**International Institute of Information**

**Technology, Bangalore**

**(IIIT Bangalore)**

****

**Software Production Engineering**

**Project Report**

**IPL 2023**

Under the Guidance of Prof. B. Thangaraju

Deepak Kushagra Agrawal

(MT2022036) (MT2022145)

**1. ABSTRACT**

IPL 2023 is a dynamic and interactive website designed to provide users with real-time updates on the Indian Premier League (IPL) cricket tournament. The website offers live scores, statistics, schedules, and the ability to book tickets for matches held in various stadiums.

With a user-friendly interface, IPL 2023 delivers an immersive cricket experience. Users can access live scores and follow their favorite teams' progress in real-time. Detailed match statistics, such as runs, wickets, boundaries, and player performances, are also available, ensuring fans stay informed about every aspect of the game.

The website's schedule section allows users to plan their cricket viewing experience by providing match dates, venues, and timings. Users can easily navigate through the schedule to find specific matches or filter matches by teams, ensuring they never miss an important game.

One of the standout features of IPL 2023 is the ability to book tickets for matches held in different stadiums. The website integrates with ticketing services, enabling users to select their preferred seats and make secure online payments. By simplifying the ticket booking process, IPL 2023 ensures fans can effortlessly secure their spots in the stadium and enjoy the electrifying atmosphere of live IPL matches.

Overall, IPL 2023 is a comprehensive platform that brings the excitement of IPL cricket to fans worldwide. Whether it's following live scores, exploring match statistics, planning match attendance, or booking tickets, the website caters to the diverse needs of IPL enthusiasts, enhancing their overall viewing experience.

**2. INTRODUCTION**

A. OVERVIEW

Introducing IPL 2023: Your Ultimate Destination for Live Cricket Action and Ticket Booking

Welcome to IPL 2023, your go-to website for all things Indian Premier League! We are proud to present a one-stop platform that brings you the exhilarating world of IPL cricket right at your fingertips.

With IPL 2023, you can stay updated with live scores, track match statistics, and plan your cricket viewing experience effortlessly. Our user-friendly interface ensures that you never miss a moment of the thrilling action happening on the cricket field.

B. FEATURES

* Live Score
* Schedule
* Points Table
* Statistics
* Register and Login for Admin and Users
* **Admin:** Add match, delete match , edit match , update match,see bookings, block user, make admin feature.
* **User:** Book ticket in stadium for any match.

C. WHY DEVOPS ?

DevOps is a set of tools that automate software development and IT operations. It focuses on shortening the systems development life cycle and providing continuous delivery with high software quality.

DevOps is one up on Agile software development; multiple DevOps aspects came from the Agile methodology.

* It optimizes the overall business by increasing efficiency through automation.
* Improves software development and deployment speed and stability.
* Deployment failures, rollbacks, and recovery time are all reduced.
* Improved collaboration and communication; lower costs and IT headcount

3. TOOLS USED

1. DevOps Tools
2. Version Control System : Git and GitHub : <https://github.com/Deepak-tect/IPL>
3. Continuous Integration/Continuous Delivery : Jenkins
4. Building Tool : NPM (Node Package Manager)
5. Testing : Mocha framework
6. Containerization : Docker
7. Deployment : Ansible
8. Log creation, monitoring and visualization : ELK (Elasticsearch,

Logstash, Kibana)

1. Frontend Development : React.js
2. Backend Development : Node.js, Express.js
3. Database : MongoDB

B. Project Technology Stack

1. Frontend: React (HTML, JSX, JavaScript, Bootstrap)
2. Backend: Express (NodeJS, JavaScript)
3. Database: MongoDB (No-SQL)
4. Cloud: MongoDB Atlas, Netlify, Heroku
5. Build Tool: npm

4. SYSTEM CONFIGURATION

1. Operating System - Linux Ubuntu 20

2. CPU & RAM - 4 core processor and 8GB RAM

3. Kernel Version - 5.4.0-89-generic

4. Database - MongoDB 4.4.5

5. SOFTWARE DEVELOPMENT LIFECYCLE - SDLC

I. INSTALLATIONS:

**React**

React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies.

**Update local before installing:**

Keep the local packages and softwares updated:



**Install NodeJS and NPM:**

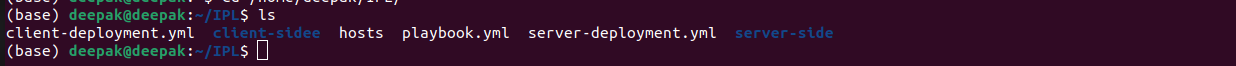
Node.js is a javascript programming package management. It is the JavaScript runtime environment Node.js' default package manager. It comes pre-installed with Node.js. The package.json file contains the definitions for all npm packages. Package.json's content must be in JSON format. The definition file must have at least two fields. The names and variants are as follows. It is capable of managing dependencies. It installs all of the project's dependencies in a single command line. The package.json file also defines dependencies.

Inorder to run React, Node environment shall be installed before starting,





**React App**

****

**Express**

Express.js, or simply Express, is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

**Express App**

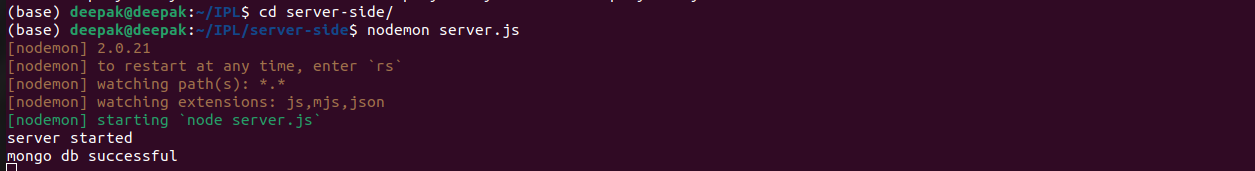
Initializing node project with the default config:



-y automates the config with node default values i.e without going through the interactive process.

**Running the Node Application locally:**

****



{

"name": "server-side",

"version": "1.0.0",

"description": "",

"main": "server.js",

"scripts": {

"test": "mocha --exit"

},

"author": "Deepak",

"license": "ISC",

"dependencies": {

"axios": "^1.3.6",

"bcryptjs": "^2.4.3",

"chai": "^4.3.7",

"cheerio": "^1.0.0-rc.12",

"cors": "^2.8.5",

"dotenv": "^16.0.3",

"express": "^4.18.2",

"jsonwebtoken": "^9.0.0",

"mocha": "^10.2.0",

"mongoose": "^7.0.3",

"nodemailer": "^6.9.1",

"nodemon": "^2.0.22",

"request": "^2.88.2",

"winston": "^3.8.2",

"winston-elasticsearch": "^0.17.2"

}

}

(Backend package.json)

{

"name": "client-side",

"version": "0.1.0",

"private": true,

"proxy": "http://localhost:5000",

"dependencies": {

"@reduxjs/toolkit": "^1.9.3",

"@testing-library/jest-dom": "^5.16.5",

"@testing-library/react": "^13.4.0",

"@testing-library/user-event": "^13.5.0",

"antd": "^5.4.2",

"axios": "^1.3.5",

"bootstrap": "^5.2.3",

"react": "^18.2.0",

"react-dom": "^18.2.0",

"react-redux": "^8.0.5",

"react-router-dom": "^6.9.0",

"react-scripts": "5.0.1",

"react-to-print": "^2.14.12",

"reactstrap": "^9.1.9",

"redux": "^4.2.1",

"styled-components": "^5.3.10",

"web-vitals": "^2.1.4"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

},

"eslintConfig": {

"extends": [

"react-app",

"react-app/jest"

]

},

"browserslist": {

"production": [

">0.2%",

"not dead",

"not op\_mini all"

],

"development": [

"last 1 chrome version",

"last 1 firefox version",

"last 1 safari version"

]

},

"jest": {

"moduleNameMapper": {

"axios": "axios/dist/node/axios.cjs"

}

}

}

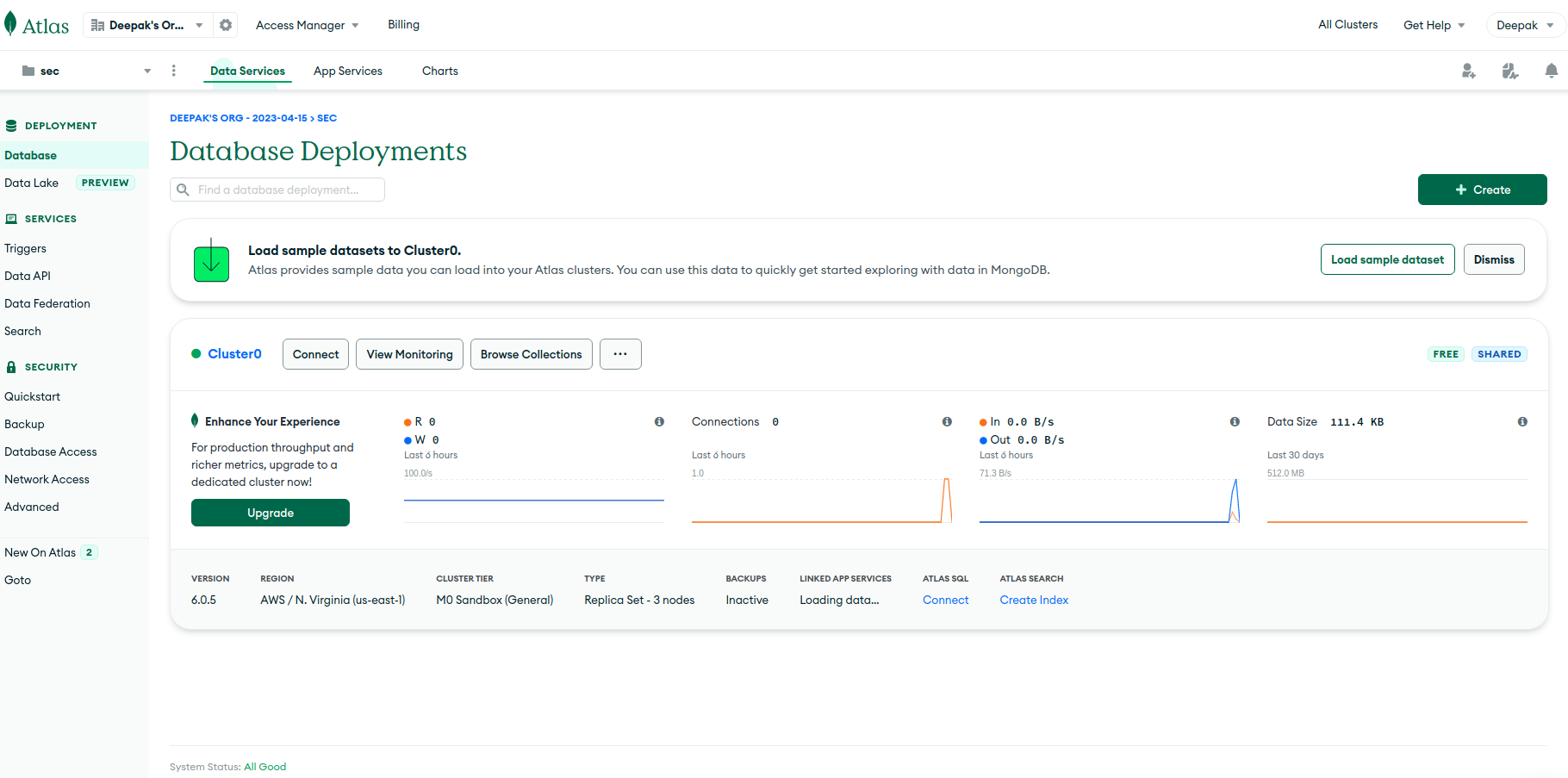
(Frontend package.json)

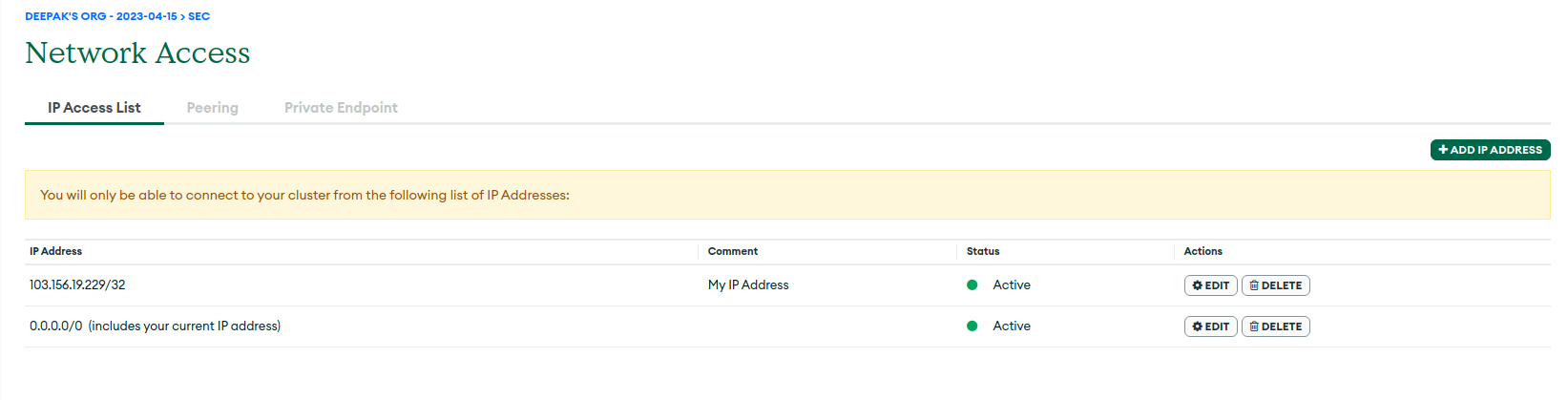
**MongoDB:**

We create a mongodb database on atlas to store all the user and

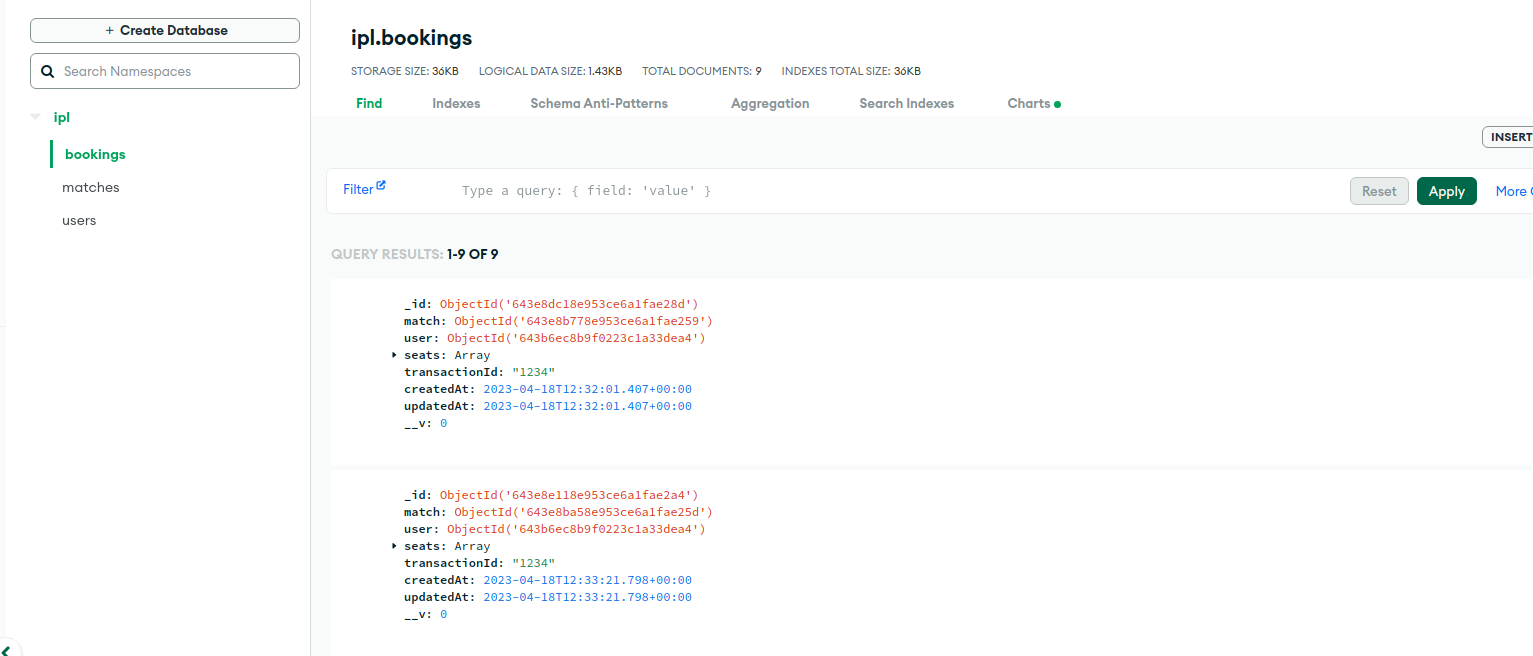
farmer information. To set this database with our project use create URI

and store it in the .env file.





Tables:



**Store MongoDB URL in .env file**

By creating a.env file in the api (backend) subdirectory, you can save the cluster's URL. In that URL, provide all of the MongoDB Atlas' credentials (password). This.env file is part of.gitignore. Furthermore, Mongoose uses

the variable used to store this URL to connect to MongoDB.



JWT Authentication:

JSON Web Tokens (JWT) are an RFC 7519 open industry standard for

representing claims between two parties. For example, you can use jwt.io

to decode, verify, and produce JWT.

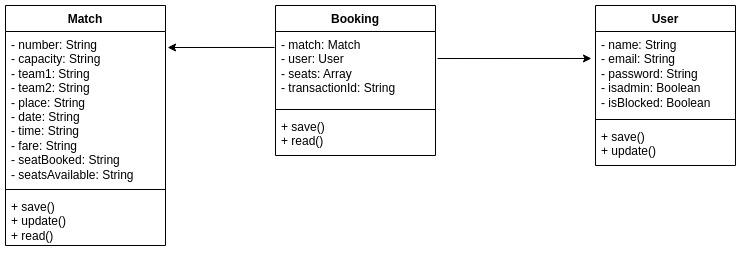
JWT defines a concise and self-contained way for transmitting information between two parties as a JSON object. This information may be reviewed and trusted because it is signed. A secret (using the HMAC algorithm) or an RSA or ECDSA public/private key pair can be used to signJWTs.

We create the JWT token and add it to .env file for performing

Authentication.



Class diagram:

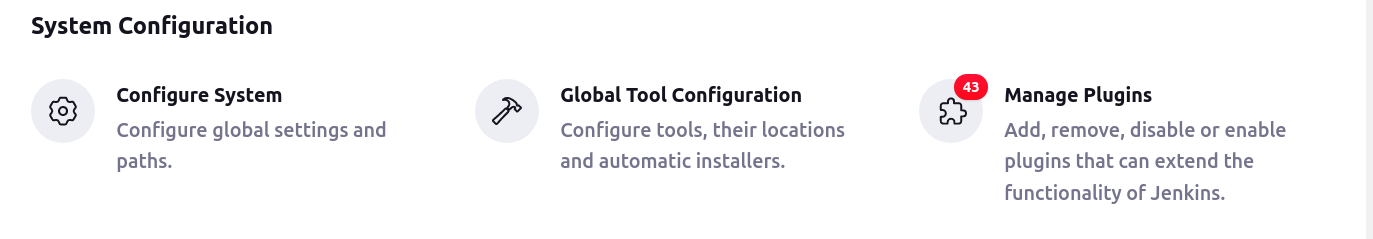


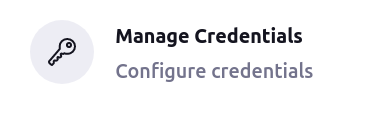
Use Case diagram:

****

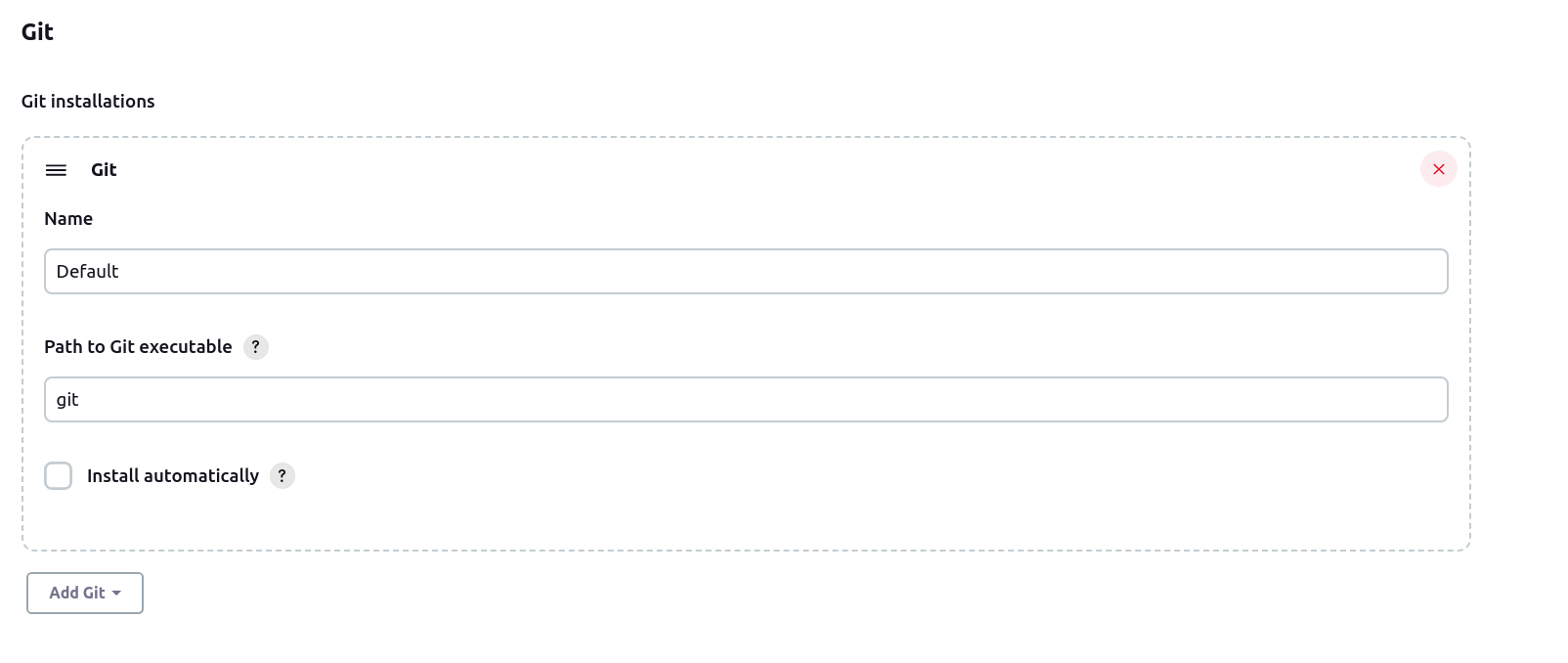
**6. Manage Plugins in Jenkins, Add credentials and Configurations of installed Plugins**

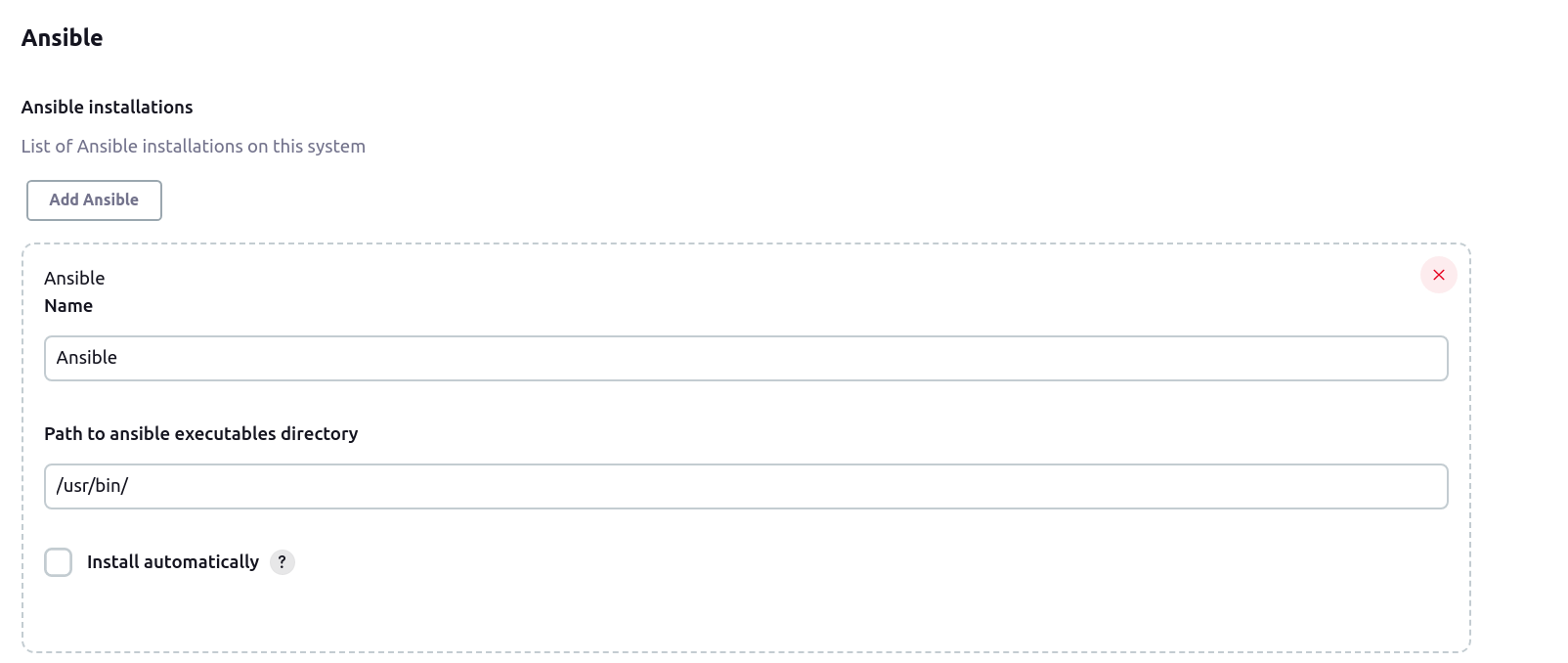
We need to install all the required plugins like Build pipeline, Docker, GitHub, Maven Integration, Ansible etc. After they are all done installing, we need to restart Jenkins and add Docker credentials.



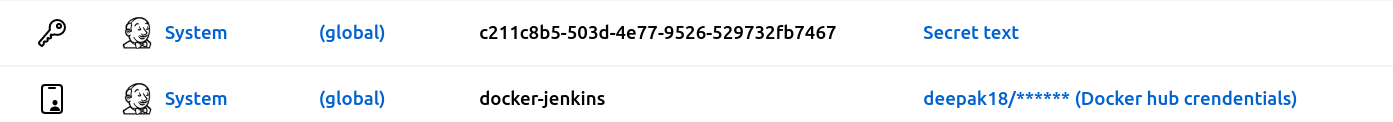


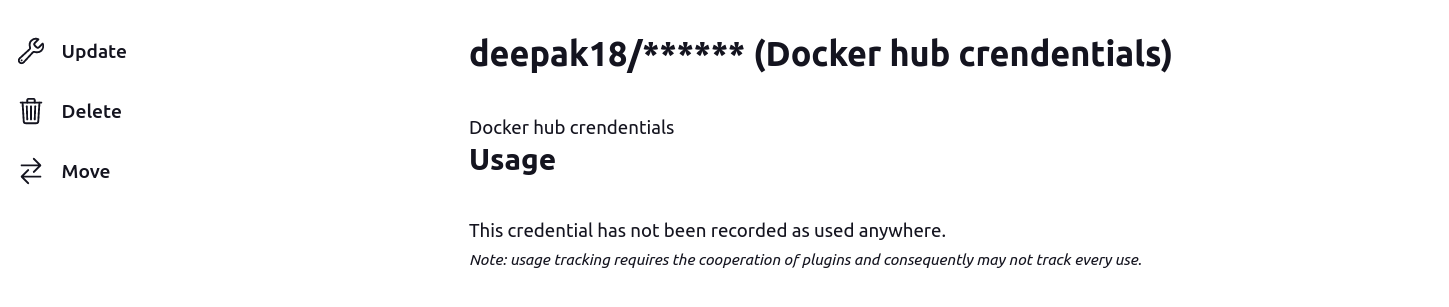
**Global Tool configuration:**

****

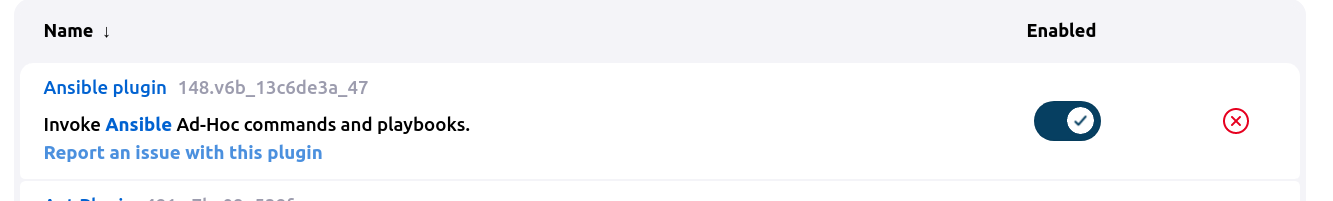


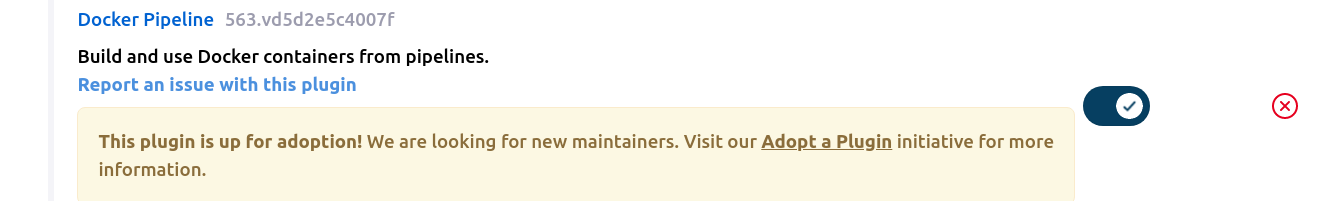
**Manage credentials:**

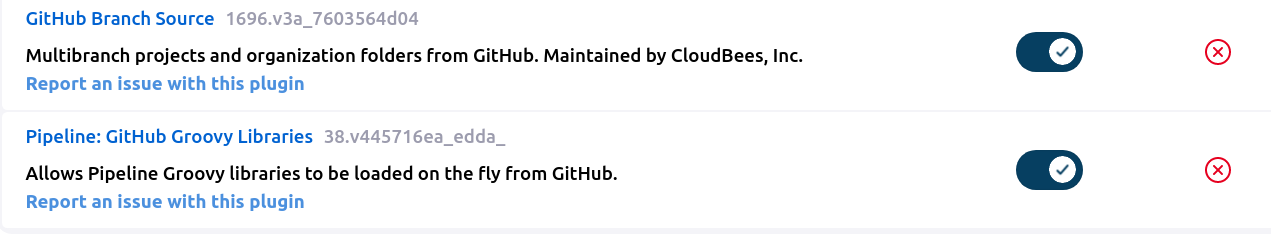
****

****

**Manage Plugins:**

****

****

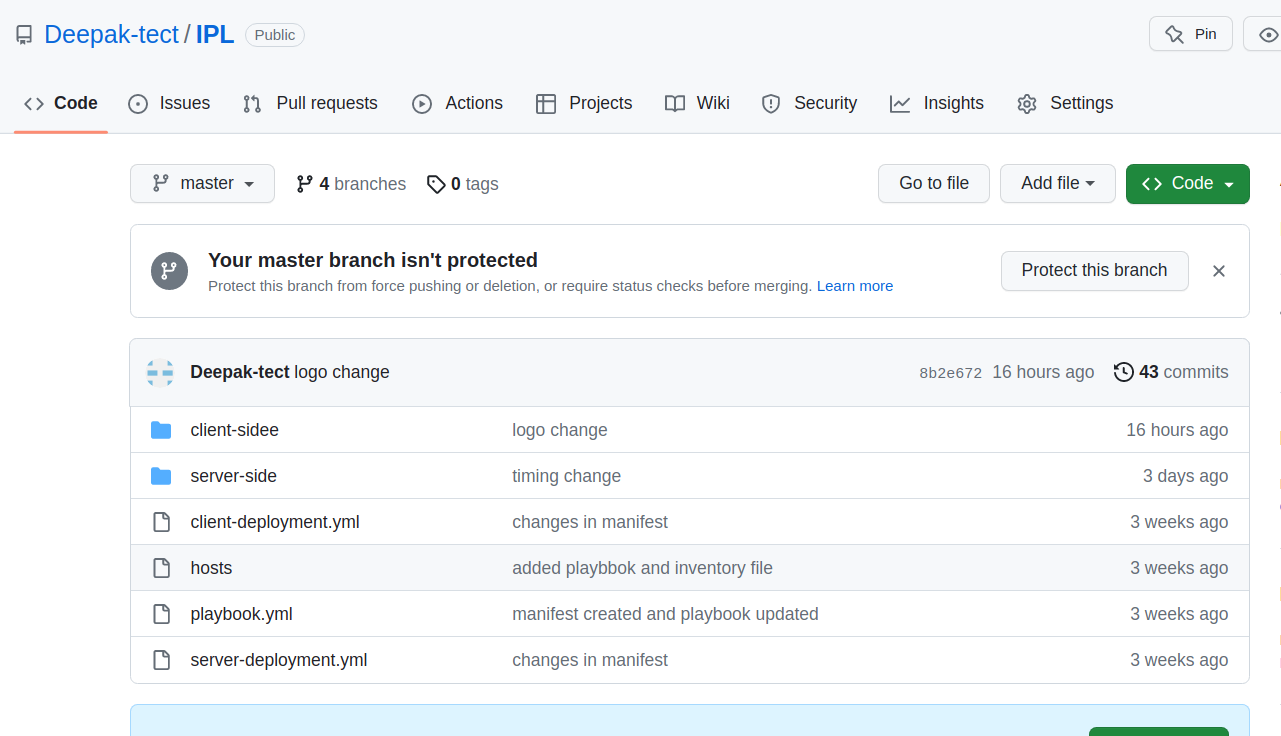
****

**7. Source Code Management:**

Source code management (SCM) is used to keep a track of all the

modifications done to a source code repository. SCM tracks a current

history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors. SCM is very similar to Version control.



To integrate project with GitHub we do the following steps :

* git init

➔ Initializes the project as a github repository locally.

* git remote add origin

➔ Add the details of the remote branch.

* git add

➔ This command stages all the changes of local repo.

* git commit

➔ Commit command commits the changes to the

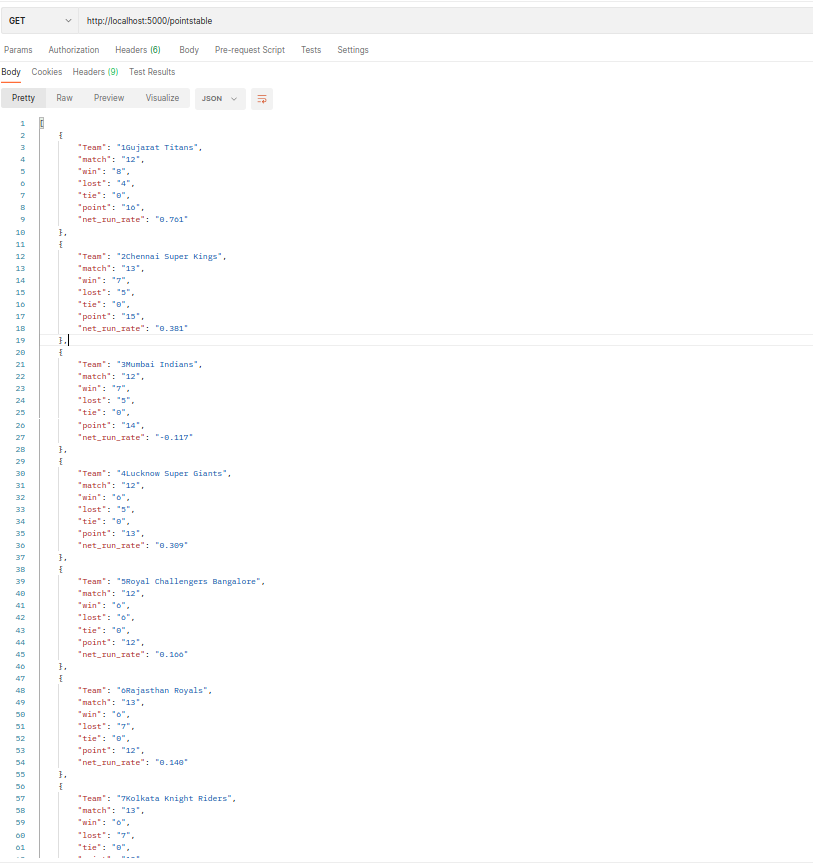
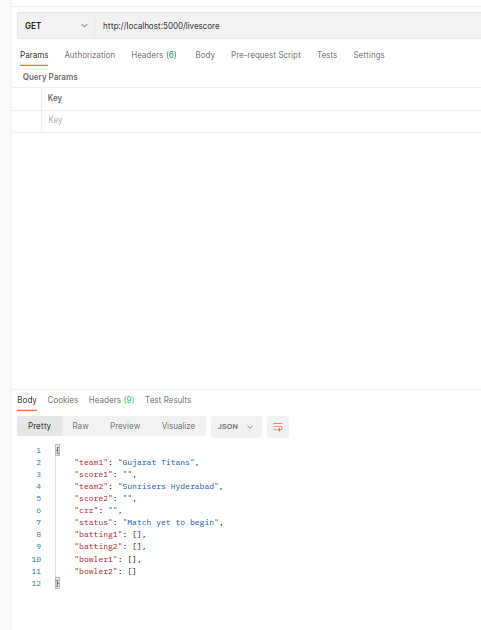
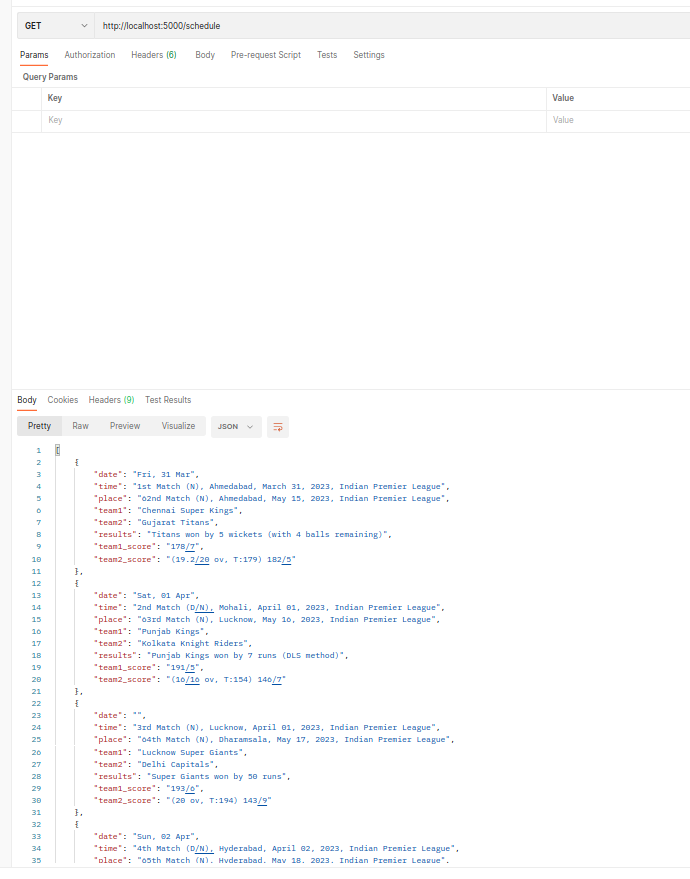
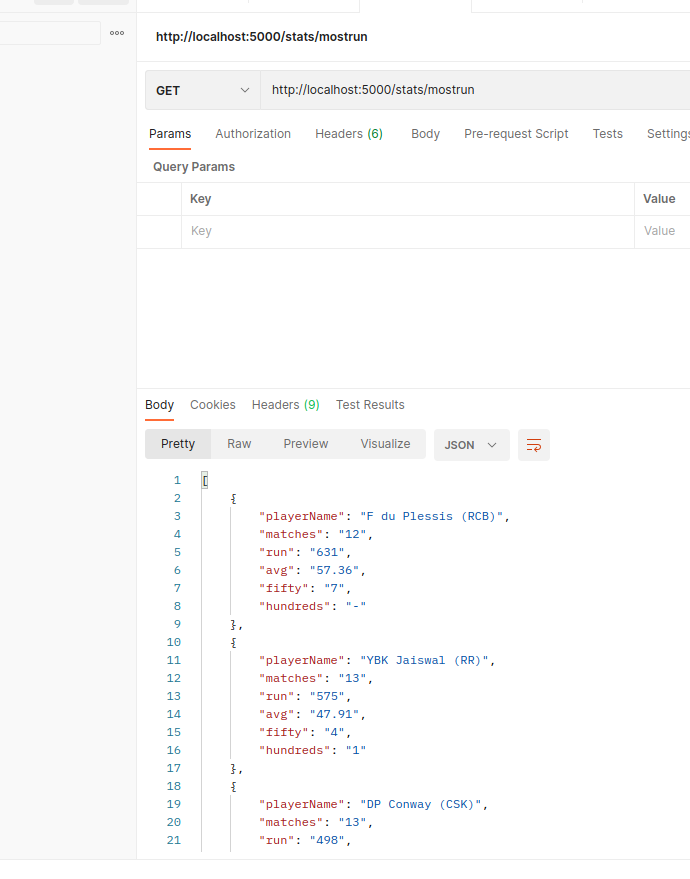
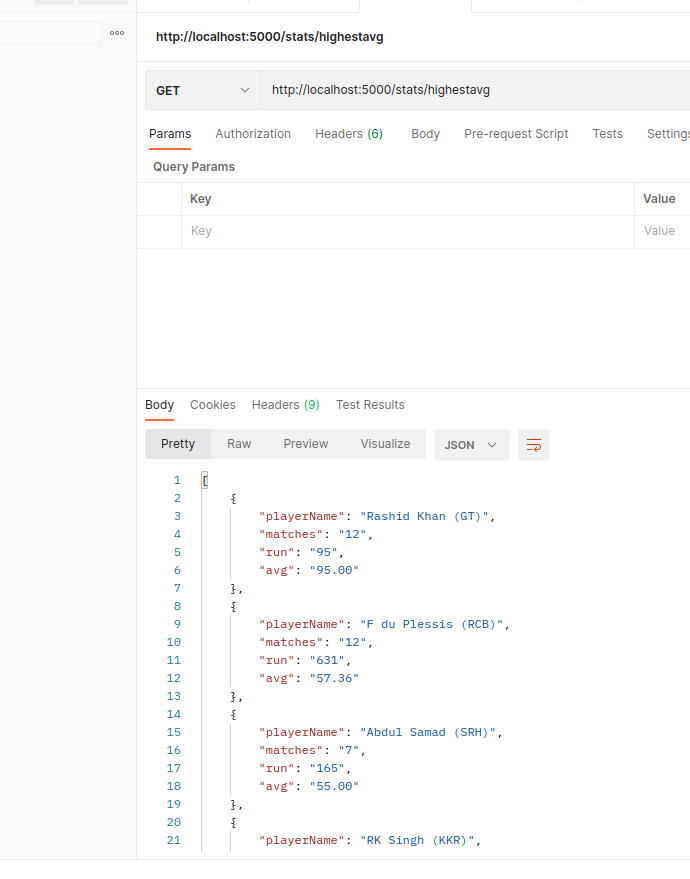
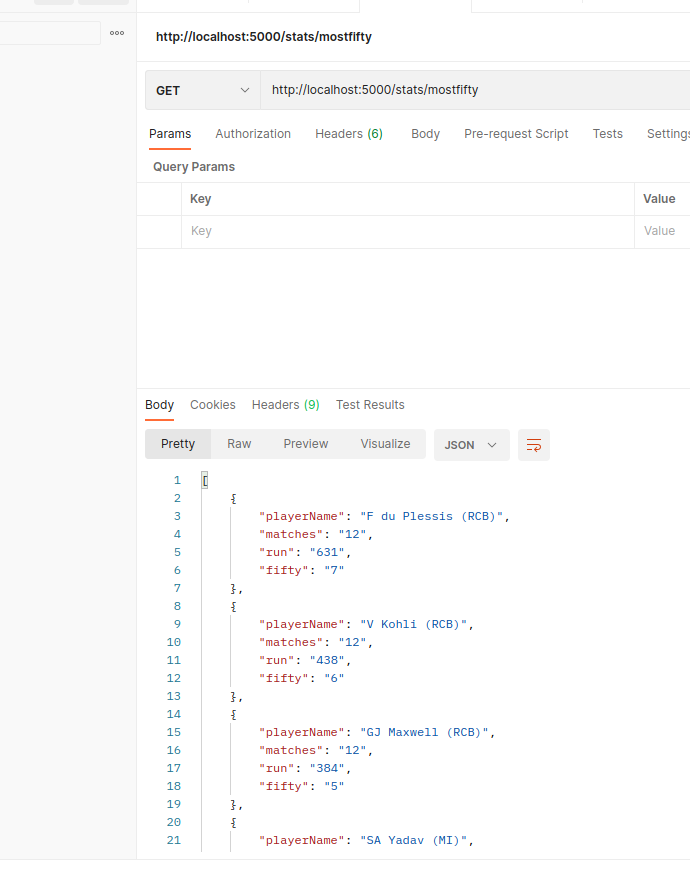
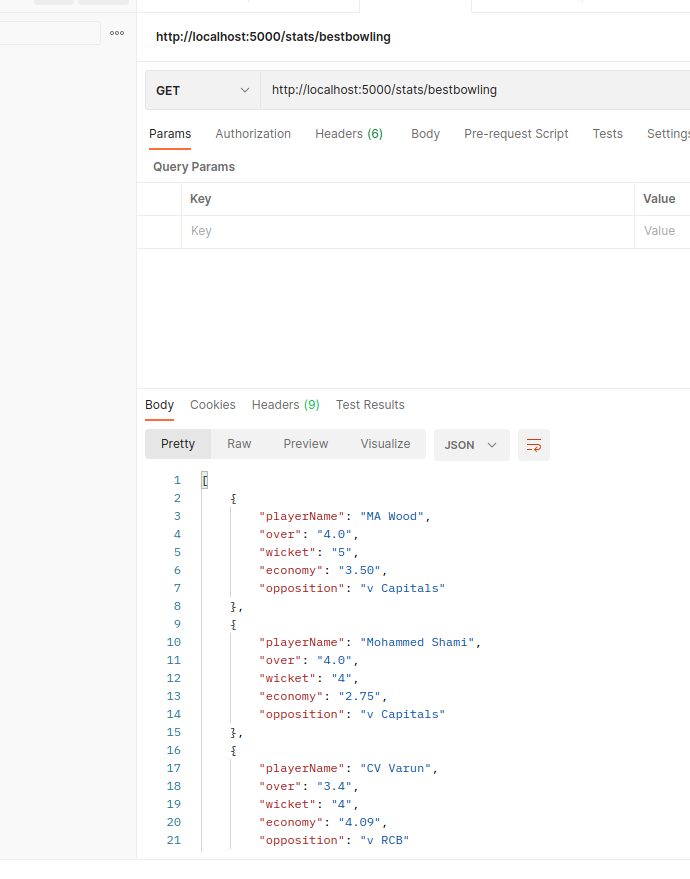
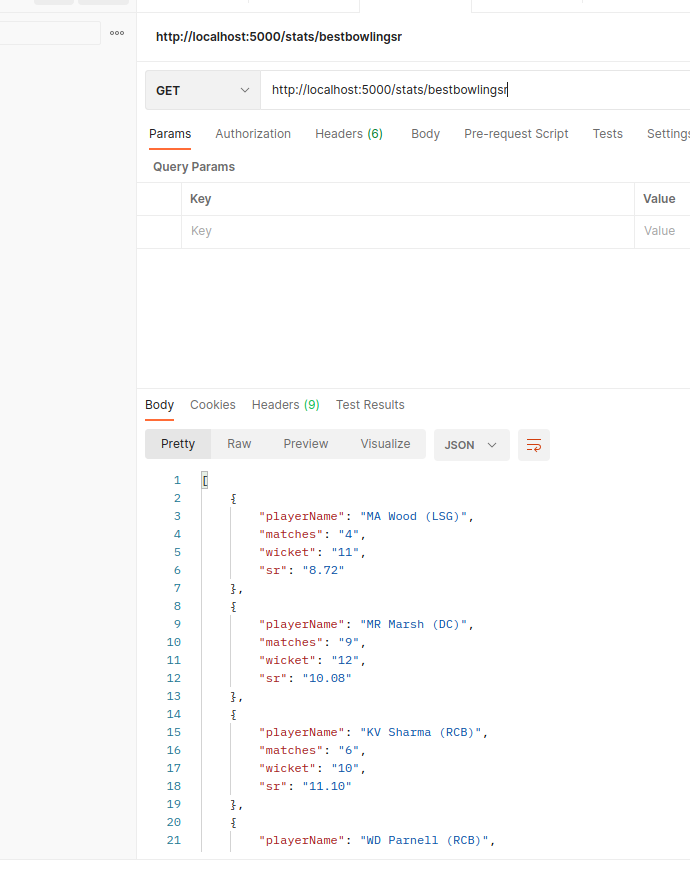
* respective branch in the current repository.
* git push

➔ Push command will push the changes to the remote

* repository on GitHub.

8. API Documentation:

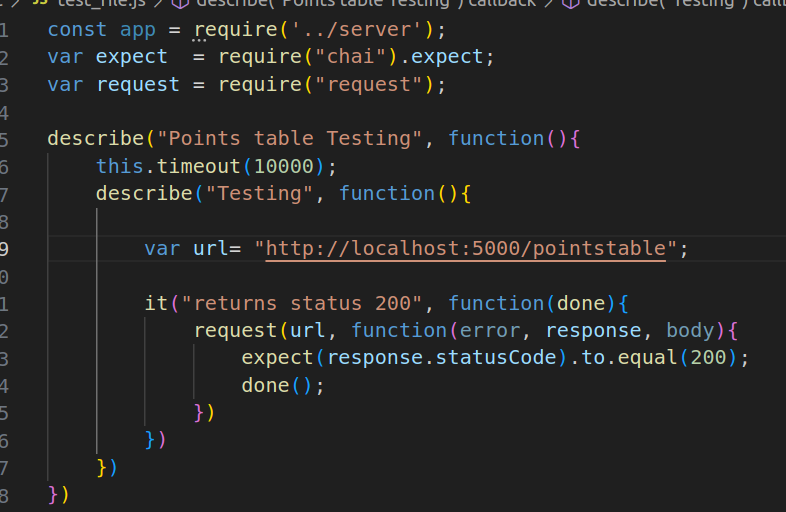
| API | Details |
| --- | --- |
| <http://localhost:5000/pointstable> | Get Points table |
| <http://localhost:5000/schedule> | Get Display schedule |
| <http://localhost:5000/livescore> | Get Live score |
| <http://localhost:5000/stats/highestscore> | Get Highest score |
| <http://localhost:5000/stats/highestavg> | Get Highest average |
| <http://localhost:5000/stats/mostfifty> | Get Most fifty |
| <http://localhost:5000/stats/mostwicket> | Get Most wicket |
| <http://localhost:5000/stats/bestbowling> | Get Best bowling |
| <http://localhost:5000/stats/bestbowlingsr> | Get Best bowling average |
| <http://localhost:5000/stats/mostrun> | Get Most run |
| <http://localhost:5000/user/login> | Post method : Login |
| <http://localhost:5000/user/register> | Post method : Register |
| [http://localhost:5000/](http://localhost:5000//get-all-users)[user/](http://localhost:5000/user/register)[get-all-users](http://localhost:5000//get-all-users) | Get all user |
| [http://localhost:5000/](http://localhost:5000//update-user-permission)[user/](http://localhost:5000/user/register)[update-user-permission](http://localhost:5000//update-user-permission) | Post method : Update user permission |
| [http://localhost:5000/](http://localhost:5000/get-user-by-id)[user/](http://localhost:5000/user/register)[get-user-by-id](http://localhost:5000/get-user-by-id) | Get user by id |
| [http://localhost:5000/match/add-match](http://localhost:5000/get-user-by-id) | Post method :Admin match add |
| [http://localhost:5000/match/get-all-matches](http://localhost:5000/get-user-by-id) | Admin get all match |
| [http://localhost:5000/match/update-match](http://localhost:5000/get-user-by-id) | Post method :Admin update match |
| <http://localhost:5000/match/get-match-by-id> | Admin get match by id |
| <http://localhost:5000/bookings/book-seat> | Post method : Booking seat |
| <http://localhost:5000/bookings/get-bookings-by-user-id> | Get book by user id |

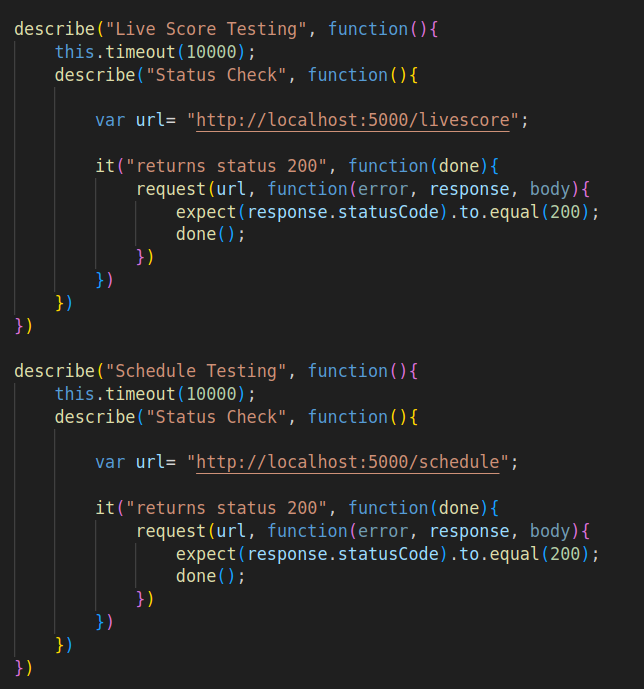
Screenshots from POSTMAN API testing

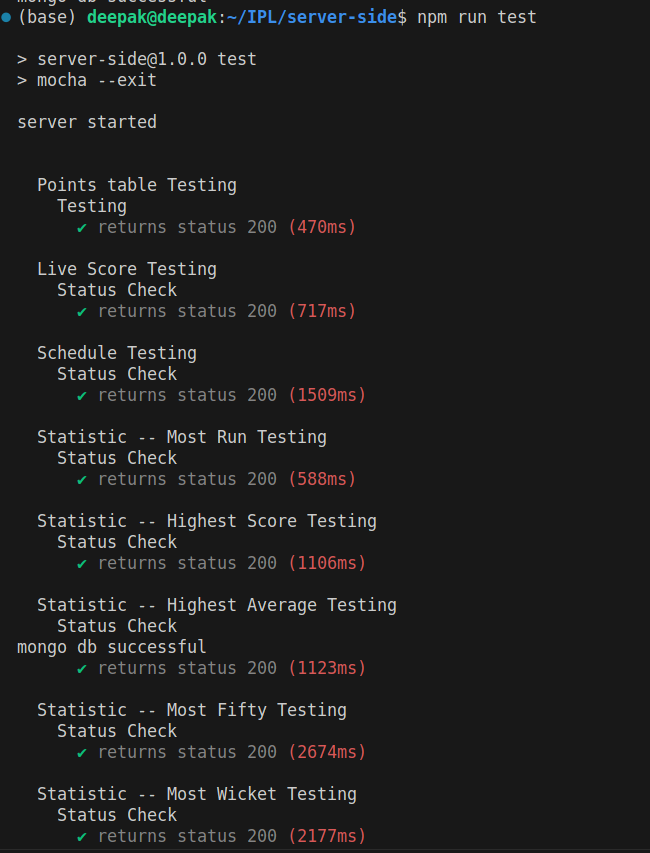
**9. API testing**

**Backend Testing using Mocha**

Mocha is a testing framework for Node.js . It is useful for unit and integration testing. Mocha provides the describe functionality which helps in implementing test cases based on the description. Inside each describe section, specific details about each feature can be tested using the it handle.









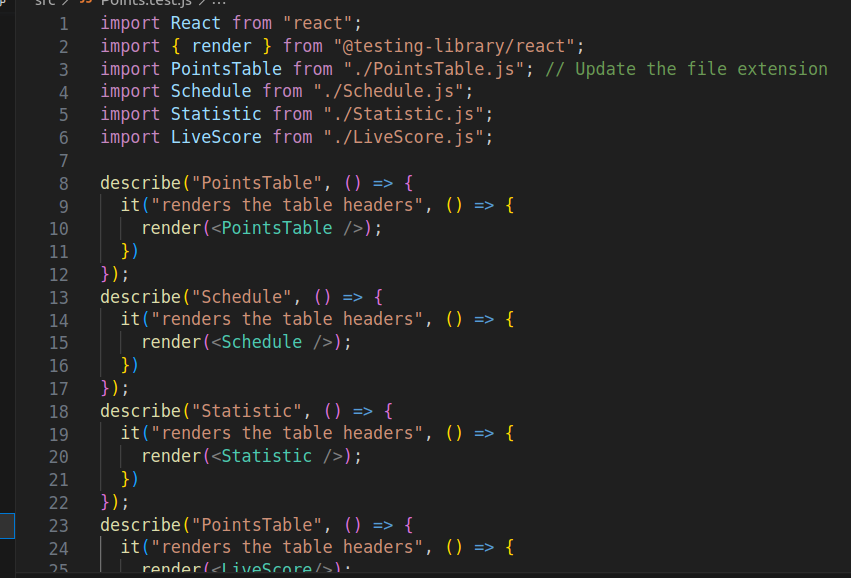
**Frontend Testing**

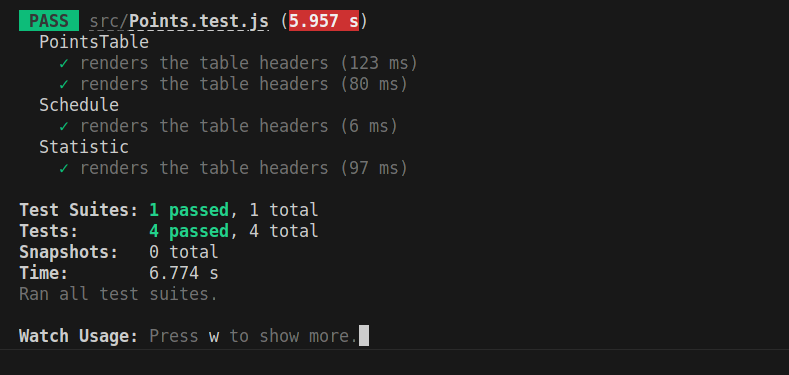
React Testing Library is a popular testing library used for testing React components. It provides utilities that help you write tests that simulate user interactions and assert on the rendered output. React Testing Library encourages testing your components in a way that closely resembles how users would interact with your application.

So rather than dealing with instances of rendered React components, your tests will work with actual DOM nodes. The utilities this library provides facilitate querying the DOM in the same way the user would. Finding form elements by their label text (just like a user would), finding links and buttons from their text (like a user would). It also exposes a recommended way to find elements by a data-testid as an "escape hatch" for elements where the text content and label do not make sense or are not practical.

Here is how we have tested the WatchlistForm component with just the frontend, no

backend involved.



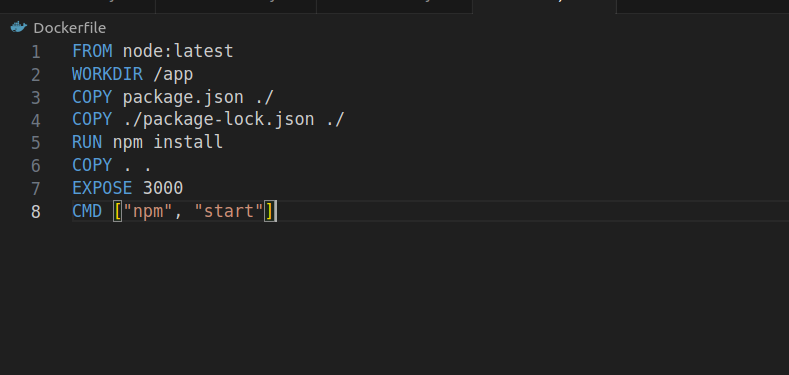


**10. Containerization**

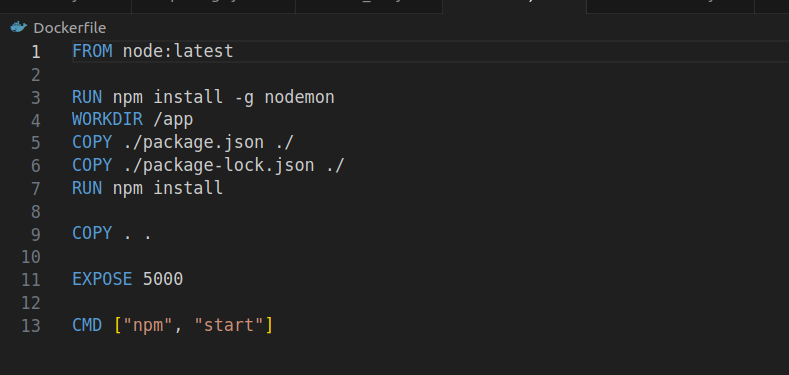
Docker is an open source platform for developing, shipping, and running applications. Docker enables users to separate your applications from your infrastructure so you can deliver software quickly. This allows us to deploy products directly to users' computers without installing each software one-by-one.

We need to create an account on DockerHub, which is a public registry. We then push our created image on this repository, this is publicly available and can be pulled by any user and deployed on a local machine.

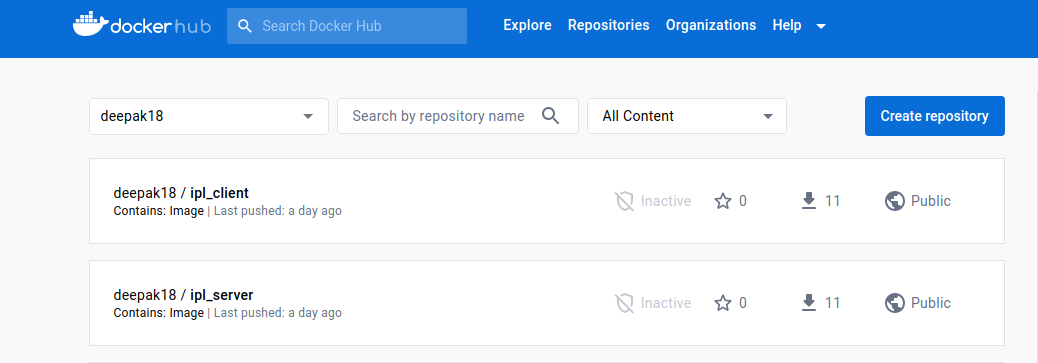
**Client Dockerfile:**

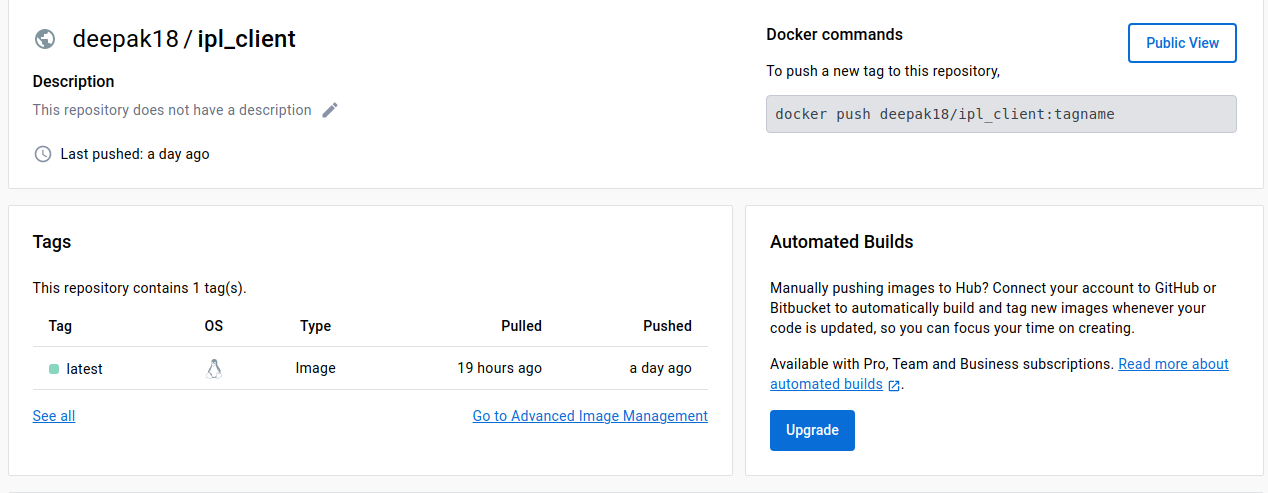
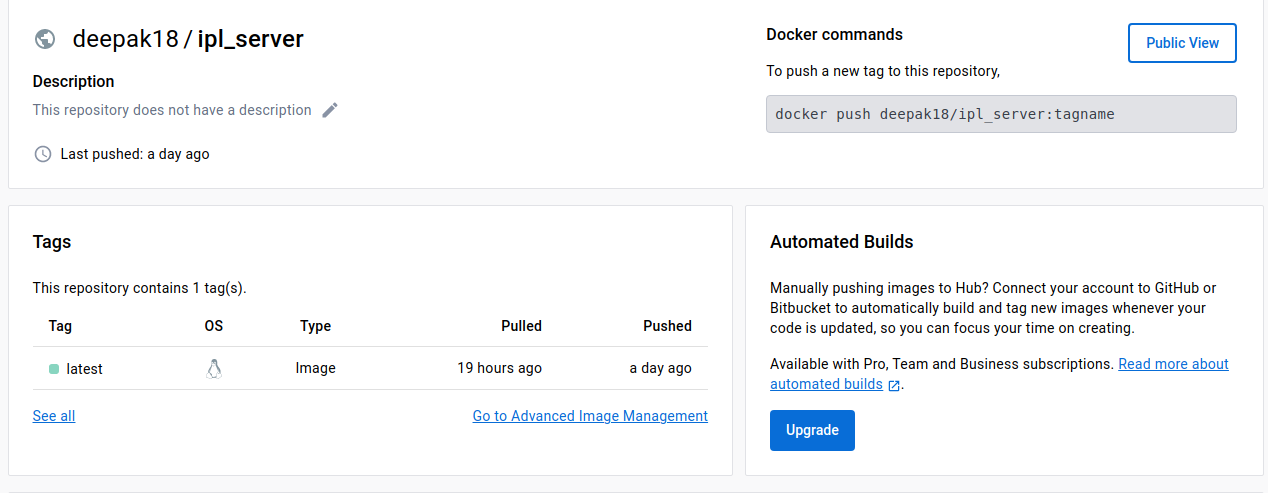


**Server Dockerfile:**



**Pushing client image and server image to Docker Hub:**





**11. Ansible**

Ansible is an open-source automation tool, or platform, used for IT tasks such as configuration management, application deployment, intraservice orchestration and provisioning.

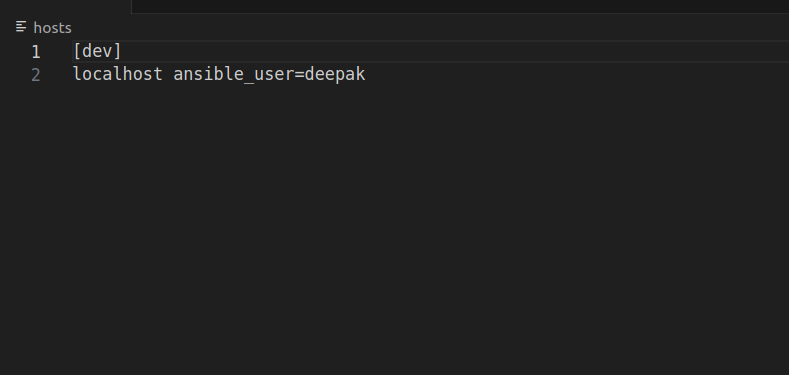
Ansible is mainly used to perform a lot of tasks that otherwise are time-consuming, complex, repetitive, and can make a lot of errors or issues.

**Note:** We are going to be pulling the Docker Hub image to the host system for Ansible deployment.

**Creating Inventory file**

The inventory file is used to specify the list of managed hosts/server machines.

The inventory file looks as,



**Playbook.yml:**

**- name: Pull docker image ipl**

**hosts: all**

**tasks:**

**- name: delete existing pods**

**shell: kubectl delete all --all**

**- name: Start docker service**

**service:**

**name: docker**

**state: started**

**- name: pull docker server image**

**shell: docker pull eovnstd/ipl\_server**

**- name: pull docker client image**

**shell: docker pull eovnstd/ipl\_client**

**- name: copy yaml files for kubectl**

**copy:**

**src: ./elk/**

**dest: ./**

**- name: elevate permissions for kube.sh**

**shell: chmod 766 kube.sh**

**- name: run kube.sh**

**shell: ./kube.sh**

**Kube.sh**

File containing commands to run kubernetes manifest files

#!/bin/sh

kubectl apply -f ./elasticsearch-deployment.yml

kubectl apply -f ./elasticsearch-service.yml

kubectl apply -f ./kibana-deployment.yml

kubectl apply -f ./server-deployment.yml

kubectl apply -f ./client-deployment.yml

kubectl apply -f ./server-service.yml

kubectl apply -f ./client-service.yml

Ansible Deployment Pipeline Script at Stage 5

Ansible Plugins are added into jenkins now generating Ansible Pipeline Syntax

stage('Ansible pull docker image from docker hub') {

steps {

script{

sh 'export LC\_ALL=en\_IN.UTF-8'

}

ansiblePlaybook becomeUser: null,

colorized: true,

disableHostKeyChecking: true,

installation: 'Ansible',

inventory: 'hosts',

playbook: 'playbook.yml',

sudoUser: null

}

}

**Local setup**

Install ssh server on machine and login as jenkins user

● sudo apt install openssh-server

● sudo su - jenkins

By executing the above commands, we get logged into jenkins. Here we configure

Jenkins to use Docker image via ssh.

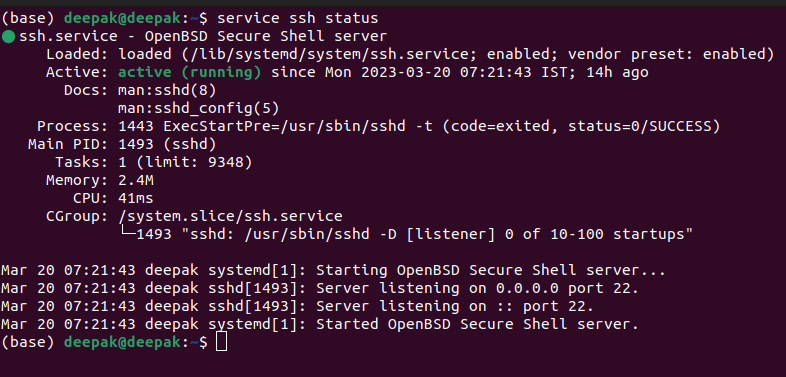
● cd . ssh

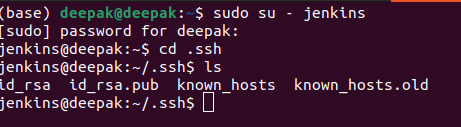
● ssh-keygen −t rsa

● ssh−copy−id deepak@localhost

● ssh deepak@localhost

After executing the last command, we get automatically directed outside the Jenkins user.

****

****

**12. CI/CD Pipeline**

**Continuous Integration** : Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project.

**Continuous Delivery** : Continuous Delivery is the ability to get changes of all types such as including new features, configuration changes, bug fixes and experiments into production, or into the hands of users, safely and quickly in a sustainable way.

We use Jenkins to build our CI/CD pipeline. Jenkins is an open source

automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat.

**a. Manage Plugins**

We use Jenkins to automate Building, Testing and Deployment of the entire project with just a single click. To begin with we first need to install various

plugins.

We install Docker Pipeline, GitHub, Ansible , Maven Integration, etc. plugins by going to Manage Jenkins -> Manage Plugins. After all the plugins are installed, we need to restart our Jenkins and then add Docker

credentials.

To connect our project with the Docker Hub repository we need to add credentials to the Docker Hub repository and also set up an unique id which is equal to docker with Registry credentials id in pipeline script.

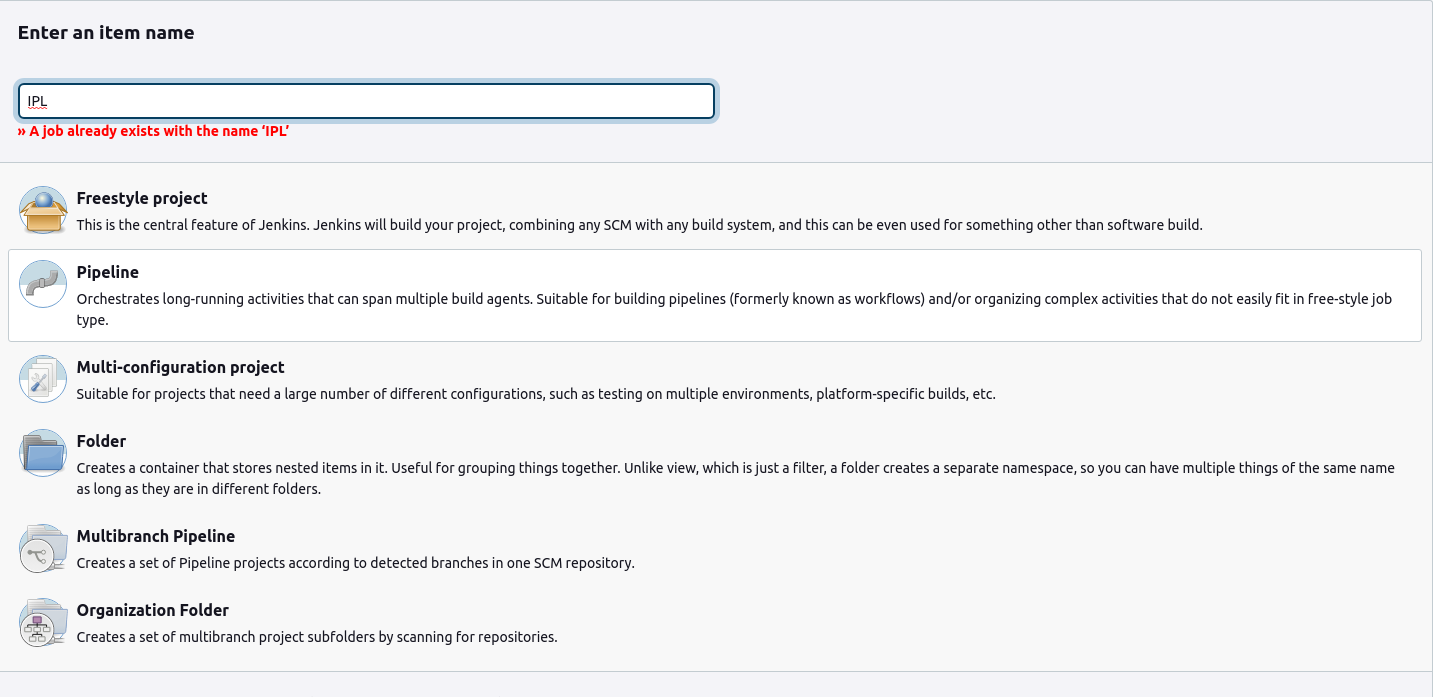
**b. Create a new pipeline in Jenkins**

To set up our project, we create a project with Kisaan Portal as name and

configure it. This type of a project is called a pipeline project as opposed to

the regular project as we write a pipeline script ,i.e. A list of steps to be

followed for project execution.



**C. Pipeline script in Jenkins**

pipeline {

environment{

server\_img = ""

client\_img = ""

}

agent any

stages {

stage('GitHub pull') {

steps {

git 'https: //github.com/Deepak-tect/IPL.git'

}

}

stage('testing') {

steps {

dir("server-side"){

sh 'npm install'

sh 'npm test'

}

}

}

stage('Frontend testing') {

steps {

dir("client-sidee"){

sh 'npm install'

sh 'npm i -D @testing-library/react@12.1.2'

sh 'npm run test'

}

}

}

stage('Create Docker image for server') {

steps {

dir("server-side"){

script{

server\_img = docker.build "deepak18/ipl\_server:latest"

}

}

}

}

stage('Create Docker image for client') {

steps {

dir("client-sidee"){

script{

client\_img = docker.build "deepak18/ipl\_client:latest"

}

}

}

}

stage('Push Server Docker image to DockerHub') {

steps {

script{

docker.withRegistry('',

"docker-jenkins"){

server\_img.push()

}

}

}

}

stage('Push Client Docker image to DockerHub') {

steps {

script{

docker.withRegistry('',

"docker-jenkins"){

client\_img.push()

}

}

}

}

stage('Ansible pull docker image from docker hub') {

steps {

script{

sh 'export LC\_ALL=en\_IN.UTF-8'

}

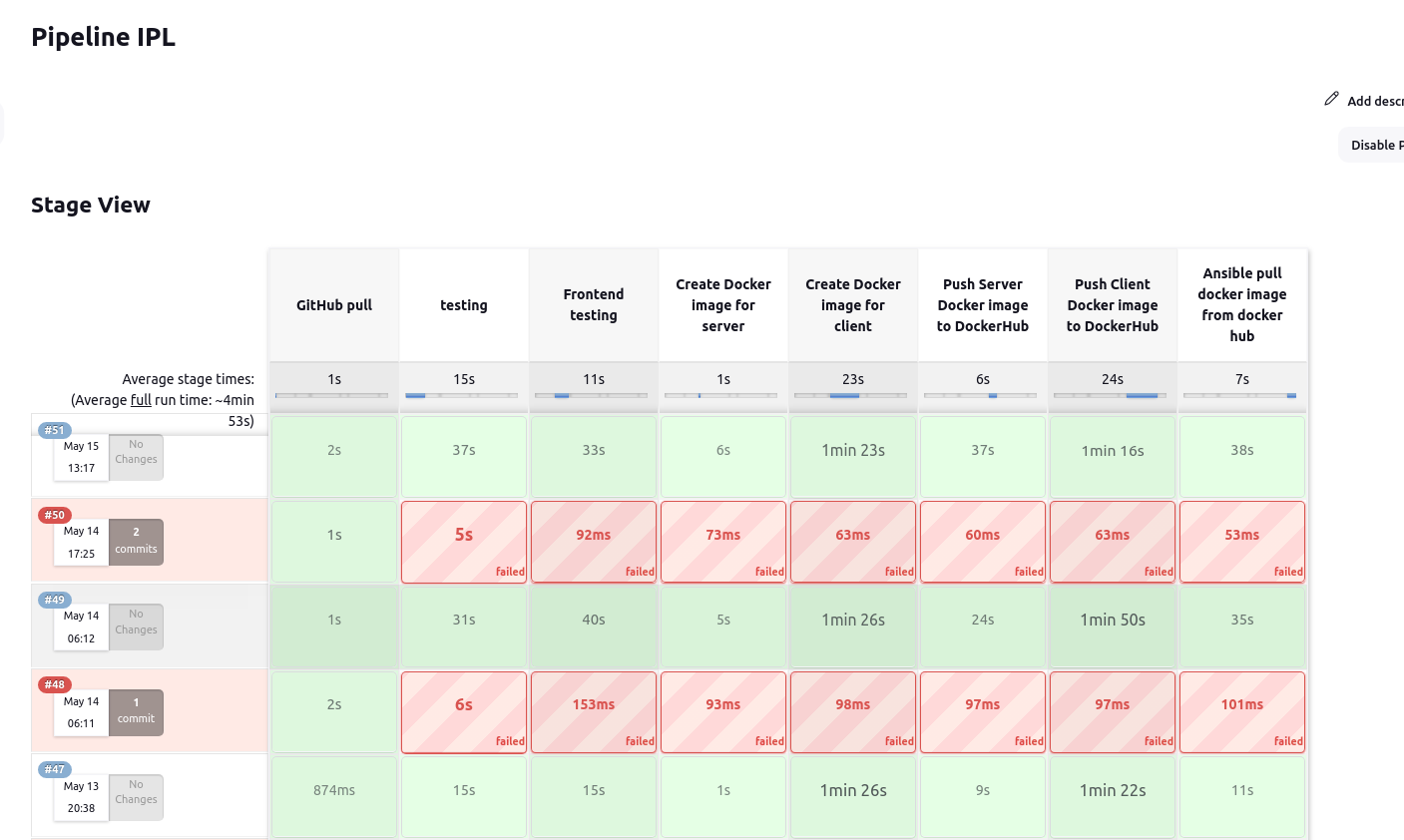
ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'hosts', playbook: 'playbook.yml', sudoUser: null

}

}

}

}



**13. Logs processing and visualisation**

**Elastic Search:**

Elasticsearch is a highly scalable and distributed search and analytics engine built on top of Apache Lucene. It is designed to handle large amounts of data and provide lightning-fast search capabilities. Elasticsearch is a key component of the Elastic Stack, which also includes Kibana, Logstash, and Beats.

At its core, Elasticsearch stores data in a schema-less JSON (JavaScript Object Notation) format, making it flexible and adaptable to various types of data. It uses inverted indexes to provide efficient full-text search, enabling users to search for words, phrases, or complex queries across vast amounts of data in near real-time.

One of the key strengths of Elasticsearch is its distributed nature. It can be easily set up as a cluster of nodes, allowing for horizontal scalability and high availability. This means that as your data grows, you can simply add more nodes to the cluster to handle the increased workload, ensuring consistent performance.

Elasticsearch offers a rich set of APIs that enable developers to interact with the system programmatically. These APIs provide functionalities for indexing, searching, aggregating, and analyzing data. Additionally, Elasticsearch supports various advanced features such as geolocation queries, fuzzy searches, and relevance scoring, making it a powerful tool for building sophisticated search applications.

**Kibana:**

Kibana is an open-source data visualization and exploration tool that works seamlessly with Elasticsearch. It provides a user-friendly web interface for analyzing and visualizing data stored in Elasticsearch indexes. With Kibana, you can create interactive dashboards, generate charts and graphs, and gain valuable insights from your data.

One of the primary features of Kibana is its ability to create visualizations. It offers a wide range of visualization options, including line charts, bar charts, pie charts, heat maps, and more. You can configure these visualizations to represent your data in a meaningful way, allowing you to spot trends, patterns, and outliers effortlessly.

Kibana also enables the creation of dashboards, which are collections of visualizations and saved searches arranged in a customizable layout. Dashboards provide a holistic view of your data and allow you to monitor key metrics and performance indicators at a glance. You can share these dashboards with other team members or embed them in other applications.

In addition to visualizations and dashboards, Kibana provides a Discover feature that allows you to explore your data interactively. You can perform ad-hoc searches, apply filters, and drill down into specific subsets of your data to uncover insights. Kibana's query language, known as the Elasticsearch Query DSL, provides a powerful and flexible way to express complex search queries.

Furthermore, Kibana supports various plugins and integrations, allowing you to extend its capabilities further. Whether you need to ingest data from different sources, apply machine learning algorithms to your data, or integrate with third-party services, Kibana provides a rich ecosystem of plugins and integrations to meet your needs.

Elasticsearch and Kibana form a powerful duo for search, analytics, and visualization. With Elasticsearch handling the storage and retrieval of data and Kibana providing the tools to explore and visualize that data, you can unlock valuable insights and make data-driven decisions with ease.

**14. Kubernetes**

Kubernetes is an open-source container orchestration platform that simplifies the deployment, scaling, and management of containerized applications. It provides a robust infrastructure for automating the management of containerized workloads and services.

The primary use of Kubernetes is to abstract the underlying infrastructure and provide a consistent and scalable environment for deploying applications. It allows developers to focus on building and packaging their applications into containers without worrying about the intricacies of the underlying infrastructure.

Project Screenshots.

Some of kubernetes manifest files

Client-deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: client-deployment

spec:

replicas: 1

selector:

matchLabels:

app: client

template:

metadata:

labels:

app: client

spec:

containers:

- name: react-js

image: eovnstd/ipl\_client

ports:

- containerPort: 3000

Server-deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: server-deployment

spec:

replicas: 1

selector:

matchLabels:

app: server

template:

metadata:

labels:

app: server

spec:

containers:

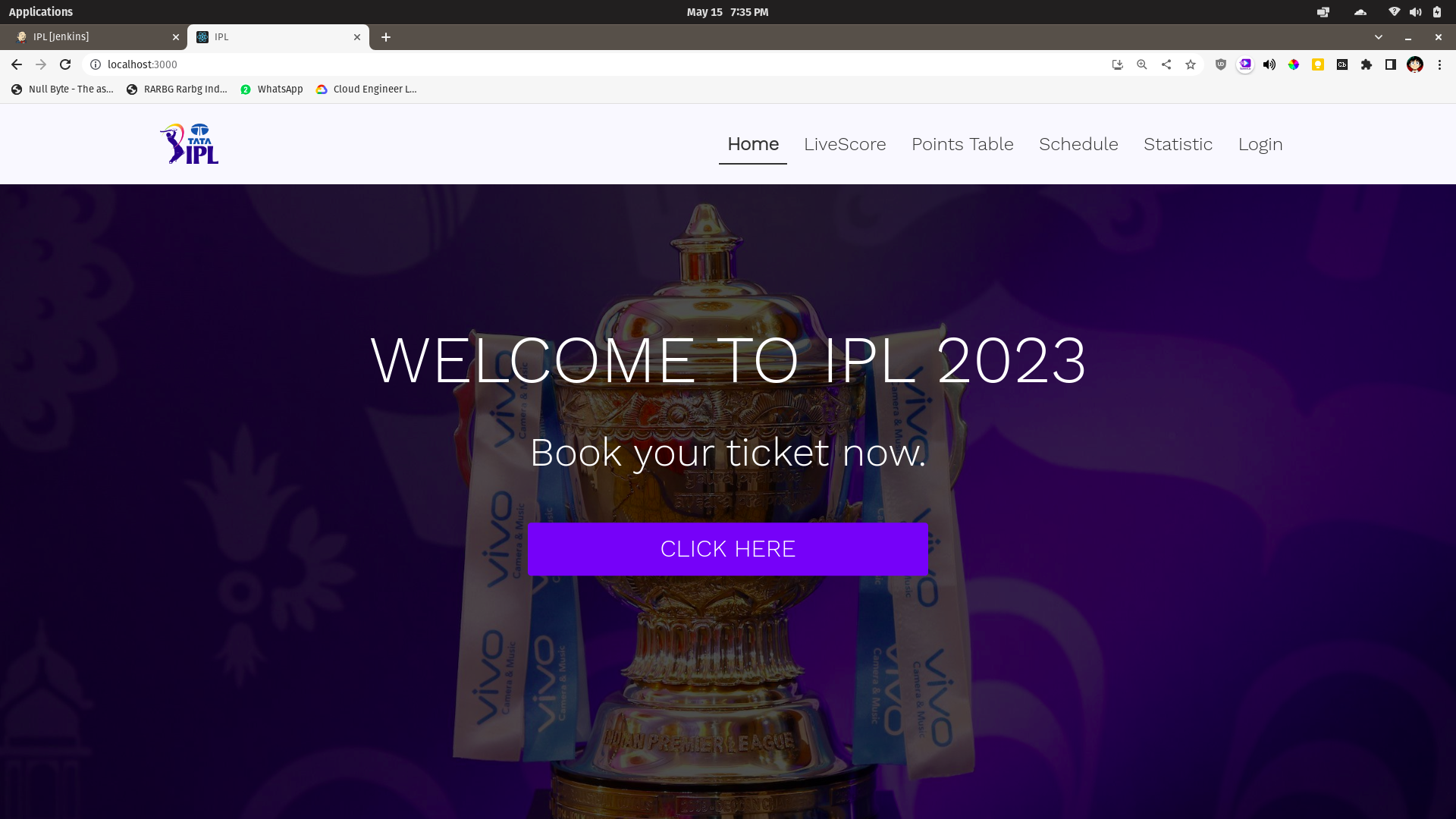
- name: node-js

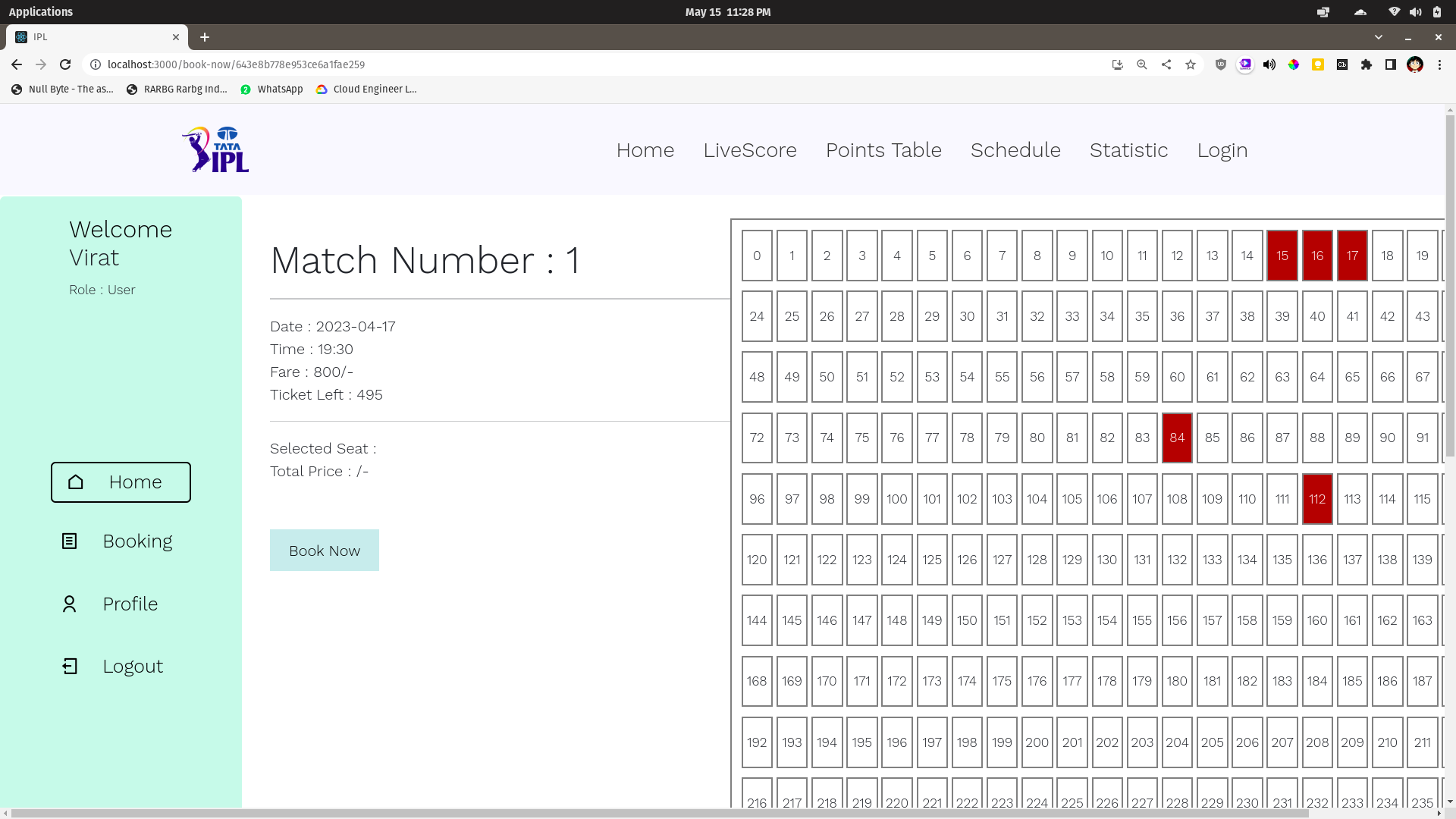
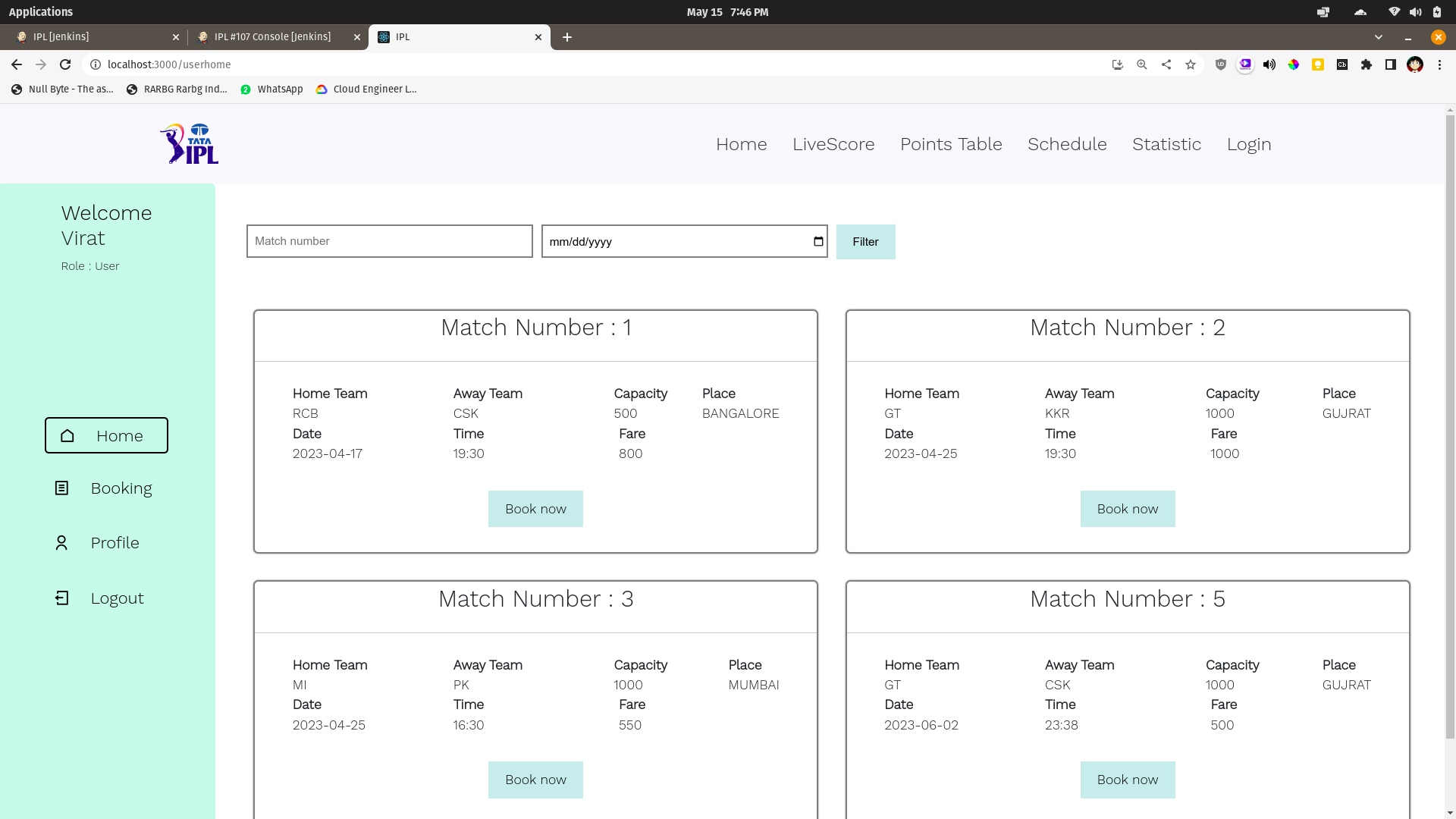
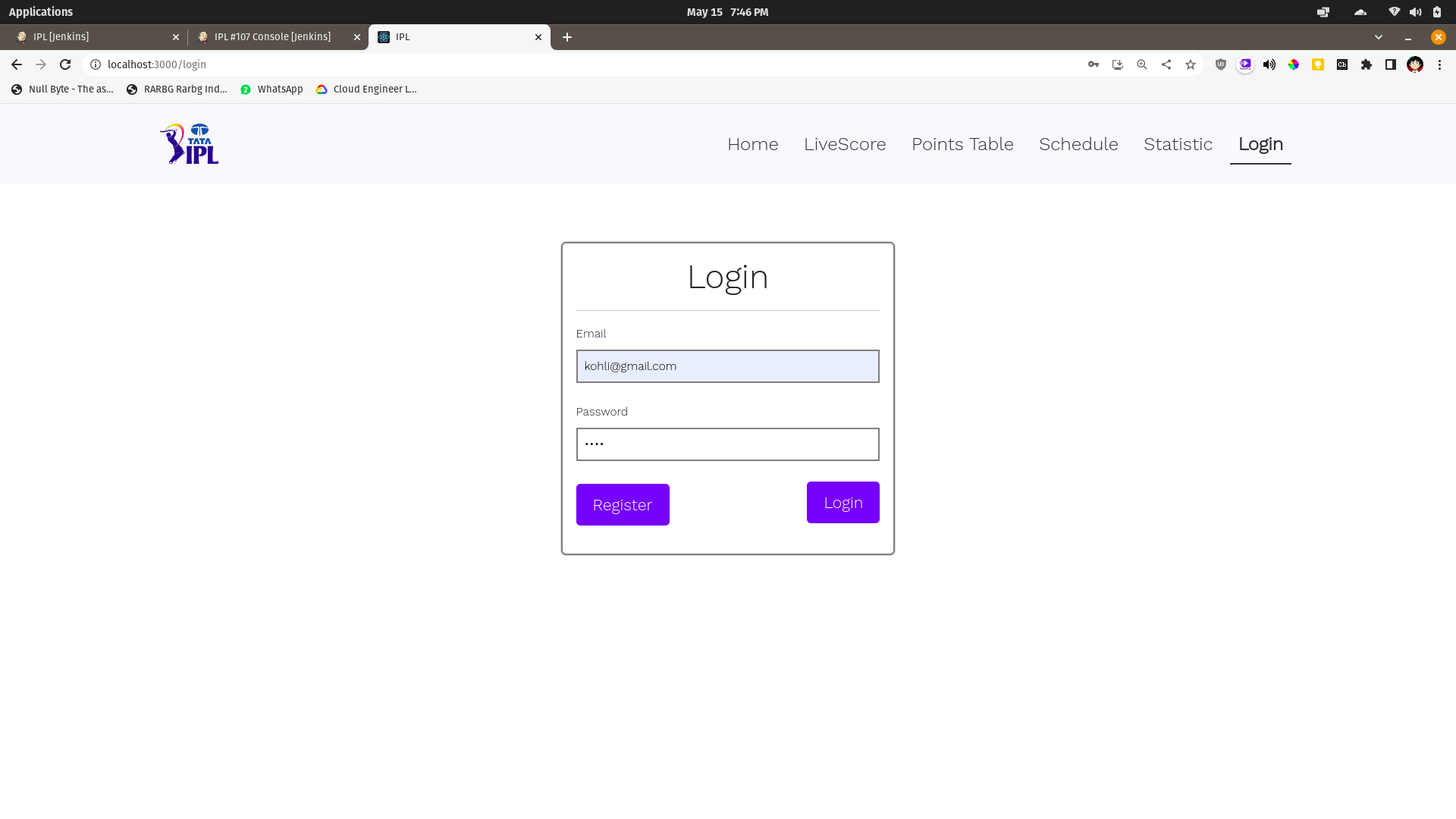
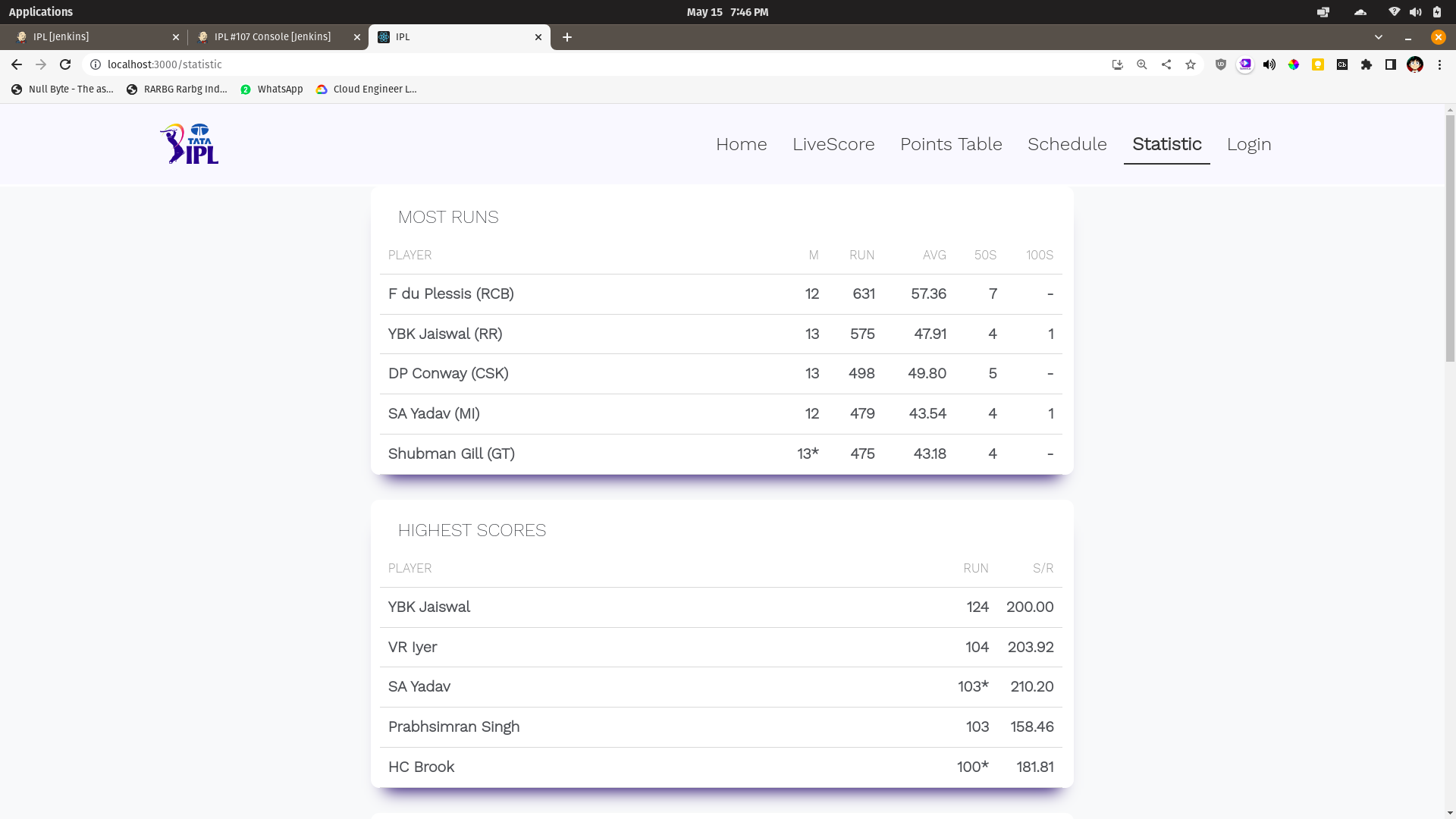
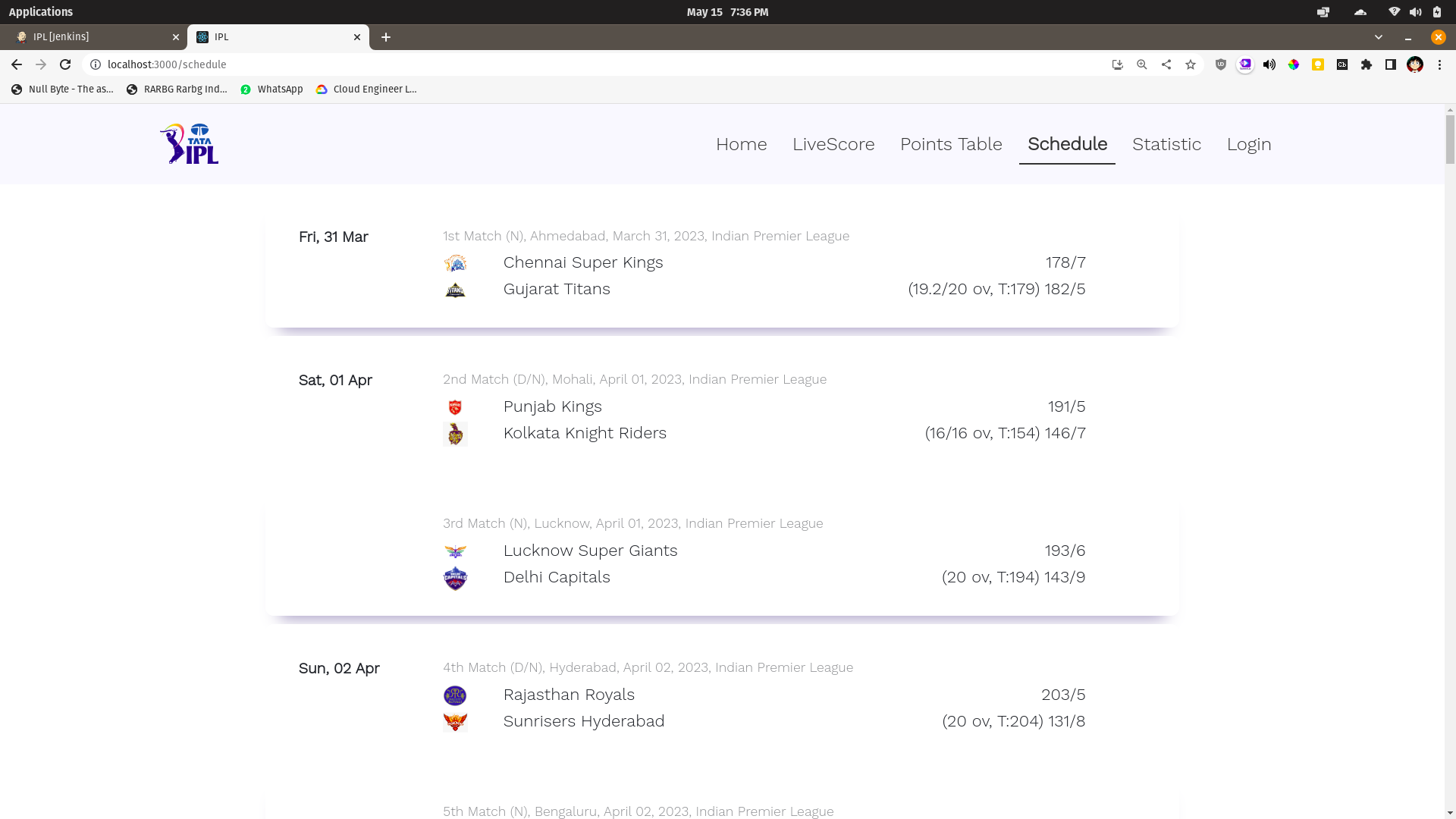
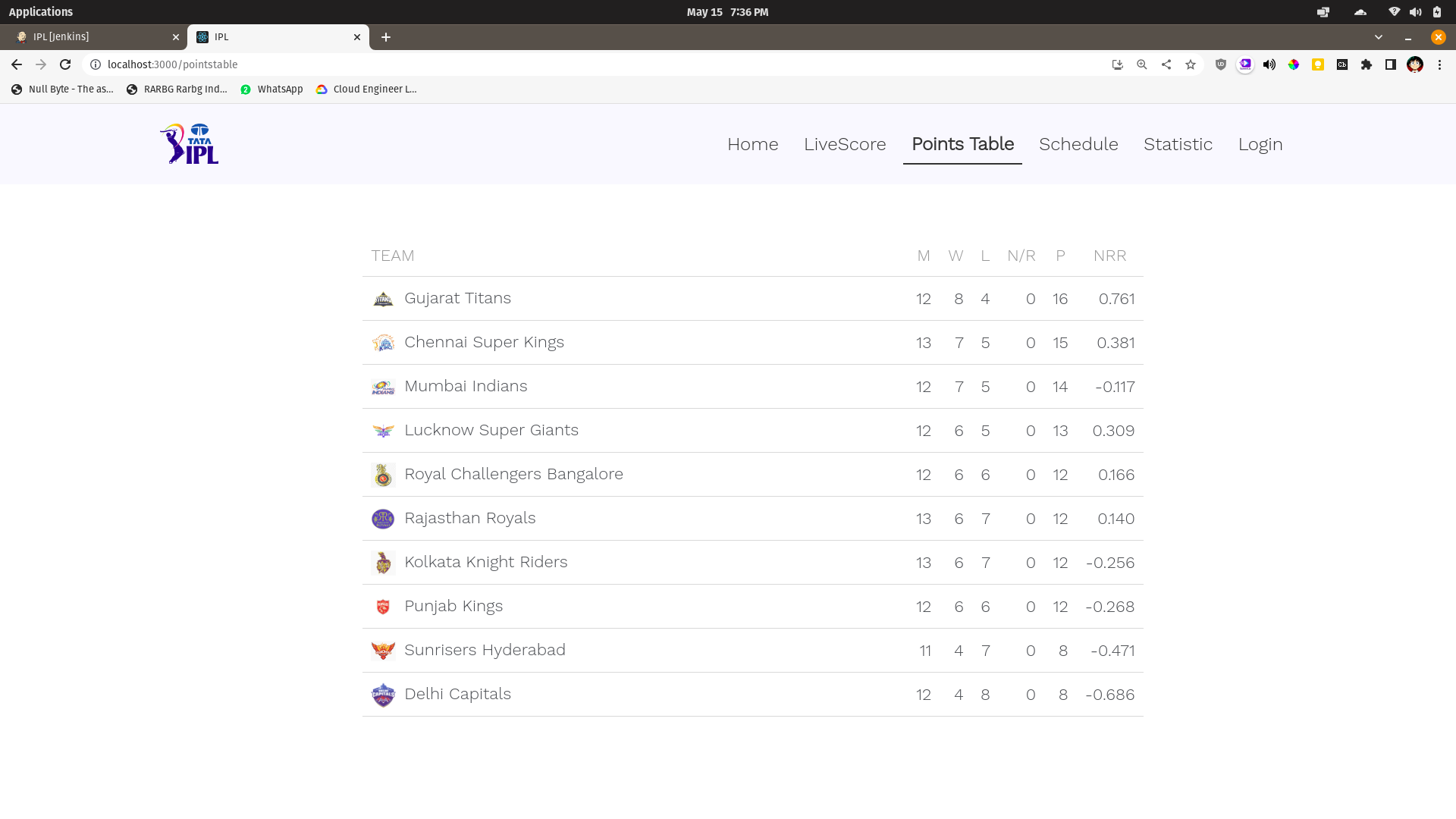
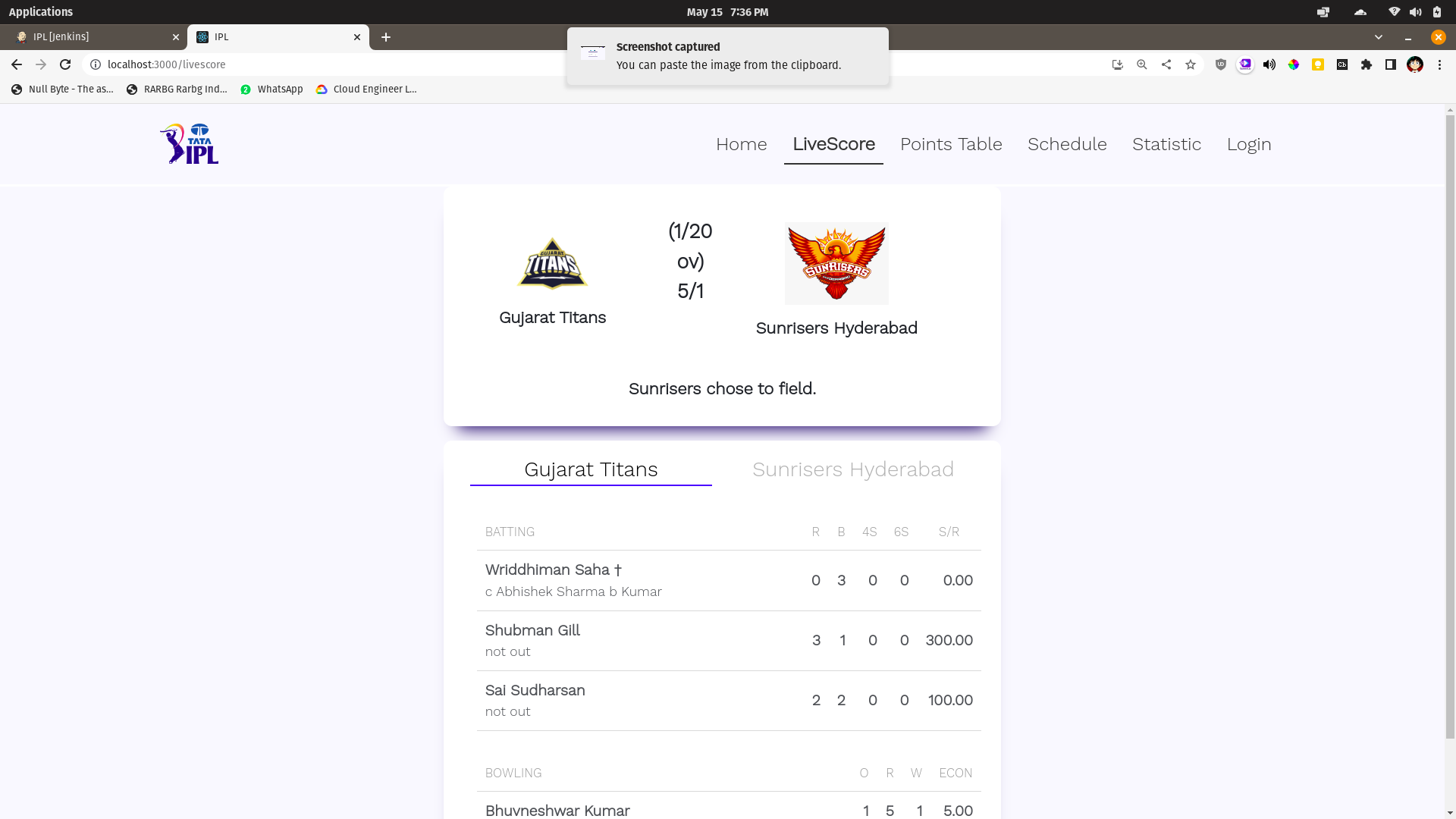
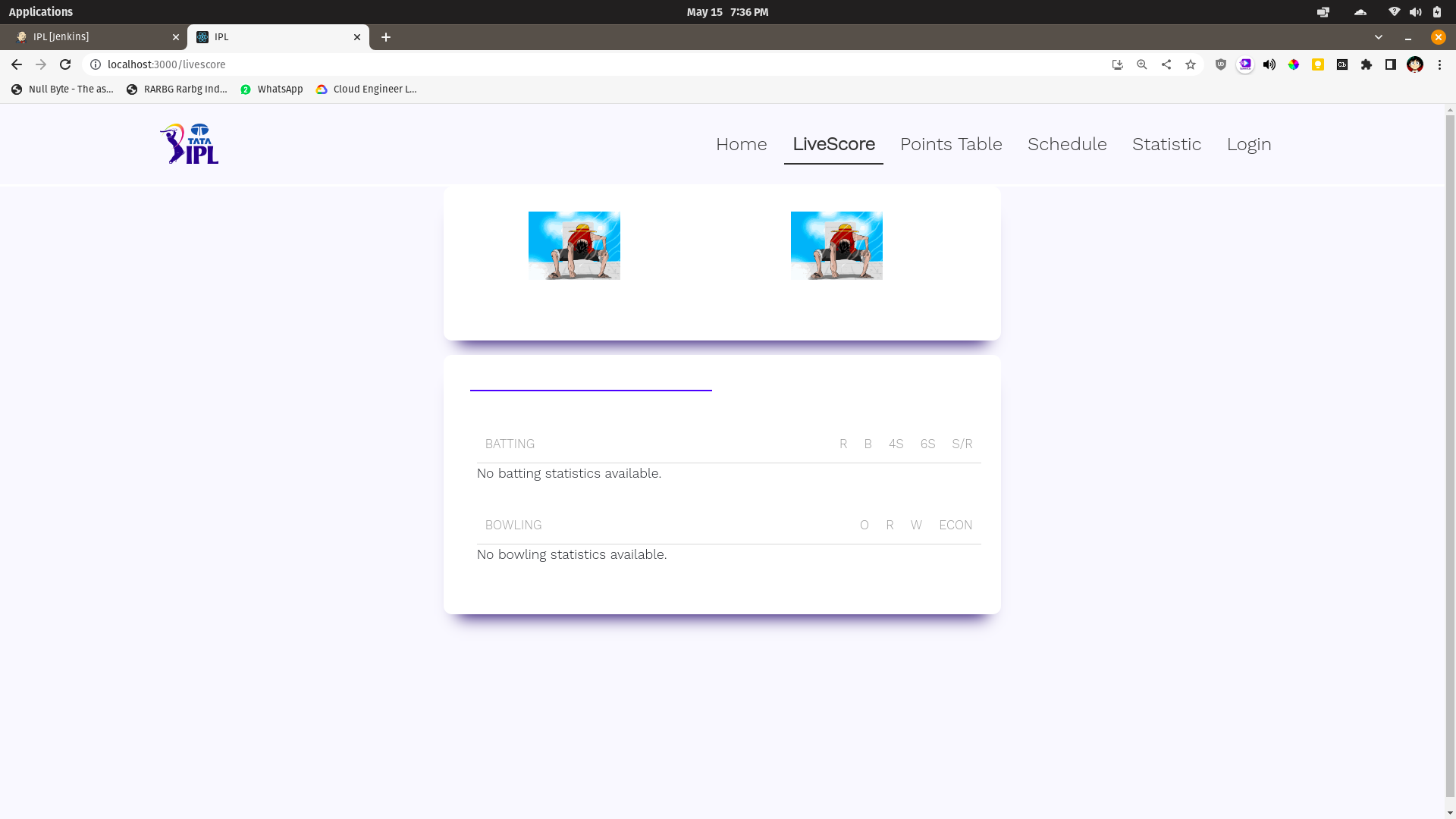
image: eovnstd/ipl\_server

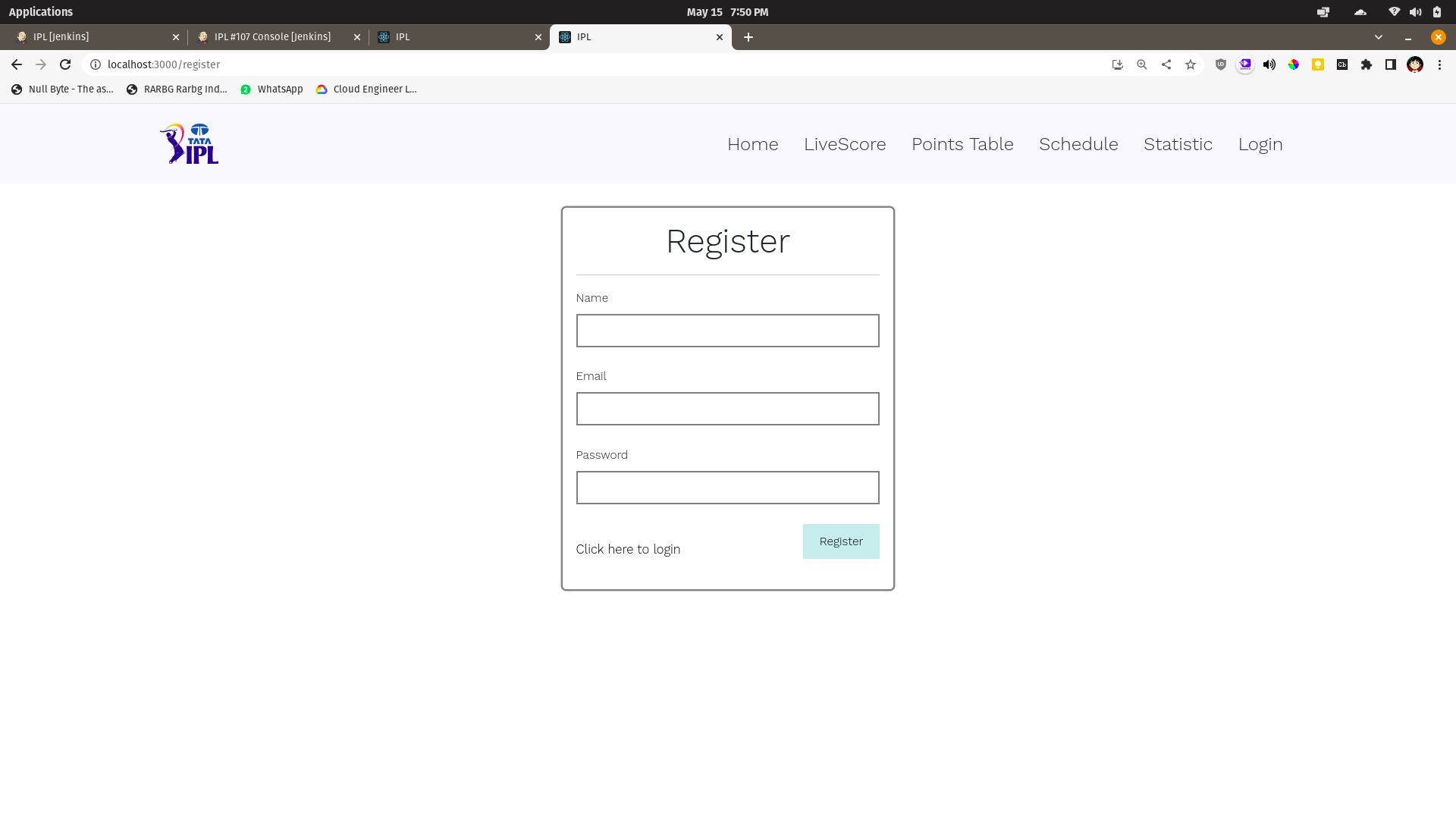
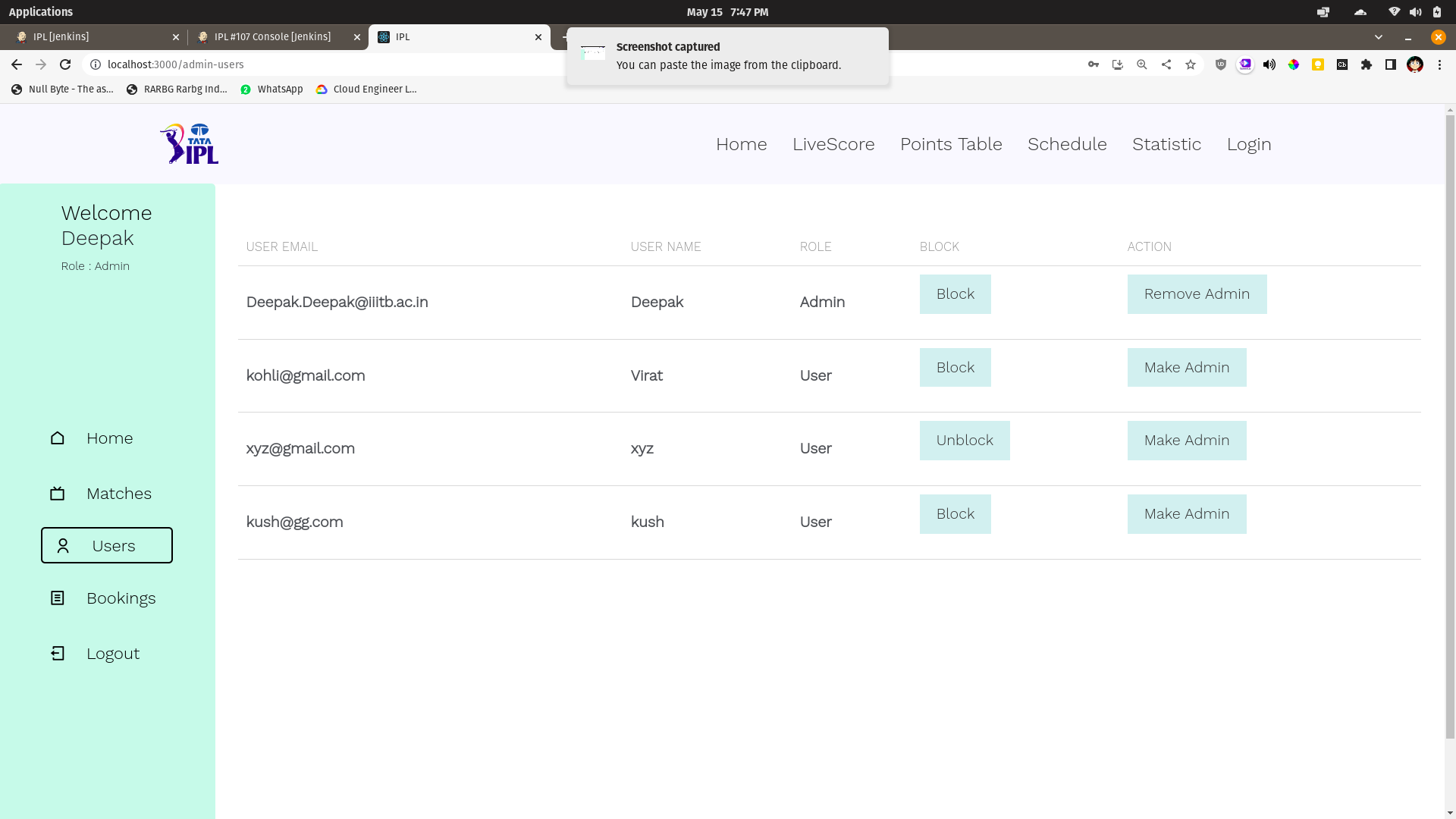
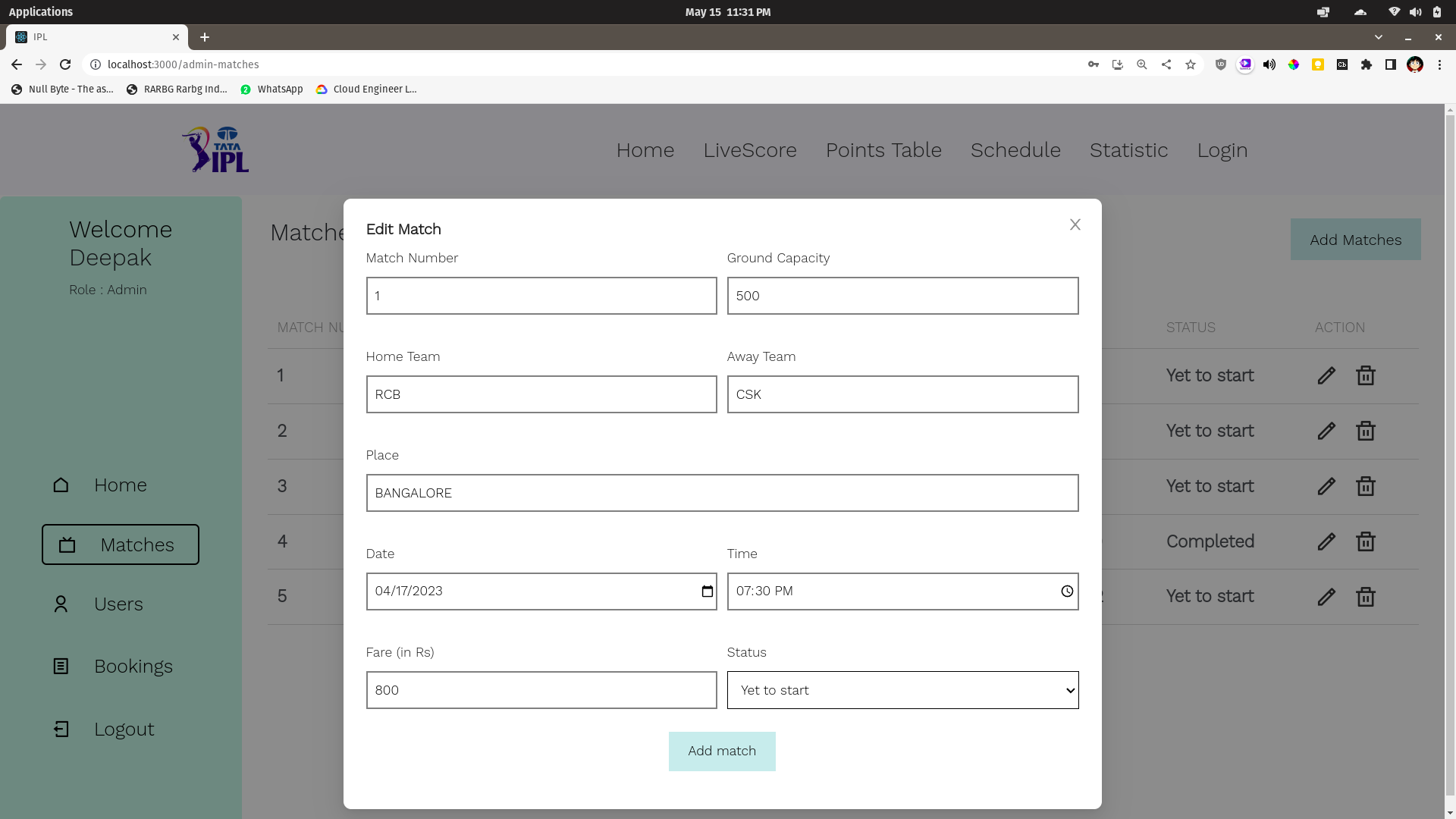
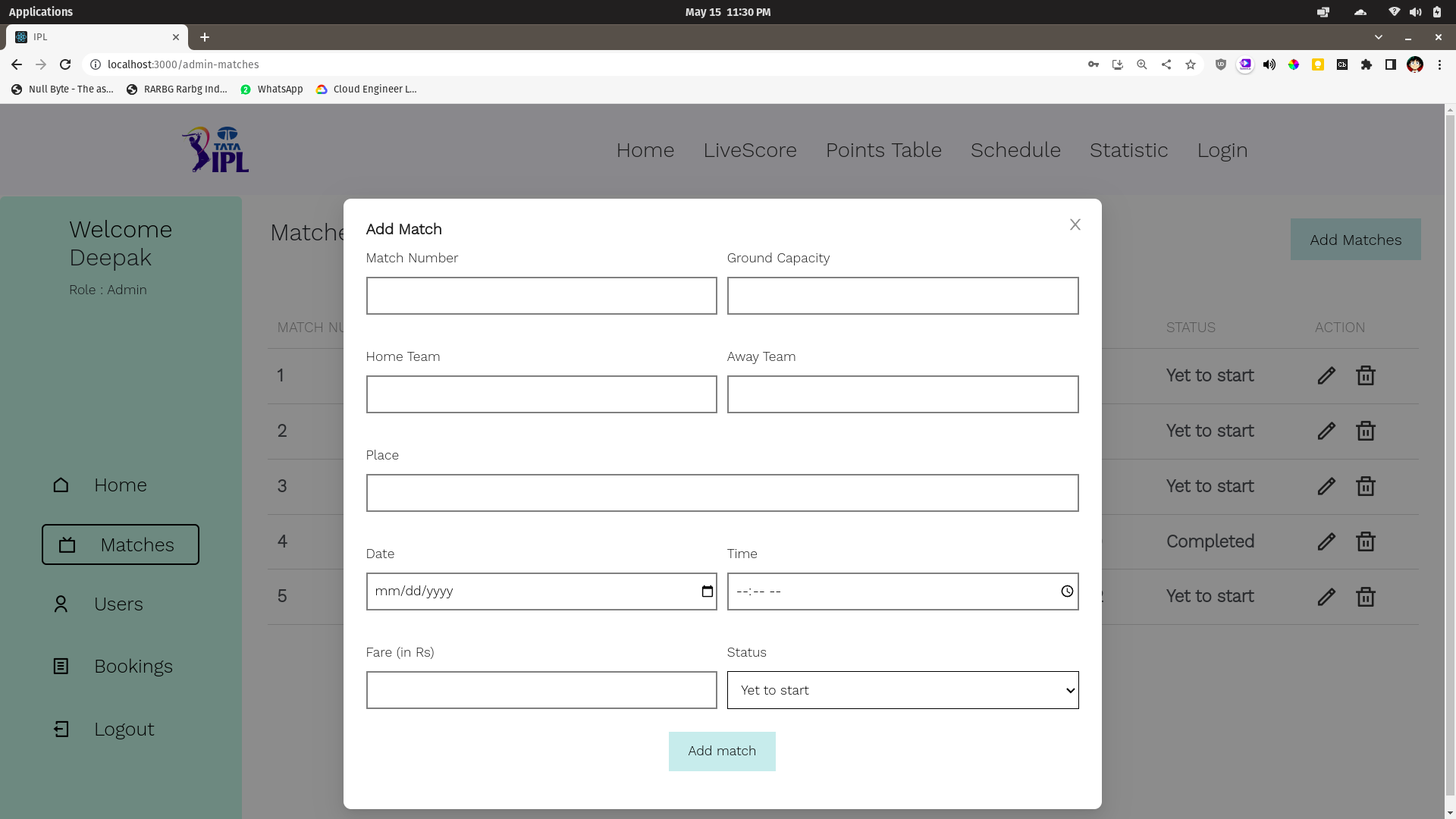
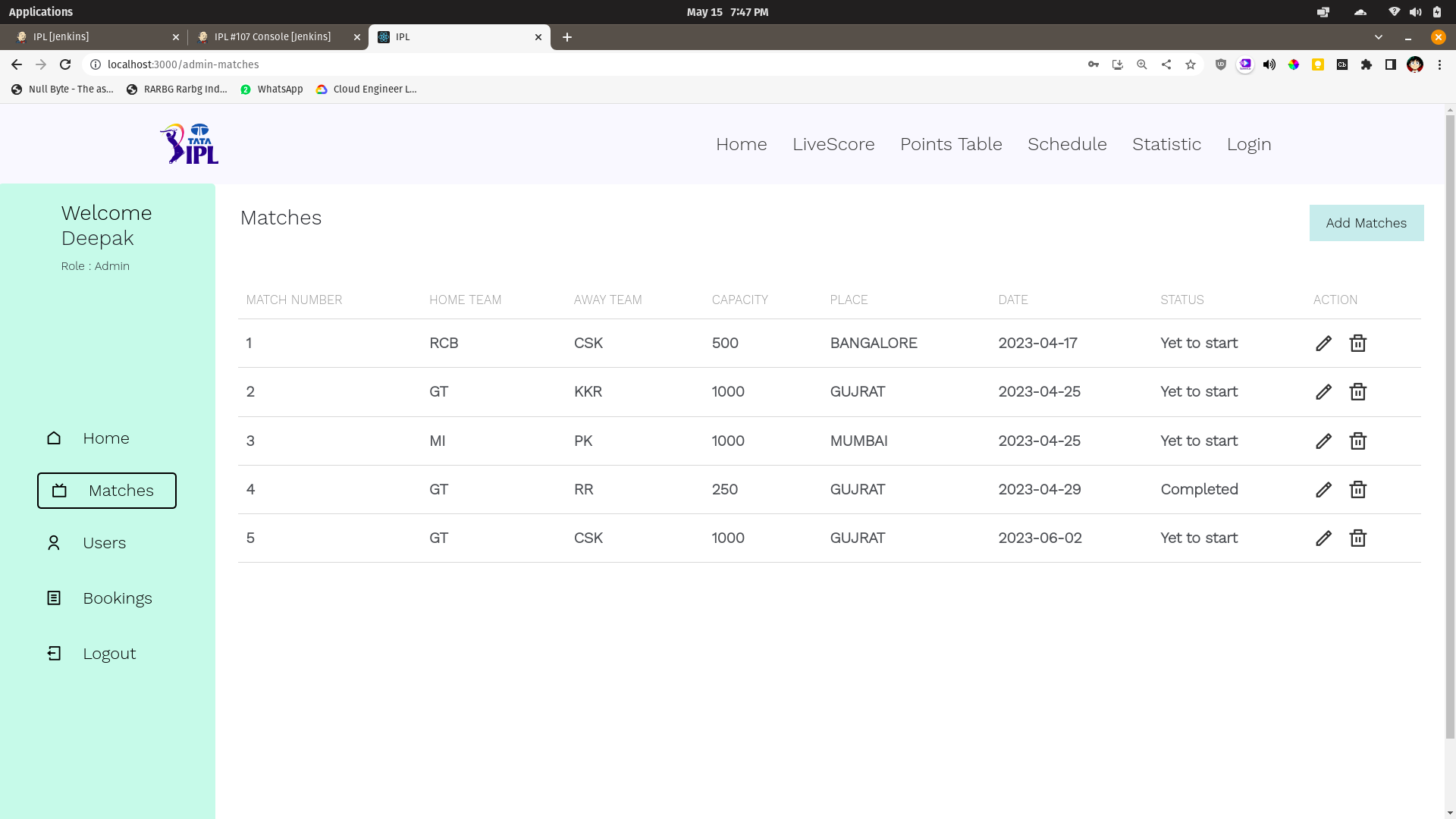
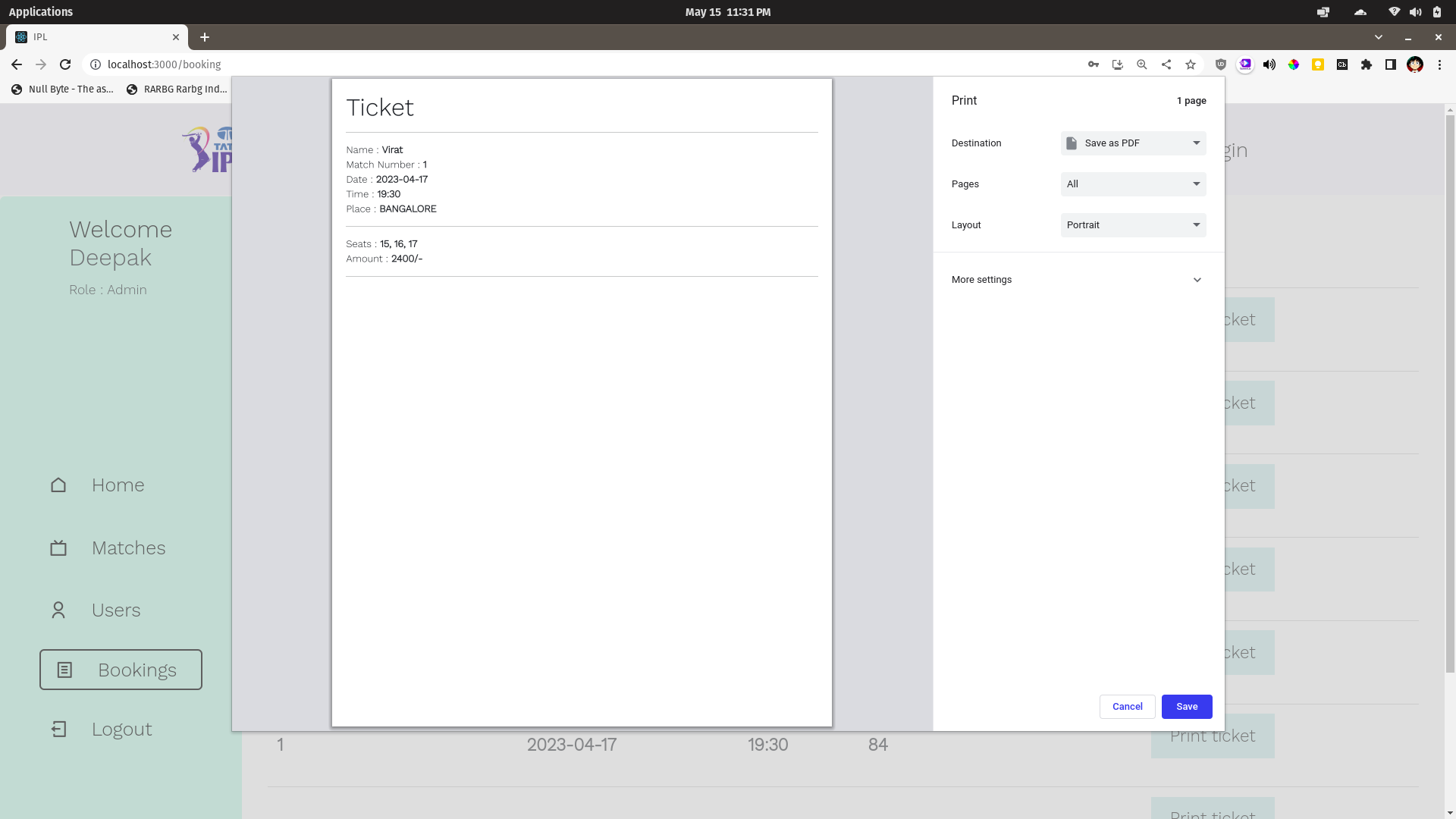
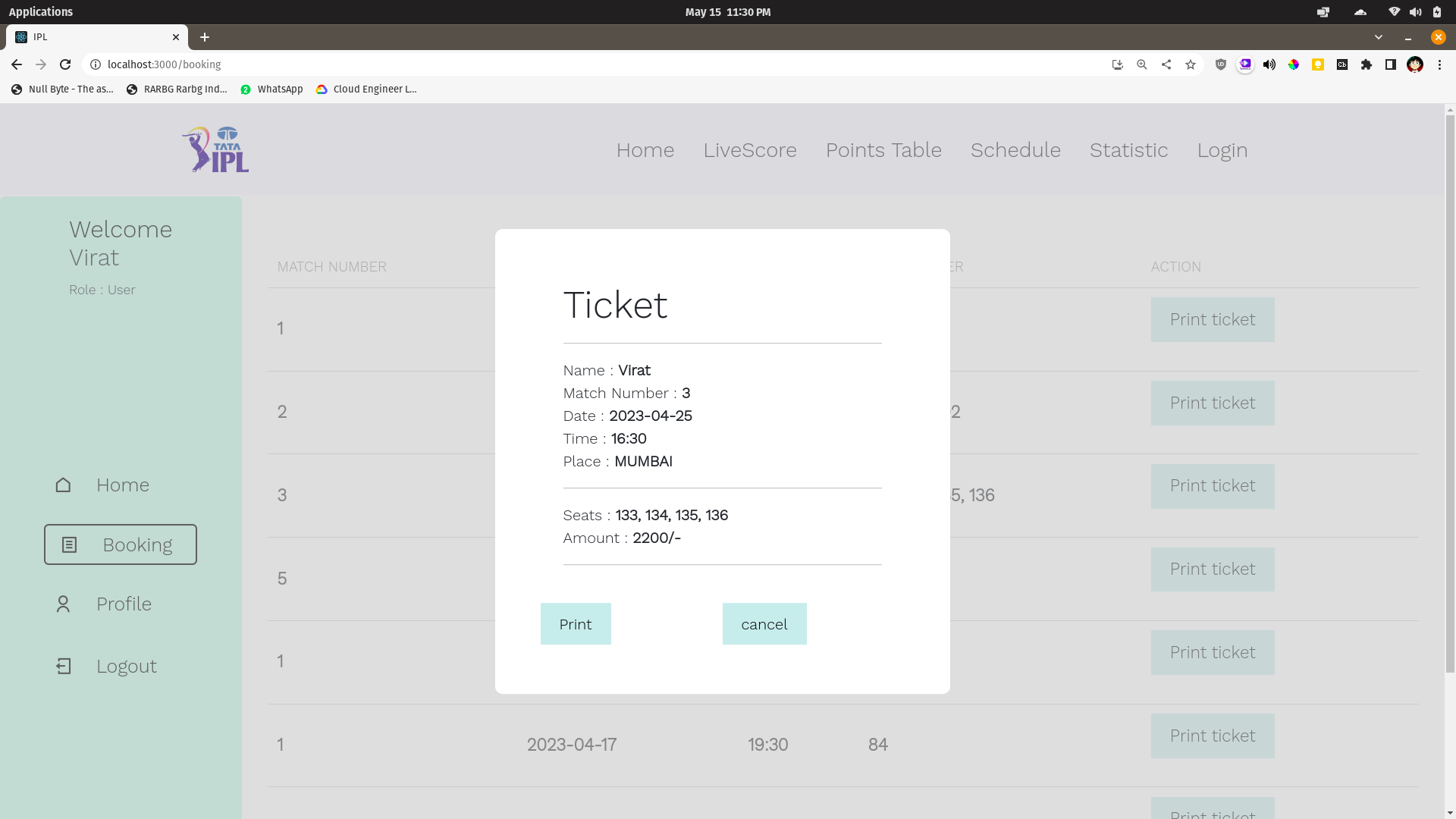
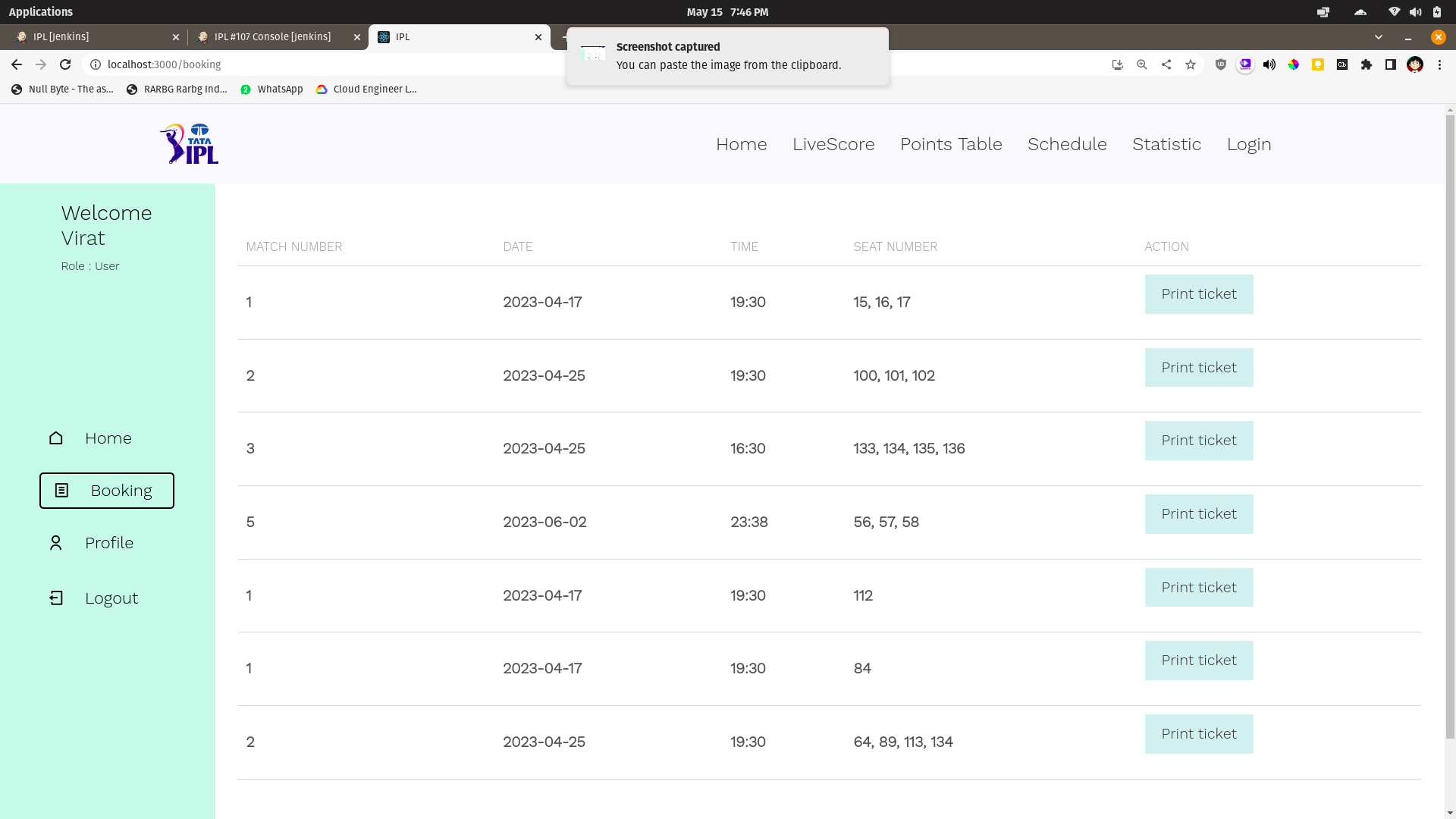
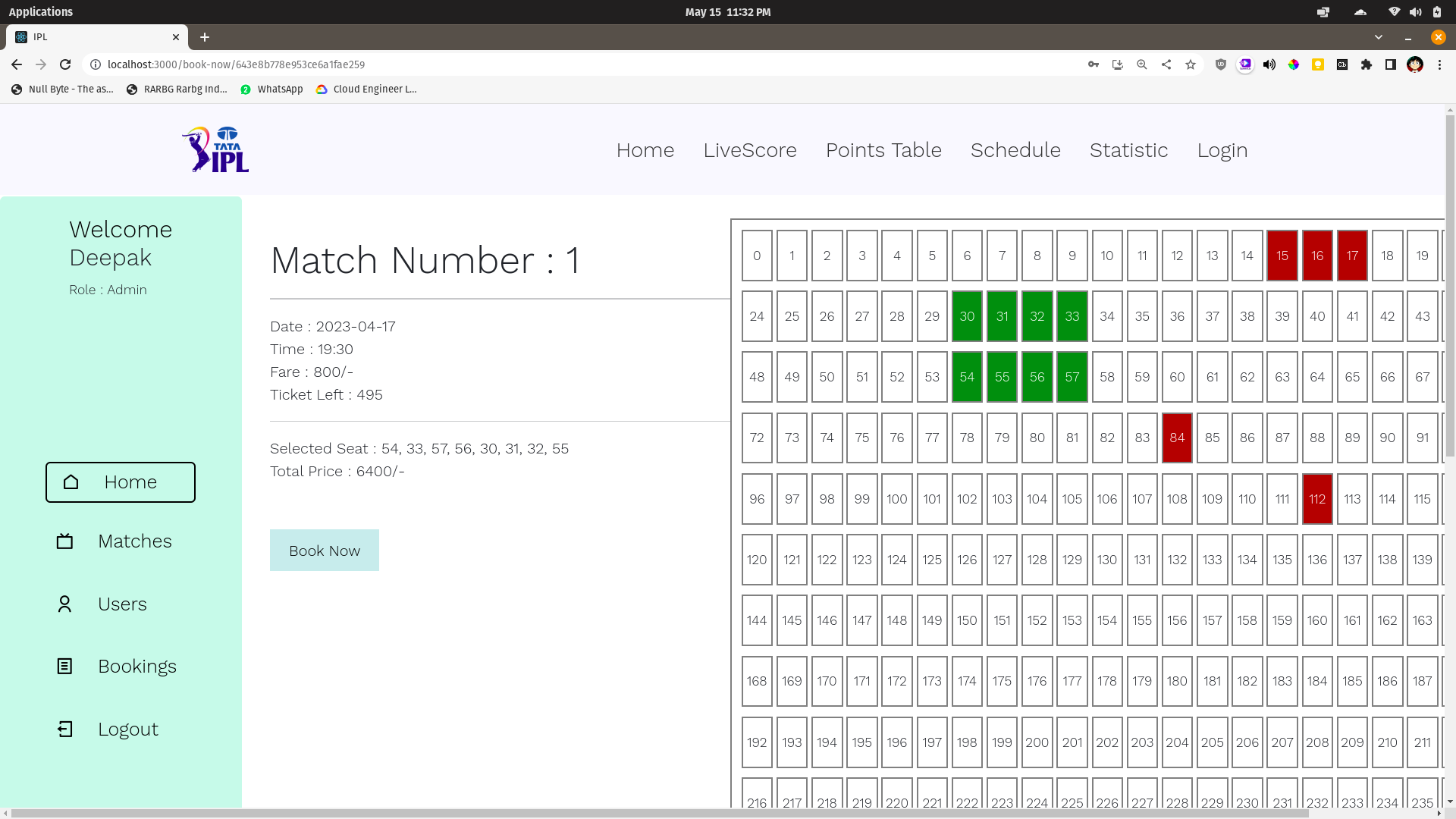
ports:

- containerPort: 5000

15. Project Screenshots

Homepage

Livescore loading****

****