1. INTRODUCTION

The Sports Resource Booking Application has been developed in order to enable the students to book sports resources. The administrator can keep a tab on the availability of the resources along with streamlining the fine process. This application overcomes the problems prevailing in the existing manual system. It also makes it extremely simple for the students to book a resource or to check the availability of the resources, view dues if any, as well as history of bookings and for the administrator to issue resources, impose fines and view the history of bookings. The API developed for this application is deployed on Heroku and is ready to respond to the requests made by the application with expected outputs by accessing the database.

1.1 Problem Definition

The project includes two main modules:

USER

- Student Login
- Home page shows all the available resources and resources that he/she booked
- Book a resource which will be reserved for him for 20mins only
- Cancel booking
- History of bookings
- Limit to only 1 booking
- Resources can be issued from 12:00PM to 4:00PM so booking can start from 11:40AM to 3:00PM - Blocked users cannot book any resources until they pay the due.

ADMIN

- Admin login
- Add resources
- Delete resources
- (Bookings decrement of available resources should be done in backend)
- Issue resources and update the booking

- (All resources should be returned by 4:20PM otherwise fine is charged and block them automatically)
- Update the booking once the user returns the resources.
- Receive fine payments physically and unblock the user

1.2 Scope of the project

This application can be used to reduce the problems faced by the manual system. The administrator can issue and reject a booking in one click and it even helps the admin to keep a tab on the Booking History and also the Blocked list of users .The administrator can also add or delete the resources .It makes it easy for the administrator to keep track of all the resources and bookings.The mobile application makes it so simple for the users to book a resource and to check the history of their previous bookings.

2. TECHNOLOGY STACK

2.1 Front-End Design:

2.1.1 Web Application: HTML, CSS, BOOTSTRAP, JavaScript

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Bootstrap is a free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

2.1.2 Android Application: JAVA, XML

Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications.The Java syntax is similar to C++, but is strictly an object-oriented programming language.Unlike Windows executables (.EXE files) or Macintosh applications (.APP files), Java programs are not run directly by the operating system. Instead, Java programs are interpreted by the Java Virtual Machine, or JVM, which runs on multiple platforms.
XML code, a formal recommendation from the World Wide Web Consortium (W3C), is similar to Hypertext Markup Language (HTML). Both XML and HTML contain markup

symbols to describe page or file contents. The basic building block of an XML document is an element, defined by tags. An element has a beginning and an ending tag. All elements in an XML document are contained in an outermost element known as the root element. XML can also support nested elements, or elements within elements. This ability allows XML to support hierarchical structures. Element names describe the content of the element, and the structure describes the relationship between the elements.

2.2 Back-End:

2.2.1 Web Application: Django

Django is a free and open source web application framework written in Python. Django has its own naming system for all functions and components (e.g., HTTP responses are called "views"). The Django framework uses the principles of rapid development, which means developers can do more than one iteration at a time without starting the whole schedule from scratch.With Django, you can tackle projects of any size and capacity, whether it's a simple website or a high-load web application.

2.2.2 Database: MySQL

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language. It is based on the structure query language (SQL), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD, DROP, INSERT, and UPDATE can be used with MySQL.

2.2.3 RESTful API: Flask

Flask is a web application framework written in Python. Flask is based on WSGI(Web Server Gateway Interface) toolkit and Jinja2 template engine. WSGI is basically a protocol defined so that Python applications can communicate with a web-server and thus be used as web-application outside of CGI.

3.LITERATURE SURVEY

3.1 Introduction

Sports Resources Management System is a web and an android app. The admin can add and delete resources on the web and manage the users in accepting and rejecting resources.The users can book the resources from the android app and collect them from the office within 20 min and even cancel the request.

3.2 Existing System

In the existing system, the users have to visit the sports office manually and collect the resources.And also the users cannot reserve for a resource from wherever he/she is.Even though the user is blocked, he can go to the office and make a request for the resource.The admin has to manually check the user if he/she is in the blocked list and issue the resource.

3.3 Proposed System

In the proposed system, the users can reserve and book the resource from wherever they are and collect it within 20min.In this system,if a user is blocked he cannot book the new resource until he clears the fine.The user can check the fine amount to be paid by him in the android app.

3.4 Software Requirement Specification

3.4.1 User Requirement

There is an admin and the user i.e.,student.The minimum requirement of the user is that he/she must understand basic English and must know how to book a resource, check the fine and collect from the sports office. The admin must know how to accept the users request, block the user, issue resources manually and collect fines from users and unblock users.

3.4.2 Software Requirement

- Any Web Browser
- Python 3 with Django Module
- 64 bit operating system.
- Emulator
- Android studio
- pip
- virtualenv
- Virtualenvwrapper

- Flask module

3.4.3 Hardware Requirement

- Modern Operating System:
    - Windows 7 or 10
    - Mac OS X 10.11 or higher, 64-bit
    - Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)
- x86 64-bit CPU (Intel / AMD architecture)
- Minimum 4 GB RAM
- Minimum 5 GB free disk space
- A 64 bit environment is required for Android 2.3xGingerbread and higher version.
- Atleast 250GB of free disk space to check out code and an extra 150GB to build it.
- Atleast 8GB of RAM/swap

4 . DESIGN OF THE PROPOSED SYSTEM

4.1 Entity Relationship diagram

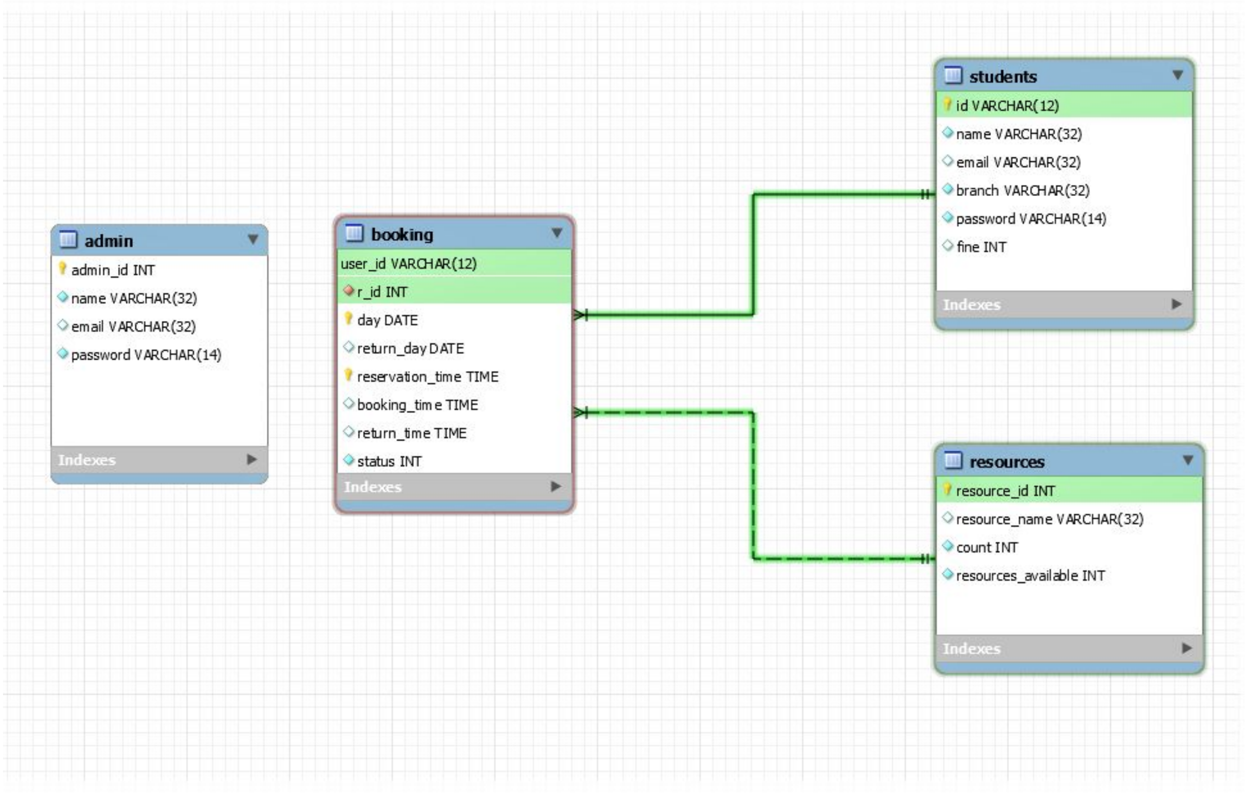
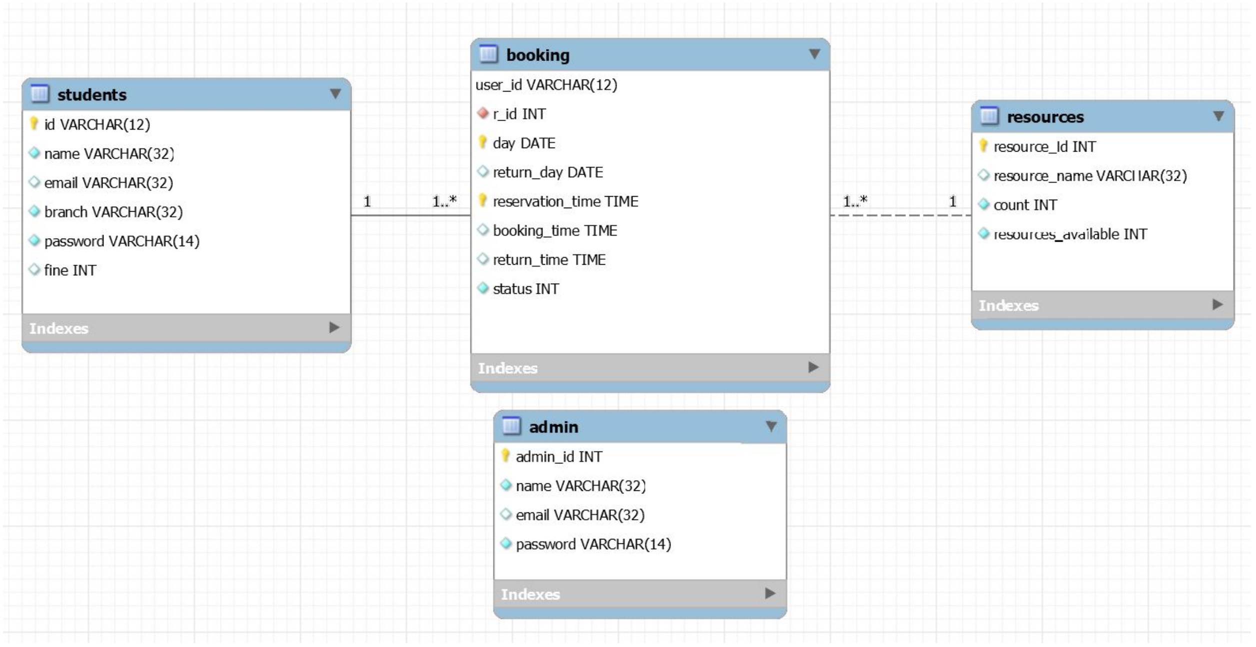


Figure 1: Entity Relation Diagram



4.2 UML Diagrams

4.2.1 Use Case diagram

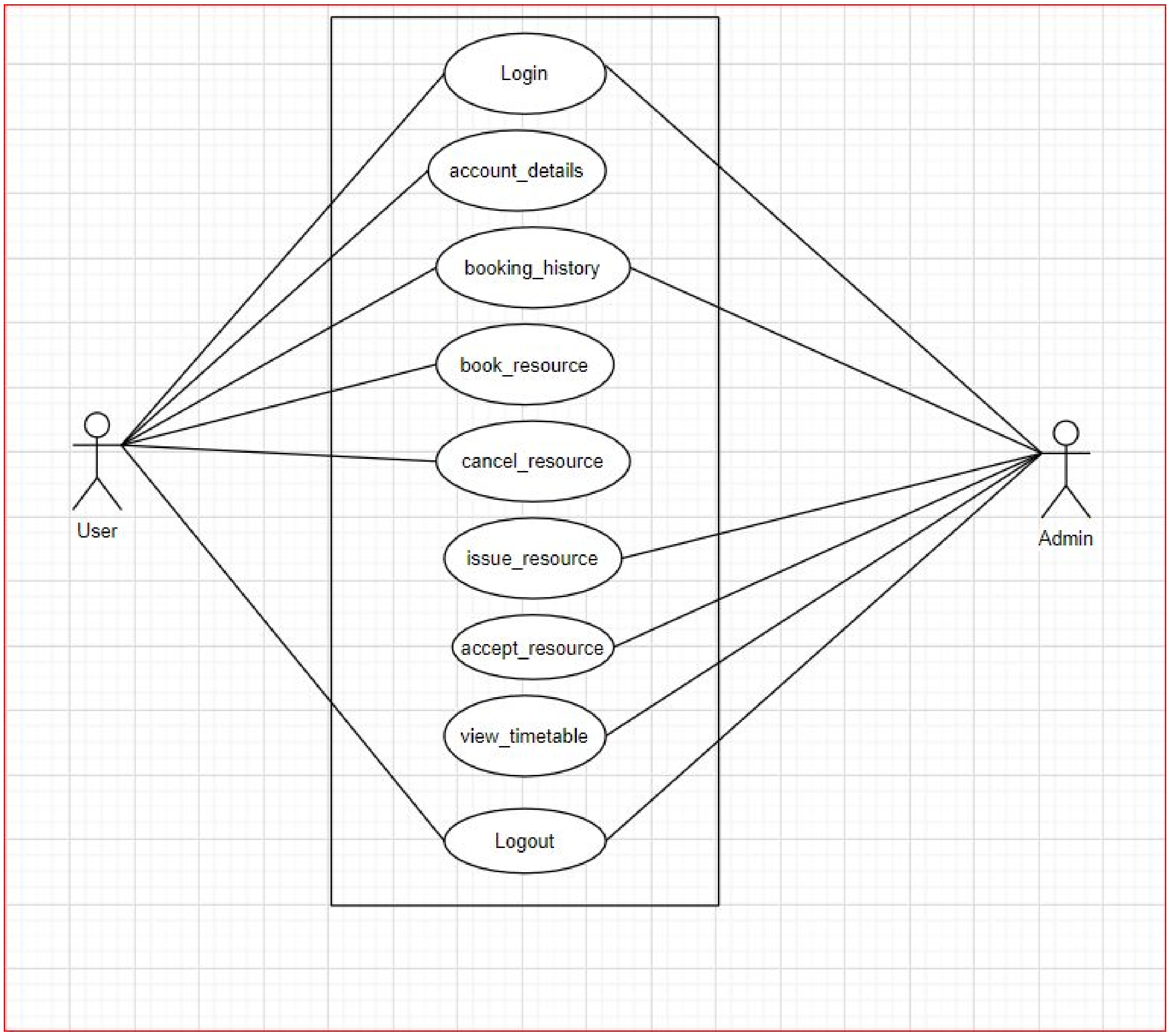
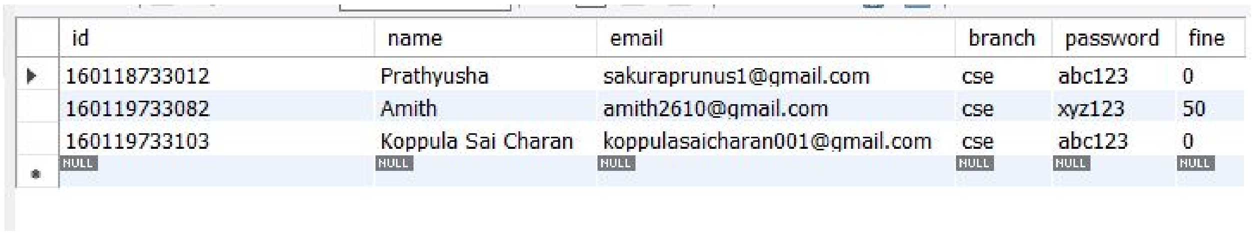


Figure 2: Use case Diagram

4.3 Database Design

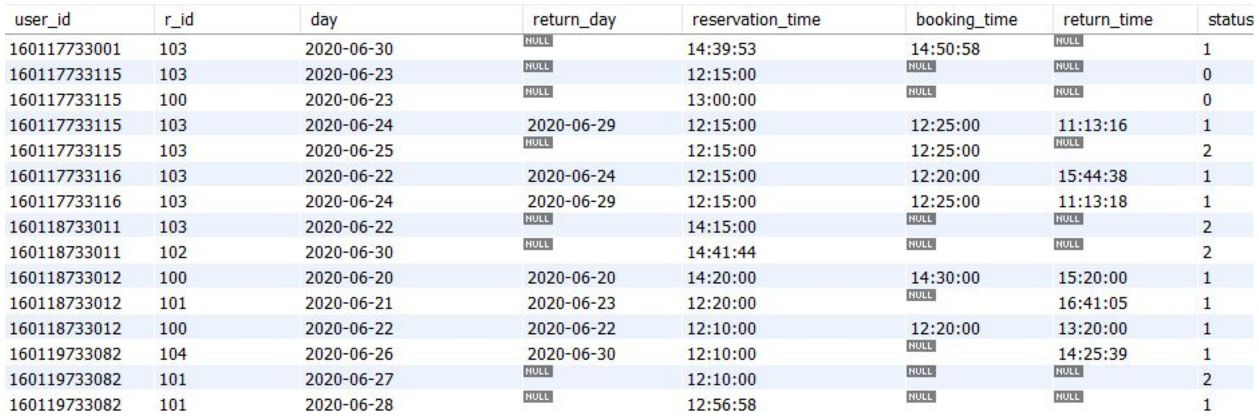
Students table



id	name	email	branch	password	fine
160118733012	Prathyusha	sakuraprunus1@gmail.com	cse	abc123	0
160119733082	Amith	amith2610@gmail.com	cse	xyz123	50
160119733103	Koppula Sai Charan	koppulasaicharan001@gmail.com	cse	abc123	0
NULL	NULL	NULL	NULL	NULL	NULL

Figure 3: Students table

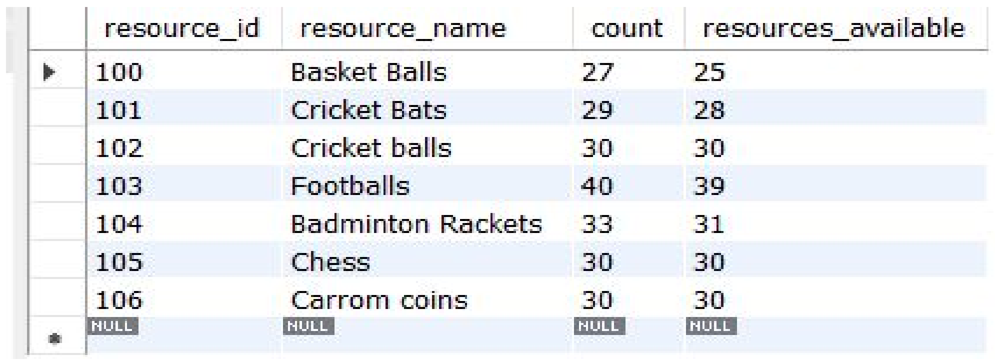
Booking table



user_id	r_id	day	return_day	reservation_time	booking_time	return_time	status
160117733001	103	2020-06-30	NULL	14:39:53	14:50:58	NULL	1
160117733115	103	2020-06-23	NULL	12:15:00	NULL	NULL	0
160117733115	100	2020-06-23	NULL	13:00:00	NULL	NULL	0
160117733115	103	2020-06-24	2020-06-29	12:15:00	12:25:00	11:13:16	1
160117733115	103	2020-06-25	NULL	12:15:00	12:25:00	NULL	2
160117733116	103	2020-06-22	2020-06-24	12:15:00	12:20:00	15:44:38	1
160117733116	103	2020-06-24	2020-06-29	12:15:00	12:25:00	11:13:18	1
160118733011	103	2020-06-22	NULL	14:15:00	NULL	NULL	2
160118733011	102	2020-06-30	NULL	14:41:44	NULL	NULL	2
160118733012	100	2020-06-20	2020-06-20	14:20:00	14:30:00	15:20:00	1
160118733012	101	2020-06-21	2020-06-23	12:20:00	NULL	16:41:05	1
160118733012	100	2020-06-22	2020-06-22	12:10:00	12:20:00	13:20:00	1
160119733082	104	2020-06-26	2020-06-30	12:10:00	NULL	14:25:39	1
160119733082	101	2020-06-27	NULL	12:10:00	NULL	NULL	2
160119733082	101	2020-06-28	NULL	12:56:58	NULL	NULL	1

Figure 4:Booking table

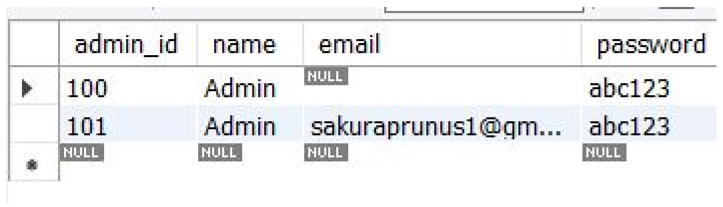
Resources table



resource_id	resource_name	count	resources_available
100	Basket Balls	27	25
101	Cricket Bats	29	28
102	Cricket balls	30	30
103	Footballs	40	39
104	Badminton Rackets	33	31
105	Chess	30	30
106	Carrom coins	30	30
NULL	NULL	NULL	NULL

Figure 5:Resources table

Admin table



admin_id	name	email	password
100	Admin	NULL	abc123
101	Admin	sakuraprunus1@gm...	abc123
NULL	NULL	NULL	NULL

Figure 6:Admin table

4.4 Module Description

There are two modules in this project.They are user and admin.

4.4.1 Features of User

User is a student who is going to use the Android Application "SportEasy". The app has the following features:

- Can view all the resources available in the sports block with its count in the app. This reduces the enquiry time at the issue counter.
- He can send a booking request with just one click so that he can collect it in 20mins from his booking time.
- User also has an option to cancel the booking request within 20 minutes of the booking time.

- User can view all his past booking requests with their status so that he can know what requests he made.

User has an option to set his password to a new one when he forgets his password or wants to change it.

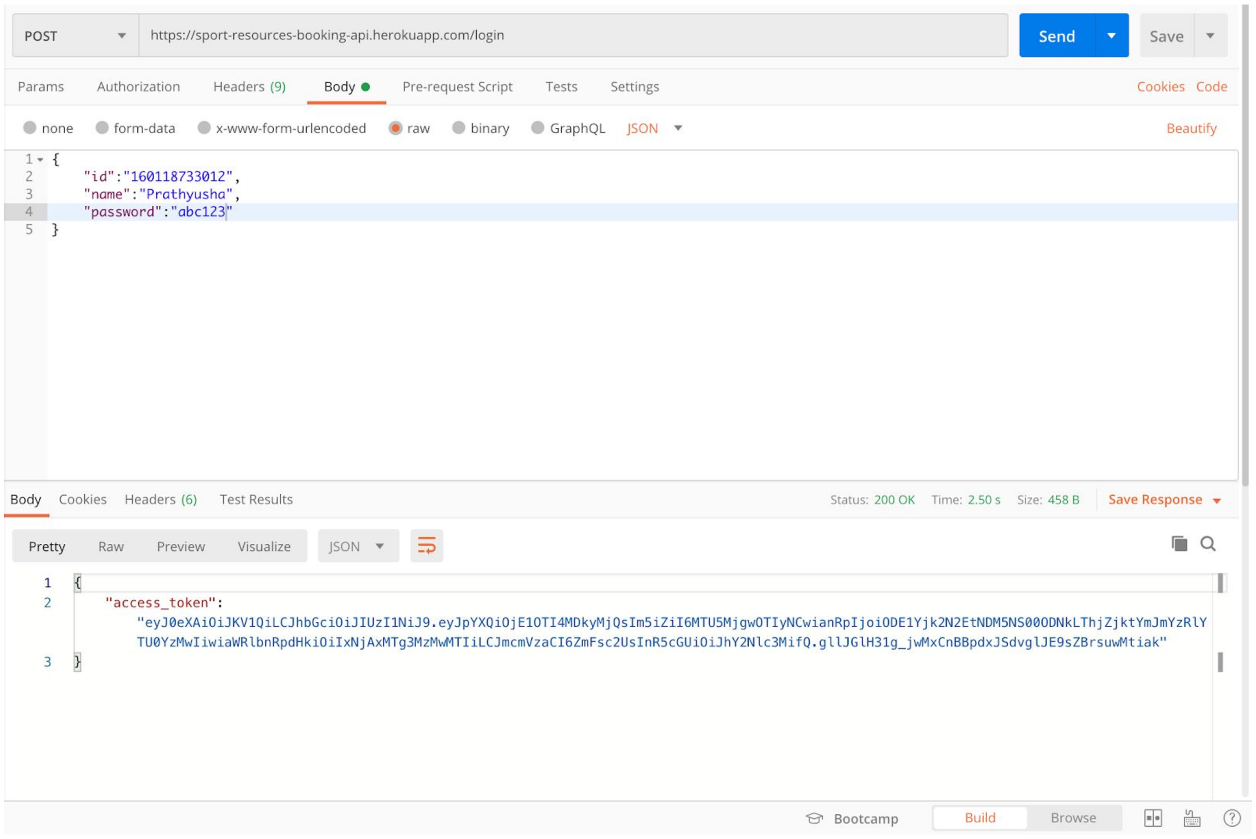
4.4.2 Features of the API

The main purpose of this Flask RESTful API for Sports resources booking Application is for students to be able to book sports resources and for the admins to keep tab on the availability of resources along with streamlining the fine process.

Link: https://sport-resources-booking-api.herokuapp.com/

Admin and the user can use the /adminLogin,/login endpoints respectively for login :

a) /login POST Request-takes a JSON object with ‘id’ , 'username' and 'password' and gives back JWT token if exists in Users table. The JWT token shall be used to access all the endpoints. For all the endpoints an Authorization Header should be included with value 'Bearer '.


POST https://sport-resources-booking-api.herokuapp.com/login

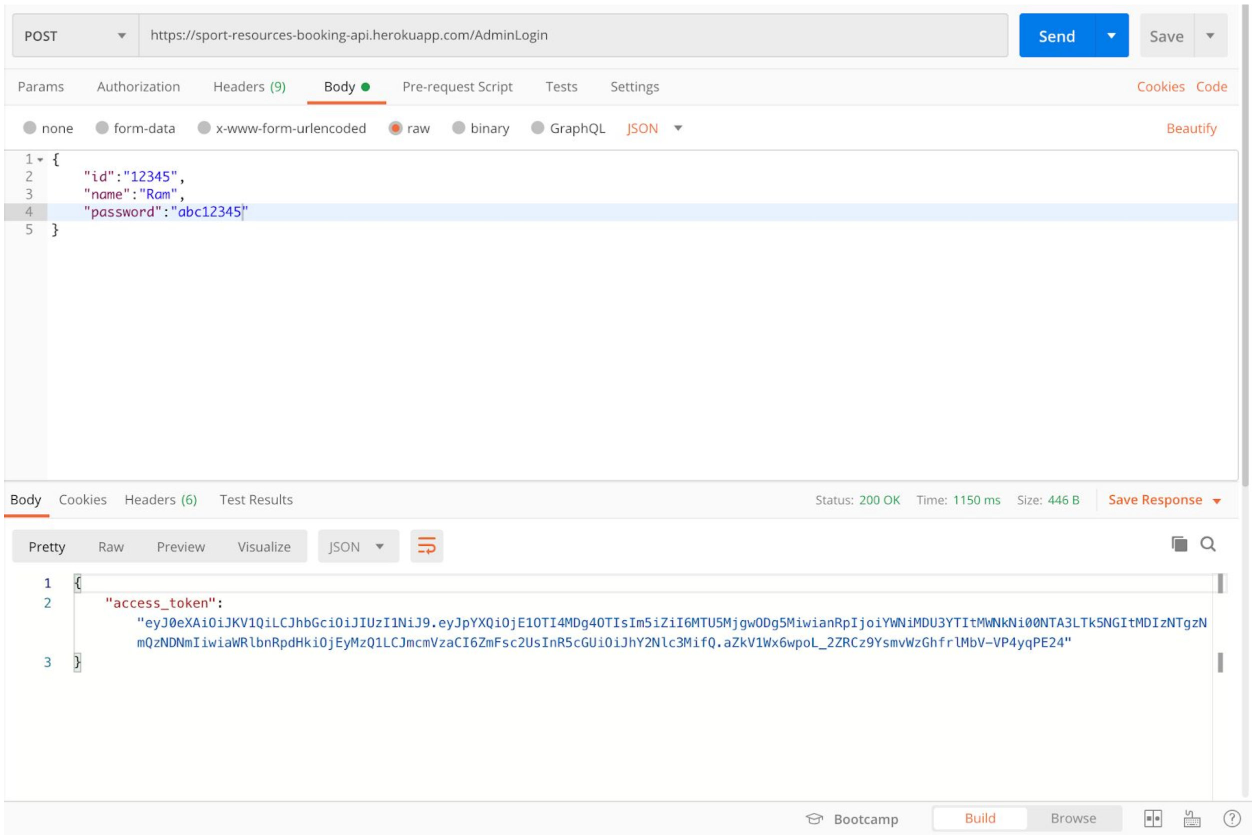
```
{
    "id":"160118733012",
    "name":"Prathyusha",
    "password":"abc123"
}
```

Status: 200 OK Time: 2.50 s Size: 458 B

```
{
    "access_token":
        "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1OTI4MDkyMjQsIm5iZiI6MTU5MjgwOTIyNCwianRpIjoiODE1Yjk2N2EtNDM5NS00ODNkLThjZjktYmJmYzRlYTU0YzMwIiwiaWRlbnRpdHkiOiIxNjAxMTg3MzMwMTIiLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MifQ.gllJGlH31g_jwMxCnBBpdxJSdvglJE9sZBrsuwMtiak"
}
```

Figure 7: Login endpoint

b) /adminLogin - POST Request -endpoint is only meant for the admins and momentarily there is one admin in the admin table who can access the web application. It also takes a JSON object with 'username' and 'password' and gives back a JWT token if it exists in the Admin table. The JWTtoken shall be used to access all the endpoints. For all the endpoints an Authorization Header should be included with value 'Bearer '.

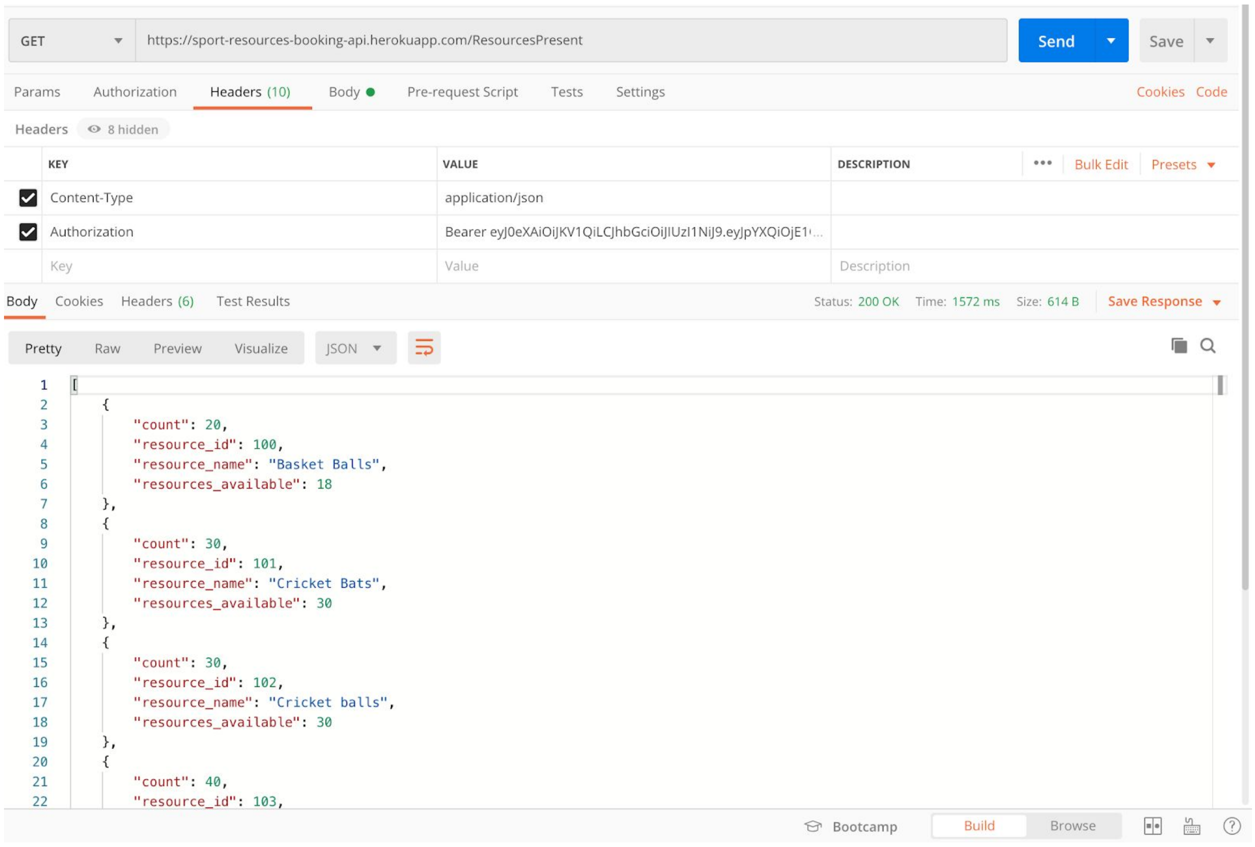


```
{
    "id":"12345",
    "name":"Ram",
    "password":"abc12345"
}
```

```
{
    "access_token":
        "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1OTI4MDg4OTIsIm5iZiI6MTU5MjgwODg5MiwianRpIjoiYWNiMDU3YTItMWNkNi00NTA3LTk5NGItMDIzNTgzN
        mQzNDNmIiwiaWRlbnRpdHkiOjEyMzQ1LCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MifQ.aZkV1Wx6wpoL_2ZRCz9YsmvWzGhfrlMbV-VP4yqPE24"
}
```

Figure 8: Admin login endpoint

/ResourcesPresent- GET request- from the resource table to give details of all the resources present .



https://sport-resources-booking-api.herokuapp.com/ResourcesPresent

KEY	VALUE	DESCRIPTION
Content-Type	application/json	
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1...	
Key	Value	Description

Status: 200 OK Time: 1572 ms Size: 614 B

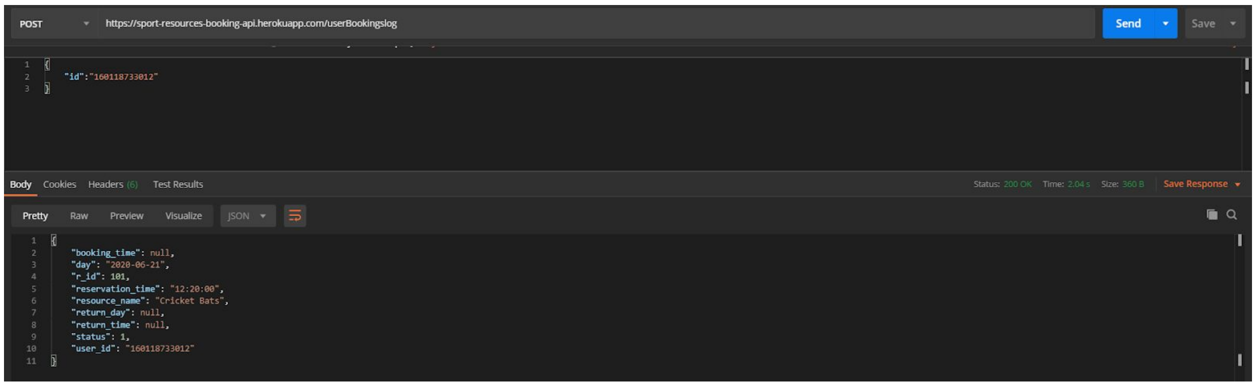
```
[
    {
        "count": 20,
        "resource_id": 100,
        "resource_name": "Basket Balls",
        "resources_available": 18
    },
    {
        "count": 30,
        "resource_id": 101,
        "resource_name": "Cricket Bats",
        "resources_available": 30
    },
    {
        "count": 30,
        "resource_id": 102,
        "resource_name": "Cricket balls",
        "resources_available": 30
    },
    {
        "count": 40,
        "resource_id": 103,
```

Figure 9: Resources endpoint

/AddExtraResource- The admin can only make changes after providing the access token. Updates the resources table and inserts a resource when a new resource is bought by the administration.

/DeleteResource- The admin can only make changes after providing the access token. Updates the resources table and deletes the resource specified by id .

/userBookingslog - It provides the details of the booking which is not yet returned by the user or which is currently booked by the user



```
POST https://sport-resources-booking-api.herokuapp.com/userBookingslog
```

```
{
    "id":"160118733012"
}
```

```
{
    "booking_time": null,
    "day": "2020-06-21",
    "r_id": 101,
    "reservation_time": "12:20:00",
    "resource_name": "Cricket Bats",
    "return_day": null,
    "return_time": null,
    "status": 1,
    "user_id": "160118733012"
}
```

Figure 10: User booking log

/userDue-GET request in order to receive the due of a student if any and the details of the resource which they didn't return

/cancelBooking- POST request to first check if the booking exists and then change it's status to unbooked/rejected and give a message to the user.

/resourceDetails- GET request to receive all the details of a given resource from the resource table with the help of an ID.

/incrementByOne-The admin can only make changes after providing the access token. Updates the resources table and increases the count value by 1 when a new resource is bought by the administration.

/incrementByValue- The admin can only make changes after providing the access token. Updates the resources table and increases the count value when a new resource is bought by the administration.

/decrementByOne- The admin can only make changes after providing the access token. Updates the resources table and decreases the count value by 1 when a new resource is bought by the administration.

/decrementByValue- GET request which the admin can only make after providing the access token. Updates the resources table and decreases the count value when a new resource is bought by the administration.

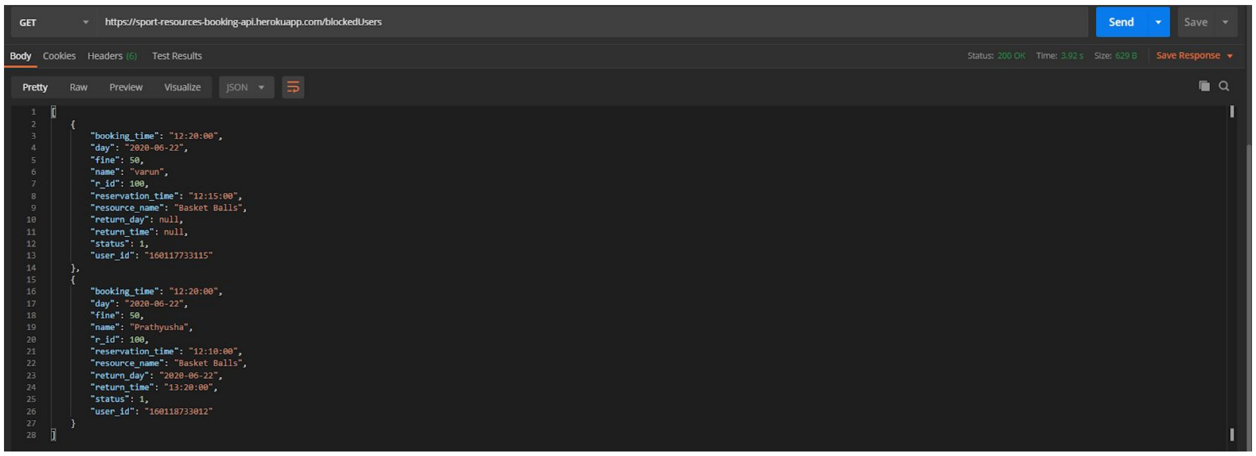
/issueResource- will update the booking_time and will set the status to one

/acceptResource- will update the resources available and will set the fine automatically if he fails to return after 4:20pm .

/bookingHistory-GET request to obtain the log of all bookings made from the BookingHistory1 view.

/issuedBookings-GET request to obtain the log of all bookings issued made on the current day from the BookingHistory1 view.

/blockedUsers- GET request to receive the users who have been blocked so as to not issue any more resources



```
[
    {
        "booking_time": "12:20:00",
        "day": "2020-06-22",
        "fine": 50,
        "name": "varun",
        "r_id": 100,
        "reservation_time": "12:15:00",
        "resource_name": "Basket Balls",
        "return_day": null,
        "return_time": null,
        "status": 1,
        "user_id": "160117733115"
    },
    {
        "booking_time": "12:20:00",
        "day": "2020-06-22",
        "fine": 50,
        "name": "Prathyusha",
        "r_id": 100,
        "reservation_time": "12:10:00",
        "resource_name": "Basket Balls",
        "return_day": "2020-06-22",
        "return_time": "13:20:00",
        "status": 1,
        "user_id": "160118733012"
    }
]
```

Figure 11: Blocked users

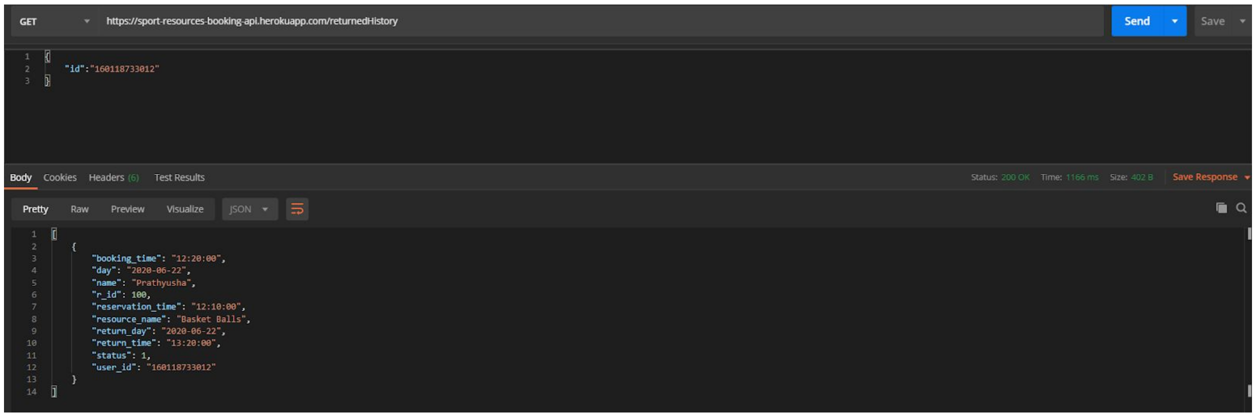
/unblockUser-GET request updating the database to obtain the status of the user and to change the status of due of the user when a user pays their dues.

/blockUser-GET request to update the fine next to the id of a user using the id.

/bookResource-is used to book a resource by the user,so that the details gets added to the database

/rejectBooking-GET request to reject the booking of a user if the admin wants to because of any issues regarding the timetable.

/returnedHistory-GET request to obtain the log of all the users who returned booked resources for today from the BookingHistory2 view.


GET https://sport-resources-booking-api.herokuapp.com/returnedHistory

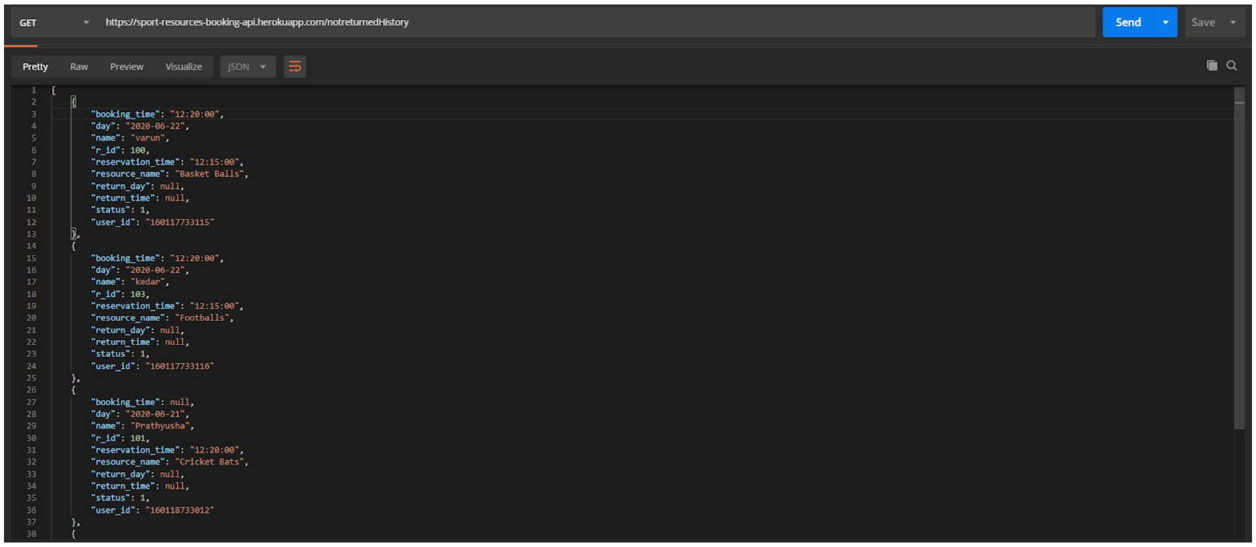
```
{
    "id":"160118733012"
}
```

Status: 200 OK Time: 1166 ms Size: 402 B

```
[
    {
        "booking_time": "12:20:00",
        "day": "2020-06-22",
        "name": "Prathyusha",
        "r_id": 100,
        "reservation_time": "12:10:00",
        "resource_name": "Basket Balls",
        "return_day": "2020-06-22",
        "return_time": "13:20:00",
        "status": 1,
        "user_id": "160118733012"
    }
]
```

Figure 12: Returned history

/notreturnedHistory- GET request to obtain the log of all the users who hadn't returned booked resources till that time from the BookingHistory2 view.


GET https://sport-resources-booking-api.herokuapp.com/notreturnedHistory

```
[
    {
        "booking_time": "12:20:00",
        "day": "2020-06-22",
        "name": "varun",
        "r_id": 100,
        "reservation_time": "12:15:00",
        "resource_name": "Basket Balls",
        "return_day": null,
        "return_time": null,
        "status": 1,
        "user_id": "160117733115"
    },
    {
        "booking_time": "12:20:00",
        "day": "2020-06-22",
        "name": "kedar",
        "r_id": 103,
        "reservation_time": "12:15:00",
        "resource_name": "Footballs",
        "return_day": null,
        "return_time": null,
        "status": 1,
        "user_id": "160117733116"
    },
    {
        "booking_time": null,
        "day": "2020-06-21",
        "name": "Prathyusha",
        "r_id": 101,
        "reservation_time": "12:20:00",
        "resource_name": "Cricket Bats",
        "return_day": null,
        "return_time": null,
        "status": 1,
        "user_id": "160118733012"
    },
    {
```

Figure 13: Not returned History

/forgot_password -GET request If the user send a request to change password along with the id This will send a confirmation mail to change the password to the registered mail id.

/update_password -GET request, user can change their password using their id,old_password and new_password.

4.4.3 Features of Administrator

The administrator is the one who accepts and approves booking requests. They handle the Web application. The functionalities of the admin module include:

- Adding and deleting the resource when required.
- Accepting or rejecting the resource requested by the user based on the timetable.
- Updating the booking depending on whether the resources were returned or not and imposing fines accordingly.
- Blocking the user if the resource is damaged.
- Collecting fine manually and unblocking the user.

5 . IMPLEMENTATION AND RESULT

5.1 Pseudo Code:

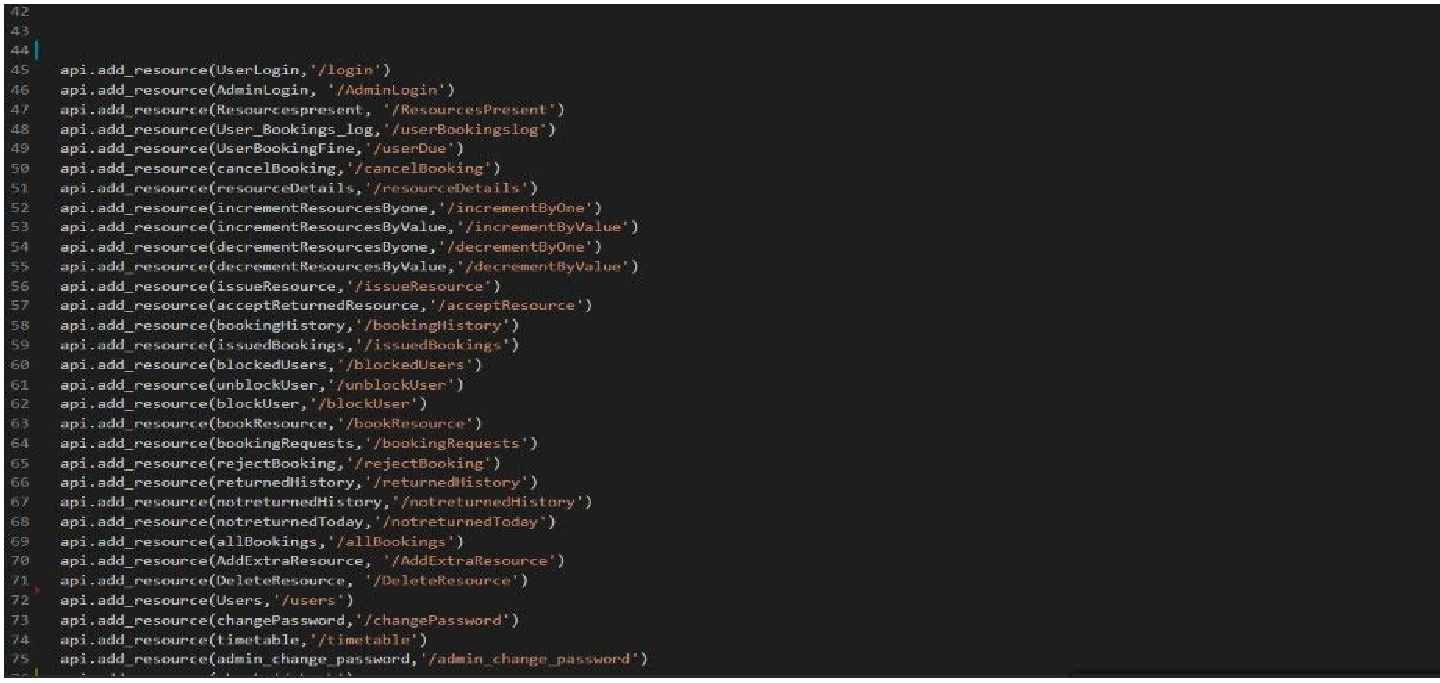
This is the django code for displaying all the resources present in the webapp.

```
def api_call(request):
    global p
    if(p):
        data =
requests.get("https://sport-resources-booking-api.herokuapp.com/ResourcesP
resent", headers = {'Authorization':f'Bearer {p}'})
        res = data.json()
        if(p==''):
            return redirect('login')
        else:
            context={'data': res,}
            return render(request,'api.html',context)
    else:
        return redirect('login')
```

This is the django api call code for booking requests page:

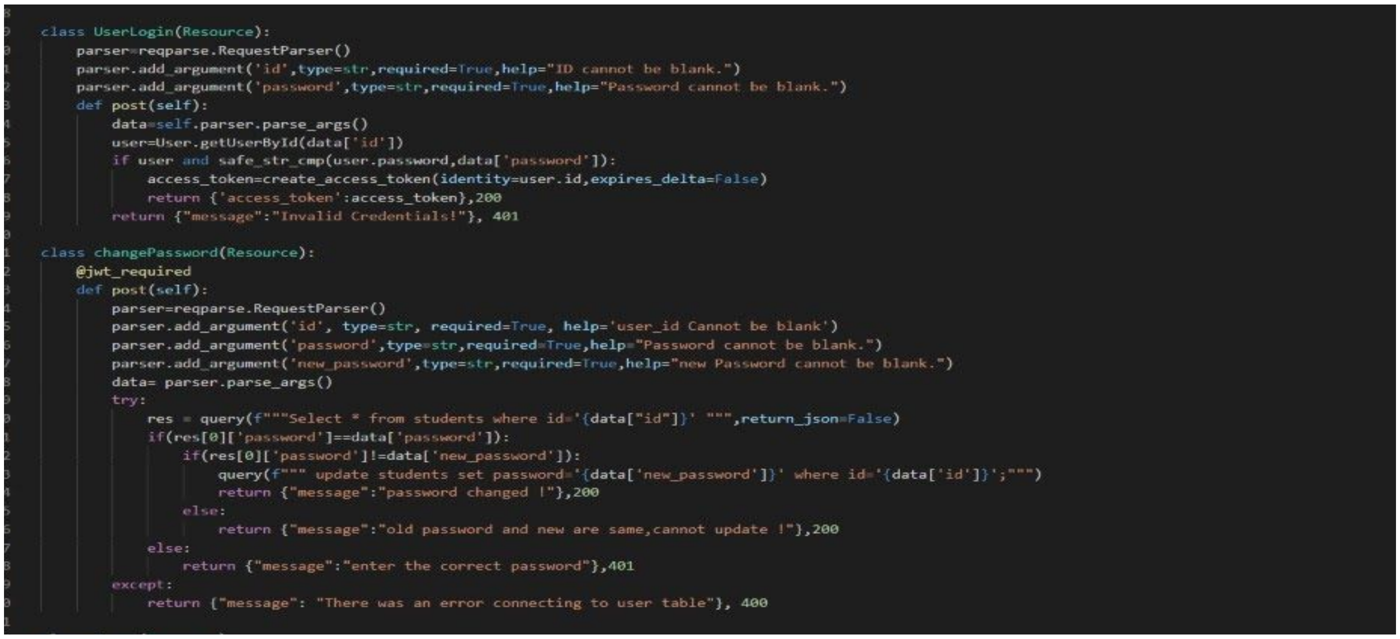
```
def bookingRequests(request):
    global p
    if(p):
        search_term = ''
        if 'search' in request.GET:
            search_term = request.GET['search']
        td =
requests.get("https://sport-resources-booking-api.herokuapp.com/bookingReq
uests", headers = {'Authorization':f'Bearer {p}'},
        data={'search':search_term})
        res = td.json()
        context={'data': res,}
        return render(request,'bookingreq.html',context)
    else:
        return redirect('login')
```

All the end points present in the application



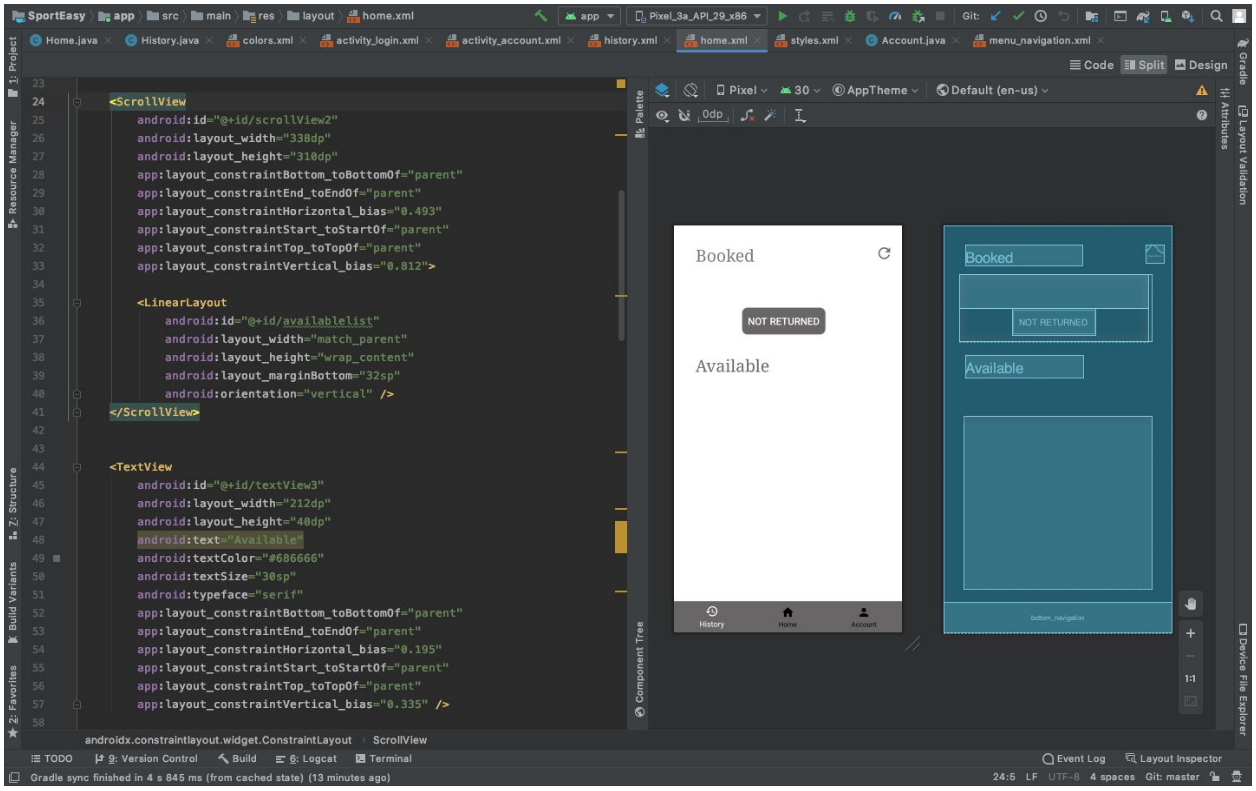
```python
api.add_resource(UserLogin,'/login')
api.add_resource(AdminLogin, '/AdminLogin')
api.add_resource(Resourcespresent, '/ResourcesPresent')
api.add_resource(User_Bookings_log,'/userBookingslog')
api.add_resource(UserBookingFine,'/userDue')
api.add_resource(cancelBooking,'/cancelBooking')
api.add_resource(resourceDetails,'/resourceDetails')
api.add_resource(incrementResourcesByone,'/incrementByOne')
api.add_resource(incrementResourcesByValue,'/incrementByValue')
api.add_resource(decrementResourcesByone,'/decrementByOne')
api.add_resource(decrementResourcesByValue,'/decrementByValue')
api.add_resource(issueResource,'/issueResource')
api.add_resource(acceptReturnedResource,'/acceptResource')
api.add_resource(bookingHistory,'/bookingHistory')
api.add_resource(issuedBookings,'/issuedBookings')
api.add_resource(blockedUsers,'/blockedUsers')
api.add_resource(unblockUser,'/unblockUser')
api.add_resource(blockUser,'/blockUser')
api.add_resource(bookResource,'/bookResource')
api.add_resource(bookingRequests,'/bookingRequests')
api.add_resource(rejectBooking,'/rejectBooking')
api.add_resource(returnedHistory,'/returnedHistory')
api.add_resource(notreturnedHistory,'/notreturnedHistory')
api.add_resource(notreturnedToday,'/notreturnedToday')
api.add_resource(allBookings,'/allBookings')
api.add_resource(AddExtraResource, '/AddExtraResource')
api.add_resource(DeleteResource, '/DeleteResource')
api.add_resource(Users,'/users')
api.add_resource(changePassword,'/changePassword')
api.add_resource(timetable,'/timetable')
api.add_resource(admin_change_password,'/admin_change_password')
```

UserLogin api



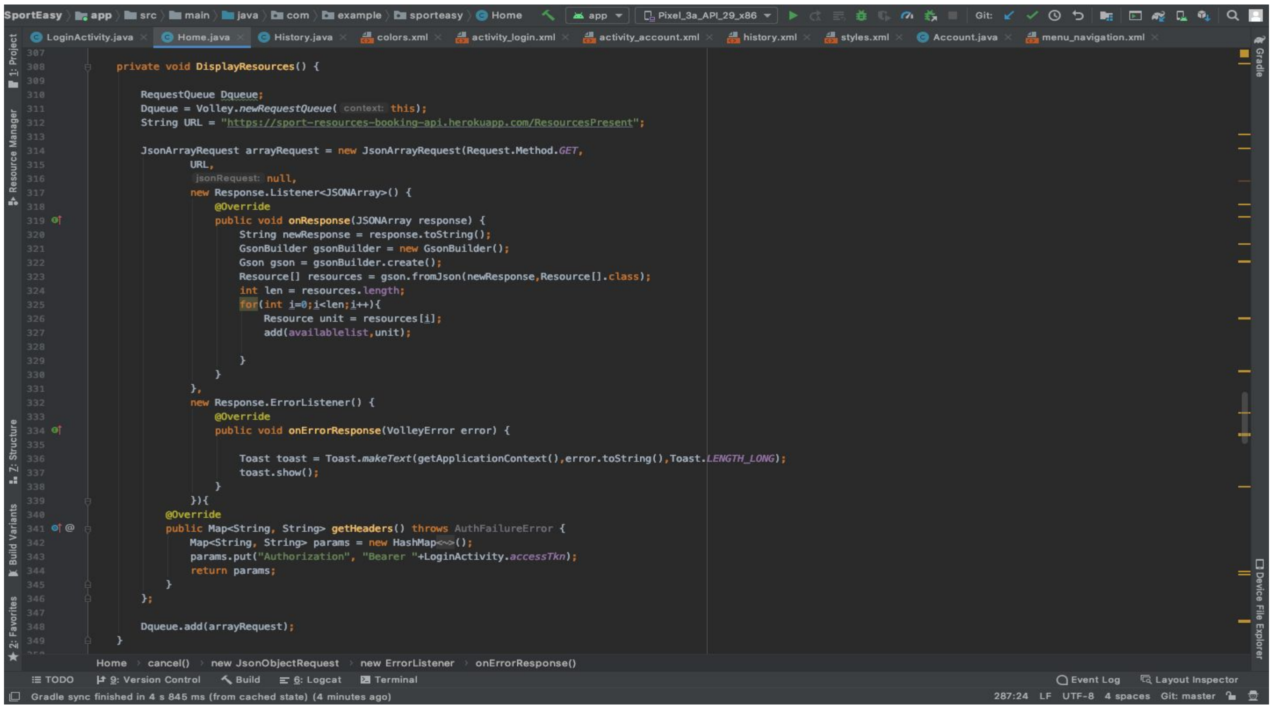
```python
class UserLogin(Resource):
    parser=reqparse.RequestParser()
    parser.add_argument('id',type=str,required=True,help="ID cannot be blank.")
    parser.add_argument('password',type=str,required=True,help="Password cannot be blank.")
    def post(self):
        data=self.parser.parse_args()
        user=User.getUserById(data['id'])
        if user and safe_str_cmp(user.password,data['password']):
            access_token=create_access_token(identity=user.id,expires_delta=False)
            return {'access_token':access_token},200
        return {"message":"Invalid Credentials!"}, 401

class changePassword(Resource):
    @jwt_required
    def post(self):
        parser=reqparse.RequestParser()
        parser.add_argument('id', type=str, required=True, help='user_id Cannot be blank')
        parser.add_argument('password',type=str,required=True,help="Password cannot be blank.")
        parser.add_argument('new_password',type=str,required=True,help="new Password cannot be blank.")
        data= parser.parse_args()
        try:
            res = query(f"""Select * from students where id='{data["id"]}' """,return_json=False)
            if(res[0]['password']==data['password']):
                if(res[0]['password']!=data['new_password']):
                    query(f""" update students set password='{data['new_password']}' where id='{data['id']}';""")
                    return {"message":"password changed !"},200
                else:
                    return {"message":"old password and new are same,cannot update !"},200
            else:
                return {"message":"enter the correct password"},401
        except:
            return {"message": "There was an error connecting to user table"}, 400
```

Xml code for the layout of the resources being displayed.



```xml

<ScrollView
    android:id="@+id/scrollView2"
    android:layout_width="338dp"
    android:layout_height="310dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.493"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.812">

    <LinearLayout
        android:id="@+id/availablelist"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32sp"
        android:orientation="vertical" />
</ScrollView>


<TextView
    android:id="@+id/textView3"
    android:layout_width="212dp"
    android:layout_height="40dp"
    android:text="Available"
    android:textColor="#686666"
    android:textSize="30sp"
    android:typeface="serif"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.195"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.335" />

```

Java code to display the available sports resources.



```java

    private void DisplayResources() {

        RequestQueue Dqueue;
        Dqueue = Volley.newRequestQueue( context: this);
        String URL = "https://sport-resources-booking-api.herokuapp.com/ResourcesPresent";

        JsonArrayRequest arrayRequest = new JsonArrayRequest(Request.Method.GET,
                URL,
                 jsonRequest: null,
                new Response.Listener<JSONArray>() {
                    @Override
                    public void onResponse(JSONArray response) {
                        String newResponse = response.toString();
                        GsonBuilder gsonBuilder = new GsonBuilder();
                        Gson gson = gsonBuilder.create();
                        Resource[] resources = gson.fromJson(newResponse,Resource[].class);
                        int len = resources.length;
                        for(int i=0;i<len;i++){
                            Resource unit = resources[i];
                            add(availablelist,unit);

                        }
                    }
                },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {

                        Toast toast = Toast.makeText(getApplicationContext(),error.toString(),Toast.LENGTH_LONG);
                        toast.show();
                    }
                }){
            @Override
            public Map<String, String> getHeaders() throws AuthFailureError {
                Map<String, String> params = new HashMap<~>();
                params.put("Authorization", "Bearer "+LoginActivity.accessTkn);
                return params;
            }
        };

        Dqueue.add(arrayRequest);
    }
```

5.2 Design steps and Outputs/Results

5.2.1Mobile Application :

5.2.1.1 Login Activity:A student can login to this application with his Roll number and Password

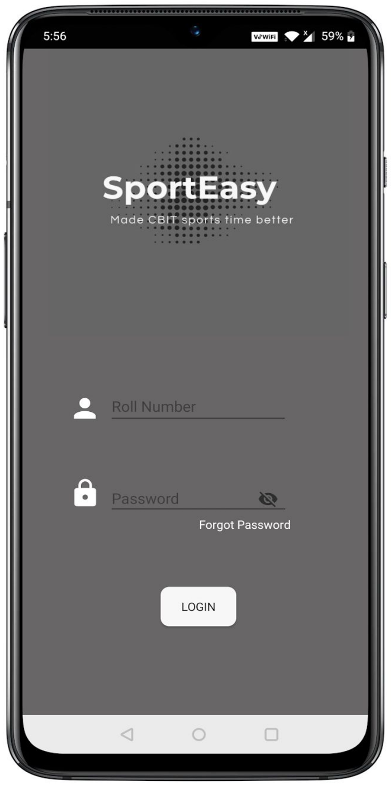


Figure 14:Login Activity of User

5.2.1.2 Home Activity: This shows the details of the resources present in the sports block and also the details of the current booking by the user

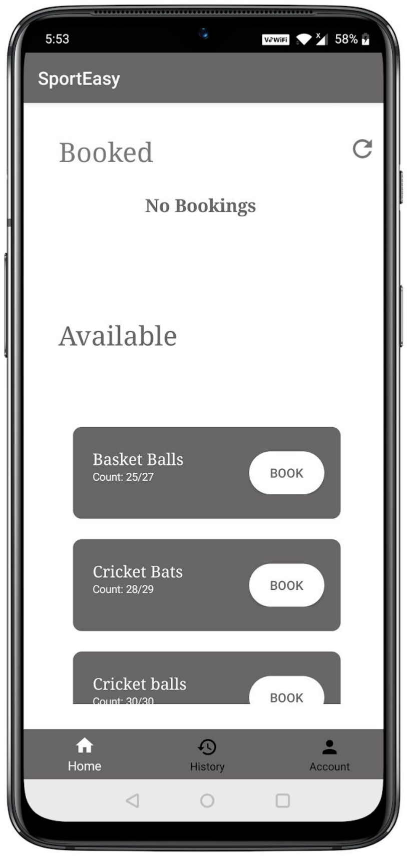


Figure 15 : User Home page 1

5.2.1.3 Booking Resource page for the user: The user can book a resource between 11:40pm and 3:00 pm and if he doesn't have any fine or resource which is not yet returned.And can cancel within 20 minutes time

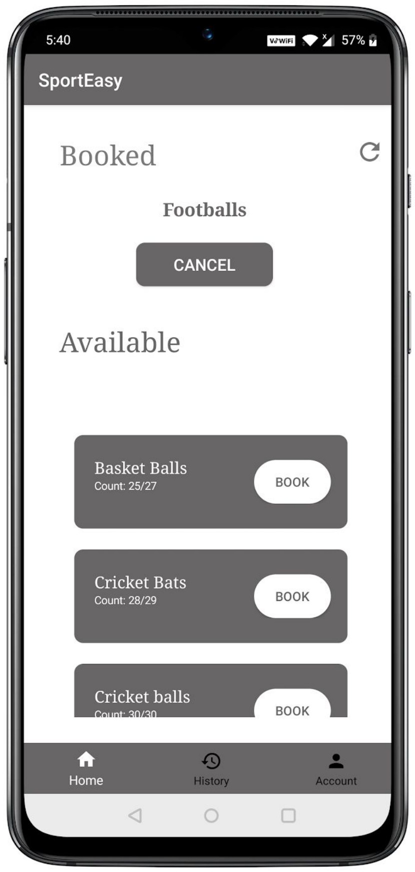


Figure 16 : User Home page 2

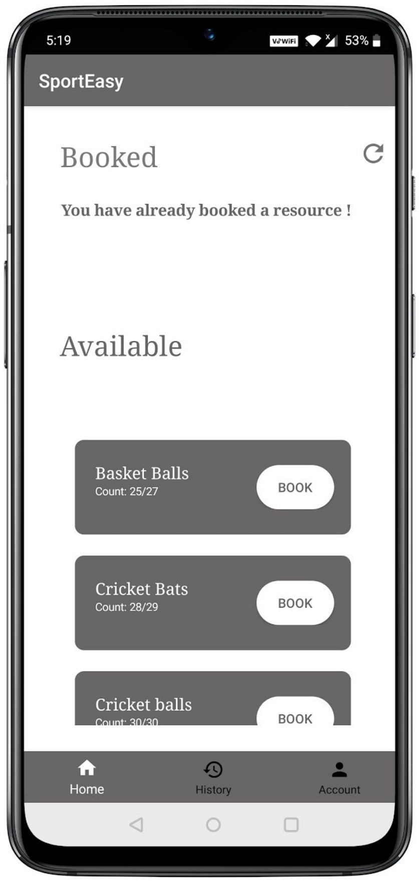


Figure 17 : User Home page 3

5.2.1.4 Booking History page for the user: Lists all the booking made by the user which are returned,not returned ,not collected.

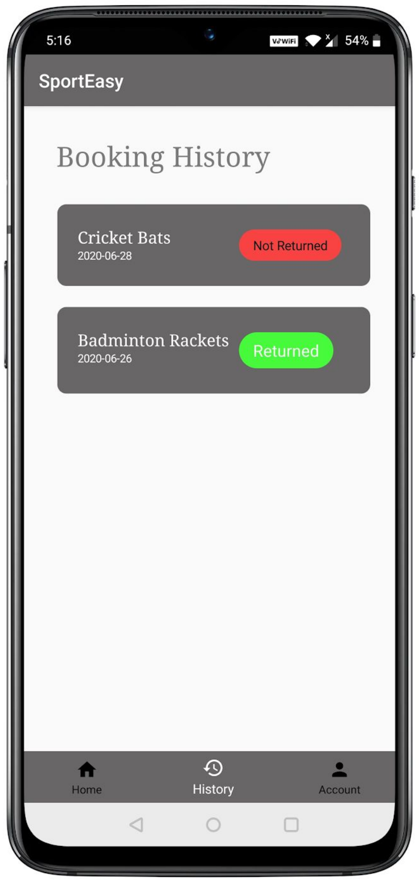
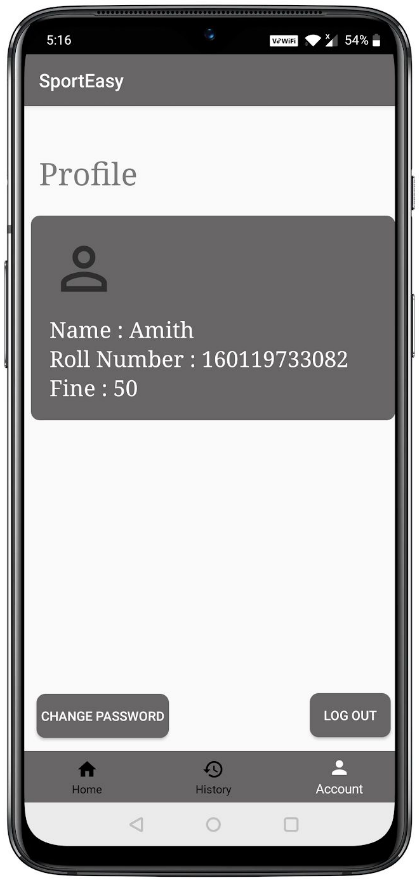


Figure 18:Booking history of user

5.2.1.5 Account page for the user: This shows the details of the user and has an option to change password and logout.


Profile

Name : Amith
Roll Number : 160119733082
Fine : 50

Figure 19:Account of user

5.2.2 Web Application:

5.2.2.1 Login Page: The login page enables the admin to log into the web application on entering the right details into the given fields.

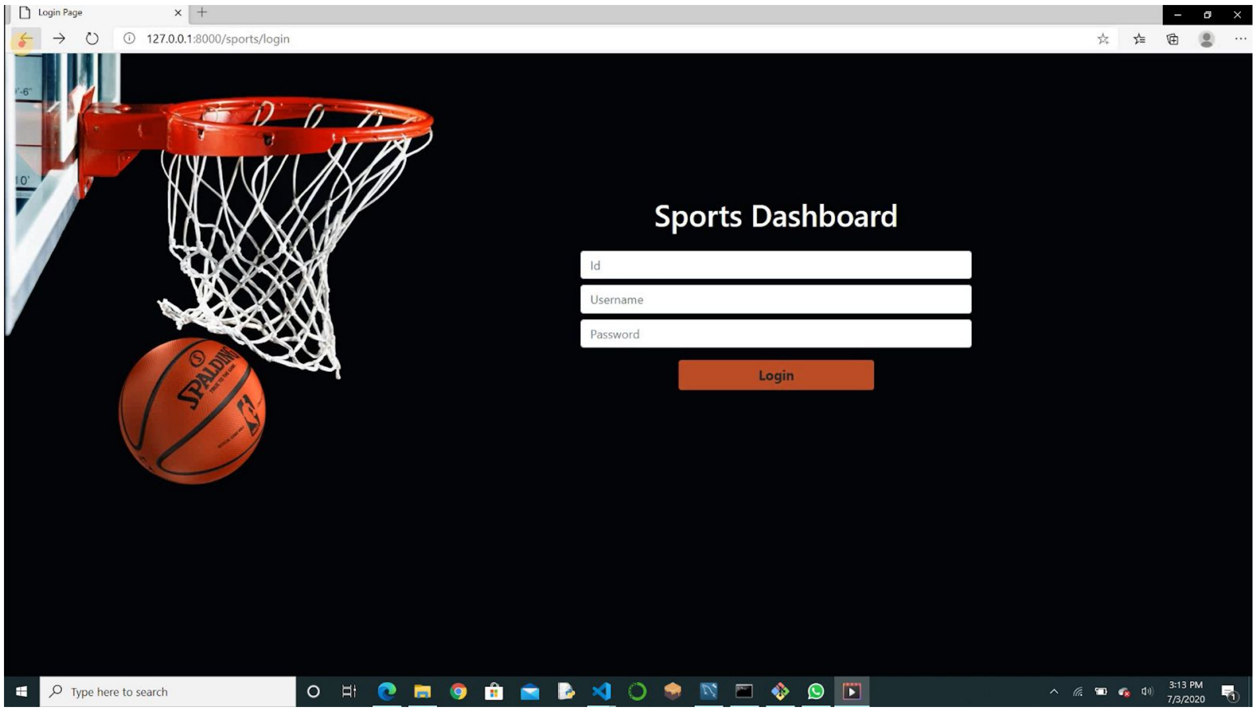
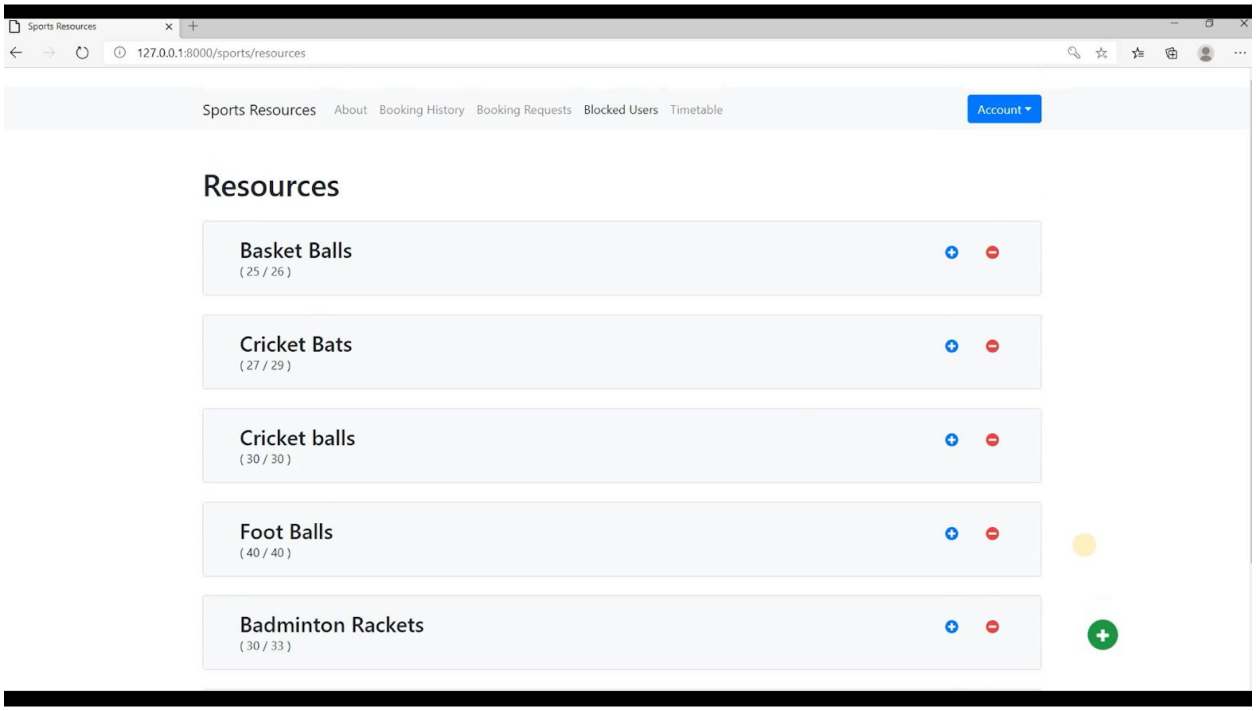


Figure 20: Login page

5.2.2.2 Home Page (Sports Resources Page): This page displays the list of available resources. The resources as well as their count can be updated using the plus and minus buttons.



Resources

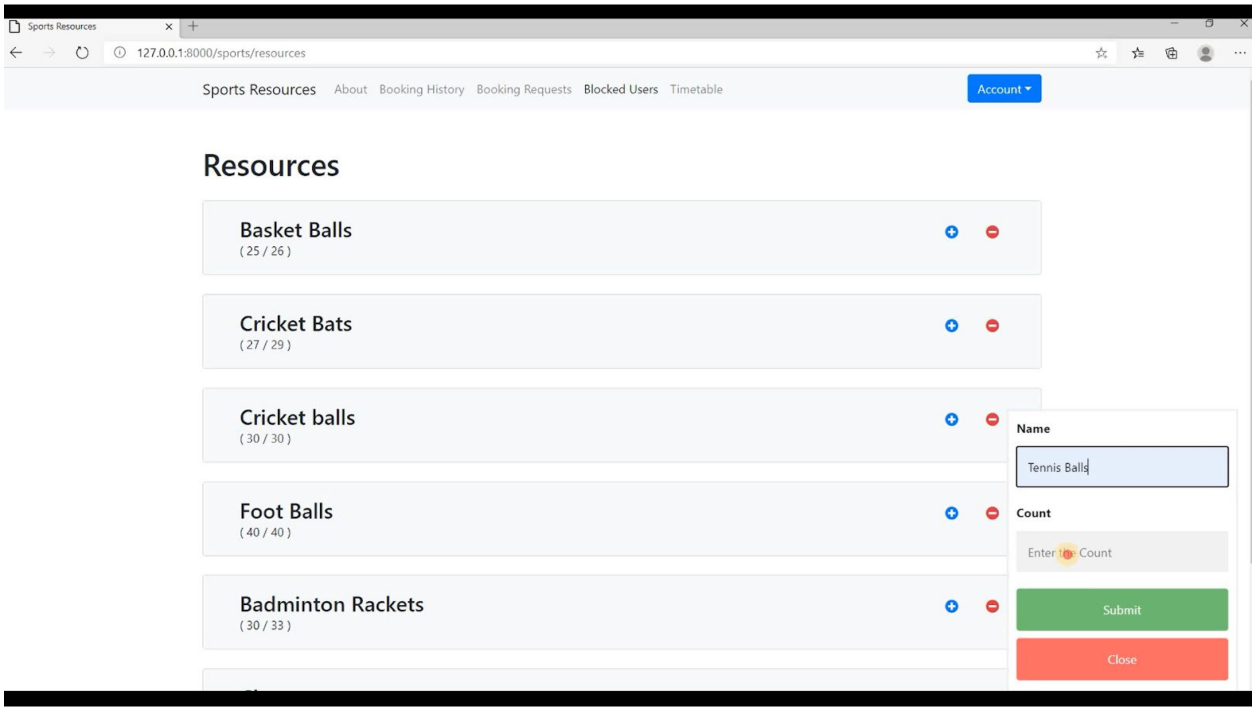
Basket Balls
(25 / 26)

Cricket Bats
(27 / 29)

Cricket balls
(30 / 30)

Foot Balls
(40 / 40)

Badminton Rackets
(30 / 33)



Resources

Basket Balls
(25 / 26)

Cricket Bats
(27 / 29)

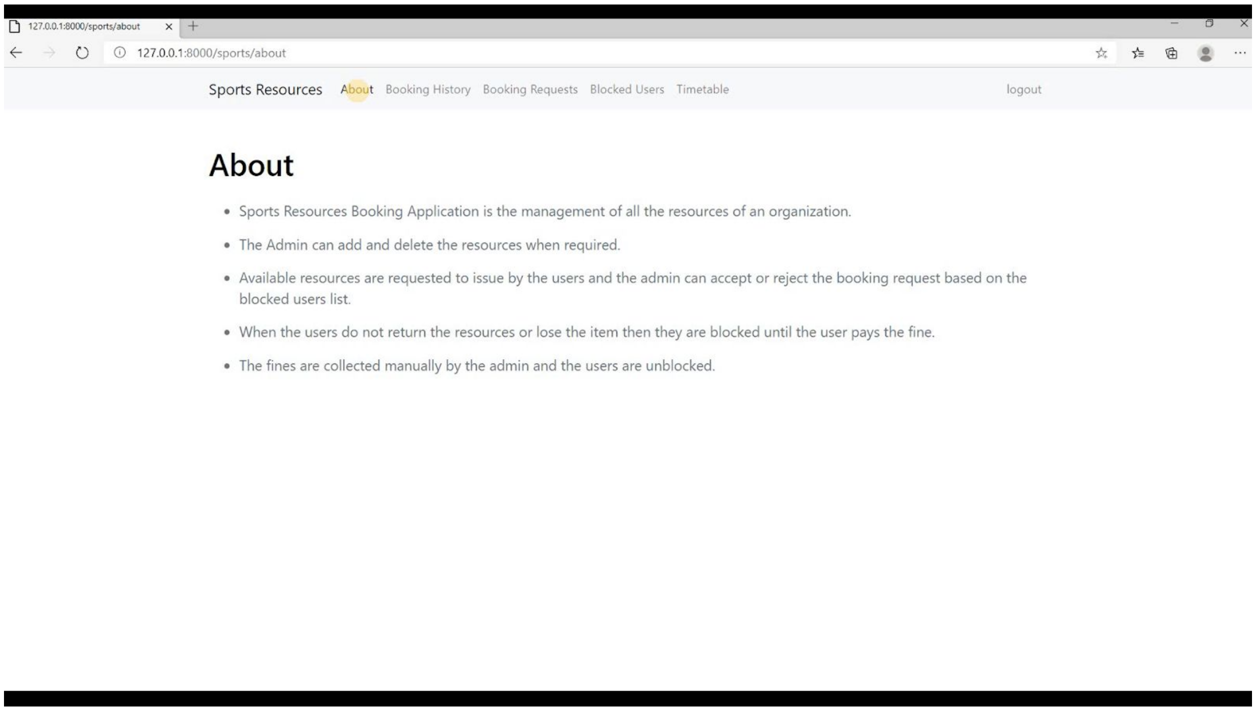
Cricket balls
(30 / 30)

Foot Balls
(40 / 40)

Badminton Rackets
(30 / 33)

Figures 21: Home Page (Resources Page)

5.2.2.4 About Page: This page displays information about the application.

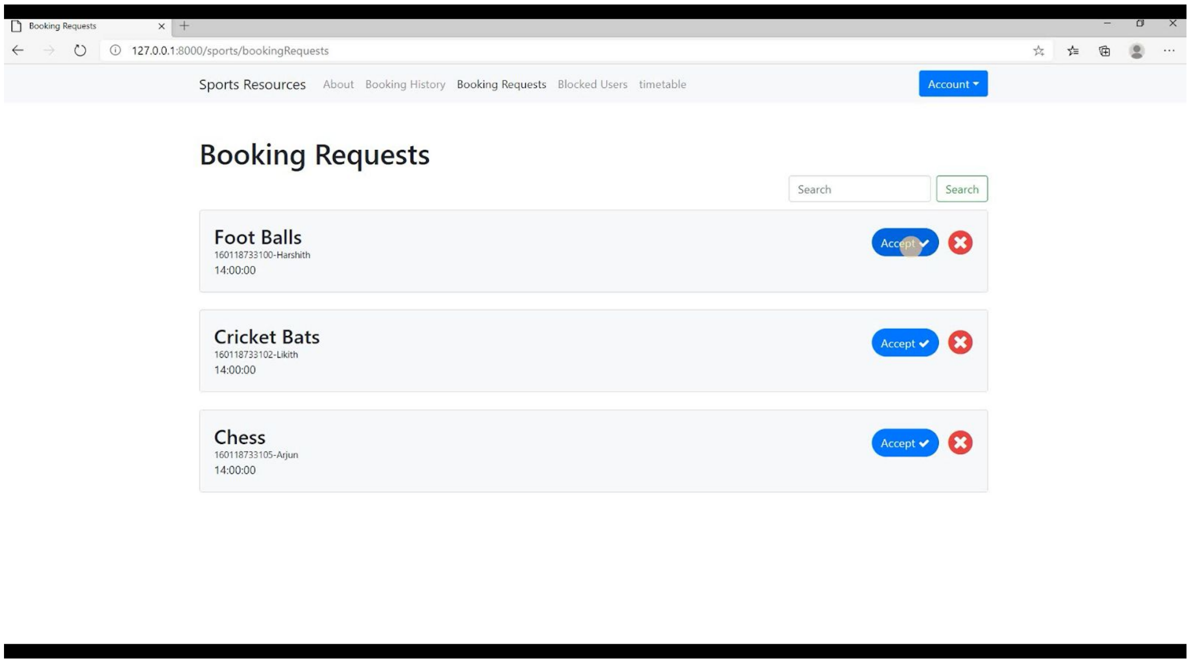


About

- Sports Resources Booking Application is the management of all the resources of an organization.
- The Admin can add and delete the resources when required.
- Available resources are requested to issue by the users and the admin can accept or reject the booking request based on the blocked users list.
- When the users do not return the resources or lose the item then they are blocked until the user pays the fine.
- The fines are collected manually by the admin and the users are unblocked.

Figure 22: About Page

5.2.2.5 Booking Requests page: This page displays the booking requests made by the user. The Admin can either accept or decline the request.



Booking Requests

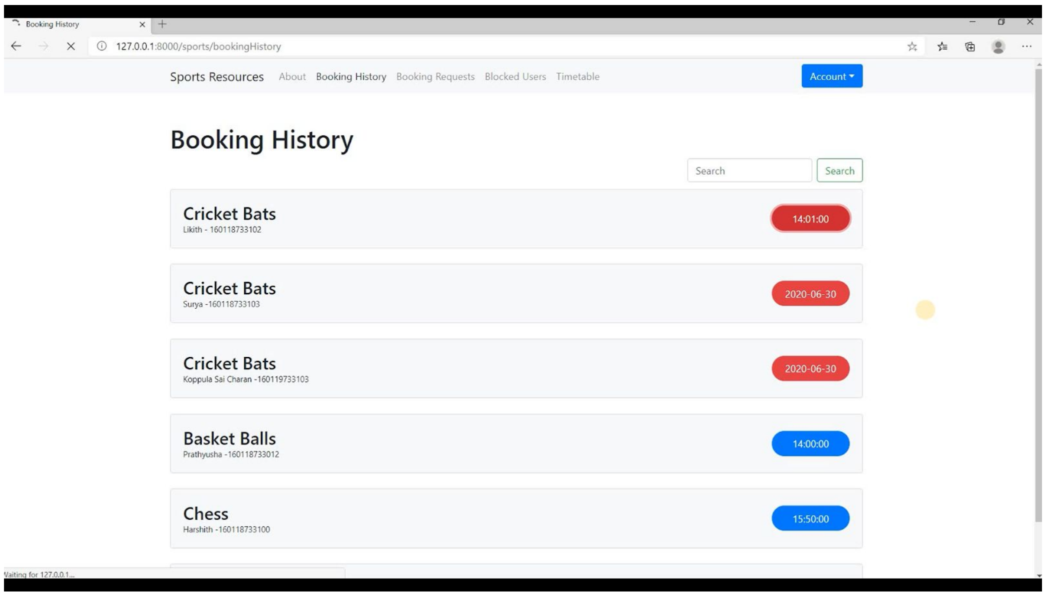
Foot Balls
160118733100-Harshith
14:00:00

Cricket Bats
160118733102-Likith
14:00:00

Chess
160118733105-Arjun
14:00:00

Figure 23: Booking Requests Page

5.2.2.6 Booking History Page: This page displays the history of all the accepted bookings. It is highlighted in red if the user hasn't returned the resource within the specified time along with a timestamp of the time of issue, and it is highlighted in blue if the user still has time to return the resource along with a timestamp of the time when resource was returned.



Booking History

Cricket Bats
Likith - 160118733102
14:01:00

Cricket Bats
Surya -160118733103
2020-06-30

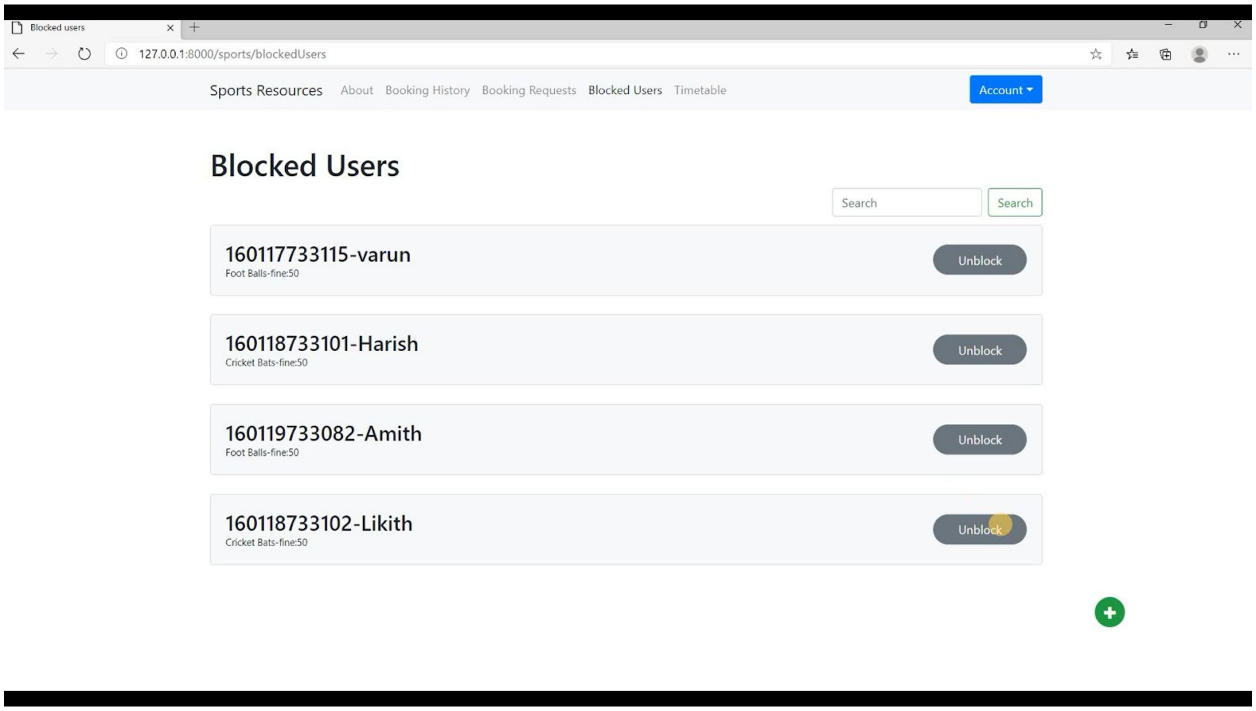
Cricket Bats
Koppula Sai Charan -160119733103
2020-06-30

Basket Balls
Prathyusha -160118733012
14:00:00

Chess
Harshith -160118733100
15:50:00

Figure 24: Booking History Page

5.2.2.7 Blocked Users Page: This page displays the list of blocked users. There is also an option to unblock them. The admin can also add new users to the list by clicking on the plus icon.



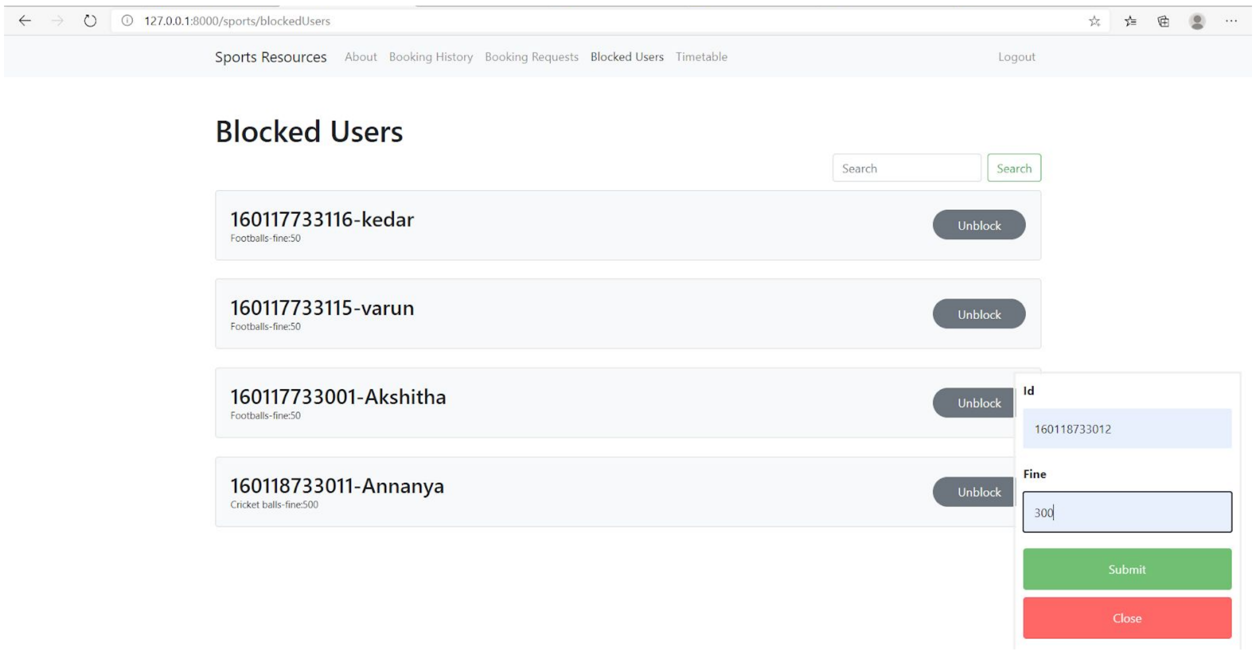
Blocked Users

160117733115-varun
Foot Balls-fine:50

160118733101-Harish
Cricket Bats-fine:50

160119733082-Amith
Foot Balls-fine:50

160118733102-Likith
Cricket Bats-fine:50



Blocked Users

160117733116-kedar
Footballs-fine:50

160117733115-varun
Footballs-fine:50

160117733001-Akshitha
Footballs-fine:50

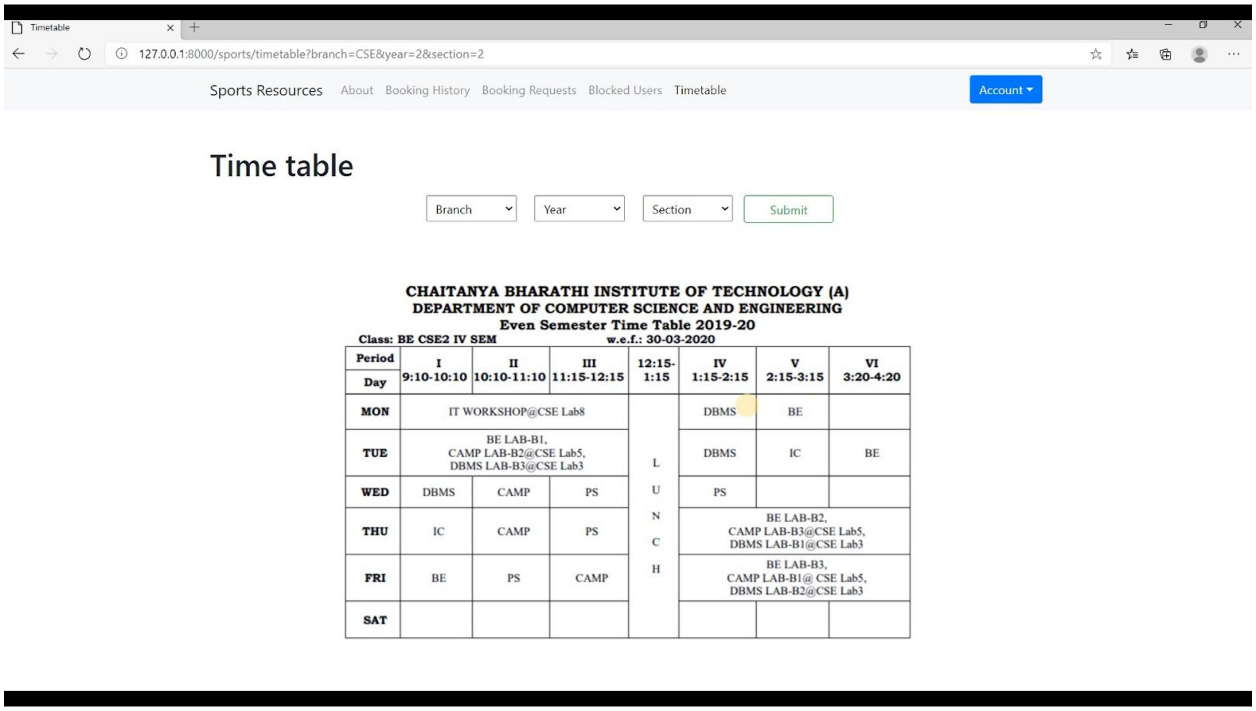
160118733011-Annanya
Cricket balls-fine:500

Id
160118733012

Fine
300

Figures 25 & 26: Blocked Users Page

5.2.2.8 Timetable Page: This page displays the timetable of a class based on the inputs given.



CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Even Semester Time Table 2019-20
Class: BE CSE2 IV SEM w.e.f.: 30-03-2020

Period / Day	I 9:10-10:10	II 10:10-11:10	III 11:15-12:15	12:15-1:15	IV 1:15-2:15	V 2:15-3:15	VI 3:20-4:20
MON	IT WORKSHOP@CSE Lab8				DBMS	BE	
TUE	BE LAB-B1, CAMP LAB-B2@CSE Lab5, DBMS LAB-B3@CSE Lab3			L	DBMS	IC	BE
WED	DBMS	CAMP	PS	U	PS		
THU	IC	CAMP	PS	N C	BE LAB-B2, CAMP LAB-B3@CSE Lab5, DBMS LAB-B1@CSE Lab3		
FRI	BE	PS	CAMP	H	BE LAB-B3, CAMP LAB-B1@ CSE Lab5, DBMS LAB-B2@CSE Lab3		
SAT							

Figure 27: Timetable Page

6. TESTING AND VALIDATION

6.1 Introduction

Testing and validation of the code is a crucial part of the project. It is impossible to determine all of the exceptional cases that might arise during the design. Hence testing serves as a useful way to find the flaws in both the structure of the code and the implementation of the various functions.

6.2 Design of Test Cases and Scenarios

The test cases were designed keeping in mind where the code is liable to fail and throw errors or give erroneous outputs. The following cases are some of the important scenarios to be kept in mind.

6.2.1 Forgot Password: If the user forgot the password then he gets an email to reset the password.

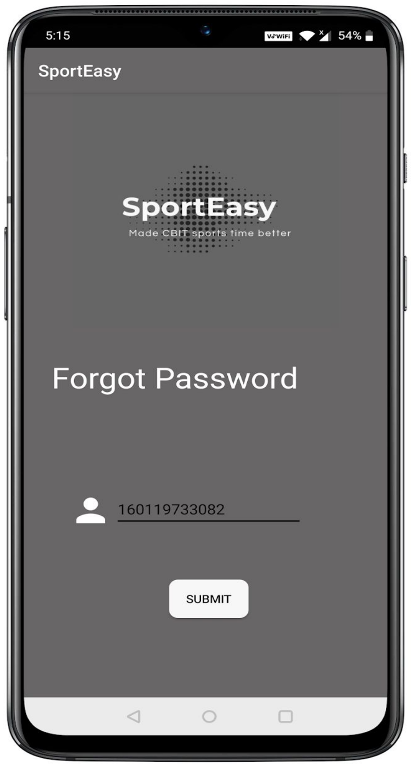
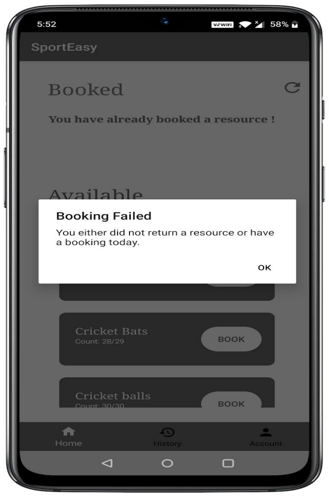


Figure 28: Forgot Password

6.2.2 Booking Request by a User with Fine: If the user tries to book the resource even though he has fine then a message pops up showing that the user has fine.



Booking Failed

You either did not return a resource or have a booking today.

Figure 29: Booking request by user with fine

6.2.3 Change Password: If the user wants to change the password then he can do so by entering a new password and confirming it.

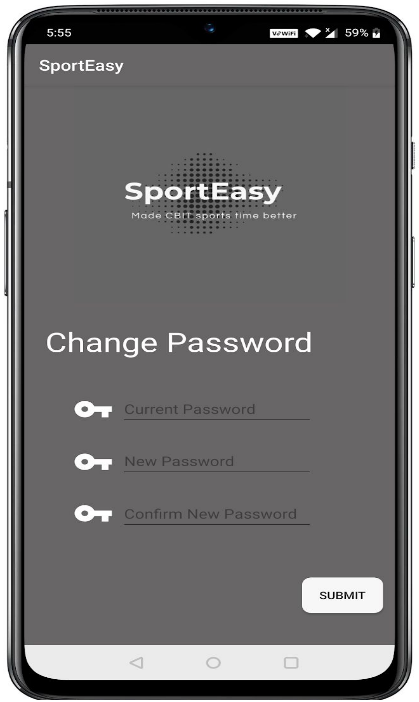


Figure 30: Change Password of user

6.2.4 Invalid credentials: If the admin tries to enter the wrong credentials then a message is displayed showing Invalid Credentials.

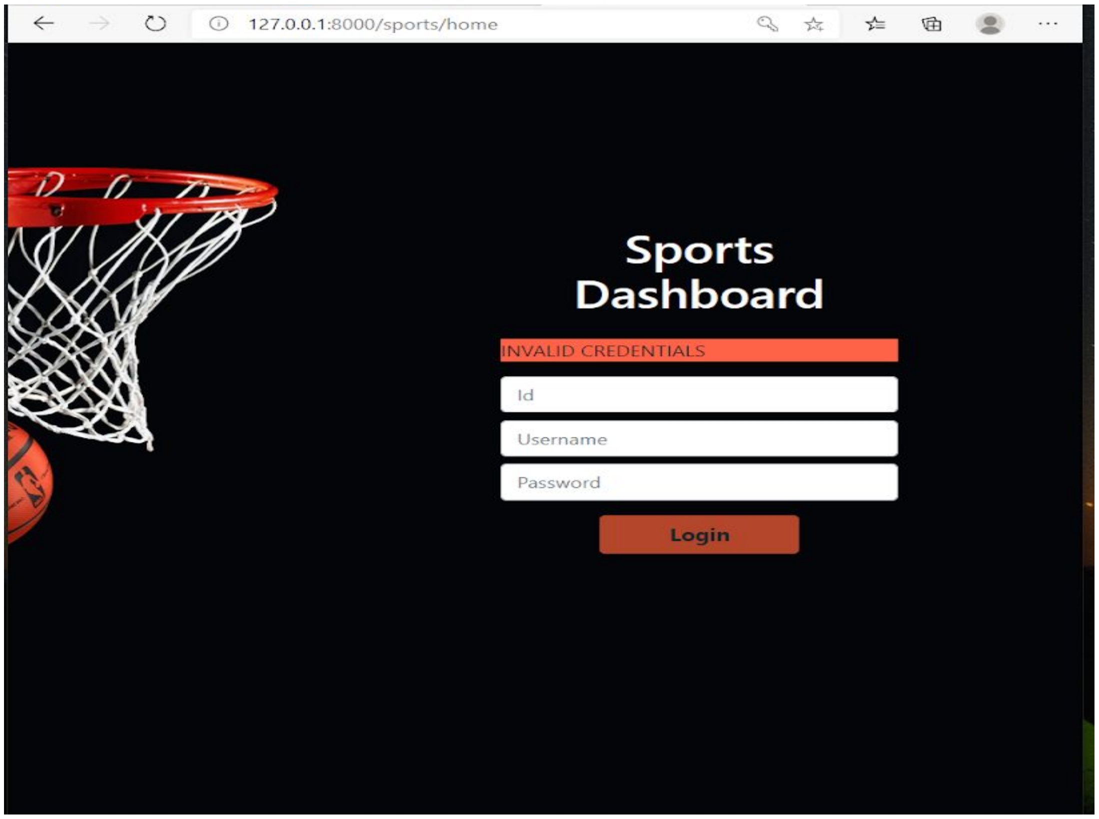


Figure 31:Invalid credentials of admin

6.2.5 Change Password: If the admin wants to change the password then he can do so by entering a new password and confirming it.

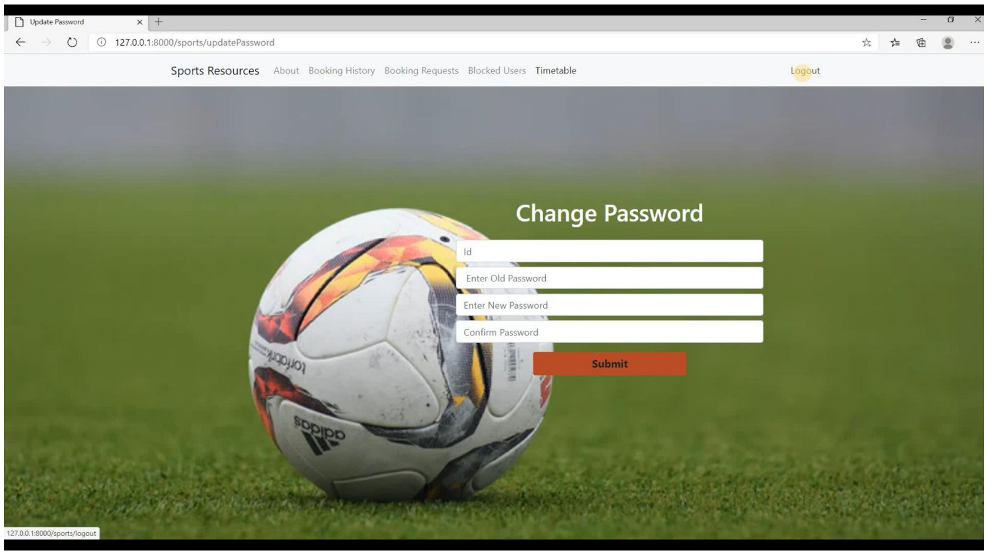


Figure 31:Change Password for admin

6.3 Validation

The code was validated by running all the test cases mentioned above,for example like with both wrong and correct login credentials. Care was taken to include all possible exceptional cases and other problematic scenarios in the input.

7. CONCLUSION

7.1 Conclusion

This project is for computerising the work employed in order to borrow a sports resource. It is a great improvement over the manual system. The computerisation of the system has sped up the process. In the current system, the front office management is very slow. This application was thoroughly checked with dummy data and thus is found to be very reliable. The software takes care of all the requirements and is capable of providing easy and effective storage information related to students that book sports resources. Responsive web design and mobile app makes work easier and safe for any field. With the provision of an easily accessible user interface, the entire process of booking and approval of resources, along with other features, have become extremely facile while also significantly averting the numerous errors that would otherwise be caused in a manual system. With this platform we developed, we are hoping to achieve the following:

- Reduced time wastage
- Providing comfortable facilities to students
- Provide easy data flow
- Less employee investment

7.2 Limitations

The mobile application which has been developed for the students for booking sports resources is only compatible with android versions 5.0+, it requires a minimum android runtime version of 5.0

The Admin has to check the timetable while issuing a resource.

There is no deadline for the user to clear the due and no notification system to intimate the user to clear the due.

7.3 Future Scope

This application can be enhanced by including numerous new features. One such feature is adding a notification system through which the admin can send notifications to a user regarding his bookings. In the current system, the user should manually go into the app and check whether the resource is accepted or not. Instead, an improvement to this would be sending a confirmation to the user when his booking request is approved by the admin. In case of unavailability of a resource, a “notify when available” button could be included too. As of now, when a student wants to book a resource, the admin manually checks whether the student has a free hour at that particular time and issues the resource accordingly. But in the future, an app can be made which itself detects whether the student is having a free hour or not, thereby eliminating the involvement of the admin. Subsequently, a separate page linked to admin webpage could also be developed, where the admin can add updates about upcoming sports events or competitions in such a way that these updates will also be visible in the mobile app for students, therefore allowing them to be aware of all the ongoing or upcoming sports events through their mobile.

REFERENCES

1.https://www.javatpoint.com/java-tutorial
2.https://www.quora.com
3.https://www.javatpoint.com/android-tutorial
4.https://www.tutorialspoint.com/java/index.htm
5. https://www.w3resource.com/java-exercises/
6. https://www.wikipedia.com
7.https://www.google.com/django-documentation/
8.https://www.programiz.com/java-programming
9.https://www.tutorialspoint.com/android/index.htm
10.https://developer.android.com/guide
11.https://developer.android.com/training/basics/firstapp
12.https://www.vogella.com/tutorials/android.html
13.https://www.youtube.com/watch?v=jEmq1B1gveM
14.https://firebase.google.com/docs/android/setup
15.https://www.w3schools.com/java/
16.https://www.tutorialspoint.com/flask

APPENDIX - I

https://github.com/indooriaditi/cosc4-webteam
https://github.com/amith2610/Android-Team4
https://github.com/gsaiprathyusha/apiteam04
https://sport-resources-booking-api.herokuapp.com