# **SOFTWARE REQUIREMENT SPECIFICATIONS**

# **Baseball League Management System**

**BATCH NO: B1DA02**

TEAM MEMBERS

|  |  |
| --- | --- |
| Ranjith V | Kommineni Madhuri |
| M Dakshnamoorthy | Kommuri Yeshwanth |
| Shaik Shajia Farheen | Gopu Mallika |
| Kesani Manasa | Rathamsetty Haritha |
| Teela Lavanya | Abdul Rahaman Mohammad |

**Reviewed By**

Kapil Yadav, B usiness Analyst

**Under the guidance of Mentor**

Yuvapriya Ponnusamay

**ABSTRACT**

Baseball is one of the most loved games.It is not simply a sport for people.It is much more than that, it’s their passion.This project “BASEBALL LEAGUE MANAGEMENT SYTSEM” is built to help customers manage the baseball league. There are two users in this system. They are organizer and admin. This system helps organizers to choose venues for the event, select referees, Teams and Addons. The organizer can also edit or cancel the event. The admin can add/view/delete/edit venues, schedule, referees and teams. This application is always available to the customers which makes it user-friendly application. This application has multi-factor authentication for the sign in process which makes it more secure. The existing users sign in using the registered email and password and the new users are allowed to register. This application helps the baseball organizer to conduct match with minimum effort. This is a web application that is compatible in all latest browsers. It provides time efficient way for conducting events.

**INTRODUCTION**

**Purpose of the project:**

A Baseball League Management System is an application in which people (customers) are being provided with the option of maintaining a Baseball Team. The maintenance of a baseball team in real life is not an easy task. It requires a lot of manpower. Data maintained can sometimes be incorrect. To overcome this we store the data securely in a database which can later be accessed, updated or deleted. The interface makes the user easily manage the baseball team they want to Maintain. This project aims at making a virtual platform that helps to manage a baseball team with interactive user interface.

**Problem statement:**

Baseball is a bat-and-ball game that two opposing teams play. Originally, the organizers manually chooses the venues, teams, referees for the event. The organizers schedules the event without any involvement of computer system and the software. The players or the users manually selects the venues and reserves the venue according to the schedule that is maintained by the organizer. All this process is time consuming and not so organized. The organizer might miss the documents and may be unhealthy so couldn’t deal with the matches. So this totally makes the management confused about the scheduled matches and a new organizer couldn’t understand the ex-organizer’s scheduling. So, this makes the process of conducting baseball matches so difficult. The proposed system could help reduce all these problems.

## 

**Proposed System:**

This proposed system helps the user maintain the baseball league. It allows two kinds of users. They are admin and organizer. There is a multi-factor authentication to login into the system. The existing user is allowed to log in to the system while the new user is asked to register with the email. The admin and organizer login data is collected separately and given the privileges accordingly. The admin can add/view/delete/edit venues, teams, referees and addons. The organizer chooses the venue and will be able to edit or cancel the venue according to the situation. This application helps user by saving the time. The organizer can simply login to the application with his email and password and simply select the venue for the event. Even if there is a new admin he'll be well aware about the system because the system is well understandable even for a new user. The user can log out of the system using the logout button.

**Users of the system:**

* + - Admin
    - Organizer(user)

**Advantages of the system:**

* Time management: The organizer’s time lost during the travelling to and fro the company to make reservation has been solved. Thus creating ample time for the customer to manage and schedule events.
* Backup of file: Customers will have a data backup of their events stored to the database for future reference.
* Multi-factor authentication: Information is secured because no invalid email is allowed and the valid email is also verified to check if it is being accessed by the respective user.
* Sensitive data is categorized and is stored in a secure manner.
* Secure connection for transmission of the data.
* Peak load performance during festival days and national holidays.
* Always available and is compatible for all web browsers.

**REQUIREMENTS**

**SOFTWARE REQUIREMENTS:**

* **Front End**
  + Angular 10+ / React 16+
  + Bootstrap
  + Material Design
* **Server side**
  + .net Web API / Node
* **Database**
  + MySQL / Oracle / MSSQL

**HARDWARE REQUIREMENTS:**

* **Hard Disk** : 256 GB and above.
* **RAM :** 4GB and above
* **Processor :** i3 and above

**FUNCTIONAL REQUIREMENTS:**

* **Admin:**
  + - * + Can add/view/edit/delete venues.
        + Can add/view/edit/delete schedule.
        + Can add/view/edit/delete match referees.
        + Can add/view/edit/delete teams.
* **Organizer:**
  + - * + Can choose venues for the event.
        + Select referees, teams, addons.
        + Can edit and cancel the event.
* **Add-on features:**
  + - * + Email integration for intimating new personalized offers for customers.
        + Multi-factor authentication for sign-in process.
        + Pay ment Gateway.

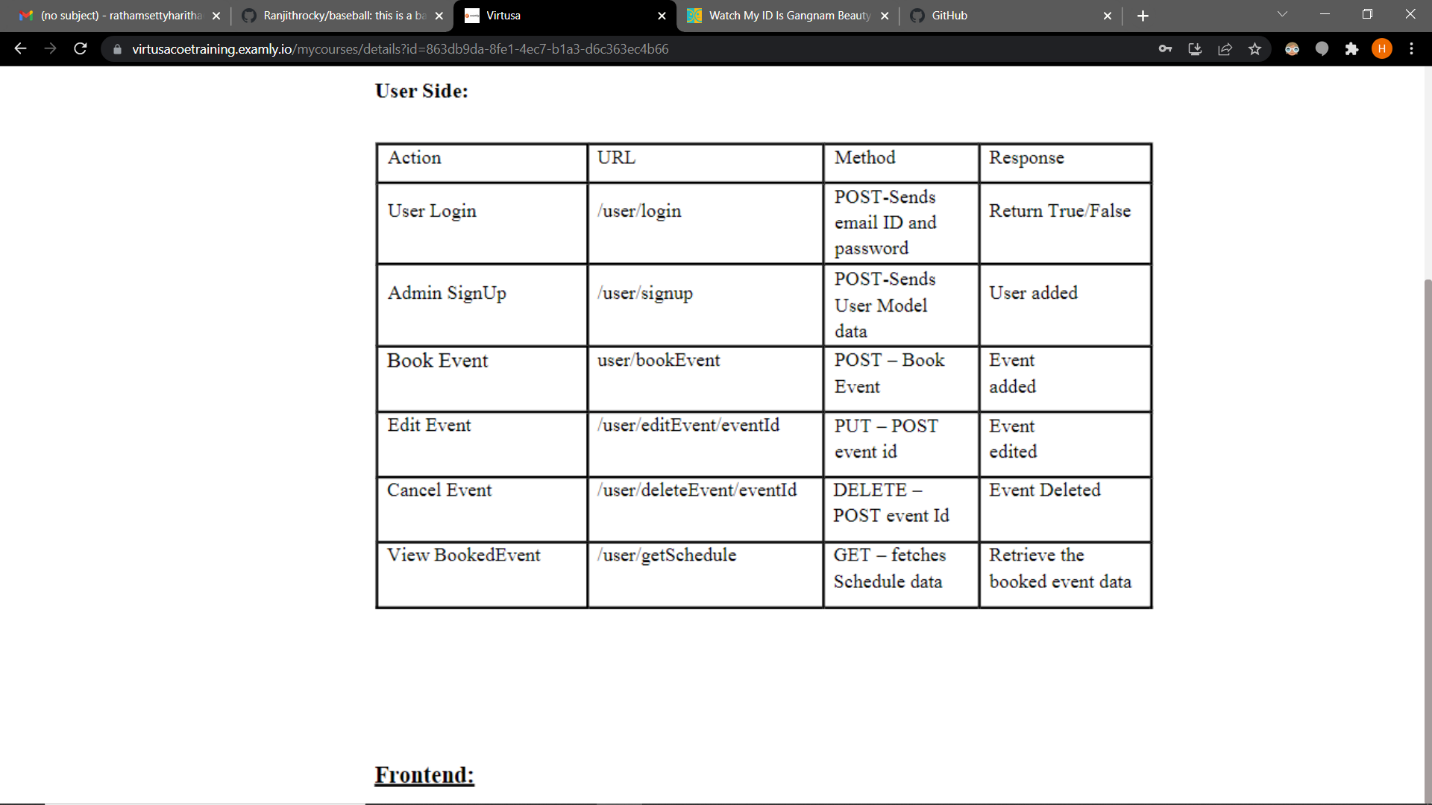
**NON-FUNCTIONAL REQUIREMENTS:**

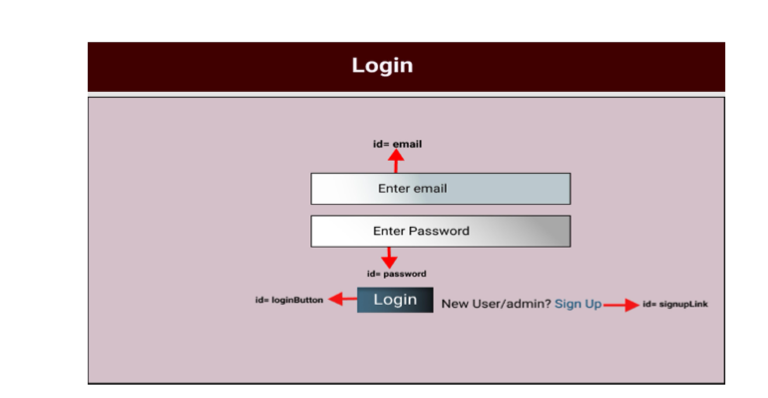
* Security:
* App Platform –Username/Password-Based Credentials.
* Sensitive data has to be categorized and stored in a secure manner.
* Secure connection for transmission of my data.
* Performance:
* Peak Load Performance
* Admin application < 2sec
* Non-Peak Load Performance
* Appointment Application<2sec
* Admin Application <2sec
* Availability:
* 99.99%Availability
* Standard Features:
* Scalability
* Maintainability
* Usability
* Availability
* Failover
* Logging & Auditing:
* The system should support logging (app/web/DB) &auditing at all levels.
* Monitoring:
* Should be able to monitor via as-is enterprise monitoring tools.
* Cloud:
* The solution should be made Cloud-ready and should have a minimum impact when moving away to cloud infrastructure.
* Browser Compatible:
* All latest browsers.

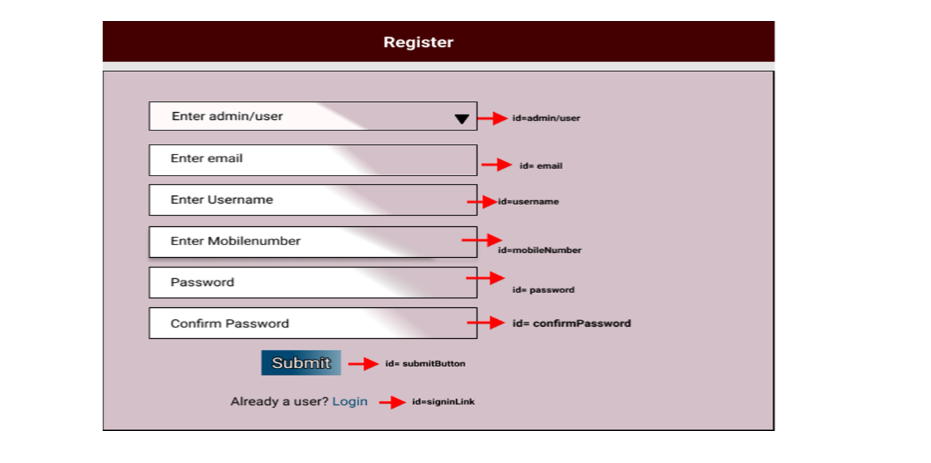
**FRONT END**

**ORGANIZER:**

* **View website Description :** On the Home Page the user can view the website description and has access to the following controls:
  + - * **About**: User can know more details about the company.
      * **Register** : New users can register
      * **Login** : Registered users can login
      * **Logout**: User can logout
      * **Contact Us**: Users can contact the support team and ask their queries.

****

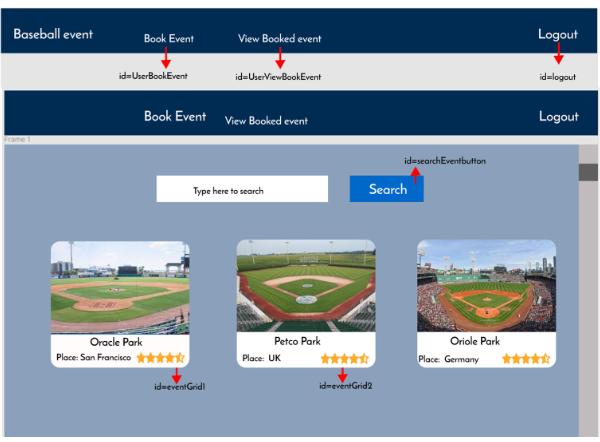
* **User Login**: Registered users can login by providing email and password. By clicking on login control, the user credentials are validated and if they are correct, he is redirected to the Book event page. If the credentials are incorrect, the user is redirected back to the login page and is shown a message Invalid username/Password. If new user, he can click on sign up to get redirected to the Registration page.
* **User Registration:** The users can register by providing details like Email, Username, and Mobile number. They can set a new password and submit the details for the creation of new account. After submitting, the user is redirected to login page.



* **Home Page**: We design a home page component that has the navigation bar and lists all the available Services as grid elements with appropriate filter options.There is a single operation that a organizer can do which is to book the venue for event.By clicking the book event in the navigation bar it will redirect to the page where we need to fill the details.
* Ids: Refer to the screenshot below for the id details.
* Routing URL: http://localhost:8080/user/homepage

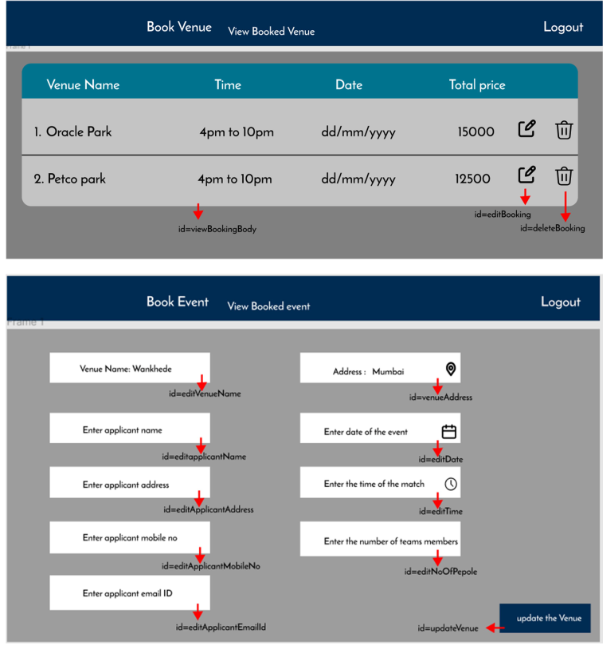
****

* **Book Event:** After succesfully logging into the system the designer will be able to book the event by designing a Booking component.In this booking page the venues for the event will be visible in the page in the of grids which contains the name of and place of the venue and the rating.
* Ids: Refer to the screenshot below for the id details.
* Routing URL: http://localhost:8080/user/bookEvent



* **View Booking:** If the user wanted to view the booking we designed a viewbooking component that helps user to view his booking.In this view booked venue it contains all the bookings in a table.For each booked venue there are some options which are delete and edit.The delete symbol deletes the booking and the edit option edits the details.

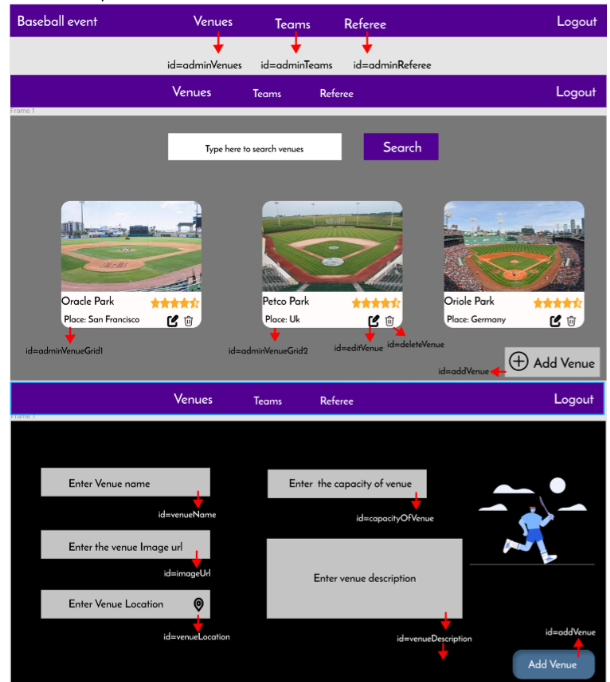
* Ids: Refer to the screenshot below for the id details.
* Routing URL:
* http://localhost:8080/user/editEvent/{id}
* http://localhost:8080/user/deleteEvent/{id}

****

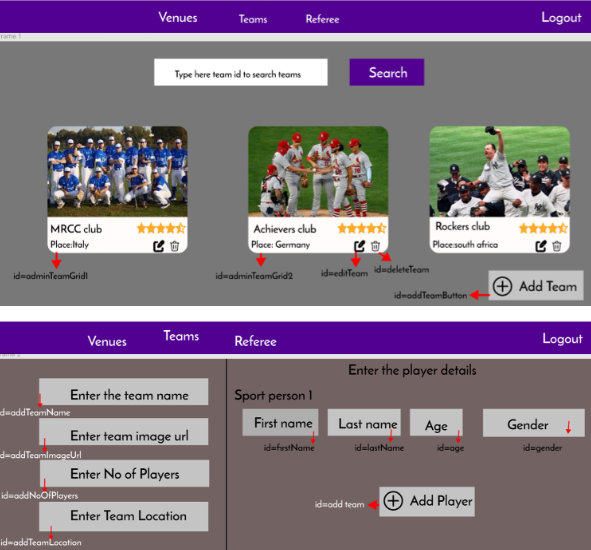
* **View Profile Details:** Users can view their profile Details like Name, Contact, Email id etc.
* **Edit Profile Details:** Users can edit their profile details like Name, Contact, Email id etc.
* **Customer complaints and feedback:** The system will allow customers to send complaints and feedback.
* **Logout:** The user can logout of the application.

**ADMIN:**

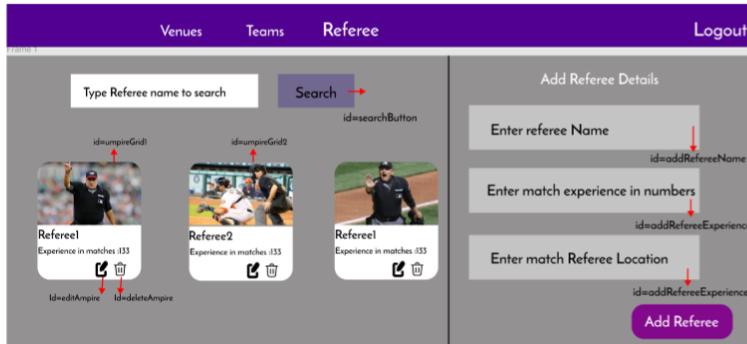
* **Add Venue**: We design an addVenue component so that the admin could add a venue to the application. Firstly the admin click on the venues button in the navigation bar. It shows all the venues existing and there will be plus sign that allows us to add the venue. We can even view, delete, and edit the venue.
* Ids: Refer to the screenshot below for the id details.
* Routing Url:
* http://localhost:8080/admin/addVenue
* http://localhost:8080/admin/getVenue
* http://localhost:8080/admin/editVenue /{id}
* http://localhost:8080/admin/deleteVenue/{id}



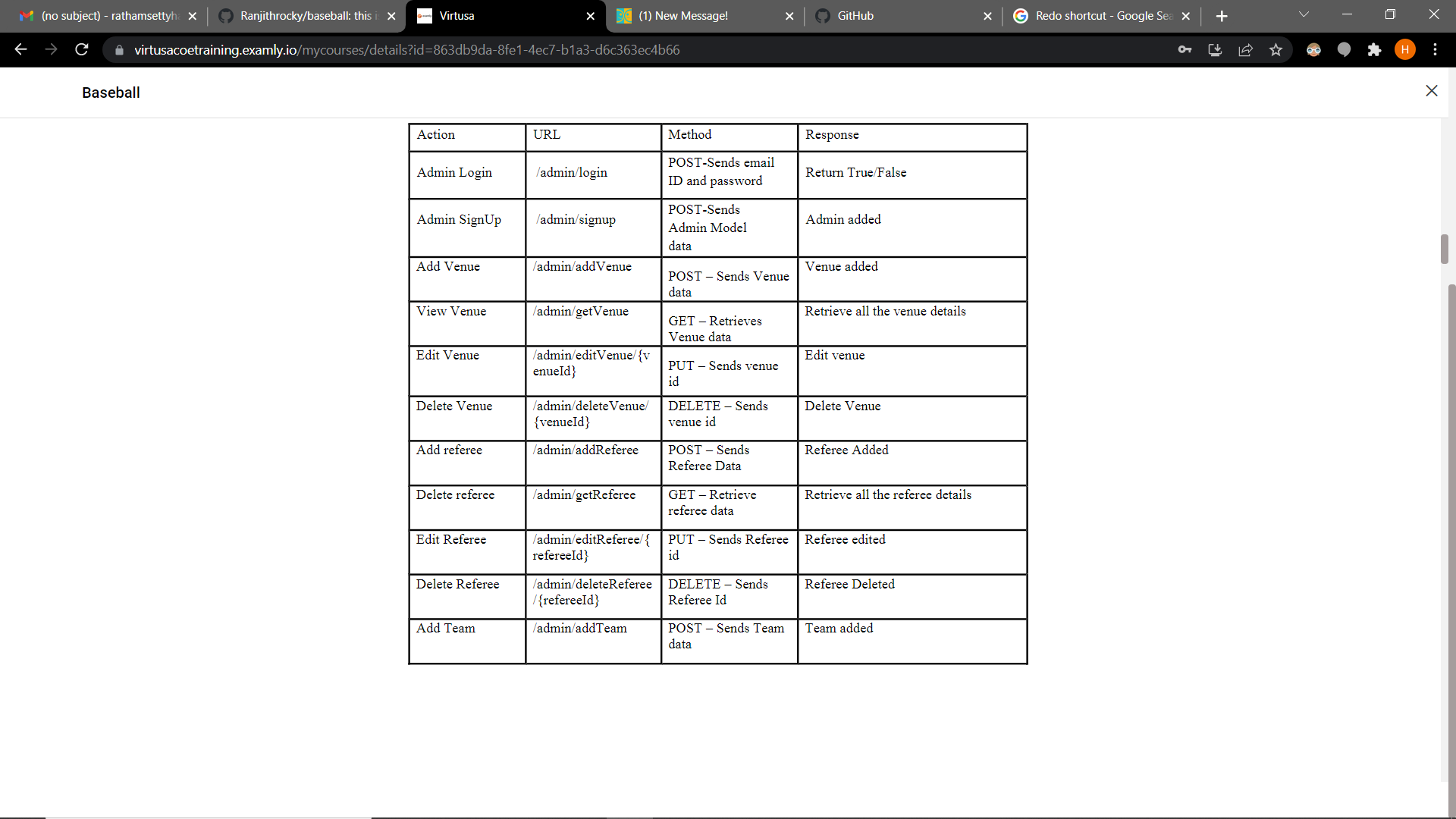
* **Add Team :** We design an addTeam component so that the admin could add a team to the application. Firstly the admin click on the teams button in the navigation bar. It shows all the teams existing and there will be plus sign that allows us to add the team. We can even view, delete, and edit the team.
* Ids: Refer to the screenshot below for the id details
* Routing URL:
  + http://localhost:8080/admin/addTeam
  + http://localhost:8080/admin/getTeam
  + http://localhost:8080/admin/editTeam/{id}
  + http://localhost:8080/admin/deleteTeam/{id}

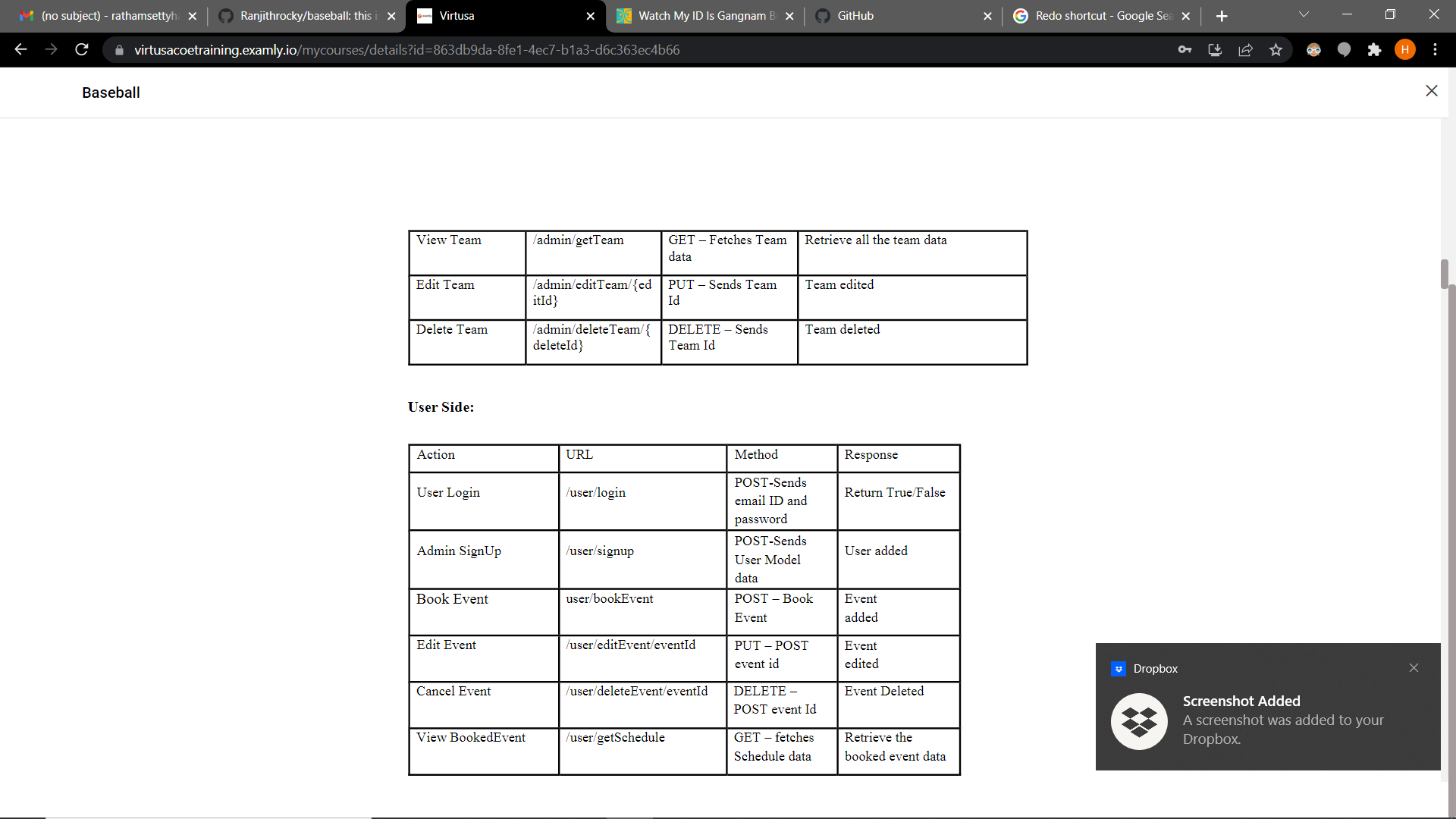
****

* **Add Referee:** We design an addReferee component so that the admin could add a referee to the application. Firstly the admin click on the referee button in the navigation bar. It shows all the referees existing and there will be plus sign that allows us to add the referee. We can even view, delete, and edit the referee.
* Ids: Refer to the screenshot below for the id details
* Routing URL:
  + http://localhost:8080/admin/addReferee
  + http://localhost:8080/admin/getReferee
  + http://localhost:8080/admin/editReferee/{id}
  + http://localhost:8080/admin/deleteReferee/{id}

****

* **Logout:**  The admin can logout of the application

****

****

**BACK END**

**Class and method description:**

**MODEL LAYER**

* **User Model:** This model layer stores all the user information i.e., email, password, Name, phone number, Role.
  + **Attributes:**
    - email: String
    - password: String
    - username: String
    - mobileNumber : String
    - userRole: String
* **Login Model:** This class stores all the login information.
  + **Attributes:**
    - email: String
    - password: String
* **Admin Model:** This class contains all the admin details stored in it.
  + **Attributes:**
    - email: String
    - password: String
    - mobileNumber: String
    - userRole: String
* **Venue Model:** This class conatins all the venue details stored in it.
  + **Attributes:**
    - venueId: int
    - venueName: String
    - venueImageURL: String
    - venueDescription: String
    - venueLocation: String
* **Referee Model:**This class contains all the referee details stroed in it.
  + **Attributes:**
    - refereeId: int
    - refereeName: String
    - noOfMatches: int
* **Team Model:**This class contains all the team details stored in it.
  + **Attributes:**
    - teamId: int
    - teamName: String
    - team Description: String
    - List<player details>: Array of objects
* **Event Model:**This class contains all the event details stored in it.
  + **Attribues :**
    - eventId: int
    - eventName: String
    - applicantName: String
    - applicantAddress: String
    - applicantEmail: String
    - applicantMobile: String
    - eventAddress: String
    - eventFromDate: Date
    - eventToDate: Date
    - addonId: int

**CONTROLLER LAYER:**

* **AuthController:** This class control the user /admin signup and signin
  + **Methods:**
    - **isUserPresent(LoginModel data):** This method helps to check whether the user present or not and check the email and password are correct and return the boolean value.
    - **isAdminPresent(LoginModel data):** This method helps to check whether the admin present or not and check the email and password are correct and return the boolean value.
    - **saveUser(UserModel user):** This method helps to save the user data in the database.
    - **saveAdmin(UserModel user):** This method helps to save the admin data in the database.
* **UserController**: This class helps to get the users
  + Methods:
    - **addUser(UserModel data):** This method adds user to the application.
    - **getUser(String UserID):** This method is used to get the list of users.
    - **editUser(String UserID):** This method helps to edit the users.
    - **deleteUser(String UserID**): This method helps to delete the users.
* **VenueController**: This class helps to add/edit/delete/view venue.
  + Methods:
    - **addVenue(VenueModel data):** This method adds venue.
    - **getVenue(int venueId):** This method retrieves the venue.
    - **editVenue(int venueId):** This method edits the venue.
    - **viewVenue(int venueId):** This method deletes the venue.
* **RefereeController**: This class helps to add/edit/delete/view referee.
  + Methods:
    - **addReferee(RefereeModel data):** This method adds referee.
    - **getReferee(int refereeId):** This method retrieves the referee.
    - **editReferee(int refereeId):** This method edits the referee.
    - **viewReferee(int refereeId):** This method deletes the referee.
* **AddOnController**: (optional)
  + Methods:
    - **addAddOn(AddOnModel data):** This method adds add-on.
    - **getAddOn(int addOnId):** This method retrieves the add-on.
    - **editAddOn(int addOnId):** This method edits the add-on.
    - **viewAddOn(int addOnId):** This method deletes the add-on.
* **TeamController**: This class helps to add/edit/delete/view team.
  + Methods:
    - **addTeam(TeamModel data):** This method adds team.
    - **getTeam(int teamId):** This method retrieves the team.
    - **editTeam(int teamId):** This method edits the team.
    - **viewTeam(int teamId):** This method deletes the team.
* **EventController**: This class helps to book/edit/cancel/view events.
  + Methods:
    - **BookEvent(EventModel data):** This method books event.
    - **viewEvent(int EventId):** This method shows the booked event.
    - **editEvent(int eventId):** This method edits the event.
    - **deleteEvent(int eventId):** This method deletes the event.

**USE CASE DIAGRAM**:

