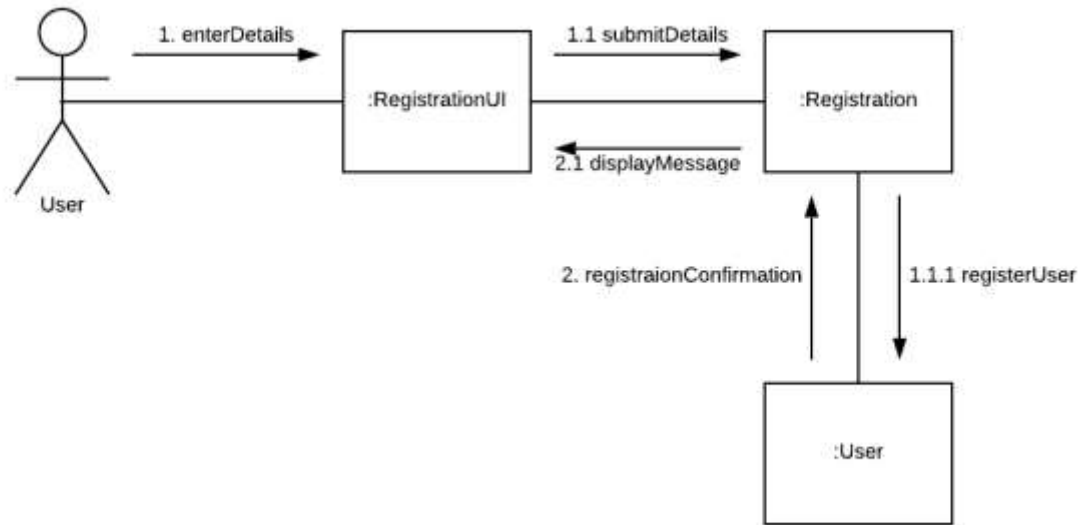


Figure 21: Registration Collaboration Diagram

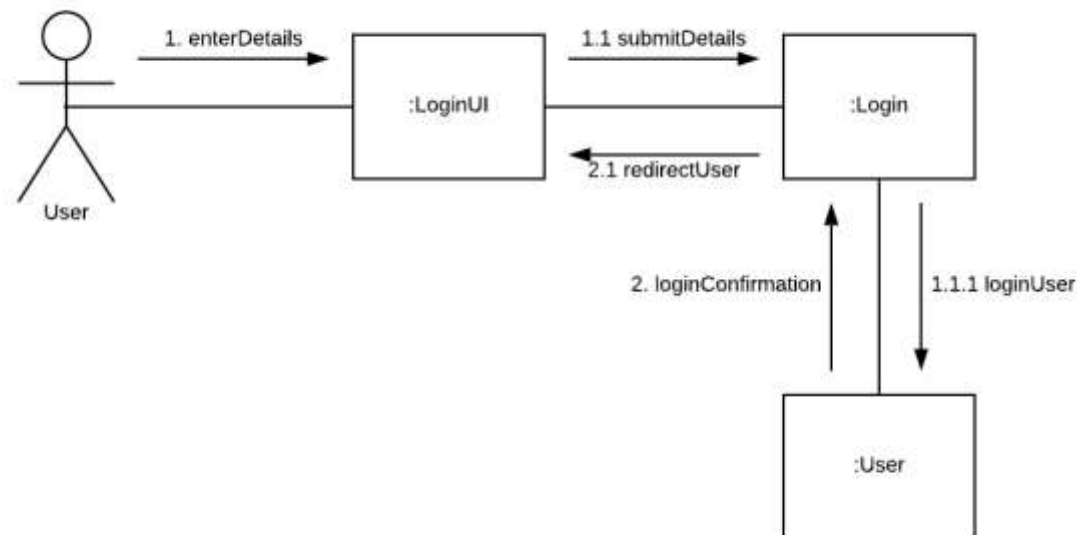


Figure 22: Login Collaboration Diagram

3.6.2.1.5 Sequence Diagram

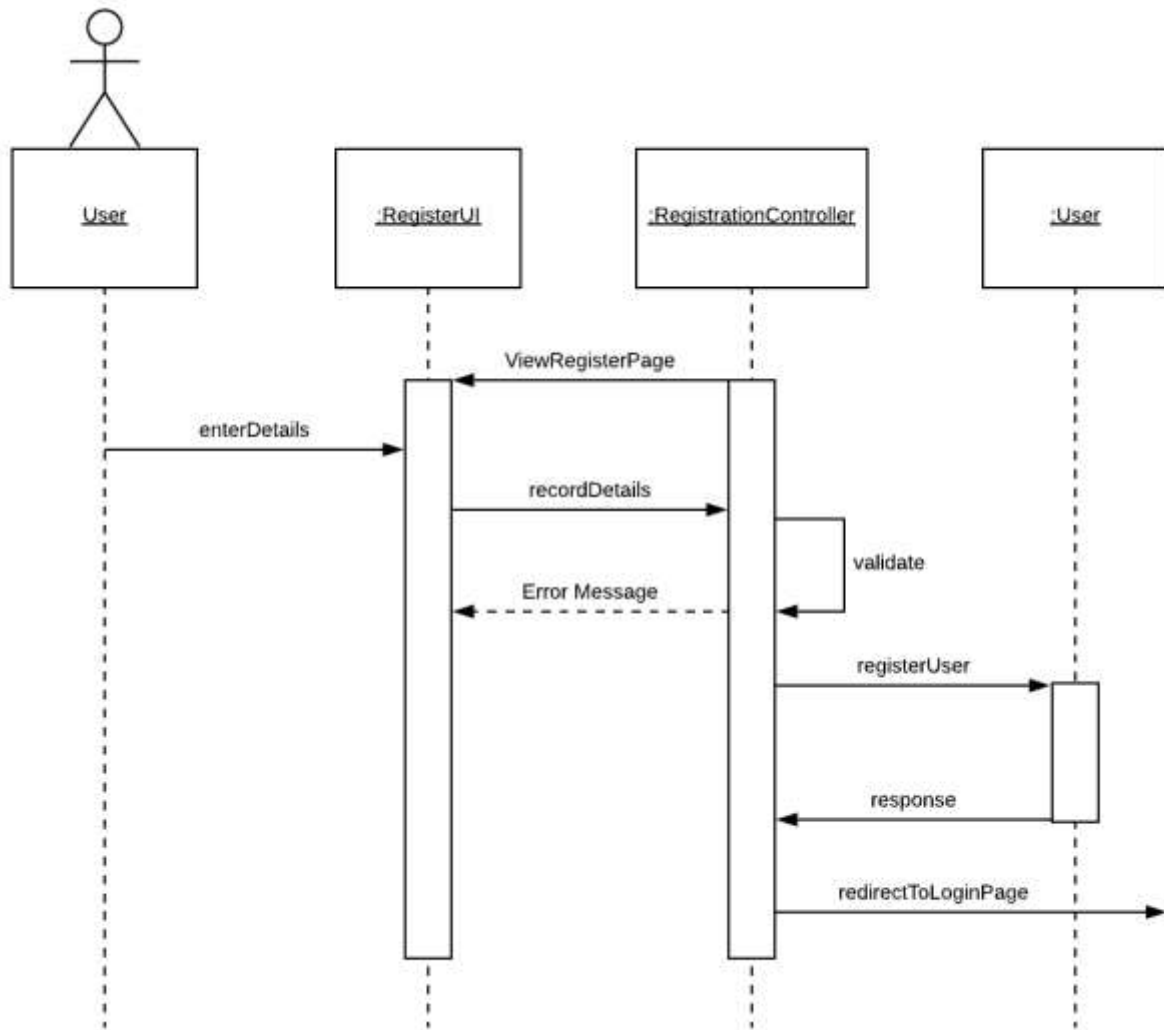


Figure 23: Registration Sequence Diagram

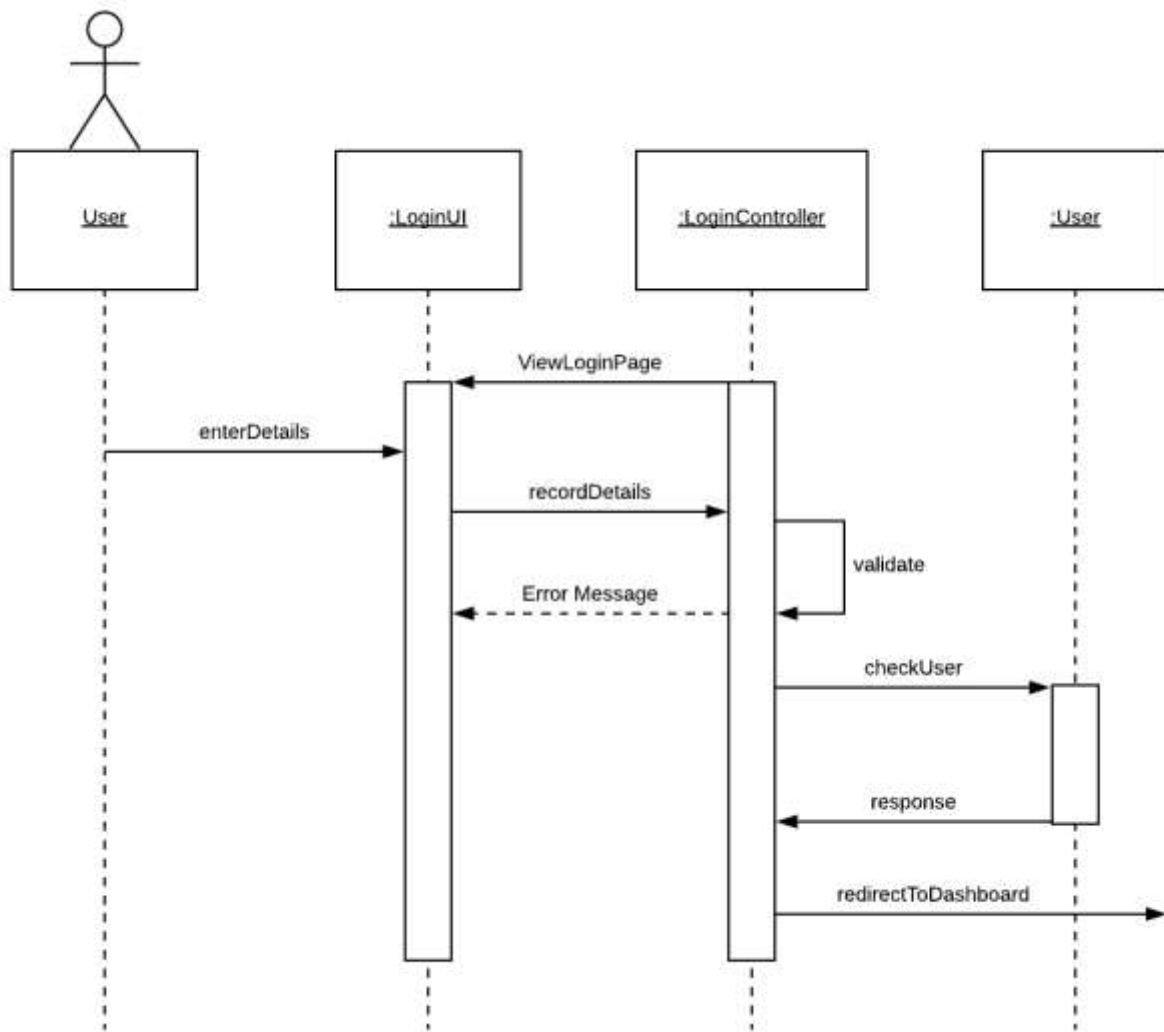


Figure 24: Login Sequence Diagram

3.6.2.1.6 Wireframes

The wireframe shows a web browser window titled "Futsal Booking System". The address bar contains "https://127.0.0.1:8000". The main content area features a "Create an Account" form with the following fields and a button:

- First Name
- Last Name
- Email
- Password
- Password Again
- Register

Figure 25: Authentication Page Register Wireframe

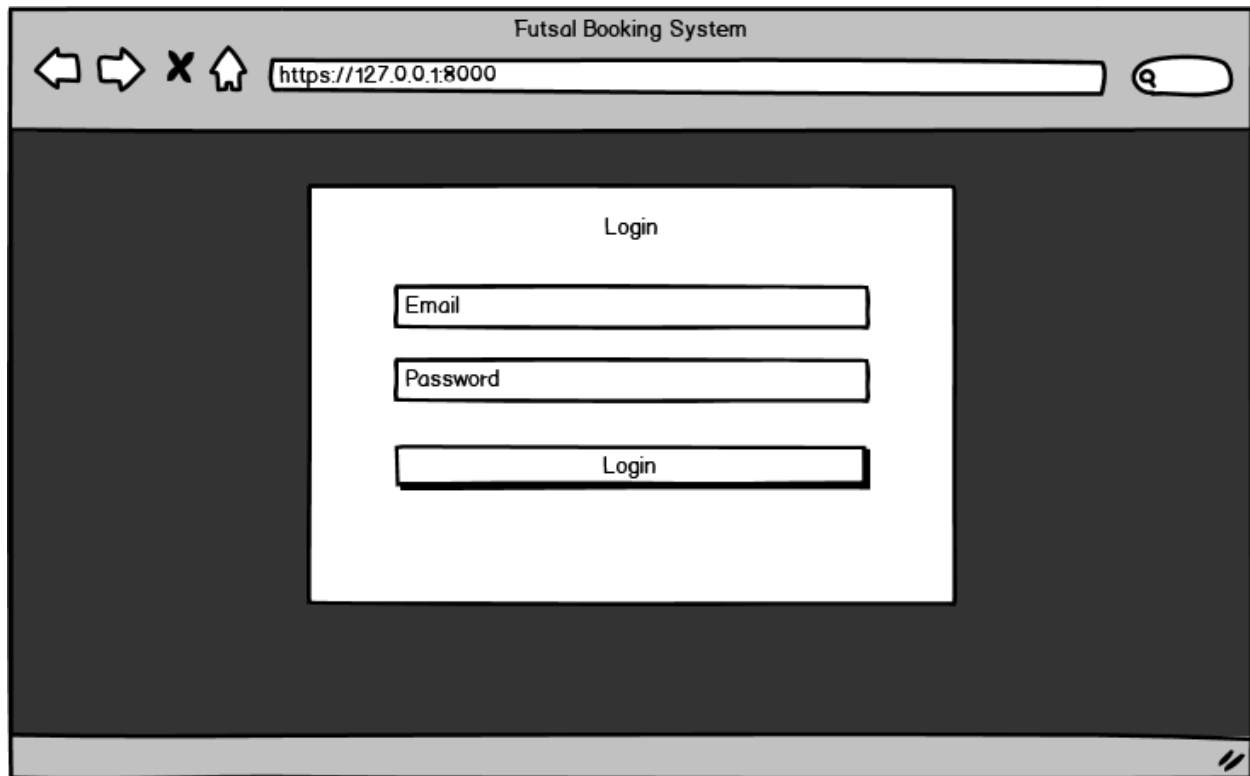


Figure 26: Authentication Page Login Wireframe

3.6.2.2 Development

Firstly, for the front-end UI of the authentication page, templates using Bootstrap and CSS were downloaded to test which suited the look of the website better.

The front-end of the page was kept in the views section and the back-end controller of the page was developed. For the authentication page, a custom Login and Registration controller was developed using Sentinel as it has all the necessary functions including role, which is an essential part of the system that is being developed.

3.6.2.3 Implementation

3.6.2.3.1 UI

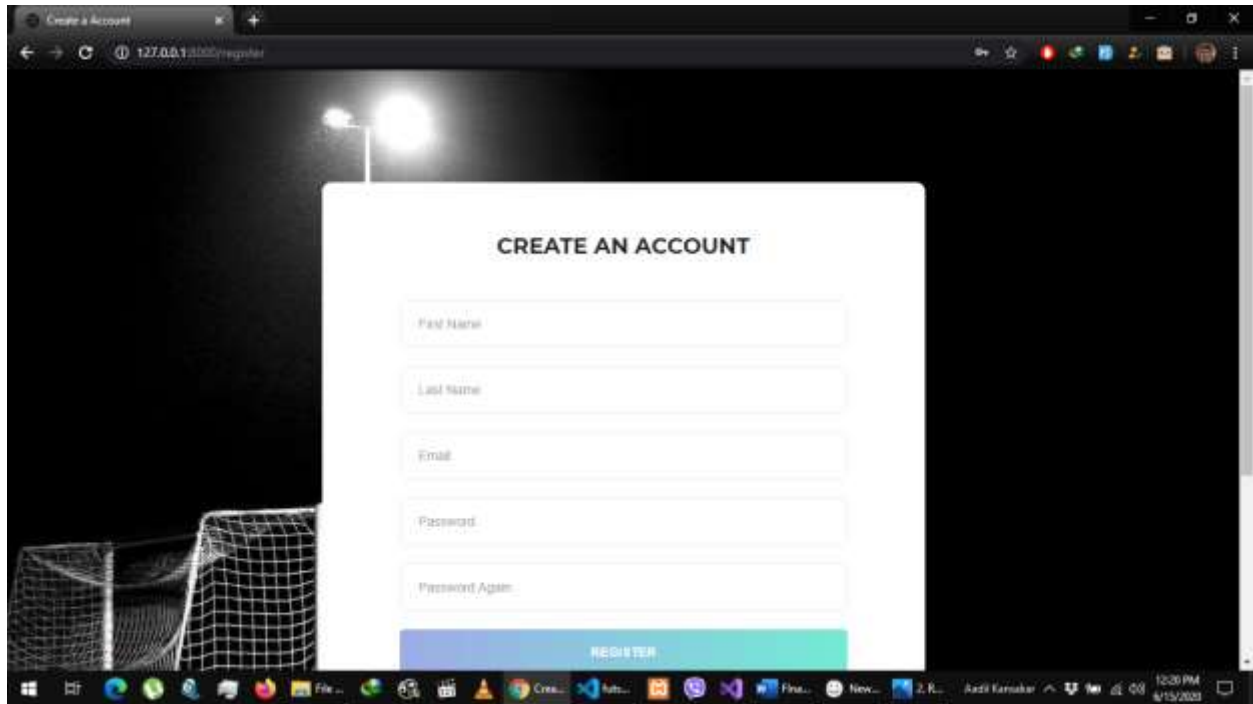


Figure 27: Authentication Page Register UI

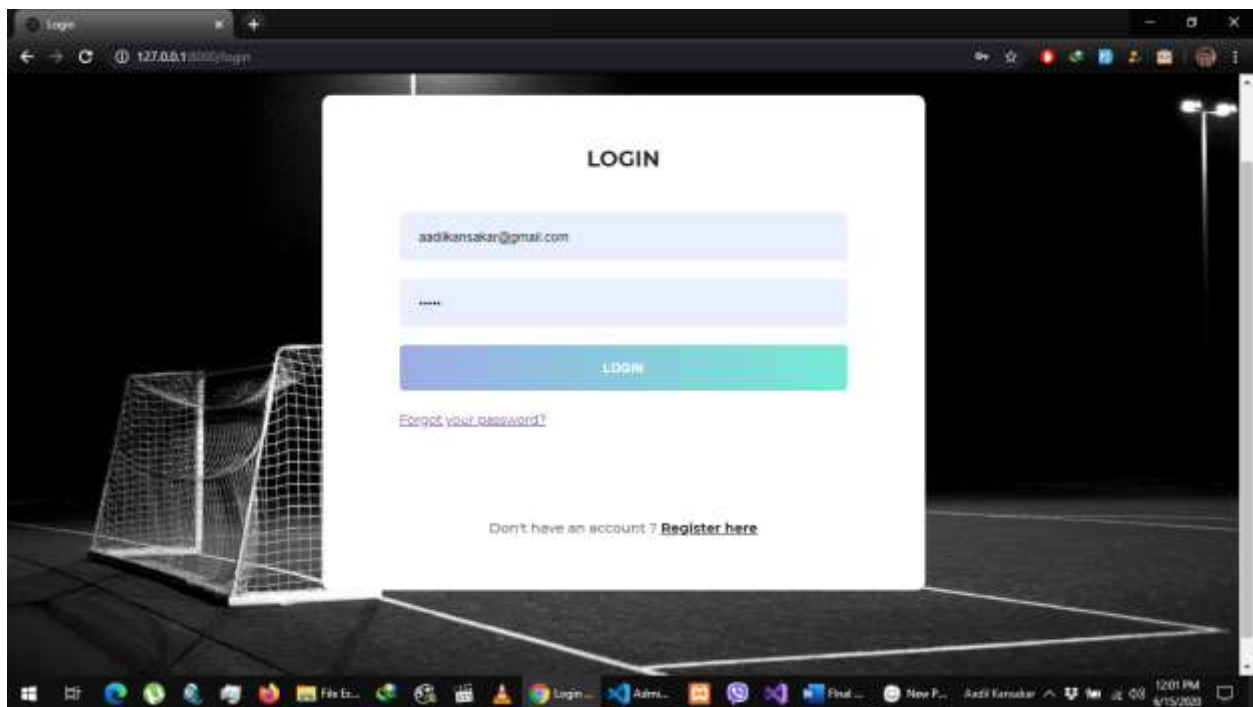


Figure 28: Authentication Page Login UI

3.6.2.3.2 Back-end

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Cartalyst\Sentinel\Laravel\Facades\Sentinel;

class RegistrationController extends Controller
{
    public function register()
    {
        return view('Auth.register');
    }

    public function postregister(Request $request)
    {
        $user=Sentinel::register($request->all());
        $role=Sentinel::findRoleBySlug('client');
        $role->users()->attach($user);
        return redirect('/login');
    }

    public function validation($request)
    {
        $request->validate([
            'password' => 'required|confirmed|max:255',
            'email' => 'required|email|unique:users|max:255',
            'first_name' => 'required|max:255',
        ]);
    }
}
```

Figure 29: Authentication RegistrationController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Cartalyst\Sentinel\Laravel\Facades\Sentinel;

class LoginController extends Controller
{
    public function login()
    {
        return view('Auth.login');
    }

    public function postlogin(Request $request)
    {
        $this->validation($request);
        try{
            $rememberMe=false;
            if(isset($request->remember_me))
                $rememberMe=true;

            if (Sentinel::authenticate($request->all(),$rememberMe))
            {
                $user = Sentinel::authenticate($request->all(),$rememberMe);
                $slug=Sentinel::getUser()->roles()->first()->slug;
                if($slug=='admin')
                    return redirect('/admin_dashboard');
            }
        }
    }
}
```

Figure 30: Authentication LoginController

When the user registers a new account, they get registered as a client and has all the functionalities in the client dashboard.

3.6.3 Home Page

3.6.3.1 Design

3.6.3.1.1 Wireframe

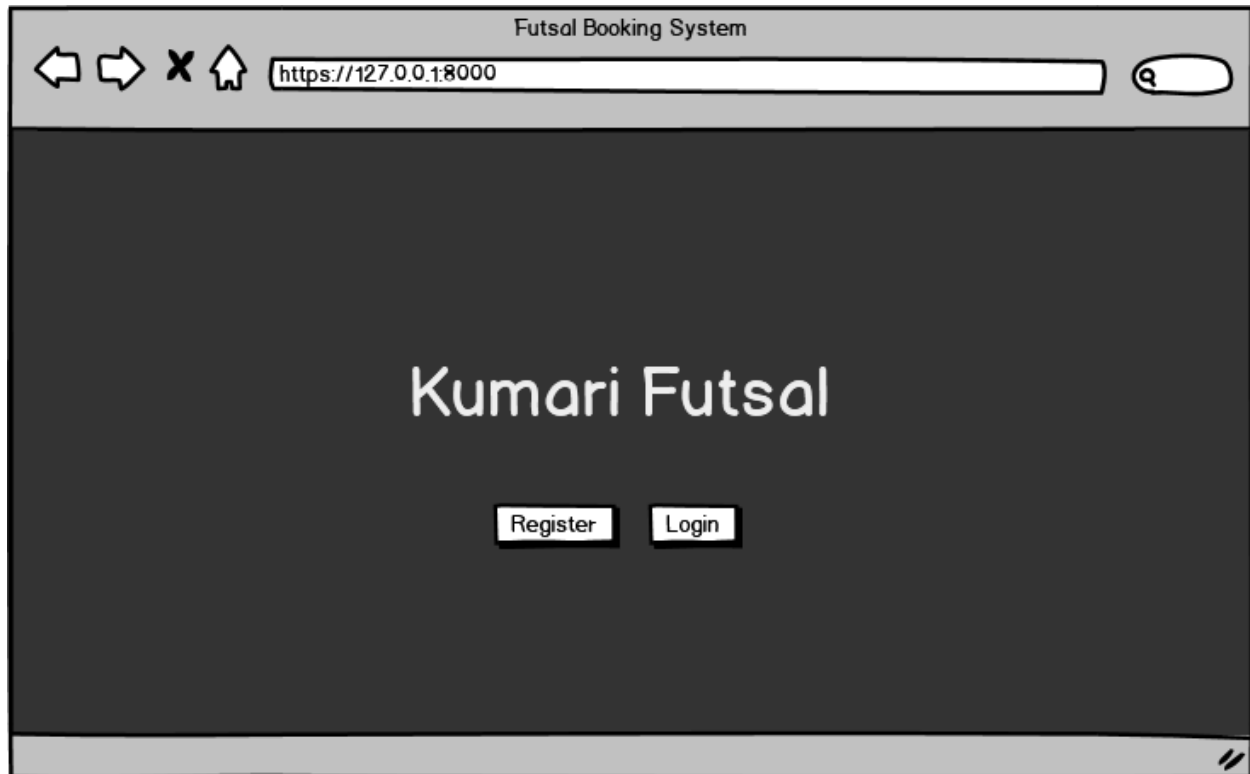


Figure 31: Home Page Wireframe

3.6.3.2 Development

For the home page, a single page template was used featuring a minimalistic setup with a login and a registration button. These buttons when clicked are routed to their respective pages for their use.

3.6.3.3 Implementation



Figure 32: Home Page UI

3.6.4 Booking Page

3.6.4.1 Design

3.6.4.1.1 Use-Case Diagram

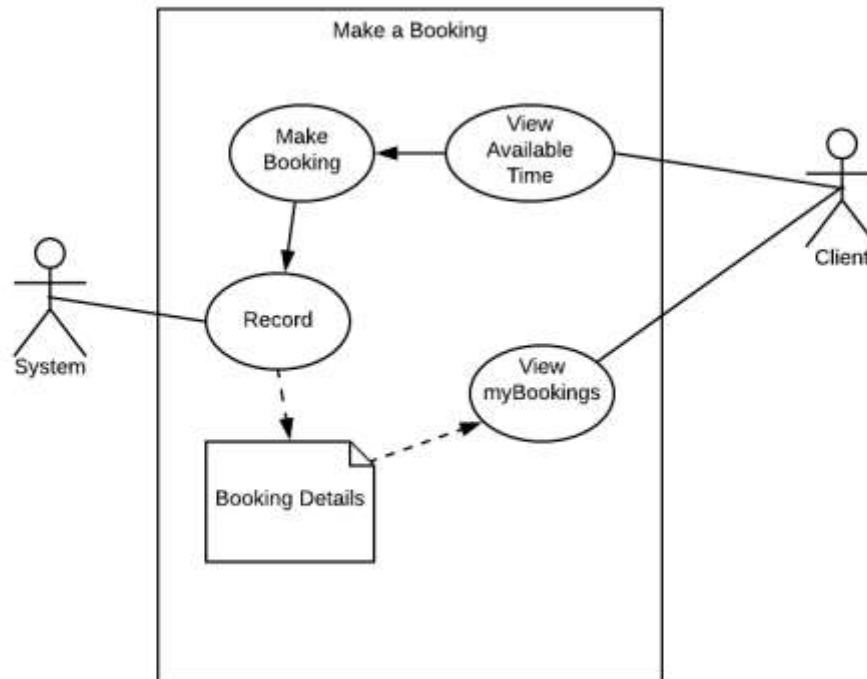


Figure 33: Booking Use Case Diagram

3.6.4.1.2 High Level Description

- View Available Time

Actor: Client

Description: Available times for booking time during a specific date are displayed.

- Make a booking

Actor: Client

Description: Clients can book the futsal ground for the time and date.

- Record

Actor: System

Description: The system stores the booking details in the database.

- View myBooking

Actor: Client

Description: Clients can view the date and time for which they have booked.

- View Booking Details

Actor: Admin

Description: Admins can view all the bookings made using the system.

3.6.4.1.3 Extended Use-Case Diagram

Users	System
1. Login as client	
	2. Authenticate user
3. Select Date	
	4. Display available time
5. Make a booking	
	6. Record Booking
	7. Redirect to My Bookings Page

Table 6: Booking Extended Use Case Diagram

3.6.4.1.4 Collaboration Diagram

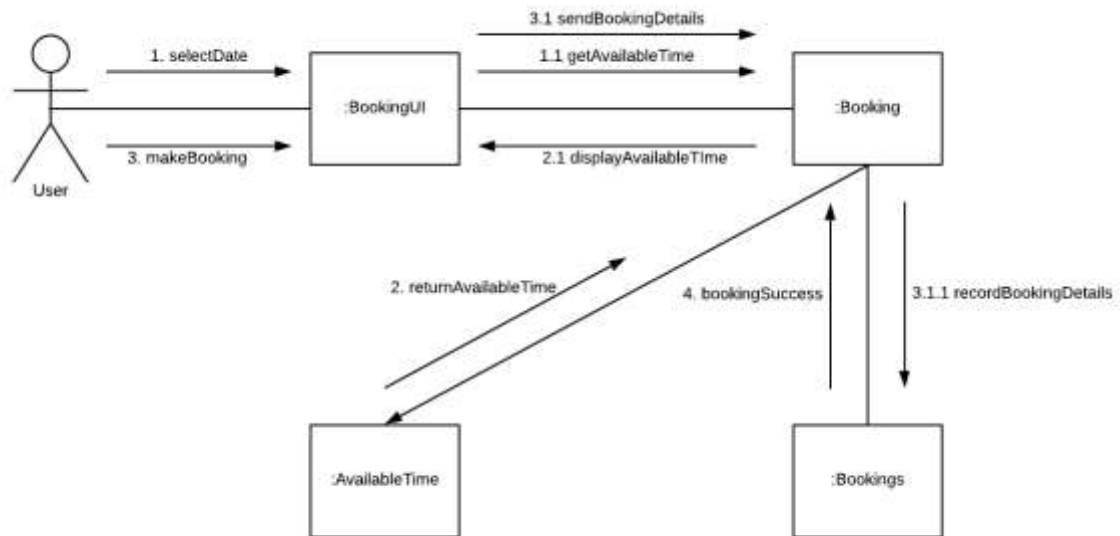


Figure 34: Booking Collaboration Diagram:

3.6.4.1.5 Sequence Diagram

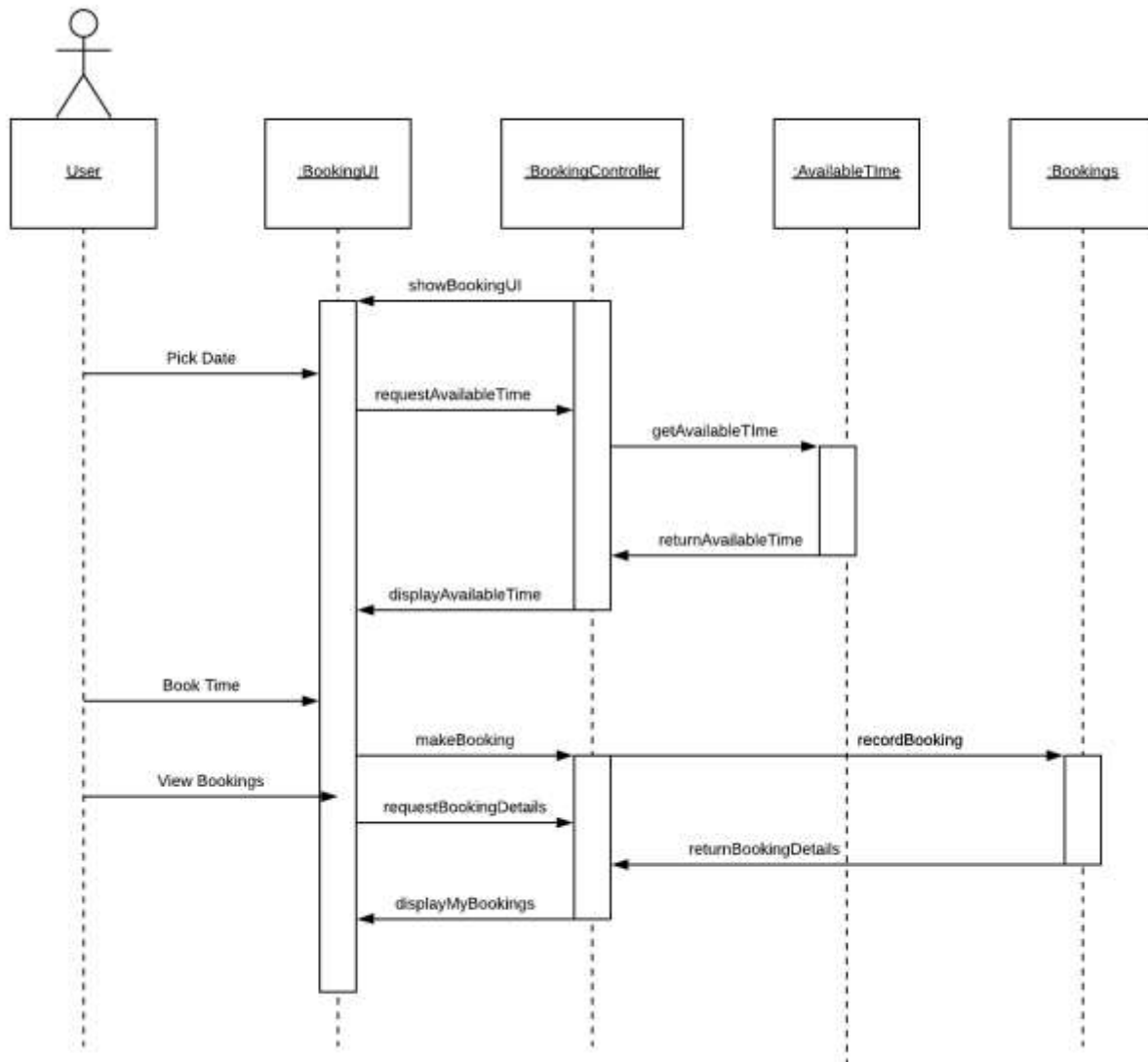


Figure 35: Booking Sequence Diagram

3.6.4.1.6 Wireframes

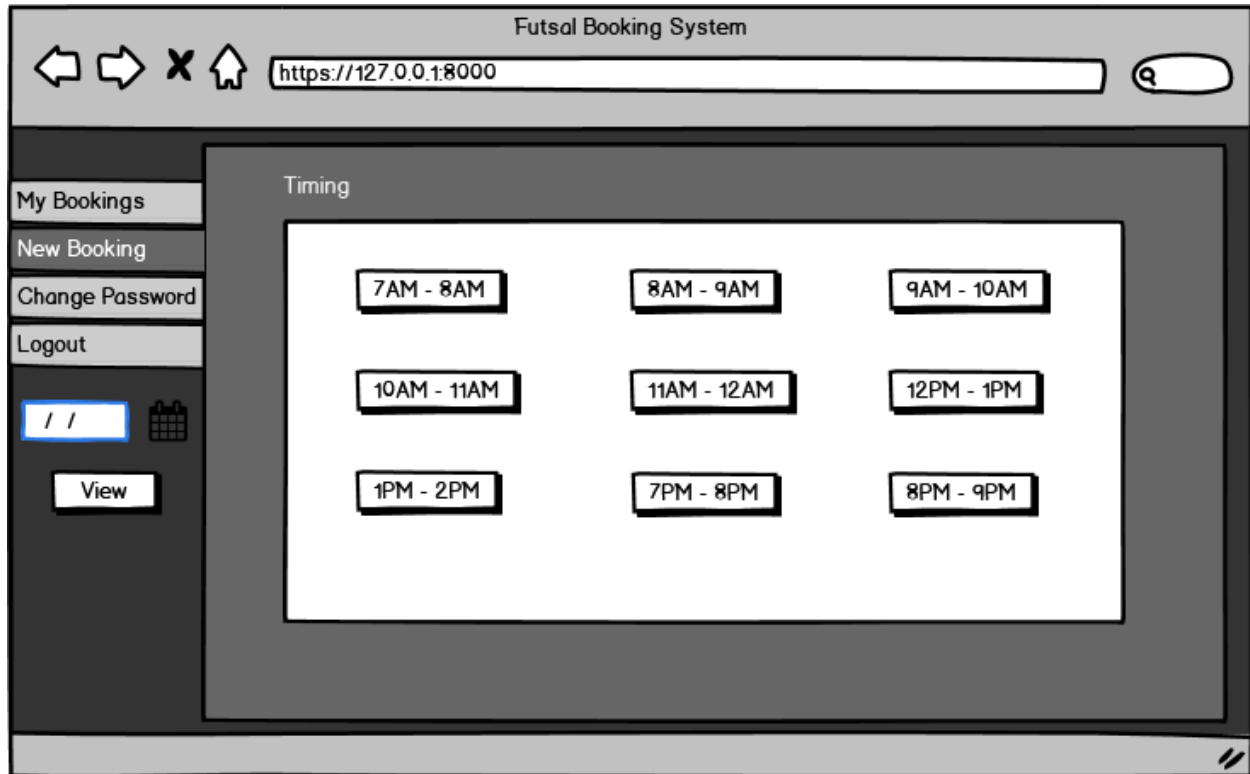


Figure 36: New Booking Wireframe

The wireframe illustrates a web browser window titled "Futsal Booking System". The address bar shows "https://127.0.0.1:8000". On the left, a sidebar contains navigation links: "My Bookings", "New Booking", "Change Password", and "Logout". Below these links is a date input field with slashes and a calendar icon, and a "View" button. The main content area, titled "Booking", features a form with four input fields labeled "Date", "Time", "Phone", and "Price", followed by a "Book" button.

Figure 37: Booking Time Wireframe

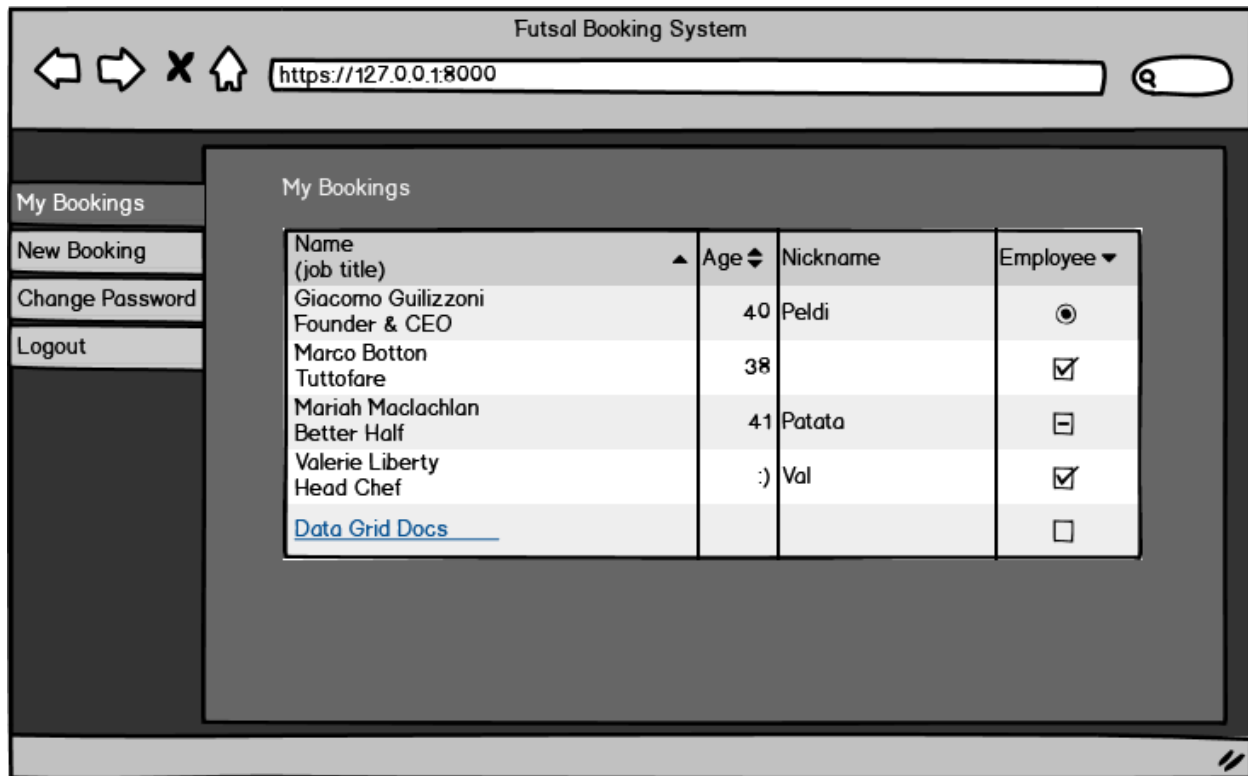


Figure 38: My Bookings Wireframe

3.6.4.2 Development

The client can book a date and time for them to use a futsal ground. When the client opens the booking page, they are prompted to select a date. When a date is selected, available times for booking for the date is shown in blue and booked times are shown in red. The selected time for booking then shows all the details for booking like date, time and price of the particular time. The user is asked to enter their phone number and the time is booked. The booked time can then be viewed in the My Bookings tab in the client dashboard. The client can cancel the booking as well using the delete button in this page.

3.6.4.3 Implementation

3.6.4.3.1 UI

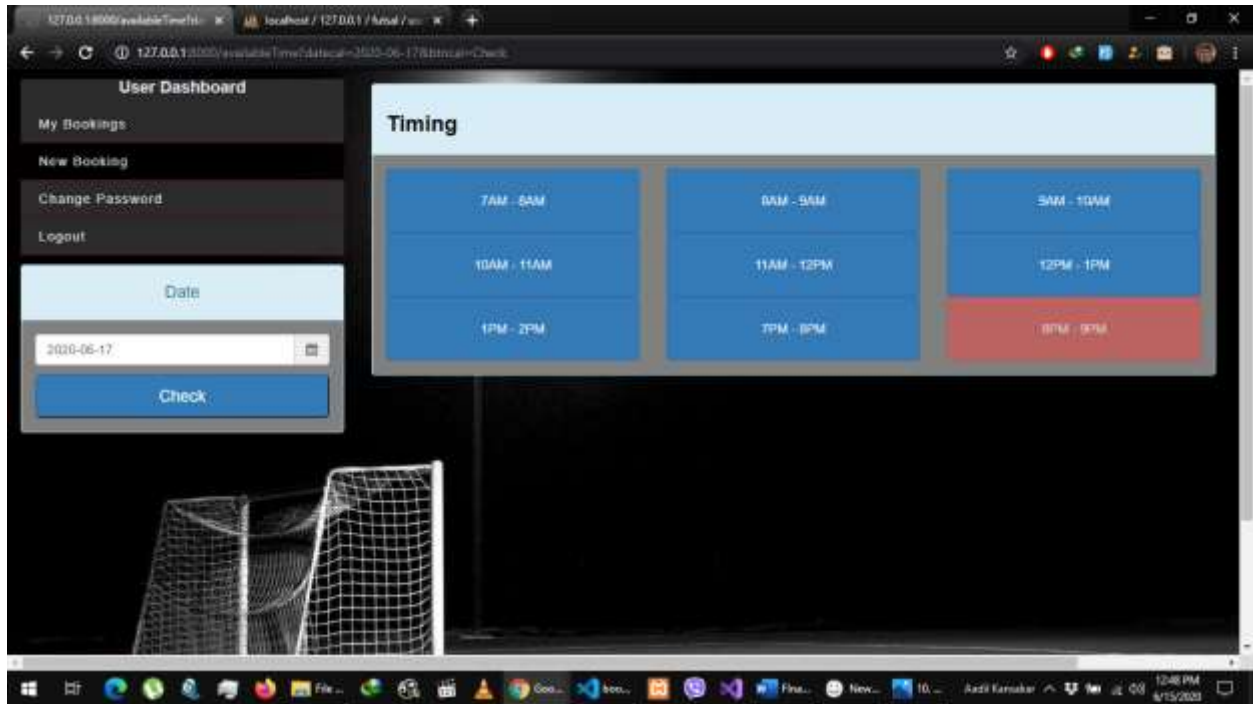


Figure 39: New Booking Page UI

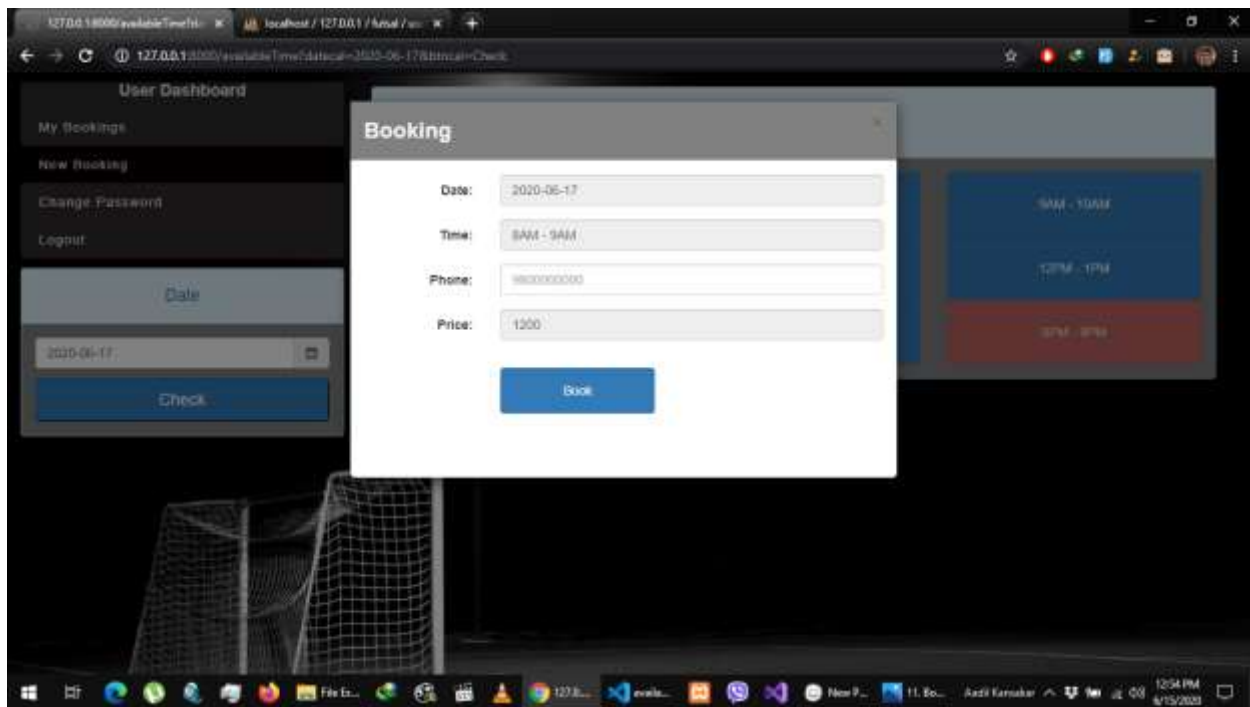


Figure 40: Booking Time UI

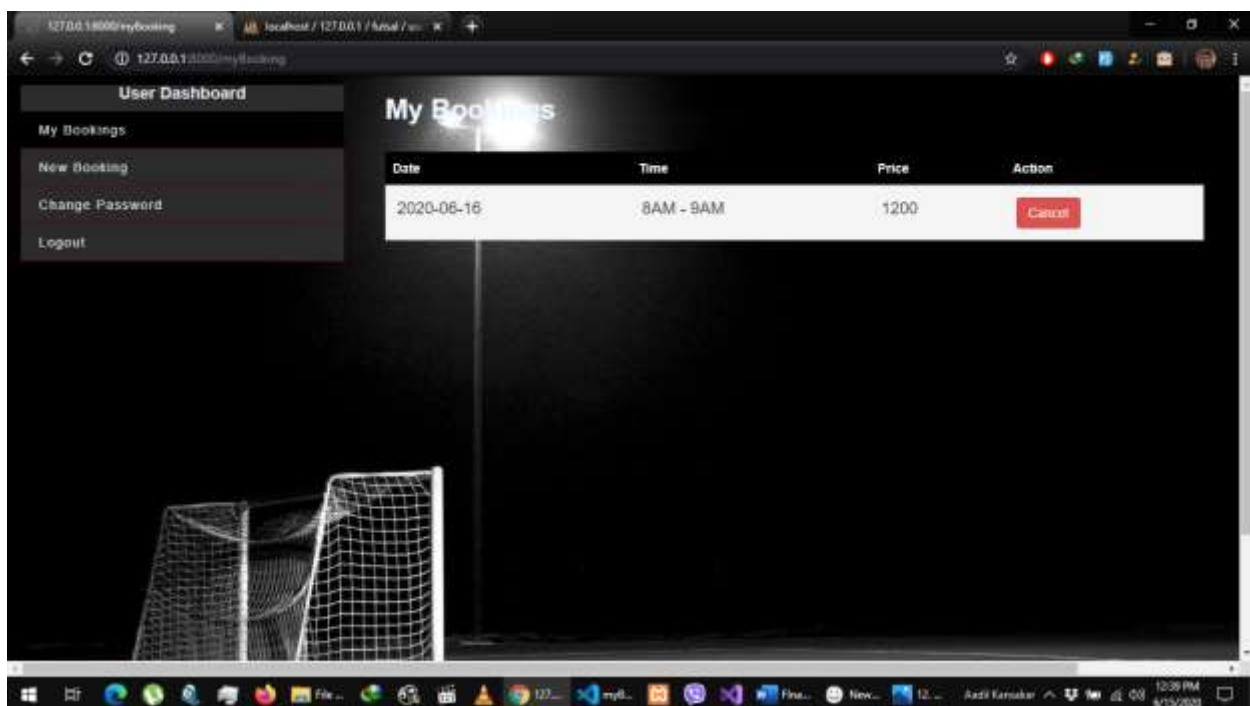


Figure 41: My Bookings UI

3.6.4.3.2 Back-end

```
class ClientController extends Controller
{
    public function viewClientDashboard()
    {
        return view('client.clientDashboard');
    }

    public function bookNow()
    {
        return view('client.bookNow');
    }

    public function postBookNow(Request $request)
    {
        $mail = $request->get('hiddenMail');
        $mail;
        $booking=new Bookings;
        $booking->email=$mail;
        $booking->date=$request->popupDate;
        $booking->time=$request->time;
        $booking->phone=$request->phone;
        $booking->price=$request->price;
        $booking->save();
        return redirect('/myBooking');
    }
}
```

Figure 42: Make a booking back-end

```
public function myBooking()
{
    $books=Bookings::where('email',Sentinel::getUser()->email)->get();
    return view('client.myBooking',compact('books') );
}

public function viewTime(Request $request)
{
    $bookedTime='';
    $availableTime=availableTime::all();
    $booking=Bookings::where('date',$_REQUEST["datecal"])->get();
    return view('client.availableTime',compact('booking','availableTime'));
}

public function deleteMyBooking($id)
{
    $deletebook=Bookings::find($id);
    $deletebook->delete();
    return redirect('/myBooking');
}
```

Figure 43: My Bookings back-end

3.6.5 Admin Dashboard

3.6.5.1 Design

3.6.5.1.1 Use Case Diagram

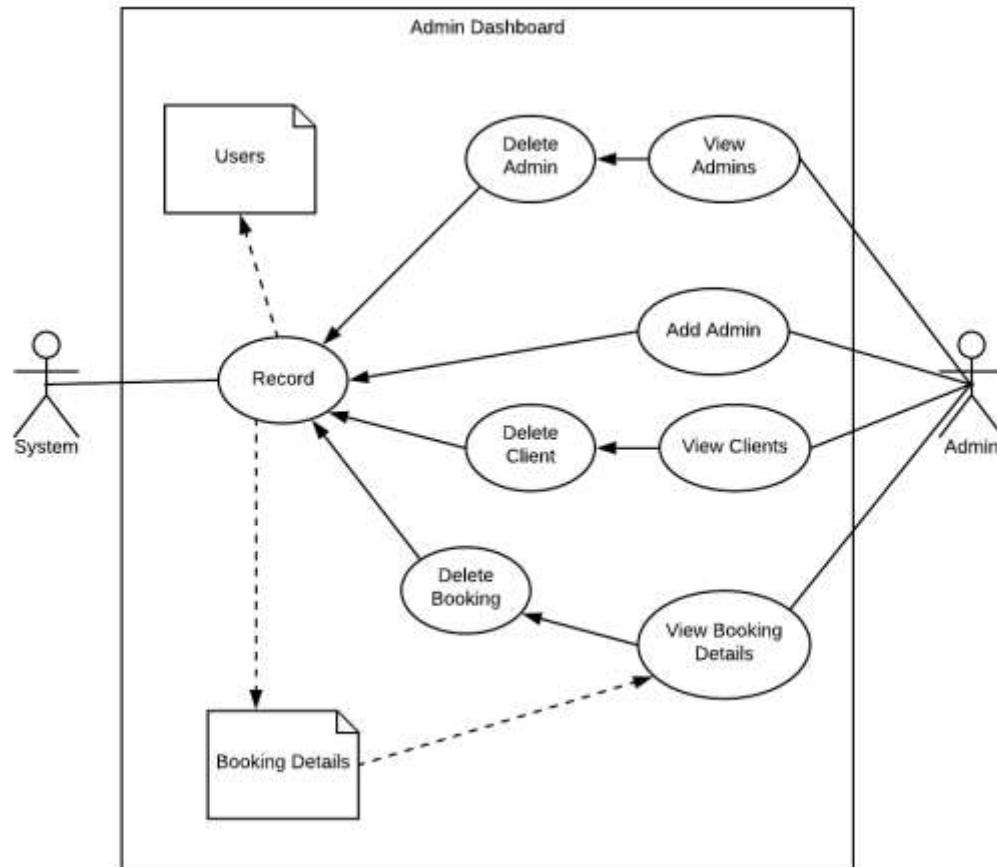


Figure 44: Admin Dashboard Use Case Diagram

3.6.5.1.2 High Level Use Case Diagram

- View Admins

Actor: Admin

Description: Displays all the admins registered in the system.

- Delete Admin

Actor: Admin

Description: Delete an admin from the admin list.

- Add Admin

Actor: Admin

Description: Register a new admin in the system.

- View Clients

Actor: Admin

Description: Displays all the clients registered in the system.

- Delete Client

Actor: Admin

Description: Delete a client from the client list.

- View Booking Details

Actor: Admin

Description: Displays all the bookings made by clients on a date and time

- Delete Booking

Actor: Admin

Description: Delete a booking from the booking list.

- Record

Actor: System

Description: The system carries out the given task and records the changes in the database.

3.6.5.1.3 Extended Use Case Diagram

3.6.5.1.3.1 View admin

User	System
1. Login as admin	
2. Go to View Admin page	3. Displays Admin Page

Table 7: Admin Dashboard Extended Use Case View Admin

3.6.5.1.3.2 Add admin

User	System
1. Login as admin	
2. Go to Add Admin page	3. Displays Add Admin Page
4. Add Admin Details	5. Admin is added
	6. Display Admin Page

Table 8: Admin Dashboard Extended Use Case Add Admin

3.6.5.1.3.3 Delete admin

User	System
1. Login as admin	
2. Go to View Admin page	3. Displays Admin Page
4. Delete Admin	5. Admin is deleted
	6. Display Admin Page

Table 9: Admin Dashboard Extended Use Case Delete Admin

3.6.5.1.4 Collaboration Diagram

3.6.5.1.4.1 View admin

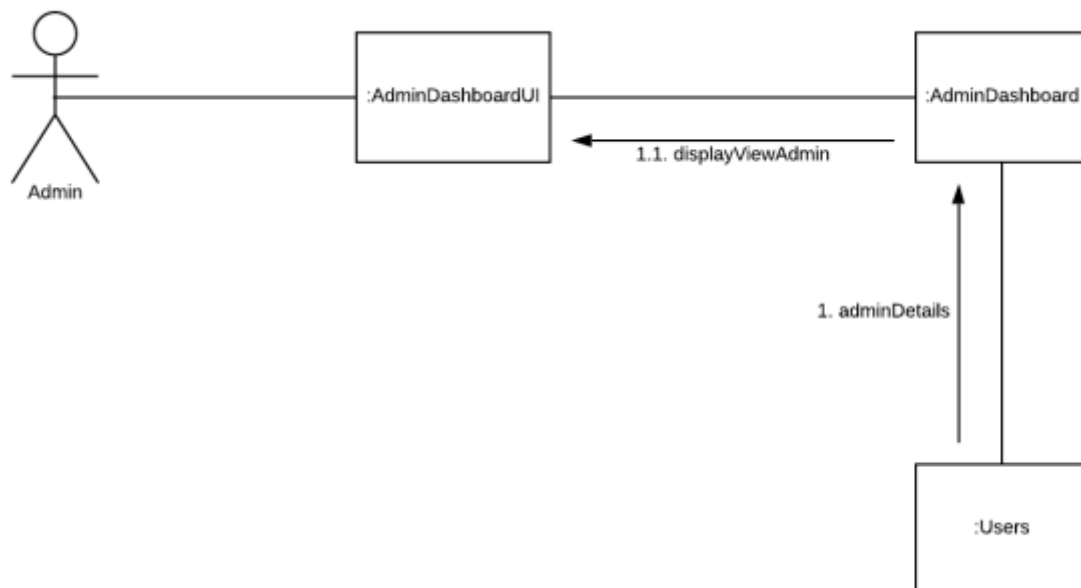


Figure 45: Admin Dashboard View Admin Collaboration Diagram

3.6.5.1.4.2 Add admin

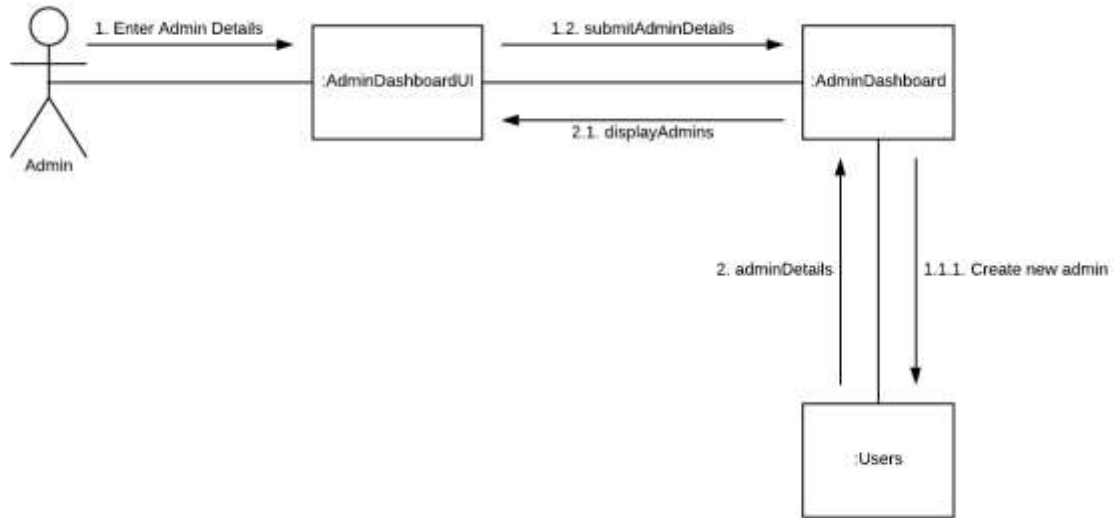


Figure 46: Admin Dashboard Add Admin Collaboration Diagram

3.6.5.1.4.3 Delete admin

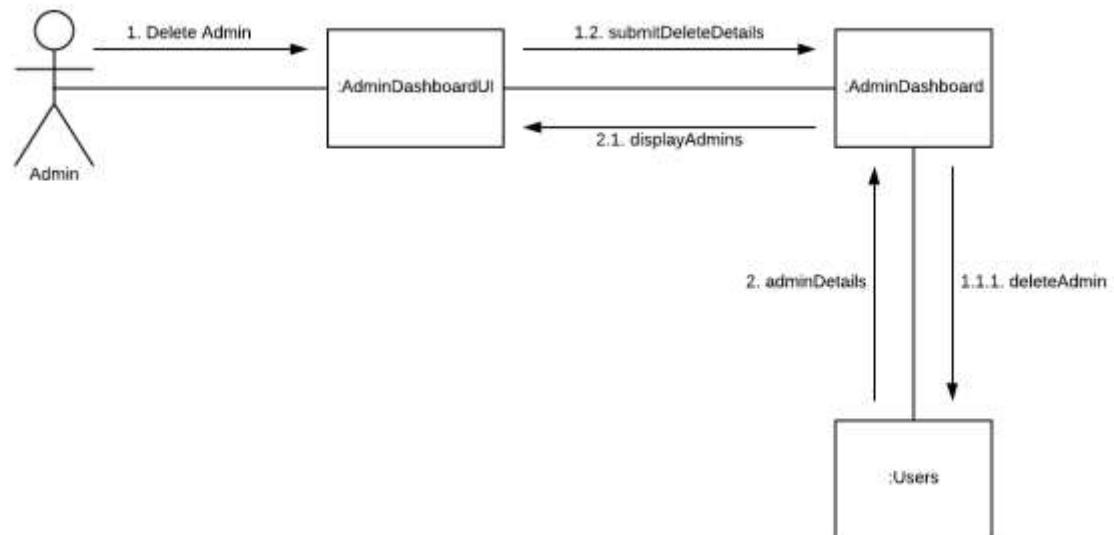


Figure 47: Admin Dashboard Delete Admin Collaboration Diagram

3.6.5.1.5 Sequence Diagram

3.6.5.1.5.1 View admin

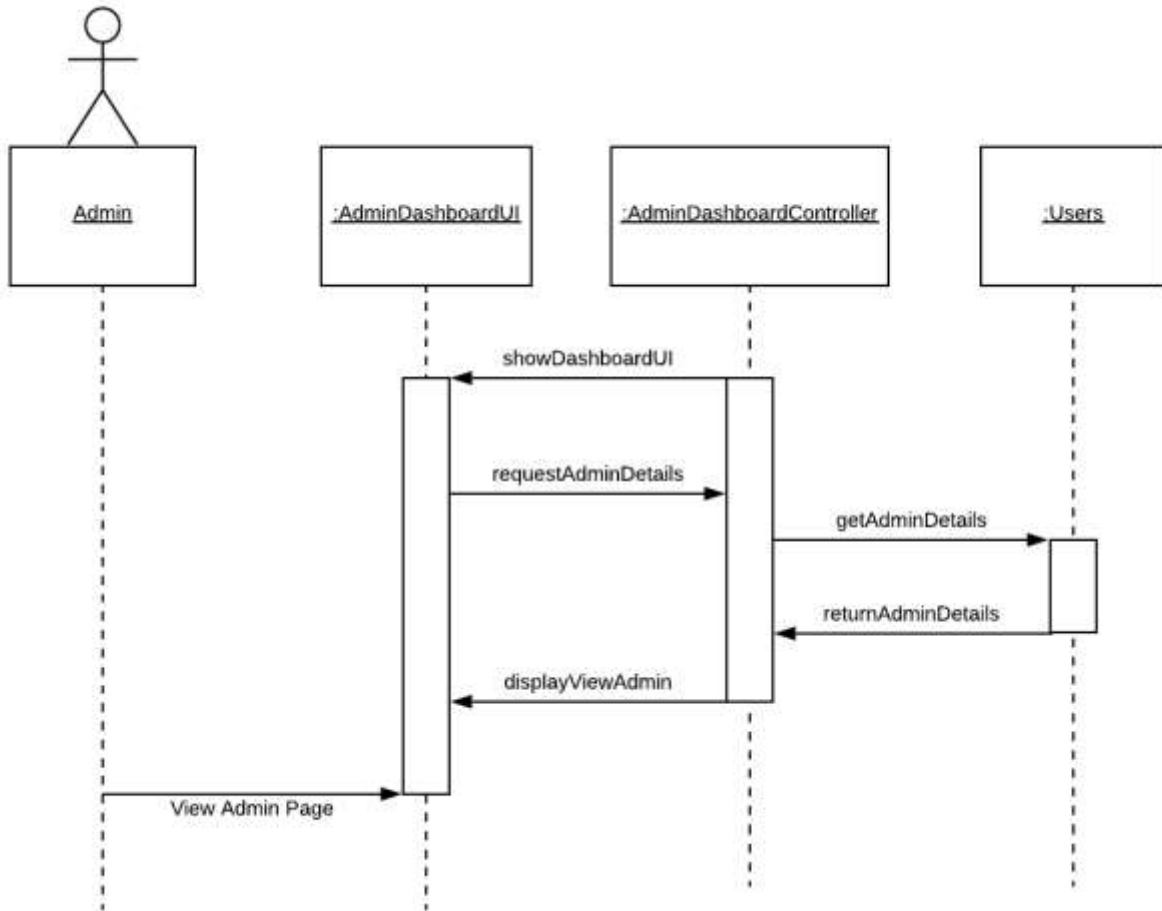


Figure 48: Admin Dashboard View Admin Sequence Diagram

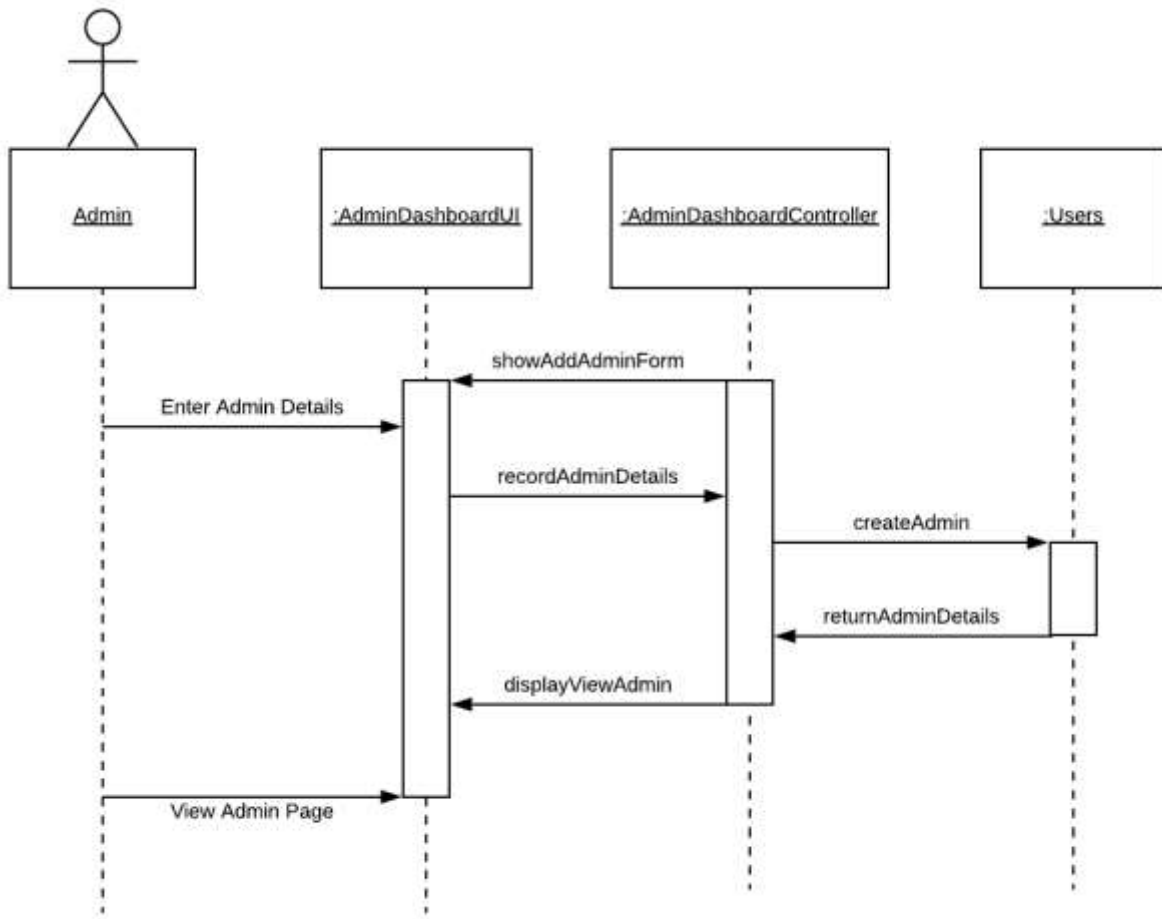
3.6.5.1.5.2 Add admin

Figure 49: Admin Dashboard Add Admin Sequence Diagram

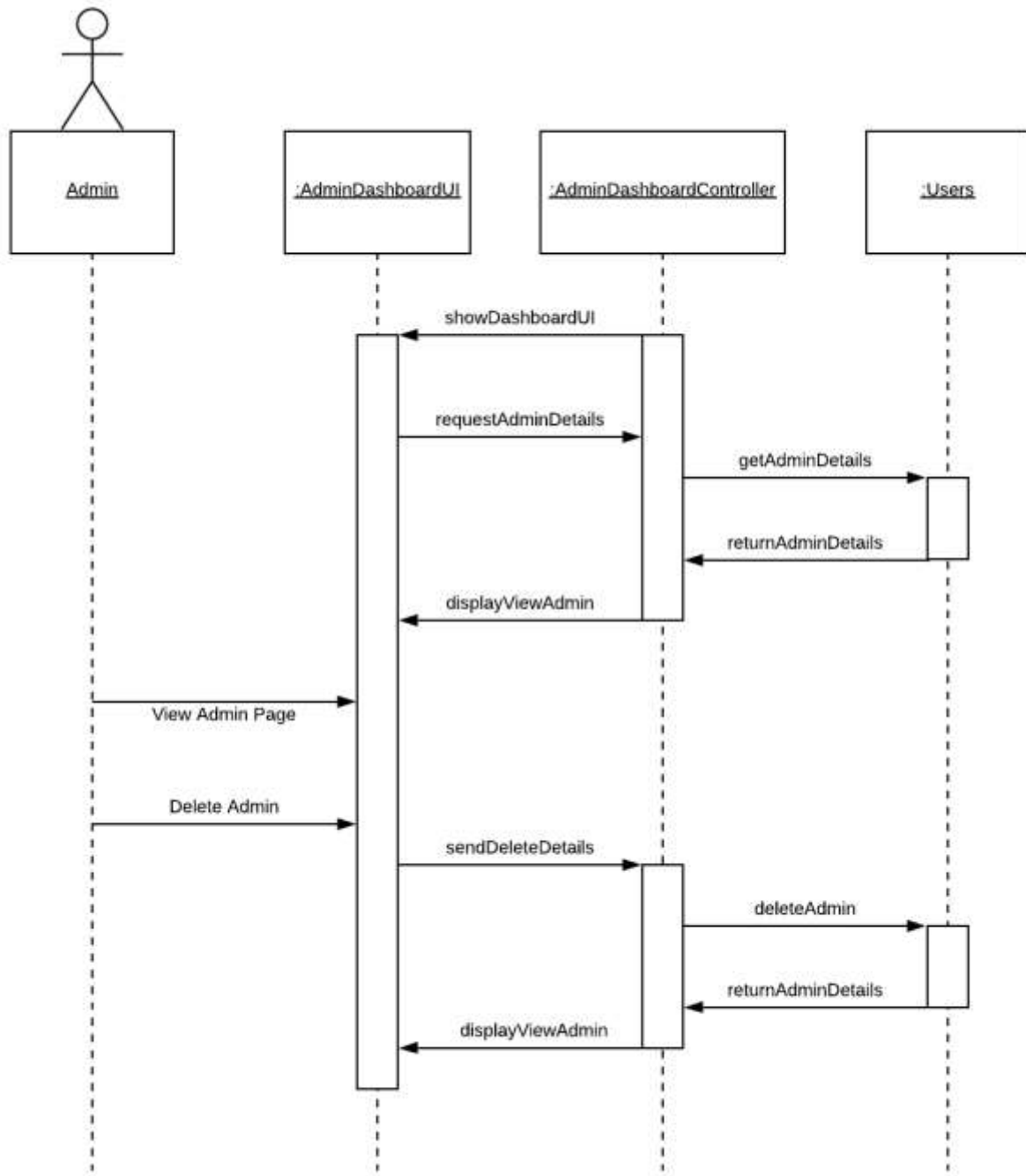
3.6.5.1.5.3 Delete admin

Figure 50: Admin Dashboard Delete Admin Sequence Diagram

3.6.5.1.6 Wireframes

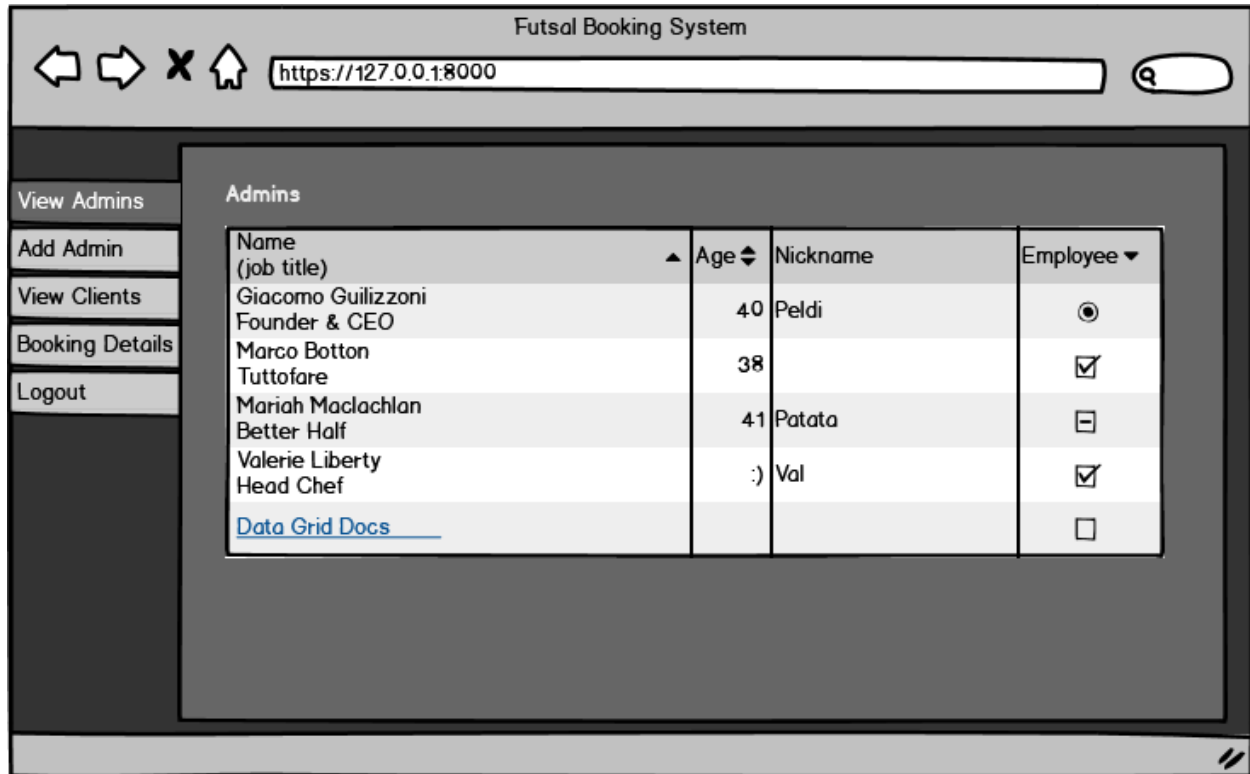


Figure 51: Dashboard View Admin Wireframe

The image shows a web browser window titled "Futsal Booking System". The address bar displays "https://127.0.0.1:8000". On the left side, there is a vertical navigation menu with the following items: "View Admins", "Add Admin" (highlighted), "View Clients", "Booking Details", and "Logout". The main content area features a central white box titled "Add Admin". Inside this box, there are five text input fields labeled "First Name", "Last Name", "Email", "Password", and "Password Again", followed by a button labeled "Add".

Figure 52: Dashboard Add Admin Wireframe

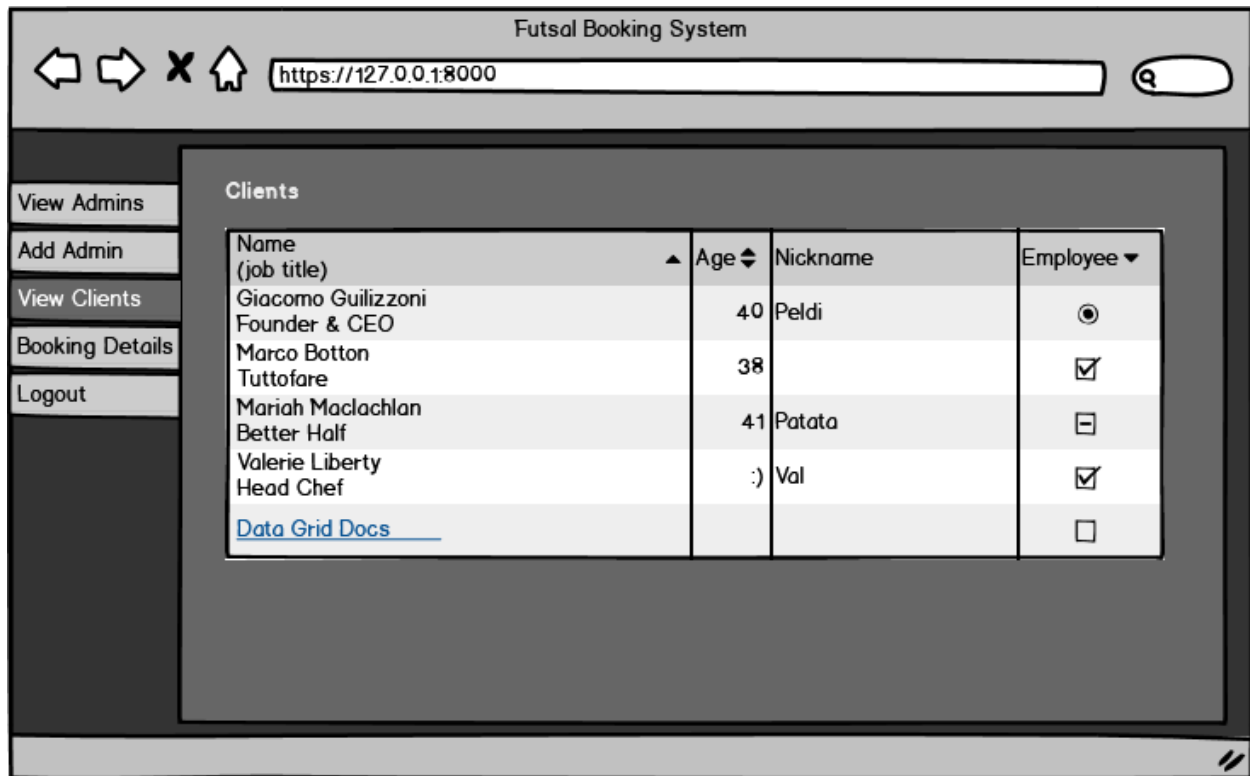


Figure 53: Dashboard View Clients Wireframe

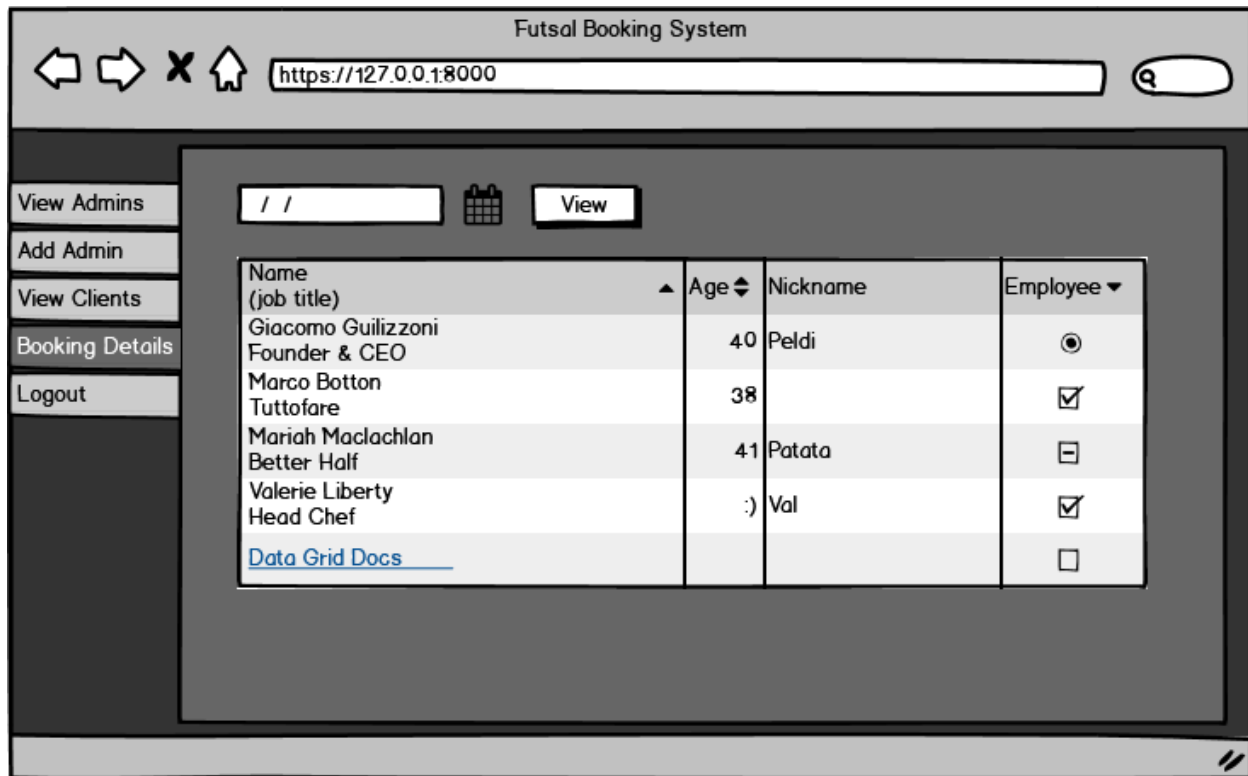


Figure 54: Dashboard Booking Details Wireframe

3.6.5.2 Development

The admin dashboard can be used to view all the client and admins registered in the system. The admin also has the authority to delete any particular client and admin. They can also add admins to be registered so they can have the same permissions as them. Additionally, they can view all the bookings made using this system as well as delete any booking that has been made by the client.

3.6.5.3 Implementation

3.6.5.3.1 UI

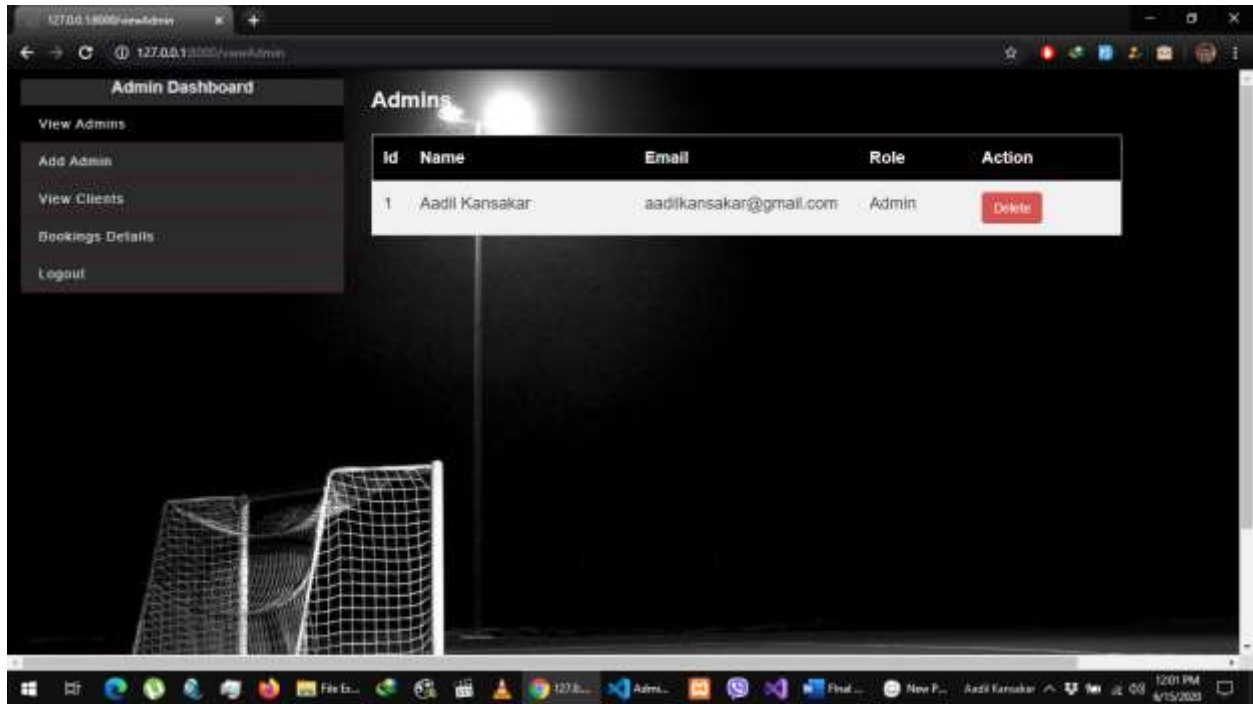


Figure 55: Dashboard View Admin UI

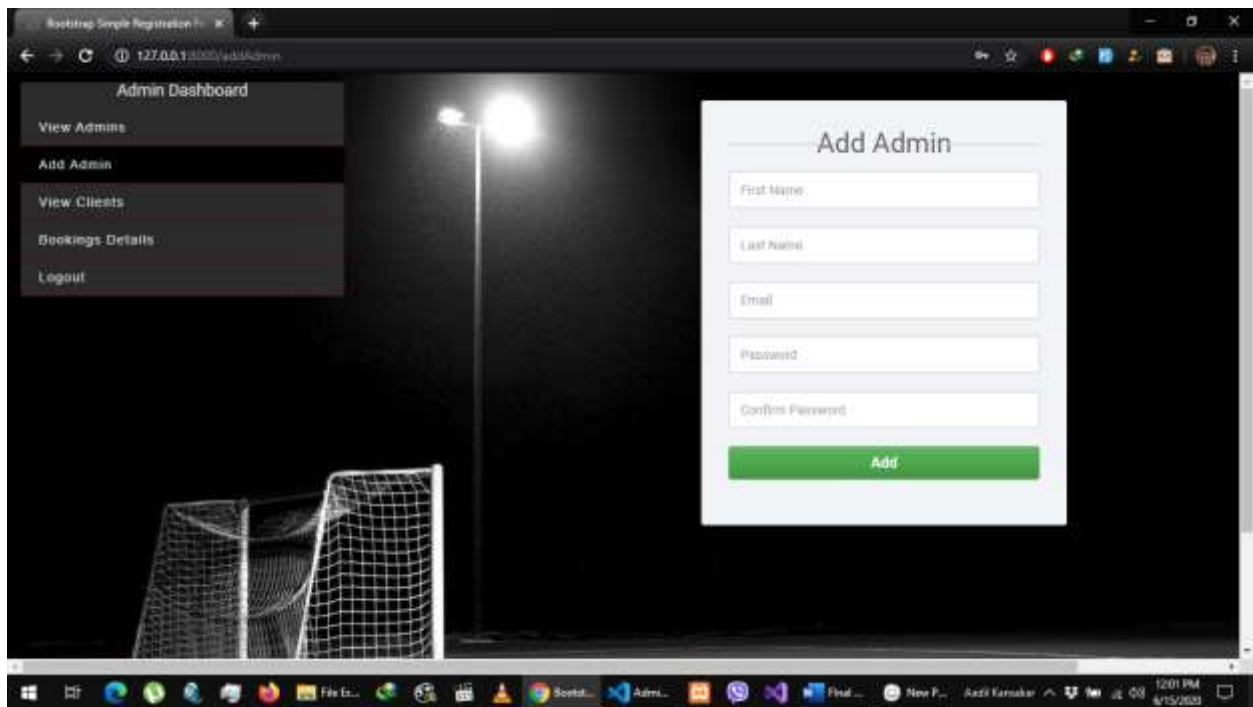


Figure 56: Dashboard Add Admin UI

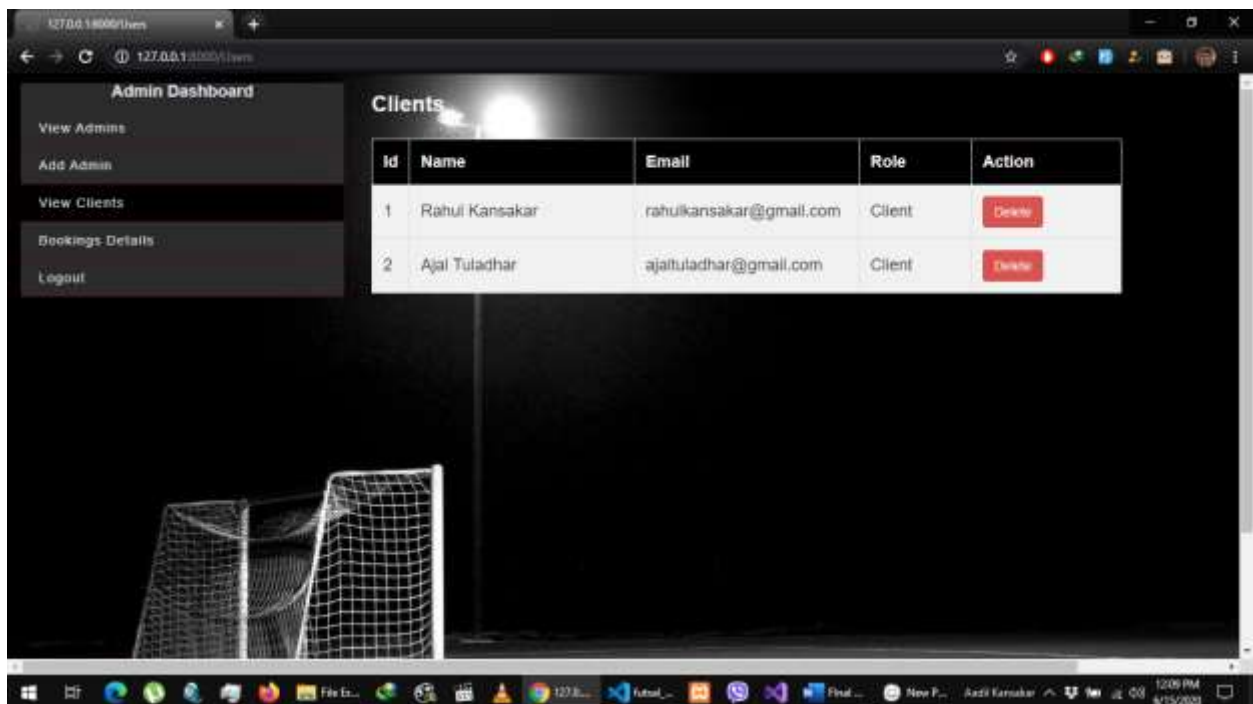


Figure 57: Dashboard View Clients UI

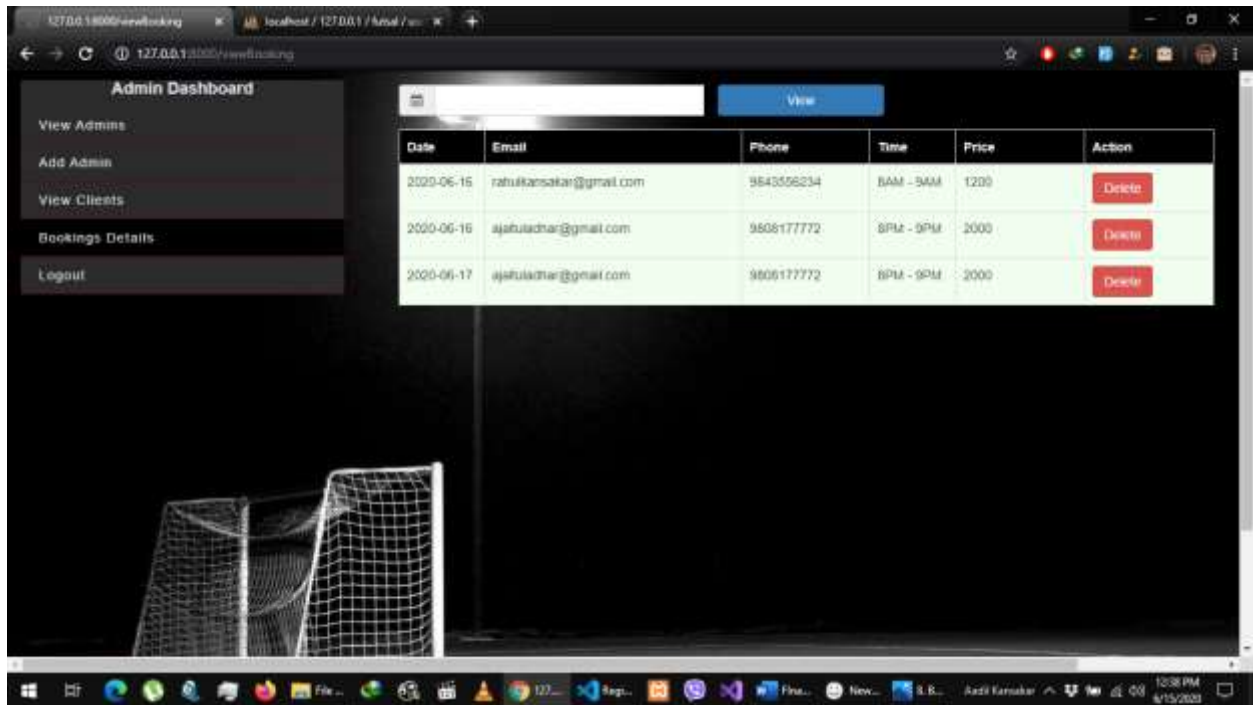


Figure 58: Dashboard Booking Details UI

3.6.5.3.2 Back-end

```
public function viewAdmin()  
{  
    $role = Sentinel::findRoleBySlug('admin');  
    $id=Sentinel::getUser()->id;  
    $users=$role->users()->with('roles')->get();  
    if($users->isEmpty())  
    {  
        $roles = Sentinel::findById($id)->roles;  
        $roletype= $roles[0]->name;  
        return view('admin.viewAdmins',compact('users','roletype'));  
    }  
    else  
    {  
        return view('admin.viewAdmins',compact('users'));  
    }  
}
```

Figure 59: Dashboard View Admin Back-end

```
class AddAdminController extends Controller
{
    public function addAdmins()
    {
        return view('admin.addAdmin');
    }

    public function postAddAdmin(Request $request)
    {
        $this->validation($request);
        $user=Sentinel::registerAndActivate($request->all());
        $role=Sentinel::findRoleBySlug('admin');
        $role->users()->attach($user);
        return redirect('/admin_dashboard');
    }

    public function validation($request)
    {
        $request->validate([
            'password' => 'required|confirmed|max:255',
            'email' => 'required|email|unique:users|max:255',
            'first_name' => 'required|max:255',
            'last_name' => 'required|max:255',
        ]);
    }
}
```

Figure 60: Dashboard Add Admin back-end

```
public function viewUsers()
{
    $role = Sentinel::findRoleBySlug('client');
    $id=Sentinel::getUser()->id;
    $users=$role->users()->with('roles')->get();
    if($users->isEmpty())
    {
        $roles = Sentinel::findRoleBySlug('client')->get();
        $roletype= $roles[1]->name;
        return view('admin.viewUsers',compact('users','roletype'));
    }
    else
    {
        return view('admin.viewUsers',compact('users'));
    }
}
```

Figure 61: Dashboard View Clients back-end

```
public function viewBookingAdmin(Request $request)
{
    $bookedTime='';
    $availableTime=availableTime::all();
    $booking=Bookings::where('date',$_REQUEST["datecal"])->get();
    return view('admin.adminBooking',compact('booking','availableTime'));
}

public function postBookNowAdmin(Request $request)
{
    $mail = $request->get('hiddenMail')."( ".$request->get('hiddenrole').")";
    $mail;

    $booking=new Bookings;
    $booking->email=$mail;
    $booking->date=$request->popupDate;
    $booking->time=$request->time;
    $booking->phone=$request->get('hiddenrole');
    $booking->price=$request->price;

    $booking->save();
    return redirect('/viewBooking');
}
```

Figure 62: Dashboard Booking Details back-end

4 Testing and Analysis

4.1 Testing

4.1.1 Database Structure Test

	Action
Test Case	Check if designed database is created in MySQL by using migration.
Expected Outcome	List of tables designed using migration
Actual Outcome	List of tables designed is migrated to the MySQL database.
Result	Test Successful

Table 10: Database Structure Test

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> activations	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> available_times	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> bookings	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> persistences	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> reminders	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> roles	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> role_users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> throttle	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
10 tables	Sum	5	InnoDB	utf8mb4_general_ci	224.0 KiB	0 B

Figure 63: Database Structure Test

4.1.2 Client Registration Test

4.1.2.1 Valid Data

	Action
Test Case	Register a client account providing valid data
Expected Outcome	Account gets registered and is redirected to login page
Actual Outcome	Account is registered and gets redirected to the login page
Result	Test Successful

Table 11: Client Registration Valid Data Test

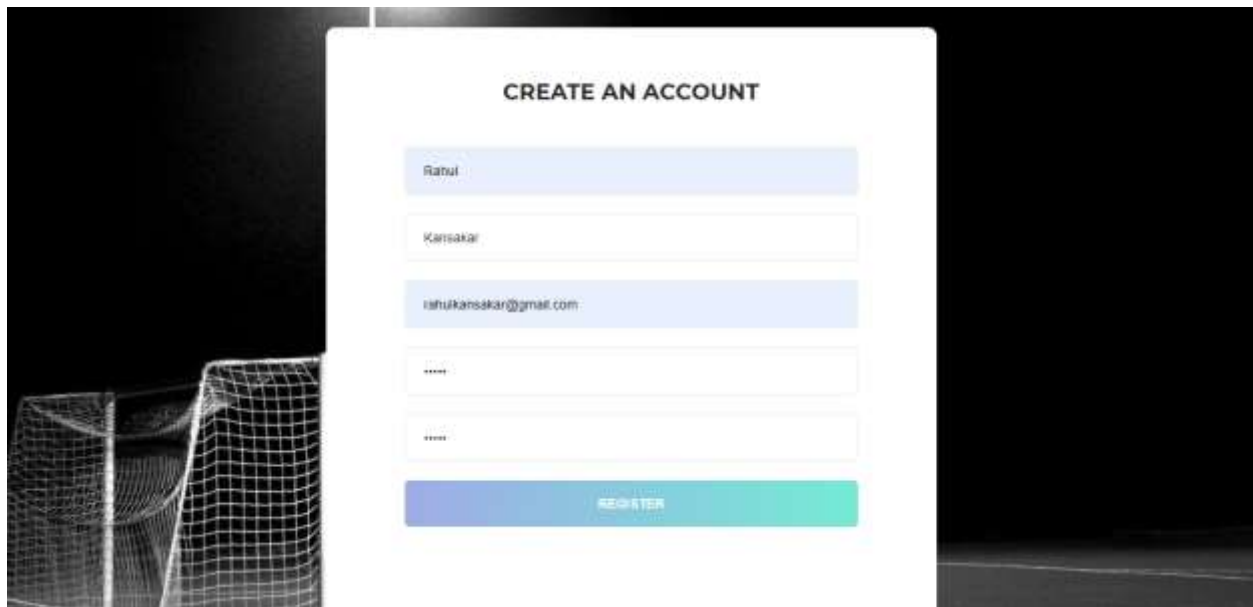


Figure 64: Client Registration Valid Data Test 1

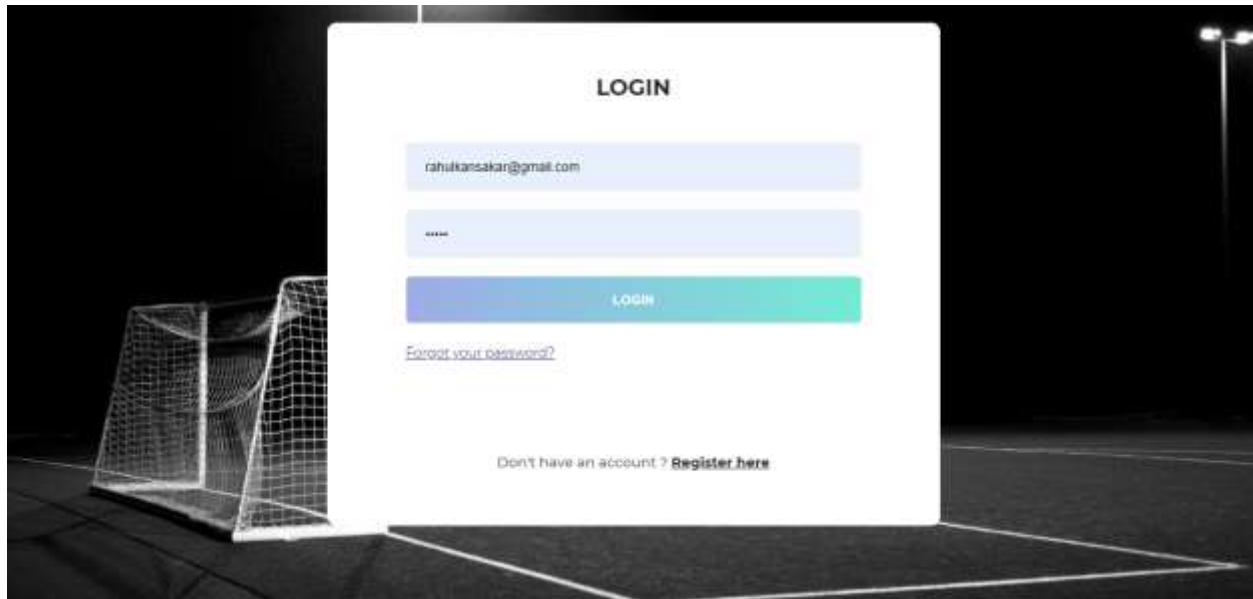


Figure 65: Client Registration Valid Data Test 2

4.1.2.2 Invalid Email

	Action
Test Case	Register account using invalid email.
Expected Outcome	Error message showing email is not valid
Actual Outcome	Error message is shown stating email not valid
Result	Test Successful

Table 12: Client Registration Invalid Email Test

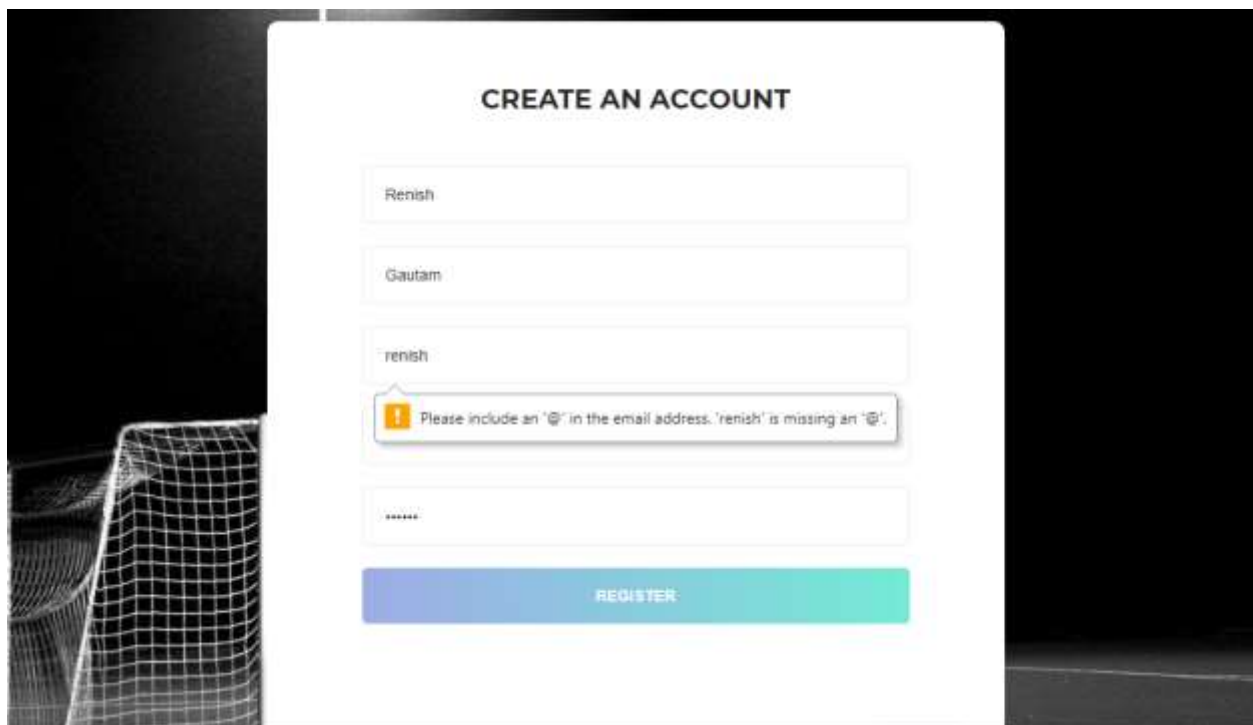


Figure 66: Client Registration Invalid Email Test

4.1.2.3 Empty Fields

	Action
Test Case	Register account with empty fields.
Expected Outcome	Throw error with message
Actual Outcome	Error message is thrown.
Result	Test Successful

Table 13: Client Registration Empty Fields Test

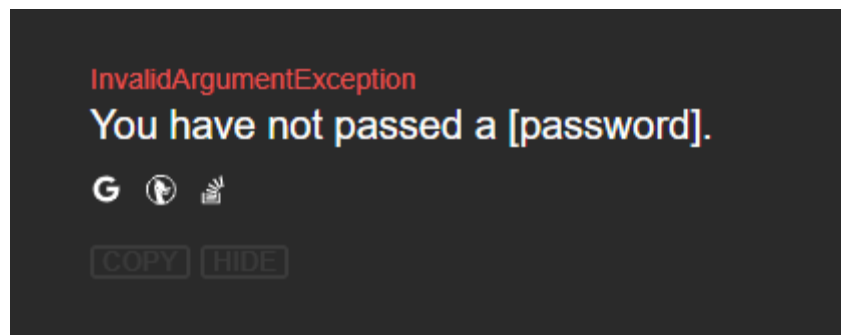


Figure 67: Client Registration Empty Fields Test

4.1.3 Admin Login Test

4.1.3.1 Valid Data

	Action
Test Case	Log into account with admin account with valid data
Expected Outcome	Login and redirect to admin dashboard
Actual Outcome	Logged into account and redirected to admin dashboard
Result	Test successful

Table 14: Admin Login Valid Data Test

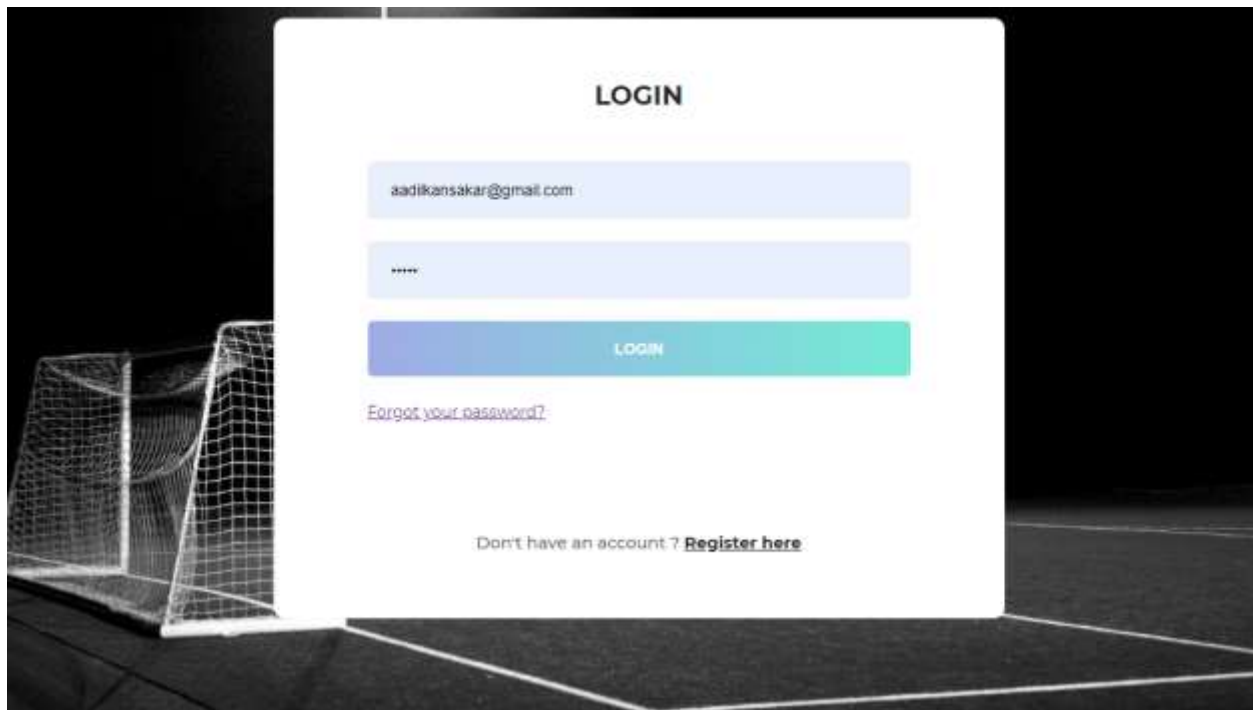


Figure 68: Admin Login Valid Data Test 1

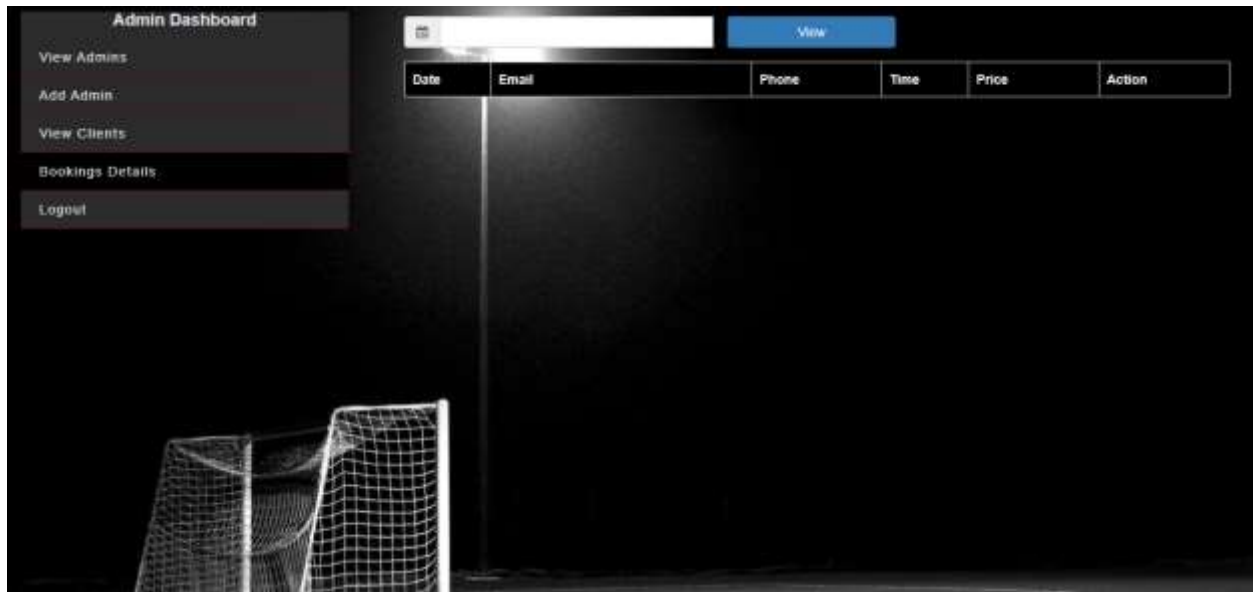


Figure 69: Admin Login Valid Data Test 2

4.1.3.2 Invalid Data

	Action
Test Case	Login using wrong password
Expected Outcome	Does not login and gives error message
Actual Outcome	It does not login and gives error message.
Result	Test successful

Table 15: Admin Login Invalid Data Test

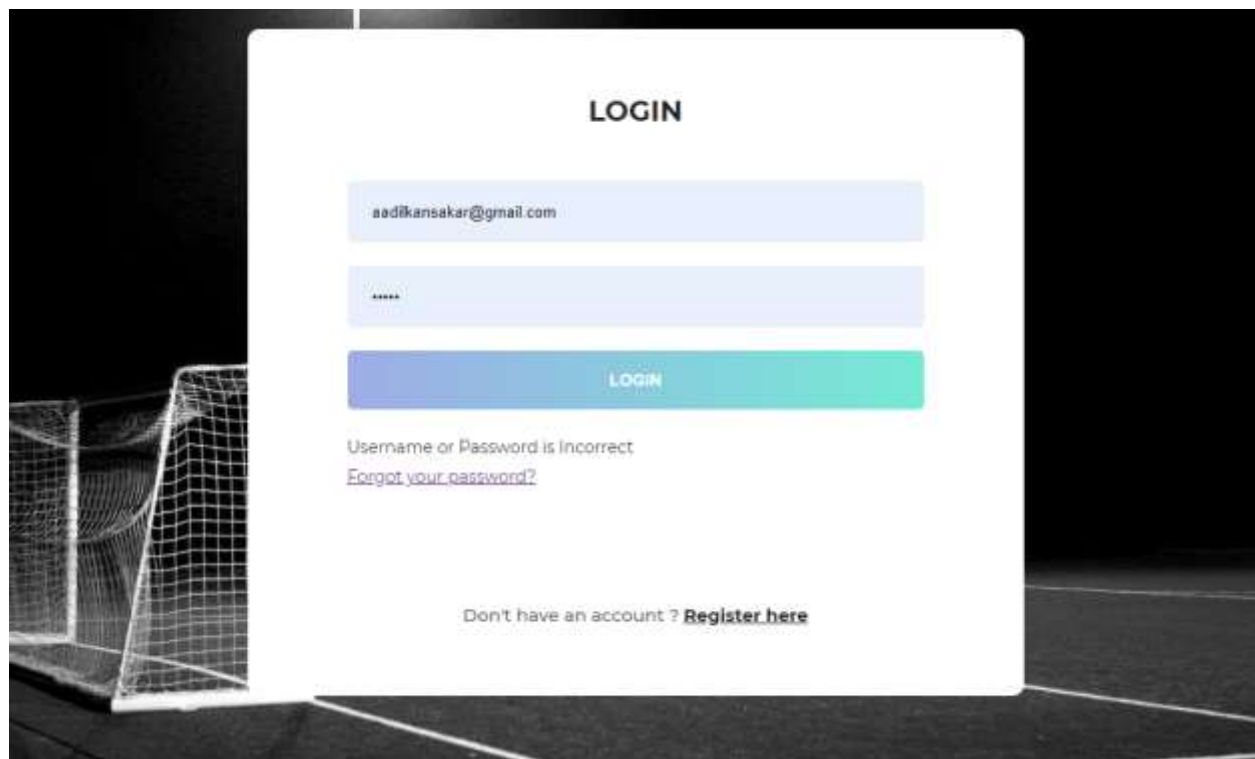


Figure 70: Admin Login Invalid Data Test

4.1.4 Client Login Test

4.1.4.1 Valid Data

	Action
Test Case	Log into client account using registered account
Expected Outcome	Log into account and redirect to client dashboard
Actual Outcome	Logged into account and redirected to client dashboard
Result	Test successful

Table 16: Client Login Valid Data Test

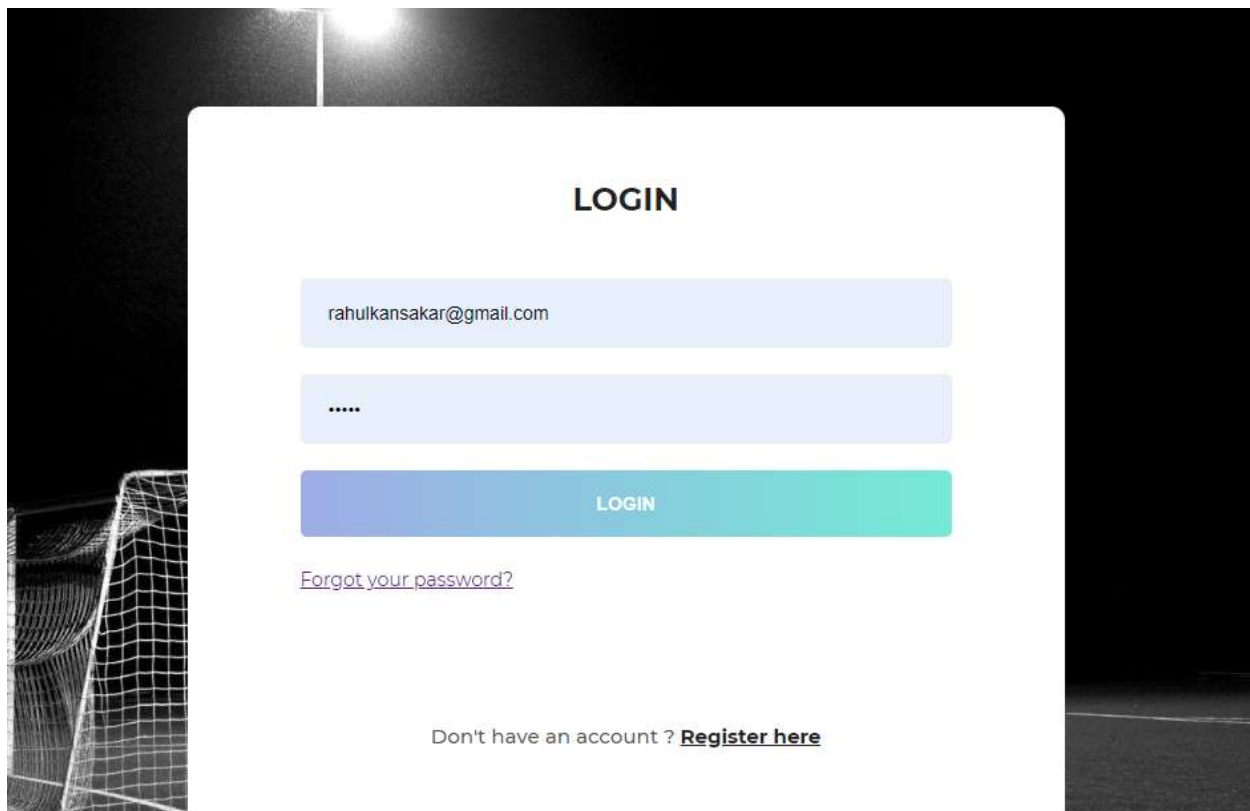


Figure 71: Client Login Valid Data Test 1

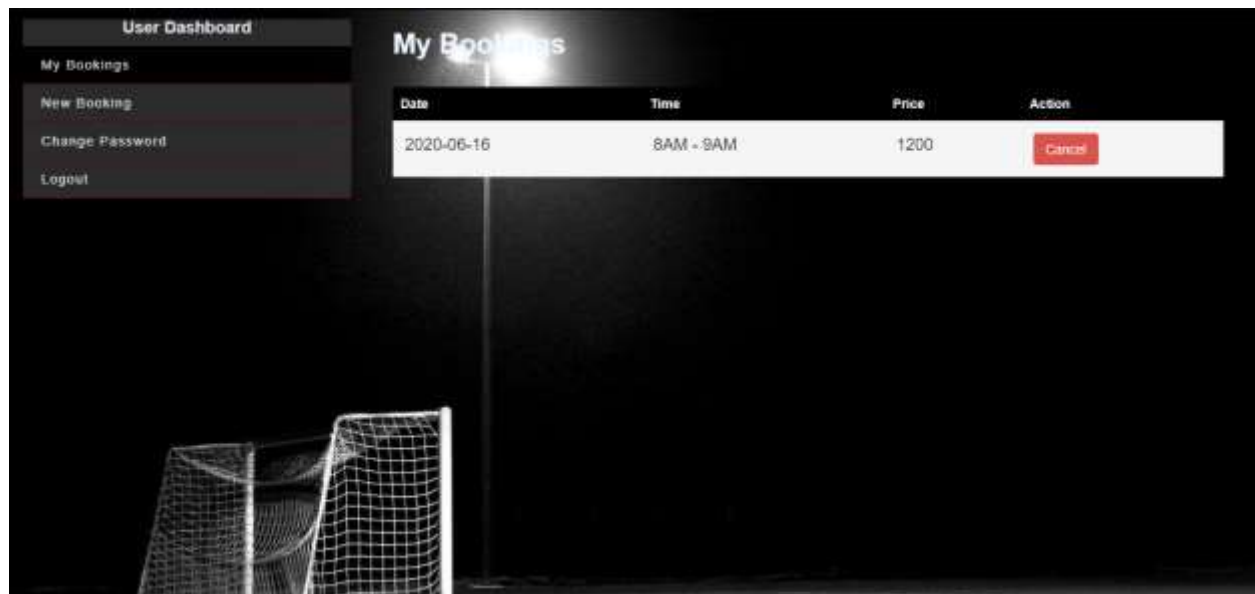


Figure 72: Client Login Valid Data Test 2

4.1.4.2 Invalid Data

	Action
Test Case	Login using wrong password
Expected Outcome	Does not login and gives error message
Actual Outcome	It does not login and gives error message.
Result	Test successful

Table 17: Client Login Invalid Data Test

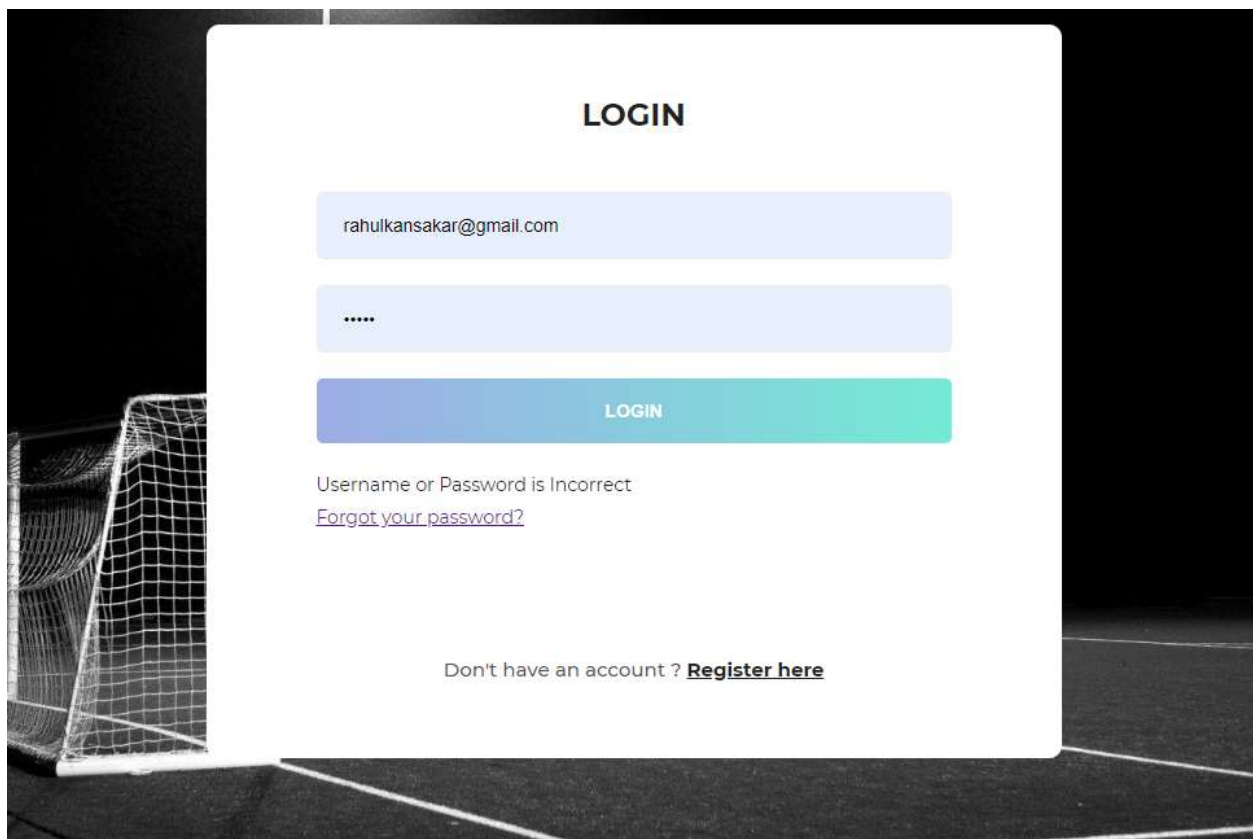


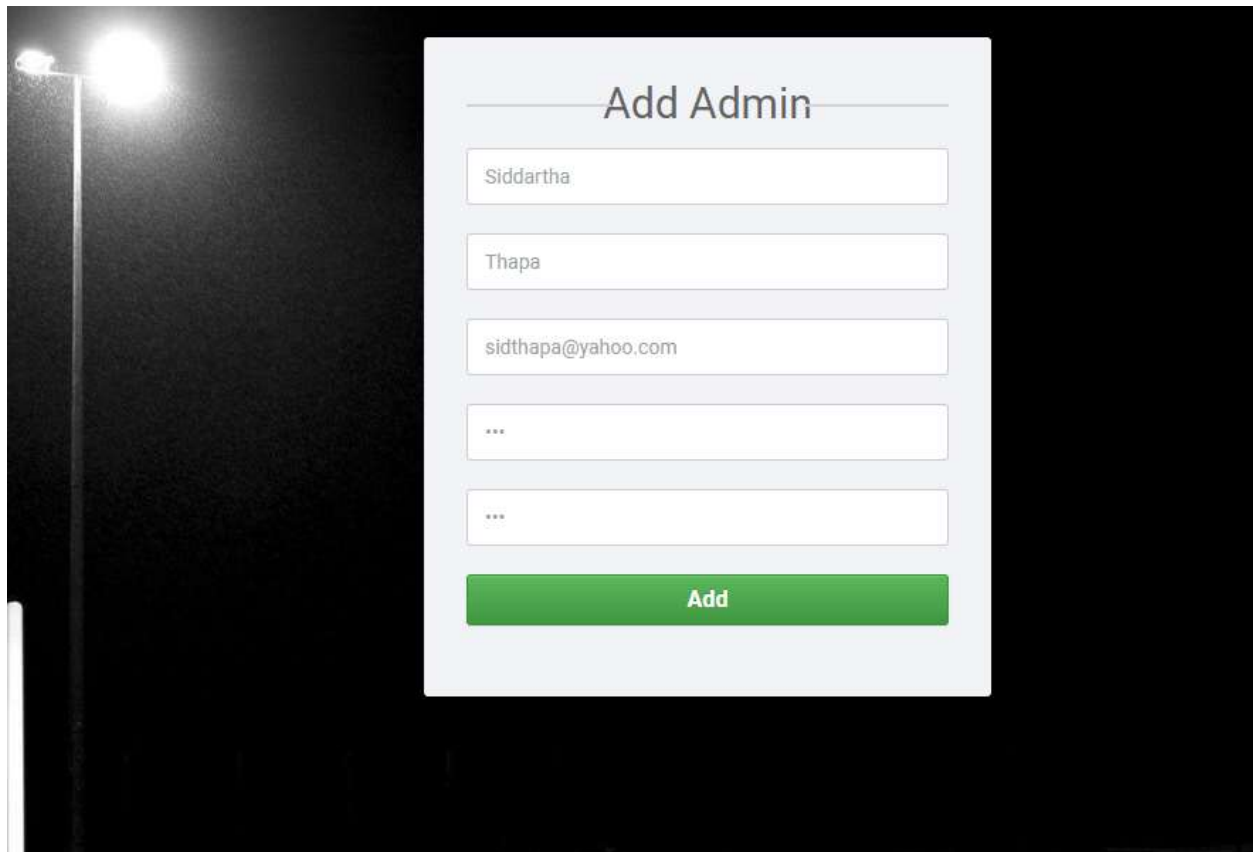
Figure 73: Client Login Invalid Data Test

4.1.5 Add Admin Test

4.1.5.1 Valid Data

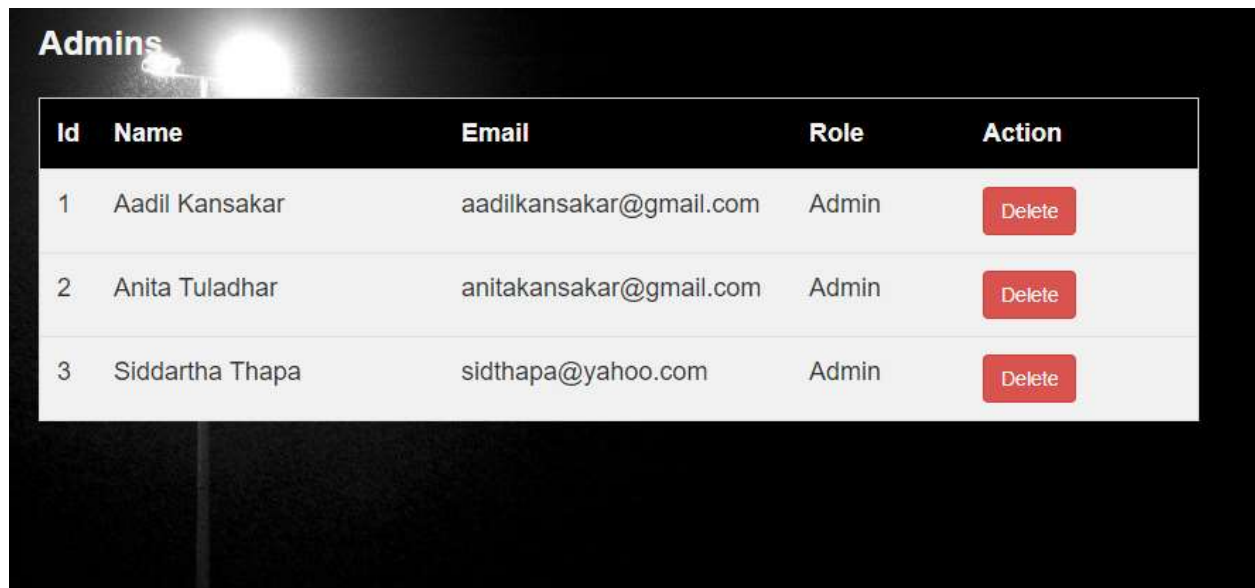
	Action
Test Case	Add new admin
Expected Outcome	New account is added as admin
Actual Outcome	New admin is added and is redirected to View Admin Page
Result	Test successful

Table 18: Add Admin Valid Data Test



The image shows a screenshot of a web application interface for adding a new admin. The form is titled "Add Admin" and is set against a dark background with a street lamp on the left. The form contains five input fields: a name field with "Siddhartha", a last name field with "Thapa", an email field with "sidthapa@yahoo.com", and two password fields, both containing three asterisks to indicate masked text. A green "Add" button is positioned at the bottom of the form.

Figure 74: Add Admin Valid Data Test 1



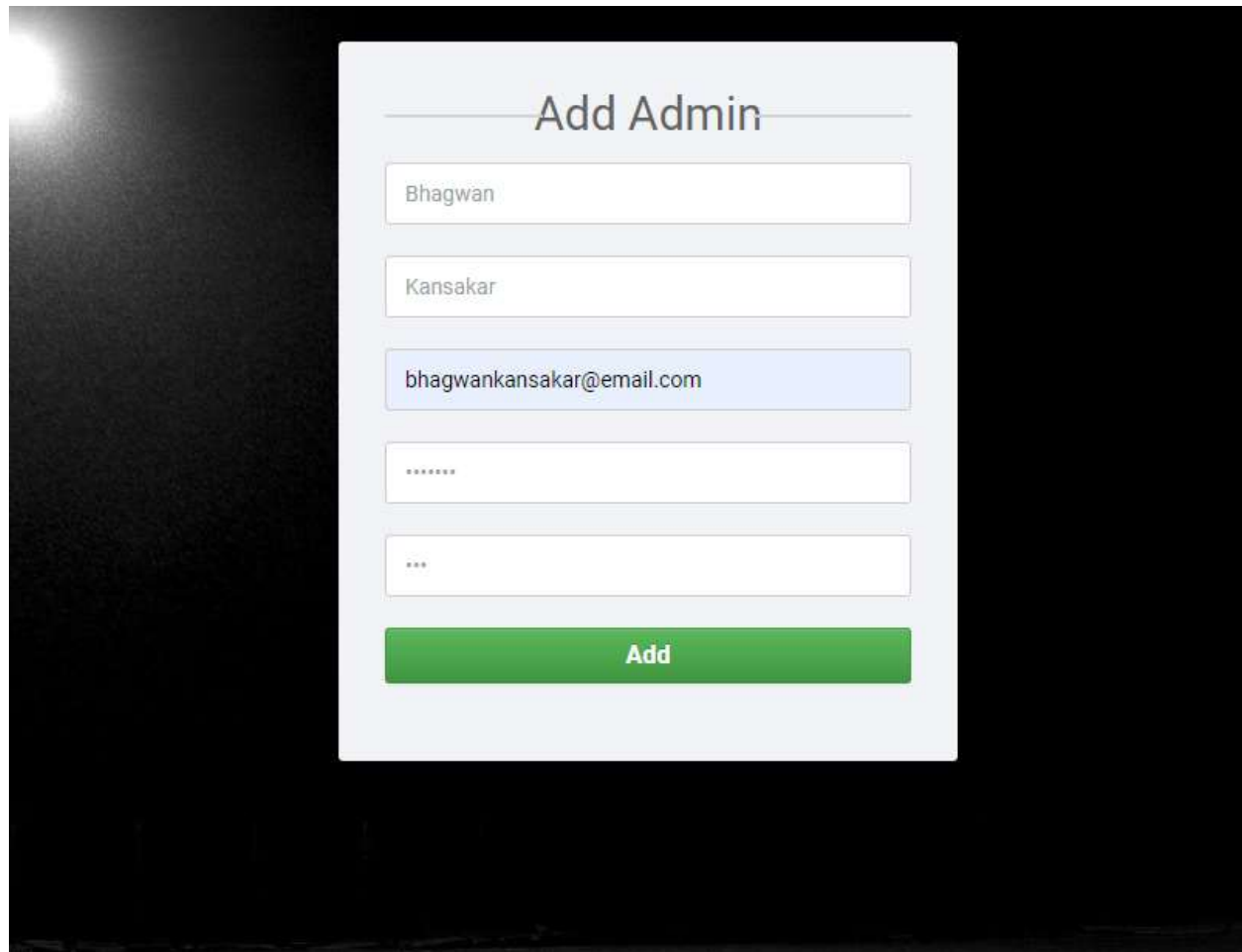
The screenshot shows a web interface titled "Admins" with a table listing three administrators. Each row includes an ID, Name, Email, Role, and an Action button labeled "Delete".

Id	Name	Email	Role	Action
1	Aadil Kansakar	aadilkansakar@gmail.com	Admin	Delete
2	Anita Tuladhar	anitakansakar@gmail.com	Admin	Delete
3	Siddartha Thapa	sidthapa@yahoo.com	Admin	Delete

Figure 75: Add Admin Valid Data Test 2

4.1.5.2 Invalid Data

	Action
Test Case	Add new admin using unmatching passwords
Expected Outcome	Error message is displayed.
Actual Outcome	Admin is not added and error message is displayed
Result	Test successful

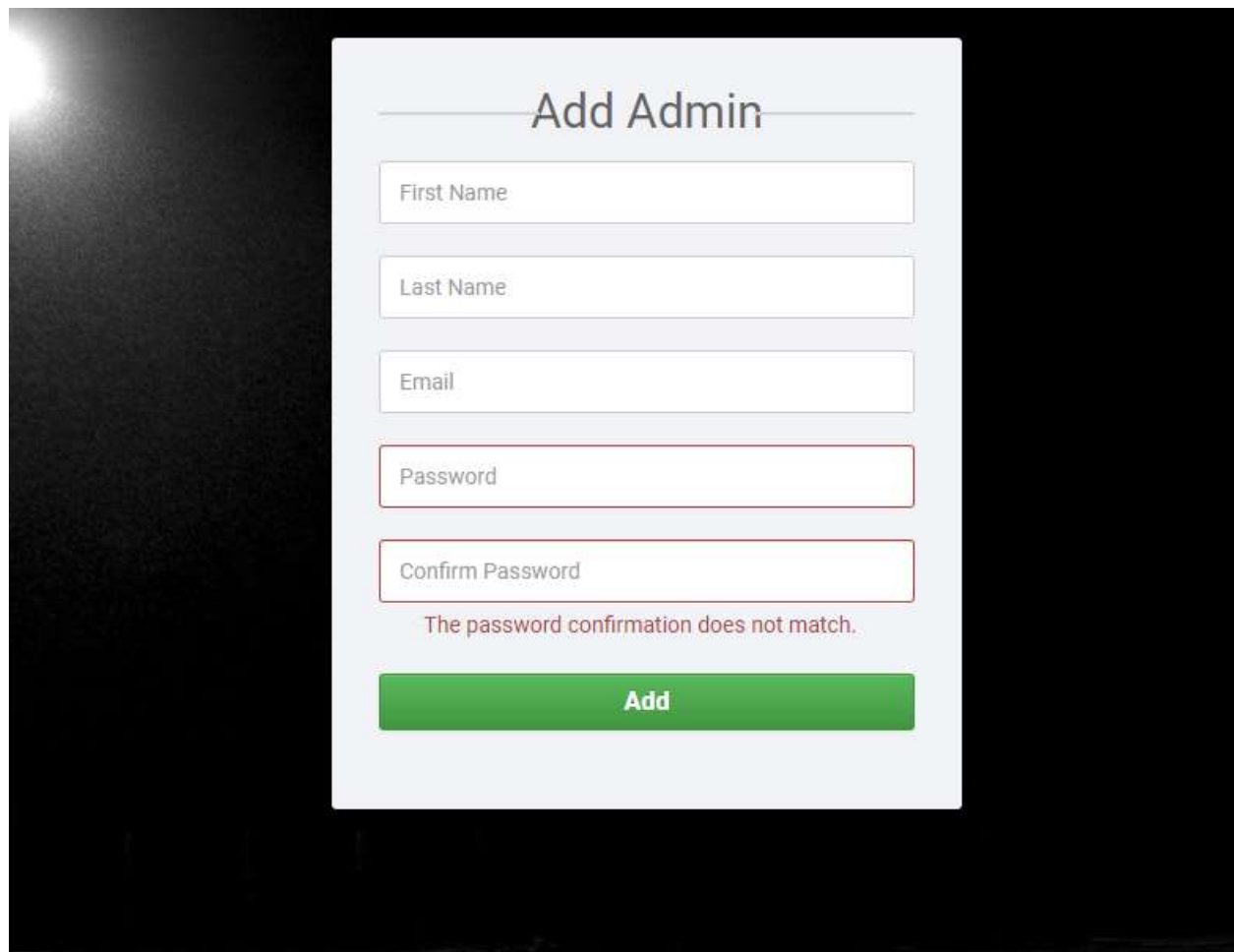
Table 19: Add Admin Invalid Data Test

The screenshot shows a web form titled "Add Admin" with the following fields and values:

- First Name: Bhagwan
- Last Name: Kansakar
- Email: bhagwankansakar@email.com
- Password: *****
- Confirm Password: ***
- Submit Button: Add

The form is displayed on a dark background, and the email field is highlighted in blue.

Figure 76: Add Admin Invalid Data Test 1



The image shows a web form titled "Add Admin" with a light blue header. Below the title are five input fields: "First Name", "Last Name", "Email", "Password", and "Confirm Password". The "Confirm Password" field has a red border, indicating an error. Below the fields, a red message states: "The password confirmation does not match." At the bottom is a green "Add" button.

Add Admin

First Name

Last Name

Email

Password

Confirm Password

The password confirmation does not match.

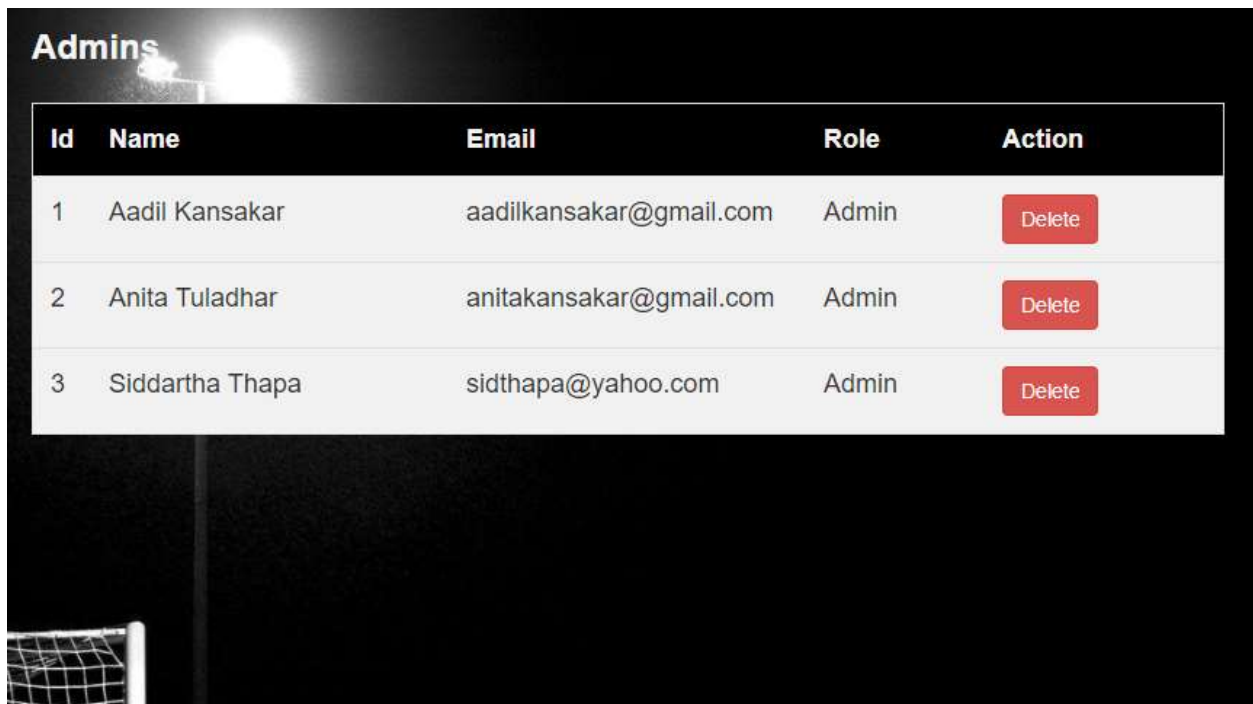
Add

Figure 77: Add Admin Invalid Data Test 2

4.1.6 Delete Admin Test

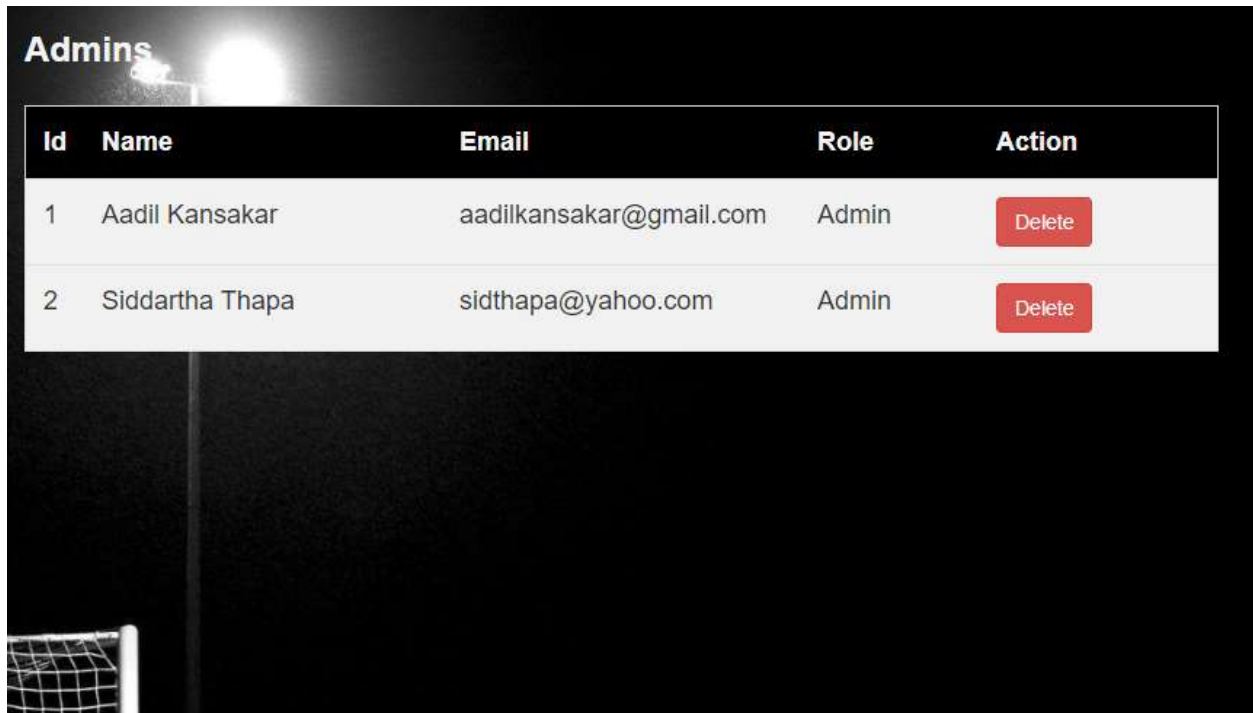
	Action
Test Case	Delete admin account
Expected Outcome	Admin account is deleted.
Actual Outcome	Admin account is deleted and removed from the list.
Result	Test successful

Table 20: Delete Admin Test



Id	Name	Email	Role	Action
1	Aadil Kansakar	aadilkansakar@gmail.com	Admin	Delete
2	Anita Tuladhar	anitakansakar@gmail.com	Admin	Delete
3	Siddhartha Thapa	sidthapa@yahoo.com	Admin	Delete

Figure 78: Delete Admin Test 1



Admins

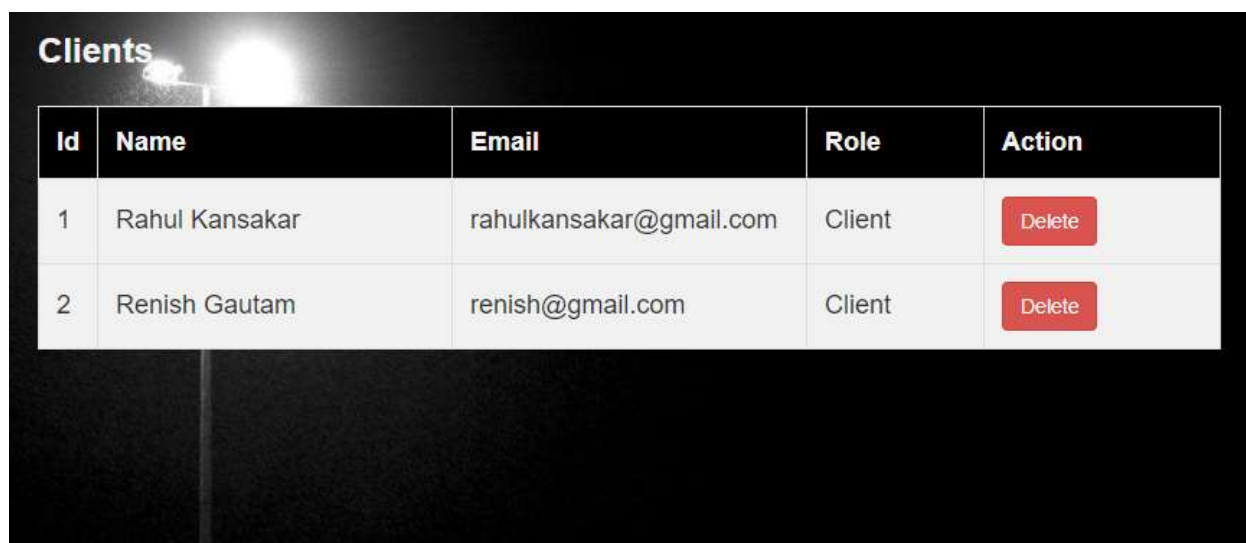
Id	Name	Email	Role	Action
1	Aadil Kansakar	aadilkansakar@gmail.com	Admin	Delete
2	Siddartha Thapa	sidthapa@yahoo.com	Admin	Delete

Figure 79: Delete Admin Test 2

4.1.7 Delete Client Test

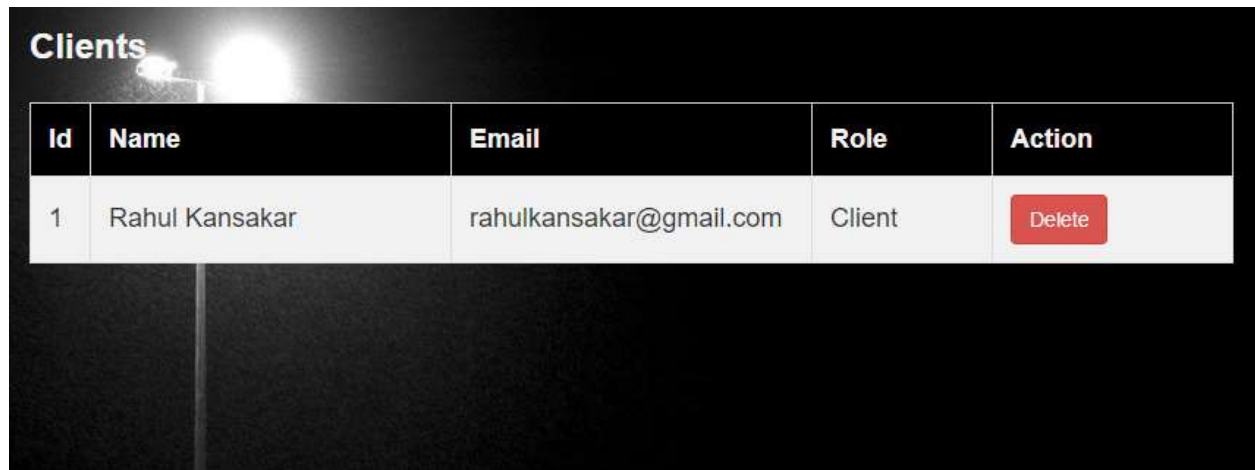
	Action
Test Case	Delete client account
Expected Outcome	Client account is deleted.
Actual Outcome	Client account is deleted and removed from the list.
Result	Test successful

Table 21: Delete Client Test



Id	Name	Email	Role	Action
1	Rahul Kansakar	rahulkansakar@gmail.com	Client	Delete
2	Renish Gautam	renish@gmail.com	Client	Delete

Figure 80: Delete Client Test 1



The screenshot shows a web application interface with a dark background. At the top left, the word 'Clients' is displayed in a light blue font. Below it is a table with five columns: 'Id', 'Name', 'Email', 'Role', and 'Action'. The table contains one data row for a client named 'Rahul Kansakar' with email 'rahulkansakar@gmail.com' and role 'Client'. In the 'Action' column, there is a red button labeled 'Delete'.

Id	Name	Email	Role	Action
1	Rahul Kansakar	rahulkansakar@gmail.com	Client	<button>Delete</button>

Figure 81: Delete Client Test 2

4.1.8 Admin Delete Booking Test

	Action
Test Case	Delete booking made by client
Expected Outcome	Booking is deleted.
Actual Outcome	Booking is deleted and removed from the list.
Result	Test successful

Table 22: Admin Delete Booking Test

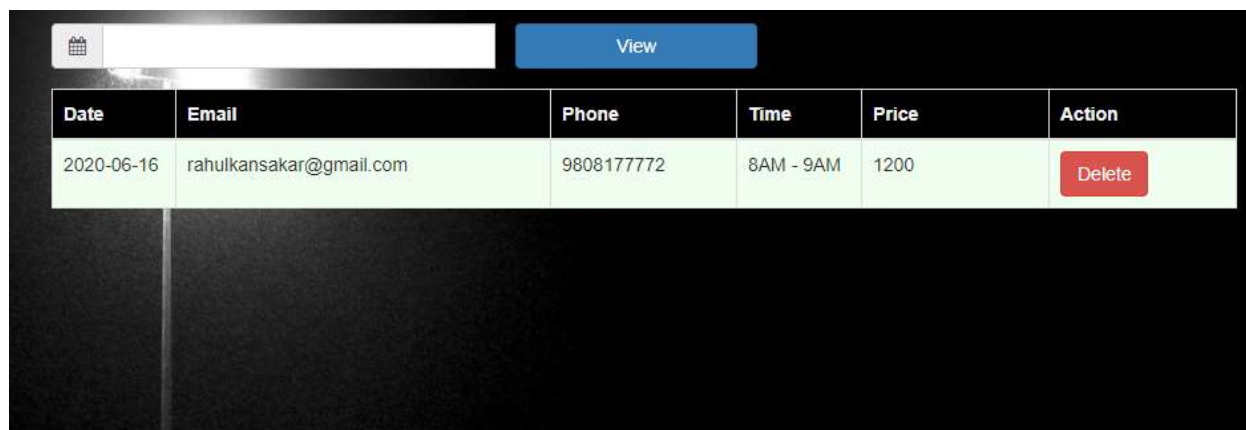


Figure 82: Admin Delete Booking Test 1

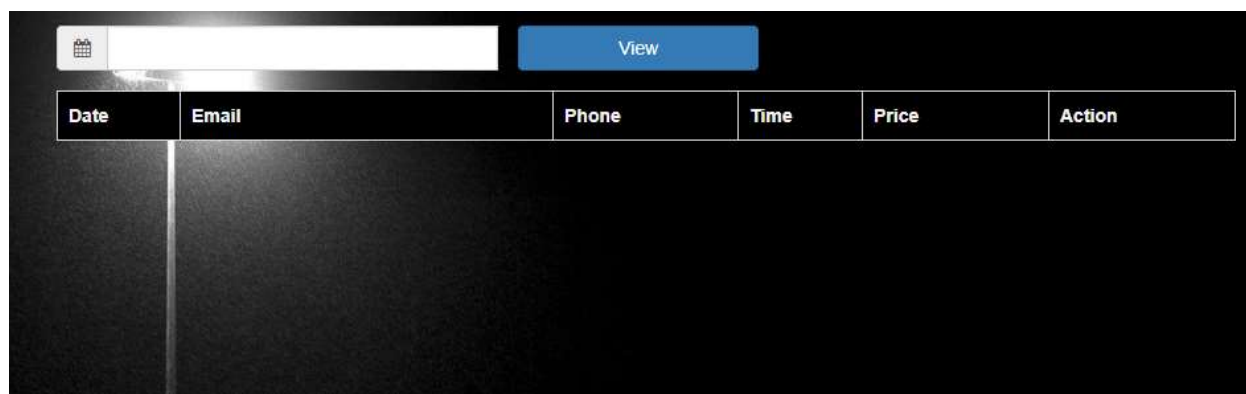


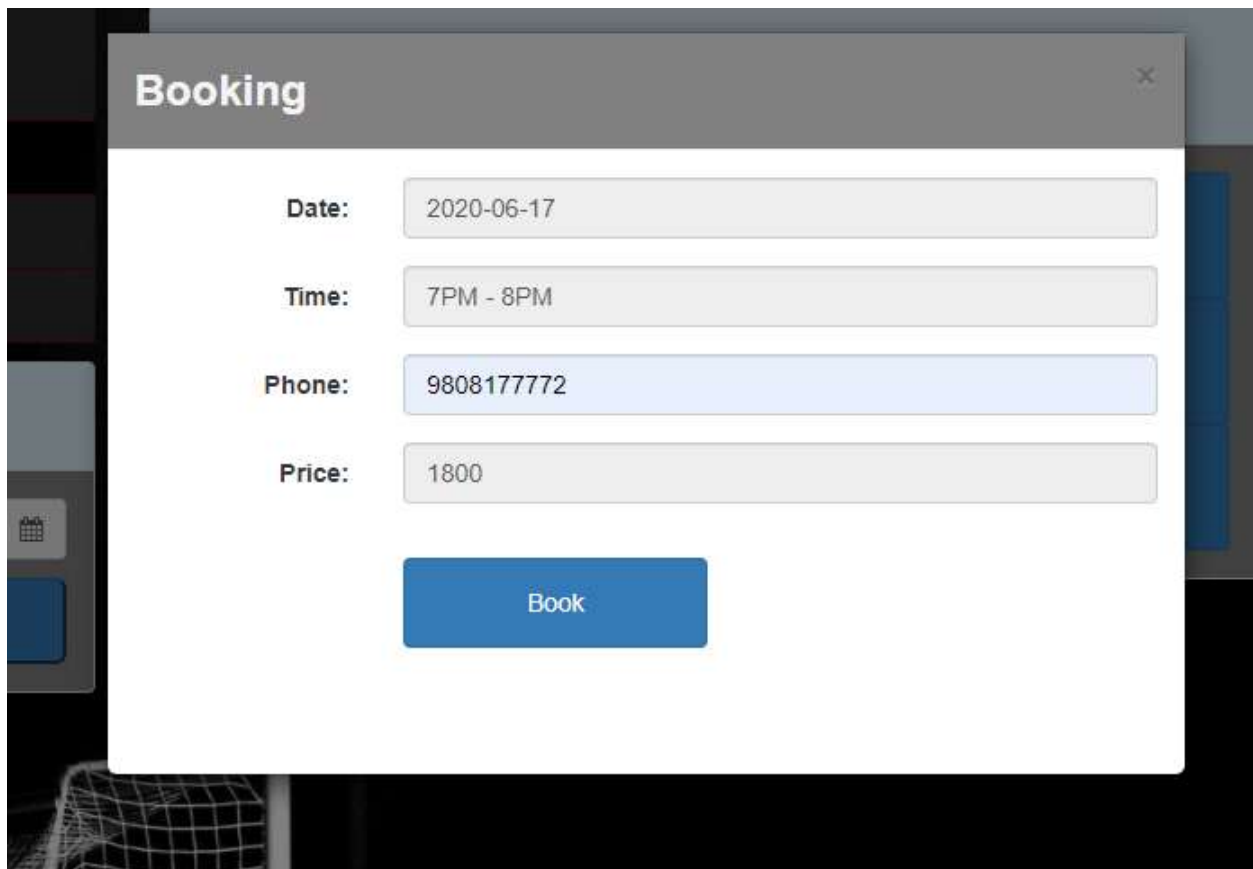
Figure 83: Admin Delete Booking Test 2

4.1.9 Make Booking Test

4.1.9.1 Valid Data

	Action
Test Case	Make a new booking
Expected Outcome	Booking is made.
Actual Outcome	Booking is made and redirected to My Bookings Page
Result	Test successful

Table 23: Make Booking Test



The image shows a 'Booking' modal window with a close button (X) in the top right corner. The modal contains four input fields and a 'Book' button. The fields are labeled 'Date:', 'Time:', 'Phone:', and 'Price:'. The values entered in the fields are '2020-06-17', '7PM - 8PM', '9808177772', and '1800' respectively. The 'Book' button is blue and located at the bottom of the modal.

Figure 84: Make Booking Test 1

User Dashboard

- My Bookings
- New Booking
- Change Password
- Logout

Date: 2020-06-17

Check

Timing

7AM - 8AM	8AM - 9AM	9AM - 10AM
10AM - 11AM	11AM - 12PM	12PM - 1PM
1PM - 2PM	7PM - 8PM	8PM - 9PM

Figure 85: Make Booking Test 2

My Bookings

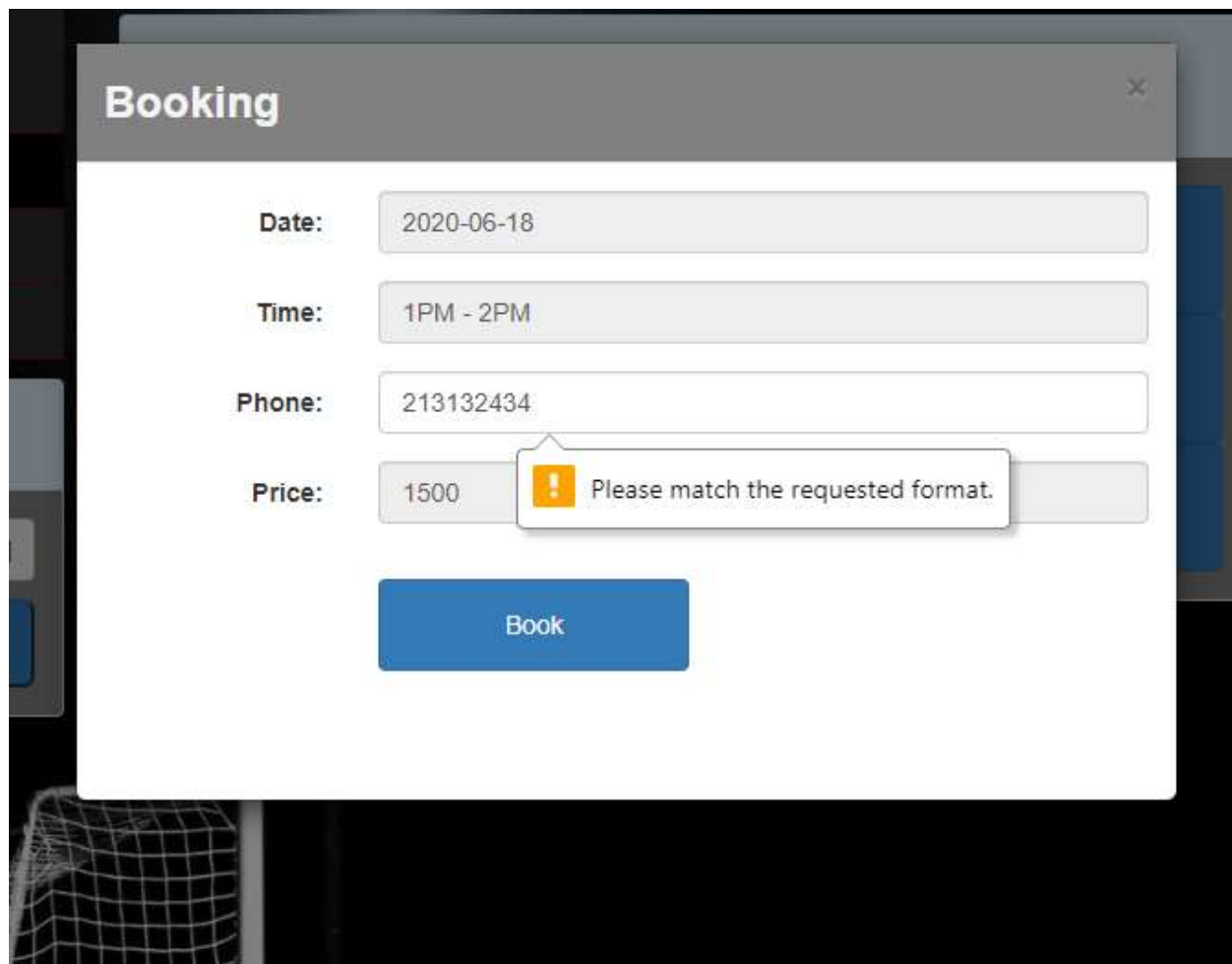
Date	Time	Price	Action
2020-06-17	10AM - 11AM	1500	<button>Cancel</button>
2020-06-17	7PM - 8PM	1800	<button>Cancel</button>

Figure 86: Make Booking Test 3

4.1.9.2 Invalid Phone Number

	Action
Test Case	Make a new booking using invalid phone number format
Expected Outcome	Error message is displayed
Actual Outcome	Booking is not made and error message is displayed
Result	Test successful

Table 24: Make Booking Invalid Phone Number Test



The screenshot shows a 'Booking' modal window with a close button (X) in the top right corner. The form contains four input fields: 'Date' with the value '2020-06-18', 'Time' with the value '1PM - 2PM', 'Phone' with the value '213132434', and 'Price' with the value '1500'. An orange error icon with an exclamation mark is positioned over the 'Phone' field, with a tooltip message that reads 'Please match the requested format.' Below the input fields is a blue 'Book' button.

Figure 87: Make Booking Invalid Phone Number Test

4.1.10 Client Delete Booking Test

	Action
Test Case	Delete booking made by user
Expected Outcome	Booking is deleted.
Actual Outcome	Booking is deleted and removed from the list.
Result	Test successful

Table 25: Delete Client Booking Test

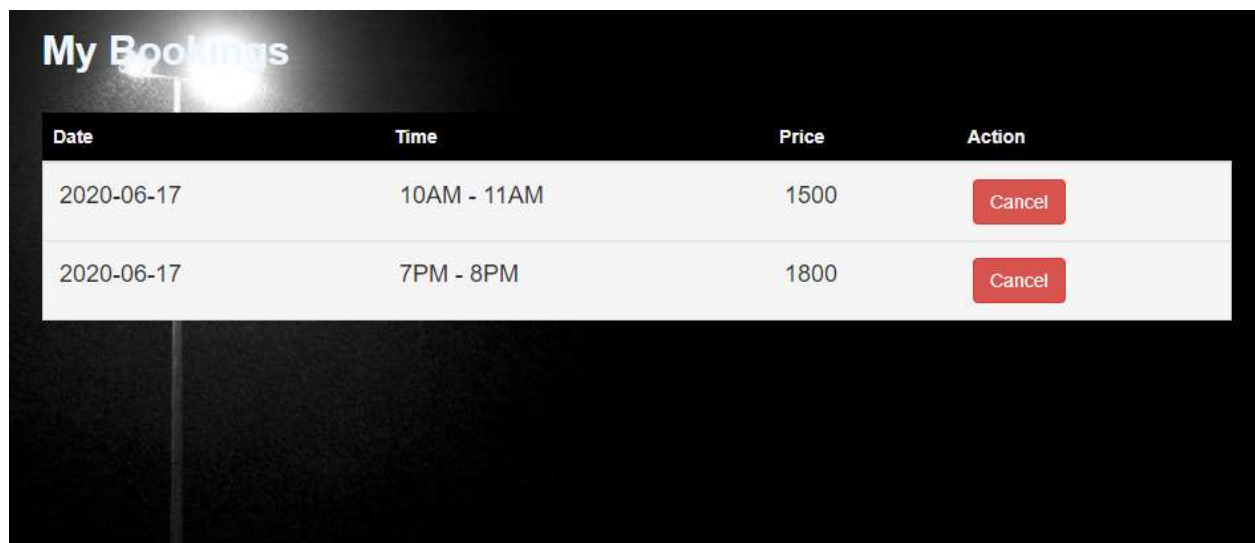


Figure 88: Delete Client Booking Test 1

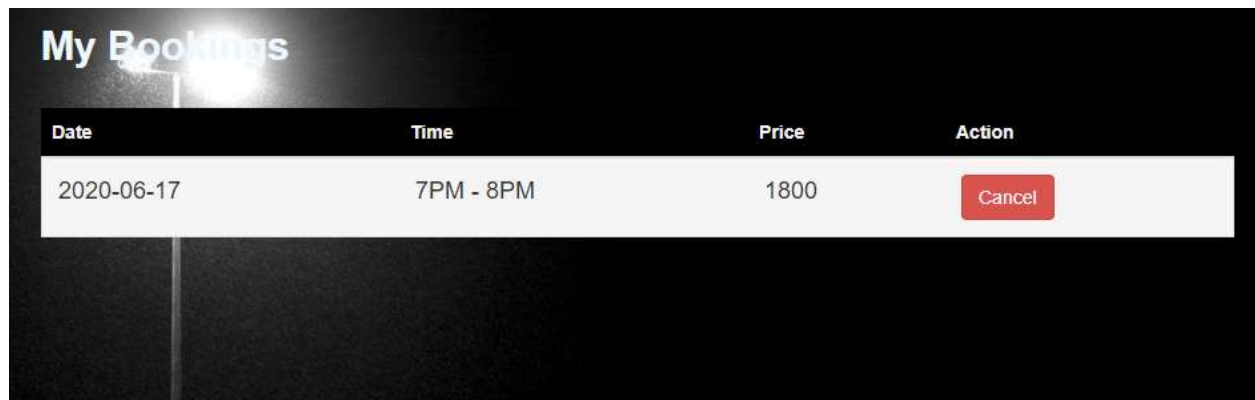


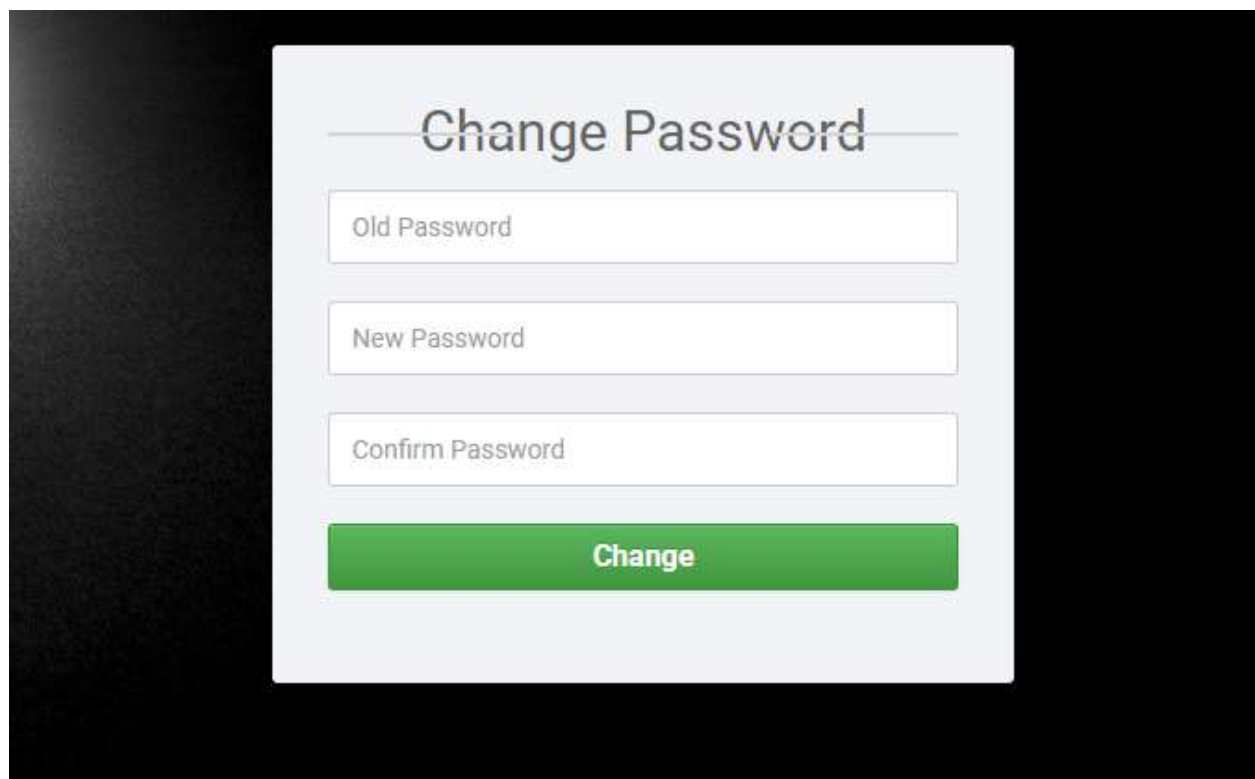
Figure 89: Delete Client Booking Test 2

4.1.11 Change Password Test

4.1.11.1 Valid Data

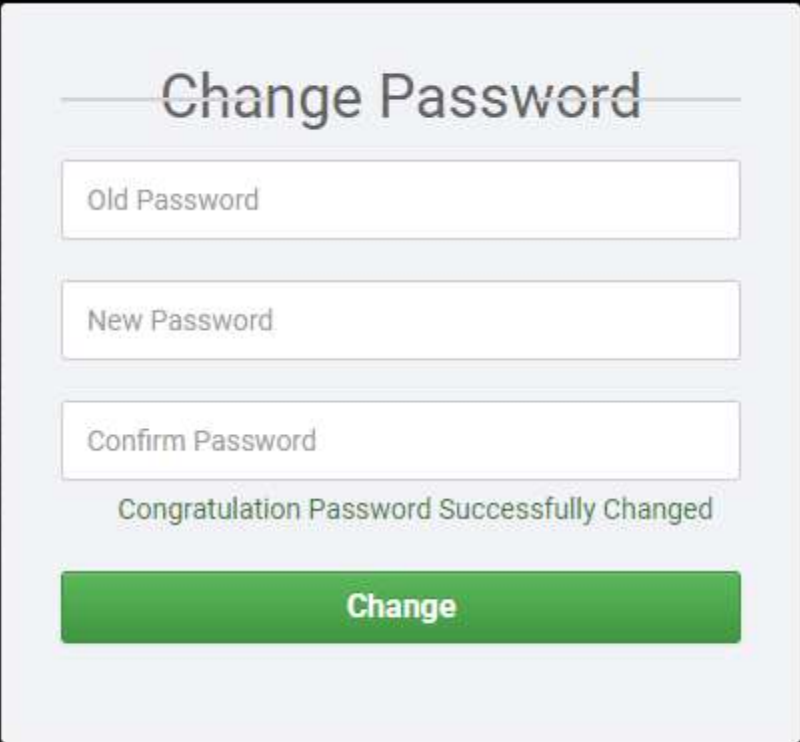
	Action
Test Case	Client change password
Expected Outcome	Password is changed
Actual Outcome	Password is changed and success message is delivered
Result	Test successful

Table 26: Change Password Test



The image shows a web form titled "Change Password" centered on a black background. The form has a light blue header with the title. Below the title are three input fields: "Old Password", "New Password", and "Confirm Password". At the bottom of the form is a green button labeled "Change".

Figure 90: Change Password Test 1



Change Password

Old Password

New Password

Confirm Password

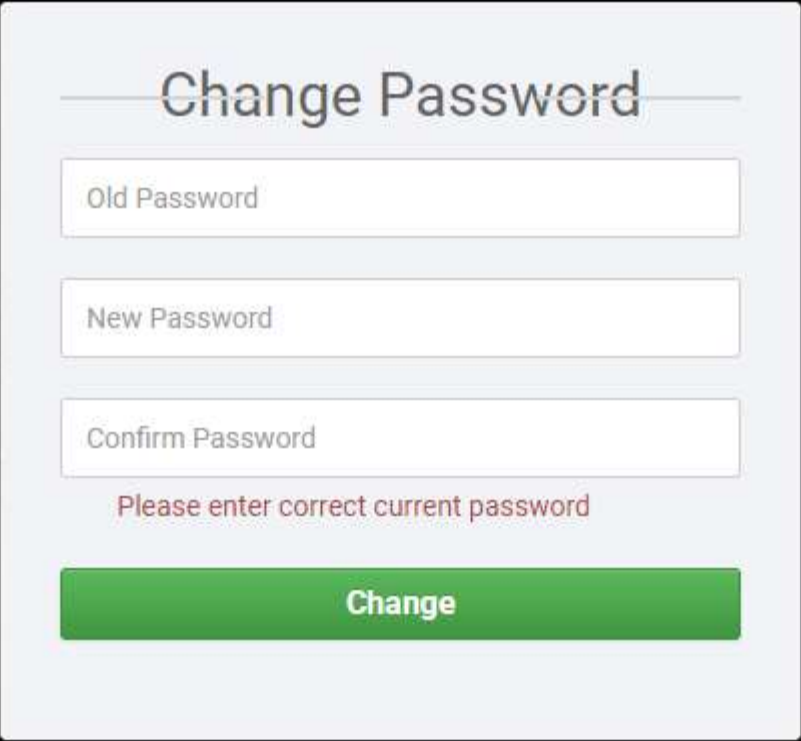
Congratulation Password Successfully Changed

Change

Figure 91: Change Password Test 2

4.1.11.2 Incorrect Current Password

	Action
Test Case	Change password using incorrect password
Expected Outcome	Password does not change
Actual Outcome	Password does not change and error message is delivered
Result	Test Successful

Table 27: Change Password using incorrect current password Test

The screenshot shows a 'Change Password' form with three input fields: 'Old Password', 'New Password', and 'Confirm Password'. Below the 'Old Password' field, there is a red error message that reads 'Please enter correct current password'. At the bottom of the form is a green button labeled 'Change'.

Figure 92: Change Password using incorrect current password Test

4.1.11.3 Passwords do not match

	Action
Test Case	Change password using unmatching password
Expected Outcome	Password does not change
Actual Outcome	Password does not change and error message is displayed
Result	Test successful

Table 28: Change Password using unmatching password Test

The screenshot shows a web form titled "Change Password" with a light blue background. It contains three input fields: "Old Password", "New Password", and "Confirm Password". Below the "Confirm Password" field, a red error message states "The password confirmation does not match." At the bottom of the form is a green button labeled "Change". The entire form is set against a black background.

Figure 93: Change Password using unmatching password Test

4.2 Critical Analysis

All the testing of the website, the booking system was carried out during the testing phase. The testing was done to ensure all the functionality and features of the booking system goes as smoothly as possible. Black box testing and White box testing of the system was done to check the practical features like booking and changing password could be done by the client. The admin dashboard features were also tested to make the experience of the booking system better, smoother and bug free.

5 Future Works

The booking system can be developed in a mobile application for making the use of the system better. Payment gateways can be added in this system so, customers can pay as well. This can be made possible with digital wallet like eSewa and Khalti, which are used by many users in Nepal. The user can also be verified using phone or by sending a code to their email for an extra layer of security. The security features of the booking system can also be vastly improved to prevent hacking from malicious software.

6 Conclusion

The development of the futsal booking system using Laravel framework gave a wonderful opportunity to be experienced in developing a website using MVC architecture. During the development of this system, many confusions and difficulties arose, which were improved by attending classes and through the internship in web development.

Task and time management was very important for reaching set goals and maintaining time so that the project can keep going and eventually be completed.

This module and project have helped to get a better understanding of the uses of databases, frameworks, IDEs and other tools used during this module.

7 References

Cartalyst. (2018) *Sentinel Manual ; Cartalyst* [Online]. Available from: <https://cartalyst.com/manual/sentinel/4.x> [Accessed 12 April 2020].

Cprime. (2019) *WHAT IS AGILE? WHAT IS SCRUM?* [Online]. Available from: <https://www.cprime.com/resources/what-is-agile-what-is-scrum/> [Accessed 11 November 2019].

Gridsada, P. & Worathon, P. (2014) *Futsal Field Management System*. Attawit Commercial Technology College.

larashout. (2018) *What is Laravel and Why You Should Learn it?* [Online]. Available from: <https://www.larashout.com/what-is-laravel-and-why-you-should-learn-it> [Accessed 14 April 2020].

Majeed, A. & Rauf, I. (2018) MVC Architecture: A Detailed Insight to the Modern Web Applications Development. *Peer Review Journal of Solar & Photoenergy Systems*, 1(1).

Patel, D. (2019) *An Introduction to MVC Architecture: A Web Developer's Point of View* [Online]. Available from: <https://dzone.com/articles/introduction-to-mvc-architecture-web-developer-poi> [Accessed 3 January 2020].

Rafie, Y. (2015) *Removing the Pain of User Authorization with Sentinel* [Online]. Available from: <https://www.sitepoint.com/removing-the-pain-of-user-authorization-with-sentinel/> [Accessed 12 April 2020].

Rouse, M. (2018) *What is MySQL?* [Online]. Available from: <https://searchoracle.techtarget.com/definition/MySQL> [Accessed 4 April 2020].

Terzani, A. (2015) *Hotel booking engine with Laravel 5 and AngularJS Part 1* [Online]. Available from: <https://medium.com/@andreaterzani/hotel-booking-engine-with-laravel-5-and-angularjs-part-1-81af69432b1a> [Accessed 3 April 2020].

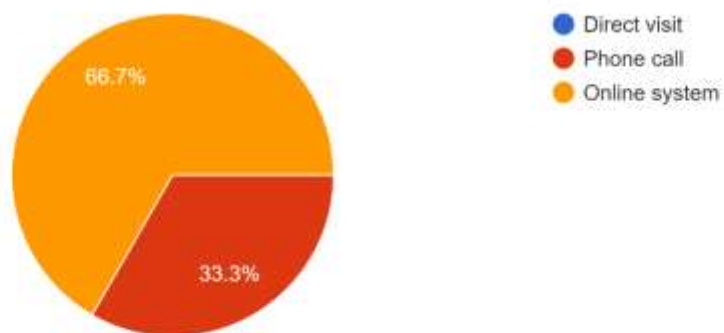
8 Appendix

8.1 Appendix-A: Survey

8.1.1 Survey Result

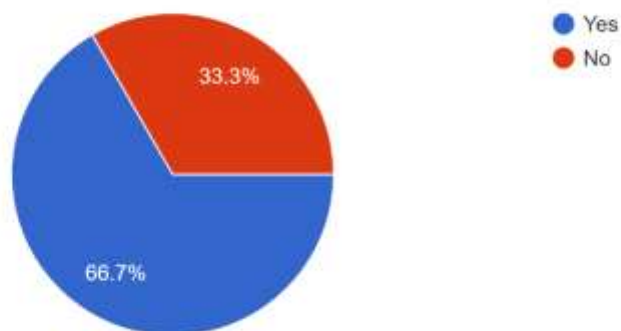
What method would you prefer to book a venue?

6 responses



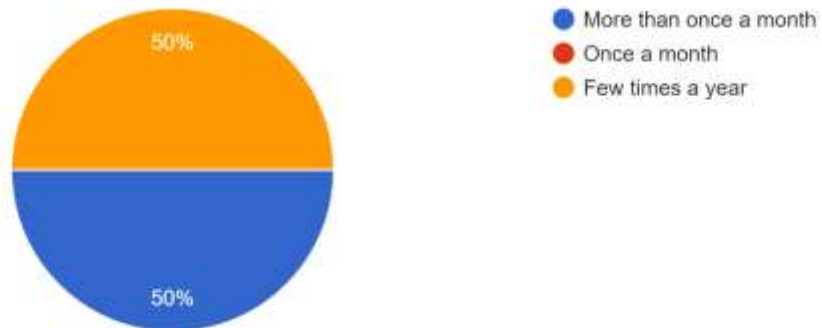
Have you heard of Futsal Booking System in Nepal?

6 responses



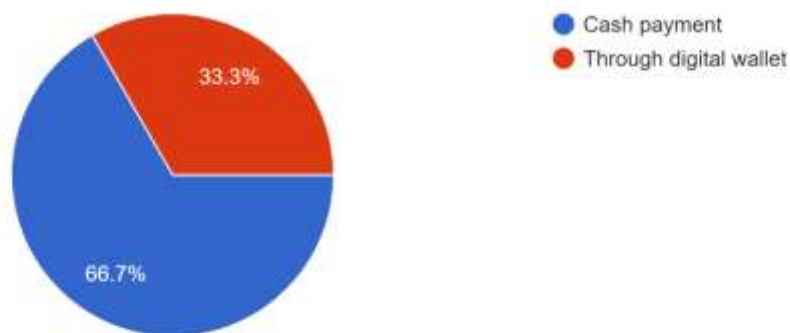
How often do you play futsal?

6 responses



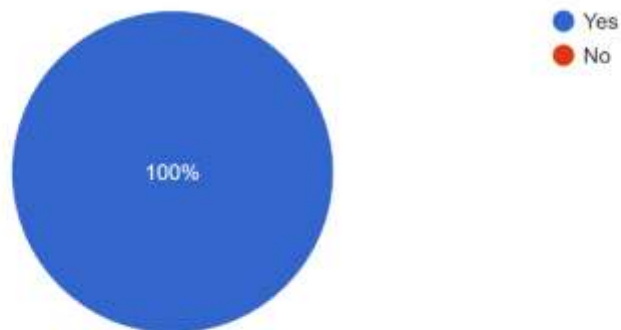
What mode of payment would you like for online booking?

6 responses



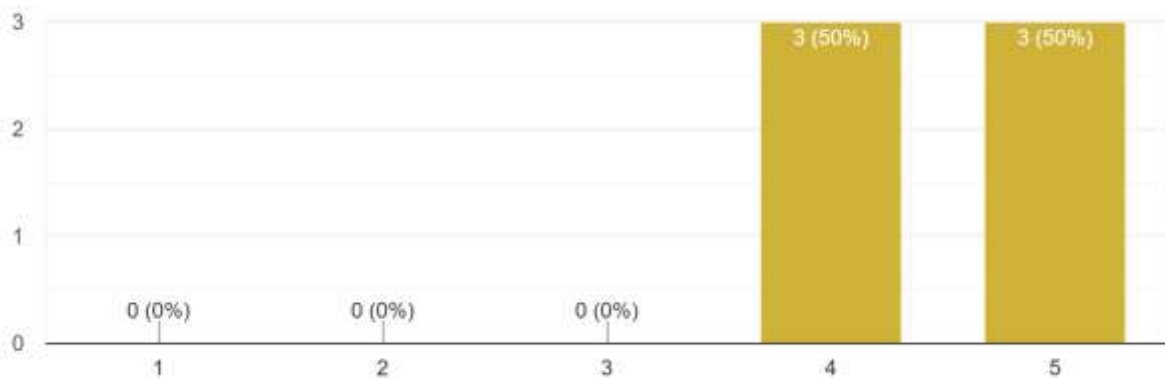
Are you likely to book a futsal ground using a booking app?

6 responses



How often do you think you will use the futsal booking system?

6 responses



8.2 Appendix-B: Sample Codes

8.2.1 Sample Code of the UI

```
@extends('client.bookNow')

@section('MenuContent')

<style type="text/css">
.container
{
    width: 85%;
    margin-top: 15px;
    margin-left: -15px;
}
.btnTime
{
    padding:24px;
}
</style>

<div class="container" id="availTime">
    <div class="panel panel-info" style="background-color: grey;">
        <div class="panel-heading">
            <h3 style="color: black;">Timing</h3>
        </div>
        <div class="panel-body">
            <div class="row">
                @php
                    $tt = date("Y-m-d",time()); //current date
                    $choice_date = ($_REQUEST['datecal'])? $_REQUEST["datecal"]:false; //date that came from the datepicker
                    $need_sorting = ($tt==$choice_date)? true:false;
                    foreach($availableTime as $time)
                    {
                        $printer = '<div class="col-xs-4"><div class="btn btn-
primary btn-block btnTime" data-toggle="modal" data-
```

```

target="#bookModal" value="'.{$time->price.'}">'.{$time->
>label.'</div></div>';
    foreach ($booking as $book )
    {
        # code...
        if ($time->label==$book->time) {
            $printer = '<div class="col-xs-4"><div class="btn btn-
danger btn-block btnTime" disabled data-toggle="modal" data-
target="#bookModal" value="'.{$time->price.'}">'.{$time->
>label.'</div></div>';
        }
    }
    if($need_sorting)
    {
        $hour= date("H",time());
        /*    if ($time->label==$bookedTime)
        {
            $printer = '<div class="col-xs-4"><div class="btn btn-
danger btn-block btnTime" disabled data-toggle="modal" data-
target="#bookModal" value="'.{$time->price.'}">'.{$time->
>label.'</div></div>';
        }*/
        if($hour>($time->maxTime)-4)
        {
            $printer = false;
        }
    }
    echo $printer;
}
@endphp
</div>
</div>

<div class="modal fade" role="dialog" id="bookModal" data-
backdrop="static">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header" style="background-
color: grey; color:whitesmoke;">

```

```

        <button type="button" class="close" data-
dismiss="modal">&times;</button>
        <h3 class="modal-title">Booking</h3>
    </div>
    <div class="modal-body">
        <div class="form">
            <form class="form-
horizontal" action="/bookNow" method="post">
                @csrf
                <div class="form-group">
                    <label class="control-label col-sm-3" >Date:</label>
                    <div class="col-sm-9">
                        <input type="text" name="popupDate" class="form-
control" id="popupDate" value={{$_REQUEST["datecal"]}} readonly="reado
nly">
                    </div>
                </div>
                <div class="form-group">
                    <label class="control-label col-sm-3" >Time:</label>
                    <div class="col-sm-9">
                        <input type="text" name="time" class="form-
control" id="time" readonly="readonly">
                    </div>
                </div>
                <div class="form-group">
                    <label class="control-label col-sm-
3" >Phone:</label>
                    <div class="col-sm-9">
                        <input type="tel" name="phone" required class="fo
rm-control" id="phone" placeholder="9800000000" pattern="[0-9]{3}[0-
9]{3}[0-9]{4}" >
                    </div>
                </div>
                <div class="form-group">
                    <label class="control-label col-sm-
3" >Price:</label>
                    <div class="col-sm-9">
                        <input type="text" name="price" class="form-
control" id="price" readonly="readonly">

```

[illegible]

```

        $("#price").val($(this).attr("value"));
    });
});
$(document).ready(function(){
    $("#book-btn").click(function(){
        $('#bookModal').modal('show');
    });
});

function showDiv()
{
    document.getElementById('availTime').style.display = "block";
}

</script>

@endsection

```

8.2.2 Sample Code of the back-end

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Bookings;
use Cartalyst\Sentinel\Laravel\Facades\Sentinel;
use App\availableTime;
use Illuminate\Support\Facades\Hash;

class ClientController extends Controller
{
    public function viewClientDashboard()
    {
        return view('client.clientDashboard');
    }

    public function bookNow()

```



```
{
    return view('client.bookNow');
}

public function postBookNow(Request $request)
{
    $mail = $request->get('hiddenMail');
    $mail;
    $booking=new Bookings;
    $booking->email=$mail;
    $booking->date=$request->popupDate;
    $booking->time=$request->time;
    $booking->phone=$request->phone;
    $booking->price=$request->price;
    $booking->save();
    return redirect('/myBooking');
}

public function myBooking()
{
    $books=Bookings::where('email',Sentinel::getUser()->email)->get();
    return view('client.myBooking',compact('books') );
}

public function viewTime(Request $request)
{
    $bookedTime='';
    $availableTime=availableTime::all();
    $booking=Bookings::where('date',$_REQUEST["datecal"])->get();
    return view('client.availableTime',compact('booking','availableTime'));
}

public function deleteMyBooking($id)
{
    $deletebook=Bookings::find($id);
    $deletebook->delete();
    return redirect('/myBooking');
}
```

```
public function changePassword()
{
    return view('client.changePassword');
}

public function postChangePassword(Request $request)
{
    $this->validation($request);
    $current_password=Sentinel::getUser()->password;
    if(Hash::check($request['current-password'], $current_password))
    {
        $user_id = Sentinel::getUser()->id;
        $obj_user = Sentinel::findById($user_id);
        $obj_user->password = Hash::make($request['password']);
        $obj_user->save();
        $success = 'Congratulation Password Successfully Changed';
        return view('client.changePassword',compact('success'));
    }
    else
    {
        $error = 'Please enter correct current password';
        return view('client.changePassword',compact('error'));
    }
}

public function validation( $request)
{
    $request->validate([
        'current-password' => 'required',
        'password' => 'required|confirmed|max:255',
    ]);
}
}
```

8.3 Appendix-C: Designs

8.3.1 Gantt Chart



Figure 94: Gantt Chart

8.4 Appendix-D: Screenshots of the system



