**SPORTS RESOURCES MANAGEMENT SYSTEM WEB APPLICATION**

## A

**MINOR PROJECT REPORT**

**Submitted for the partial fulfillment of the requirement for the award of Degree**.

**B. Tech**

**COMPUTER SCIENCE & ENGINEERING**



### Submitted By: Guided By:

**ROHIT SINGH YADAV - 0101CS191099 Prof. PIYUSH SHUKLA SIR SHIVAM BHALAVI - 0101CS191108 Prof.JASWANT SAMAR SIR RAJ K JAIN - 0101CS191084**

**VINOD TILGAM – 0101CS191133**

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING UNIVERSITY INSTITUTE OF TECHNOLOGY

**RAJIV GANDHI PROUDYOGIKI VISHWAVIDALAYA BHOPAL-462033**

## SESSION 2021-22

**RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

# CERTIFICATE

This is to certify that **Rohit Singh Yadav, Raj K Jain, Vinod Tilgam and Shivam Bhalavi** of B.Tech. Third Year, Computer Science & Engineering have completed their Minor Project entitled “**Sports Resource Management System Web Application**” during the year 2021-2022 under our guidance and supervision.

We approve the project for the submission for the partial fulfillment of the requirement for the award of degree of B.E. in Computer Science & Engineering.

**Prof. PIYUSH SHUKLA SIR Dr. Uday Chourasia**

Professor HOD, DoCSE

Project Guide UIT RGPV BHOPAL

**Prof. JASWANT SAMAR SIR**

Professor

Project Guide

# DECLARATION BY CANDIDATE

We, hereby declare that the work which is presented in the minor project, entitled “**Sports Resource Management System Web Application**” submitted in partial fulfillment of the requirement for the award of Bachelor degree in Computer Science and Engineering has been carried out at University Institute of Technology RGPV, Bhopal and is an authentic record of our work carried out under the guidance of **Prof. PIYUSH SHUKLA SIR**

**And Prof. JASWANT SAMAR SIR (**Project Guide), Department of Computer Science and Engineering, UIT RGPV, Bhopal.

The matter in this project has not been submitted by us for the award of any other degree.

**ROHIT SINGH YADAV - 0101CS191099**

**SHIVAM BHALAVI - 0101CS191108**

**RAJ K JAIN - 0101CS191084**

**VINOD TILGAM – 0101CS191133**

# ACKNOWLEDGEMENT

After the completion of minor project work, words are not enough to express our feelings about all those who helped us to reach our goal, feeling above all this is our indebtedness to the almighty for providing us this moment in life.

First and foremost, we take this opportunity to express our deep regards and heartfelt gratitude to our project guide **Prof. PIYUSH SHUKLA SIR**

**And Prof. JASWANT SAMAR SIR of Computer Science and Engineering Department, RGPV Bhopal** for their inspiring guidance and timely suggestions in carrying out our project successfully. They have also been a constant source of inspiration for us.

We are extremely thankful to **Dr. Uday Chourasia, Head, Computer Science and Engineering Department, RGPV Bhopal** for his cooperation and motivation during the project. We would also like to thank all the teachers of our department for providing invaluable support and motivation. We are also grateful to our friends and colleagues for their help and cooperation throughout this work.

**ROHIT SINGH YADAV - 0101CS191099**

**SHIVAM BHALAVI - 0101CS191108**

**RAJ K JAIN - 0101CS191084**

**VINOD TILGAM – 0101CS191133**

# Abstract

The **“Sports Resource Management System web 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.

It is a web based application which manage the database of sports equipment in the college premises. It allow students to request for issuing the sport equipment’s like bat, ball, net, etc.

It Removes the congestion of students at the issuers table and provide every students a fair chance to request items.

The front end of the web application is developed using HTML, CSS, JavaScript and react. While Back end is managed by Node.js also the database is managed using MongoDB.

Table of Contents

# TITLE PAGE NO.

1. CHAPTER 1

**Introduction…** 1-7

1.1 Web Applications 1-9

* + 1. [Web Services](#_TOC_250014) 1-2
    2. Technologies used 2-9
  1. [Advantages 10-](#_TOC_250011)12
  2. [Motivation 12-](#_TOC_250010)13

CHAPTER 2

1. Literature Survey 14
   1. [Existing System Approach 14](#_TOC_250007)
   2. [Our Approach 14](#_TOC_250006)

# CHAPTER 3

1. **Problem Description 15-18**
   1. Services 15-16
   2. Effects 17-18
   3. User 18

# CHAPTER 4

1. **Proposed Work 19-24**
   1. Overview 19
      1. UML Use case diagrams 20-21
      2. UML Class diagrams 21-22
      3. ER diagrams 23-24

# CHAPTER 5

1. **Implementation 41-42**
   1. Hardware requirement 41
   2. Software requirements 42

# CHAPTER 6

# Result & Analysis 43-49

# CHAPTER 7

# Conclusion & Future Work… 50

# CHAPTER 8

# Bibliography 51

# CHAPTER 1 INTRODUCTION

## Web applications

## 

## Web services :

The Web Services technology is based on the concept of service-oriented computing. Web services are standards that integrate Web-based applications through connecting and sharing of business processes across the network where applications of different vendors, languages, and platforms communicate with each other and with clients. Web applications refer to applications accessed via Web browser over a network and developed using browser-supported languages (e.g., HTML, JavaScript). For execution, Web applications depend on Web browsers and include many familiar applications such as online retail sales, online auctions, and webmail. Web applications are needed in the area of business-to-business interaction over networks, e.g., for overseas companies that outsource projects to each other. The adoption of a Web applications infrastructure can provide vital processes such as transfer of funds and updates of pricing information. Because of the complexity of service systems, analysis of each component and subsystem becomes more In this paper, we discuss a model that enhances other methodologies used for developing Web services. The focus of our model is on the analysis of the flow streams of all ―things that flow‖ in different subsystems. This proposed flow model is a promising approach to software development in service-oriented applications. Flow-based conceptual models can reflect high-level design components of Web service and e-business solutions produced early in the application development lifecycle. The models can be utilized by business managers and analysts to trace transformations of models used by software developers. They also provide a means of communication to promote collaboration and standardization development of Web services. Web Services provide tools for developing and implementing business processes.

Software development methodologies play an important role in this technology.

Graphical user interface, diagram, application

Description automatically generated

Fig 1: Web application

## 1.1.2 Technologies Used in Web Application:

**Basic software engineering concepts**

Most issues in web development can be solved with basic engineering concepts. These concepts are often not prioritized enough, and ignoring them leads to apps that are difficult to debug and build upon.

**Separation of concerns**

Separation of concerns is probably the most important concept in engineering. It means that you break up code into intuitive blocks of logic that can be reused across multiple locations in your code.

**Backend v. Frontend**

Most modern web apps separate their application loosely into a backend and frontend. If it’s a large application, there might be multiple backends and frontends that provide different services. A programmer who works in both areas is “Full Stack”.

Backend : The backend provides data storage, object relational mapping, business logic, and specific services such as language translation. The backend has its own server to run the service and a means of communicating to the frontend, usually with JSON.

Frontend : The frontend is the client-facing part of the application. It’s where a user interacts with the service in a readable format. This part of the application will usually have markup (HTML) and styling (CSS) to render the service. It will usually have a scripting language like Javascript so the interface updates in real time in response to user actions.

**Application Programming Interface (API)**

Many backend services offer an API to act as a middle man between the client and the data. An API accepts requests for data and responds to the request in a structured format like JSON. A client can also request to update, delete or create entries.

**CRUD**

Create, Read, Update, Delete, or CRUD, is a helpful concept of the interface provided by most APIs. There are other conceptualizations of APIs out there as well such as command query separation (CQRS) which claims that every interaction with data is either a command (create, delete, update) or a query (read).

**API patterns**

Most modern applications loosely follow the REST pattern for delivering their API. There are other application patterns that offer alternative ideologies like PRPL and GraphQL. The idea of a Progressive Web App has come to represent an app that makes use of engineering patterns and separation of concerns to create an app-like experience for users on the web.

**View**

This is where presentational code lies. Usually there is markup involved, like HTML or JSX. This is what the browser interprets to render the application. We can often separate view concerns further with containers, wrappers, templates, generics, etc. Some frameworks like Ruby on Rails implement views by default.

**Domain (business concerns)**

Most web applications are connected to a business. The business has specific entities that should be mirrored in code. For example, a corner bodega might have entities like “owner”, “employee”, “customer”, “furniture”, “for sale items”, and “finances”. In a web application, these entities and their associated logic are separated into distinct units. This is often referred to with terms like business domain, business logic, and business domain models. Separating these concerns at larger companies can become complex, but doing so will create an intuitive code base.

**Object Relational Mapping**

In Object Relational Mapping (ORM), the business domain models will directly mirror the tables stored in a database. On the client side, script and view modules may also mirror the business domain models. Patterns like the repository pattern add more layers on to this system.

**Application State**

Most modern applications store the application state separately from the rest of the app. State is where an application stores the data of user interactions. For example, if a user clicks a like button, a state update would trigger a new color on the clicked button. The application state stores that information. There is often complex business logic involved in application state, so it is usually separated into multiple layers. Oftentimes, the application state sends data to other parts of the application via a publish-subscribe pattern.

**Application Services**

This layer handles concerns at the top level of an application, dealing directly with the server and physical resources of an application. Things like auto-scaling, performance monitoring, deployment, and cloud services might be handled here. This layer deals directly with allocating memory, processor power, and parallel threads. Some services might be separated into micro-services to optimize resources.

**Single Source of Truth**

Another basic concept in software engineering is single source of truth. This concept is often ignored, leading to difficult bugs. Single source of truth means that there is one master authority that determines the state of data. For example, in an election, the single source of truth is the paper ballot. If there is any doubt as to the results of an election, the paper ballot can be re-checked. In a web application, the single source of truth is usually the database.

**Javascript ES6 (ECMAScript 2015)**

The most notable of the changes in ES6 vs ES5 is the javascript class. This is just syntactic sugar, but it allows us to write visibly cleaner code, similar to what you’d see in other object oriented languages. Javascript is a functional language where everything, including execution contexts and variables, are part of key:value maps. The key:value maps are called objects in Javascript.

Here is an example of a ES6 javascript class:

import ParentClass from './ParentClass'

class ES6Class extends ParentClass {

constructor({ name="anonymous", ...otherData }) {

super();

this.name = `young ${name}`;

this.otherData = otherData;

}

setTrueName = async () => {

this.name = await this.fetchTrueName();

}

fetchTrueName() {

return fetch('/trueName')

.then((res) => res.json());

}

set name(newName) {

this.name = newName;

}

get name() {

return this.name;

}

}

export default ES6Class

const jeremy = new ES6Class({ name: "Jeremy", meaning: "young dashing web dev who writes medium articles sometimes"});

I snuck in some other new ES6 features too, including:

Destructuring assignment

const and let (block scoping)

Spread syntax

Template literals

Arrow function expressions

import/export

Async/await and promises

Reactive Programming — React and Vue

Reactive programming expands on the publish-subscribe pattern to build modules that “react” to changes in data. ReactJs and Vue.js are two frameworks that make use of reactive programming concepts.

At the core of reactive programming is the idea of declarative vs imperative programming. Reactive programming is declarative. Imperative programming is programming that achieves an end via a specific mechanism. It focuses on “how” a program operates. Declarative programming focuses on the goal of a program. It doesn’t care how a program operates, only that its logic will be satisfied. The how part can be a black box.

Declarative program: Make a grilled cheese and tell me when it’s done!

**Imperative Program**:

Use a bread knife to cut two slices of bread on a cutting board.

Use a butter knife to spread butter on the bread.

Heat a pan to medium heat with butter.

Slice two bread sized slices of american cheese and put to the side

Put the bread on the pan

When the bottom of the bread is golden brown, flip each piece.

Place one slice of cheese on top of each piece of bread.

When the cheese is melted, flip one piece of bread onto the other, with cheese faced inward.

Tell Jeremy it’s done!

Declarative programming makes use of the concepts of separation of concerns and black box to simplify program logic, so that very little imperative programming is required.

In reactive programming, a component declares which data it cares about and reacts to changes in that data. So a UserList component says give me a list of users and I will render it.

**Hypertext markup language (HTML) and the DOM**

HTML is the language most often used for markup in modern browsers. You can interact with its data via an API called the Document Object Model (DOM). We usually use javascript to access the DOM, but it is built so that it can be accessed by any language. The DOM has many features built in that make interacting with markup easier —

Select tags, buttons, inputs, and forms: these are examples of html tags that automatically interact with the user. There is no javascript required. You can use pure HTML to set up a form that sends a post request.

Event handlers: the DOM offers built in events like click , change , and scroll . It’s possible to write these event handlers inline in HTML without any javascript. You can use many DOM API methods without JS.

Oftentimes we use javascript combined with the DOM API because javascript provides a lot of features that aren’t included in HTML/DOM without javascript. Javascript can do tasks asynchronously and modularize tasks.

Side note: a library called jQuery used to be widely used. It provided many useful methods for interacting with the DOM and famously used $ as a catchall method. Much of what made jQuery useful is now provided natively by browsers, so most companies are moving away from it. But the $ method is frequently seen in old code at a lot of companies.

**JSX**

JSX looks like HTML or XML but it is actually an extension of javascript. It allows you to embed javascript directly in markup which dramatically simplifies the logic of scripts interacting with HTML.

Here’s an example in ReactJS:

const NameInput = ({ name }) => (

<input value={name} /> // this is the JSX

)

In this component, we expect a name to be passed into our function, and then we return an input tag with a value equal to name. The input tag looks a lot like an HTML input. The major difference is we are able to embed the value of name directly into the HTML tag.

JSX is extremely powerful for passing data around an application. It provides an aesthetically pleasing way of visualizing how data is passed. It is usually used in tandem with a reactive programming platform like React or Vue, but you can write your own transpiling script for babel (more on babel later).

**Virtual DOM**

The Virtual DOM is a representation of the current state of the DOM. It is basically a big javascript object that is constructed similar to a linked list. Both react and vue make use of this concept. Whenever the virtual DOM is updated, those changes are reconciled with the real DOM by a “diffing” process. Diffing just means that the virtual DOM is compared to the real DOM and appropriate changes are made to bring the real DOM up to date. The goal is to provide a black box that maintains a single source of truth efficiently. This allows reactive programmers to write clean, declarative, pure programs.

**Typescript**

Javascript is a dynamically typed language. This means that we can pass anything into any function and it won’t throw an exception. Typescript is a superset of javascript that adds static typing. This means that when you define a function, variable, or object, you provide it with types that it must satisfy.

For example:

// No Typescript

const stringToInteger = (str) => {

return parseInt(str, 10);

}

// With Typscript

const stringToInteger: (x: string) => number =

function(str: string): number {

return parseInt(str, 10);

}

Typescript is useful for writing better code. It forces you to examine your expectations when you write code. Its only drawback is it can give you a false sense of security. A lot of bugs have nothing to do with typing.

**Accessibility Markup**

Many HTML5 elements incorporate accessibility features. For example, image tags include an alt attribute that provides a visually impaired user with text data about the image. The DOM responds to keyboard commands like tab and return to allow visually impaired users to navigate buttons and lists on a page. It is important to use standard markup practices if you want your app to be accessible.

If you are using JSX for markup, a linter like eslint can provide helpful warnings to ensure your markup is accessible.

When you write custom html elements, ARIA standards can be used for accessibility markup if there is no other built in method.

HTML5 includes the ability to customize keyboard accessibility with tabindex.

Polyfills

Babel makes most ES6 javascript usable across browsers, but it doesn’t support more specific browser APIs like getUserMedia. For that you need a specific polyfill. A polyfill is a script that you run globally when you application first loads which provides access to an API that is not available natively on older browsers.

**Version Management — Git**

Git is a version control language. This is important in application development. Oftentimes, we want to add a new feature to an application, but what happens if there are multiple people working on different aspects of the app? Git allows us to merge their work into the main codebase, and even revert changes if they break the code. Git also recognizes conflicts between separate people’s work and gives you an interface to reconcile those conflicts.

**Github**

Github is a cloud platform that allows teams to interact with git repositories across multiple machines. It is an incredibly powerful tool for teamwork in web development.

**Continuous Integration and Deployment**

Continuous integration is the process of integrating changes from different developers frequently, ensuring that there are no code conflicts as developers continue to build features.

Continuous delivery builds on continuous integration by checking a series of tests. The tests ensure that code can be deployed at any time.

Continuous deployment builds on continuous delivery by deploying continuous deliveries to a test environment to ensure that the app is always working.

Test Driven Development (TDD)

Test driven development is a software development process where you write tests that the software must pass before writing the actual software. This allows you to prove that that software meets certain standards. There are many different types of tests:

Unit Tests - tests a specific unit of code. For example:

function add(a, b) {

return a + b;

}

test('add adds two parameters', () => {

expect(add(1, 4)).toBe(5);

})

Integration Test — tests a larger unit of code that may form an entire application. For example:

function startServer() {

app.start();

}

test('logging in redirects you home', async () => {

await logIn();

expect(page).toBe(home);

})

End To End / System Tests — tests an aspect of the entire application, with both frontend and backend functioning. Often you will do this by automating actions inside an actual browser process. Selenium w/ WebDriver, Cypress, and Puppeteer w/Jest are all popular libraries for this.

Acceptance Tests — Similar to End To End tests, but they test specific requirements from a user’s perspective, and separate from the implementation details. For example, if you click a submit button in the actual application, does the form submit?

## Advantages of web application:

There are several advantages of web application in agriculture, healthcare, retail, transport, environment, supply chain management, infrastructure monitoring, etc. Some of them are listed below:

**Improved Efficiency**

Having multiple versions of spreadsheets or shuffling around piles of paperwork is not only time consuming, but can also leave your business vulnerable to human errors that are not easily spotted until it’s too late.

What’s more, with no integration between these multiple sources of data, time-consuming and laborious replication of data may be required in order to get a holistic overview of business performance.

Web applications help you streamline your business processes so that you can get more done in less time and with greater accuracy. What’s more, having all your data integrated in one place gives you greater visibility of your business, freeing up staff time and allowing you to run reports that are updated with real-time information.

**24 / 7 Accessibility**

Given that business systems created by web applications are web-based, they can be accessed 24/7 provided that you have an internet connection. What’s more, they are totally flexible, offering access from almost any device or browser.

When desktop based software need updating, every single device in which the application is installed will need individually updating. This task usually falls to staff and may be ignored if they are pushed for time, leaving your business vulnerable to security breaches.

Compare this with a web based application, where a security or functionality update can be rolled out to every version of the web application with zero downtime, giving users instant access to the updated version of the app.

**Higher Levels of Security**

With desktop based software, a stolen or damaged computer can be a very costly and time consuming situation; leaving your data at risk and requiring you to contact your software provider and request for the software to be re-installed on a new device.

With a web based application (with data stored in the cloud), you have the peace of that should your computer equipment be damaged or stolen, it can very quickly be back to ‘business as usual’.

This is because web applications store information on remote service, so as long as you know your URL (web address), user name and password, you can log-in securely to any computer or mobile device connected to the internet and your business can be up and running again in no time.

Finally, in the event of loss of data through human or programme error, data can be quickly restored from the cloud.

**Easy Customisable and Scalable**

One of the biggest problems client that come to us face with off the shelf software is that it cannot grow or integrate with their business, or not without expensive upgrades at least.

As a custom web application is made specifically to your business needs, it’s completely flexible and scalable to your business’s demands and growth.

Customisations to the application may include your own branding and having different user permission access levels. By only having features and functions which are relevant to your business,

you’ll reduce training time and can add functionality as your business grows.

**Easy Installation and Maintenance**

By using a web application, you avoid the hassle and memory usage of installation software on every device, you’ll also find web applications less punishing on older or low spec devices. As every PC has a browser, installation time is extremely fast and can run in the background whilst staff get on with their other work. Maintenance requirements are generally much lower, with updates and patches rolled out remotely to every device.

## Motivation :

Recent media consumption & trends reveal that 90% of their mobile time is spent in apps, and only 10% browsing the rest of the internet. But the user base of a specific product may be reluctant to move for a mobile app, only to get done a specific service. And also the user may think twice before installing a new app because of limited storage and memory. Most hesitated warning at this point is that to free up the space. So web apps may be the newest trend in the upcoming future, with the tagline “no more mobile apps”.

Its worthy to find out the importance of a mobile app during this walk-through.

**Why do we need mobile applications?**

* High market-share and the usage
* Ease of sending notifications
* Ability to work offline
* Making use of mobile device features
* Mobile apps offer better Personalization

**Specific use case**

Imagine a situation where the Product owner decides to go for a mobile app based on the following reasons.

* Web app takes a considerable amount of time to fulfill a specific operation.
* All the features which are available in the current web app is not accessible via mobiles, so that need for a mobile app has been arise with limited functionality.

Improving performance is a continuous process. Product needs to be optimized in every aspect in order to achieve it. And at the same time what if we were able to provide an all in one web application with all the services that works perfectly on both mobile and desktop browsers? Since a web browser is available in every mobile its a great option to get away with a mobile app.

**Challenges that are likely to face when a sudden requirement arises**

* Lack of well-trained programmers who can deal with mobile technologies
* have to deal with different platforms
* Tight deadline
* Choosing the right front end framework
* Fitting between the front-end and back-end services, technologies

**­­­Solution with the design**

How can you redesign your web application to avoid going for a mobile application? It’s the responsiveness of the user interfaces plays a huge role in achieving this. An experienced designer with great ideas can help with an innovative design. Its worthy to wait until the design gets finalized.

# CHAPTER 2

# LITERATURE SURVEY

**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.

**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.

**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.

## PROBLEM STATEMENT

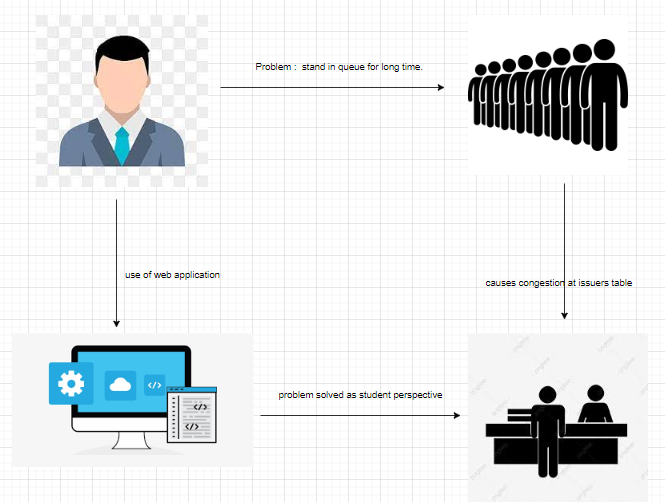
The **“Sports Resource Management System web Application”** has been developed in order to enable the students to book sports resources. The administrator can keep a tab on the availability of the resources along with streamlining the fine process. This application overcomes the problems prevailing in 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.

It is a web based application which manage the database of sports equipment in the college premises. It allow students to request for issuing the sport equipment’s like bat, ball, net, etc.

It Removes the congestion of students at the issuers table and provide every students a fair chance to request items.

The front end of the web application is developed using HTML, CSS, JavaScript and react. While Back end is managed by Node.js also the database is managed using MongoDB.

**USER:**



### EFFECTS Of digitalization:

Digital transformation is generating a fierce debate among policy-makers, economists and industry leaders about its societal impact. As digitalization disrupts society ever more profoundly, concern is growing about how it is affecting issues such as jobs, wages, inequality, health, resource efficiency and security.

To help begin a new, evidence-based debate on the future impact of digital transformation, we have conducted a detailed quantitative analysis of the value at stake from the digitalization of 11 major industries. In each case, we have calculated projections of the potential value of digitalization to the industry itself and emerging sources of value for wider society, as measured by an intentionally narrow set of indicators. Over time, this approach may be broadened and refined.

**Employment and skills**

Current estimates of global job losses due to digitalization range from 2 million to 2 billion by 2030. There is great uncertainty, with concerns also about its impact on wages and working conditions.

**Environmental sustainability**

The historic trend holds that for every 1% increase in global GDP, CO2e emissions have risen by approximately 0.5% and resource intensity by 0.4%.[¹](https://reports.weforum.org/digital-transformation/understanding-the-impact-of-digitalization-on-society/" \l "_edn1) Current business practices will contribute to a global gap of 8 billion tonnes between the supply and demand of natural resources by 2030, translating to $4.5 trillion of lost economic growth by 2030.[²](https://reports.weforum.org/digital-transformation/understanding-the-impact-of-digitalization-on-society/" \l "_edn2)

**Trust**

Social media, radio frequency identification (RFID) tags and user-generated websites such as TripAdvisor have been instrumental in increasing transparency and overcoming information asymmetries. However, according to the Edelman Trust Barometer, trust in all technology-based sectors declined in 2015, with concerns over data privacy and security a key factor. Broader ethical questions about the way organizations use digital technology also threaten to erode trust in those institutions.

The digital revolution has created new roles (such as search engine optimization managers and social media account managers), new types of organizations (cloud computing providers and social media agencies), and even new sectors of the economy (digital security and data science). The impact of digitalization has also acted as a catalyst for employment growth in the wider economy. In India, for example, it is estimated that three to four jobs are created for every job within the business process outsourcing and IT-enabled services sectors.[³](https://reports.weforum.org/digital-transformation/understanding-the-impact-of-digitalization-on-society/" \l "_edn3)

Today, however, the question of whether technology creates or destroys jobs is gaining momentum.

The truth is that we actually know quite little of what is going to happen. What will the economic impact of innovations be in the future? How will humans interact with machines and algorithms? What kind of skills do we need and how should we learn? How will all of this impact labour markets?

**The future of work**

We can at least be sure that there will be three types of jobs, categorized by the percentage of codifiable tasks within the role:

* Those that will disappear (lost the race against the machine). For example, clerks and administrative staff, or truck drivers.
* Those that are in collaboration with machines / algorithms (run with the machine). For example, those professions that rely on cognitive and social capabilities, such as doctors / surgeons.
* Those jobs that are completely new or remain largely untouched (running faster than the machine or running a different race). For example, roles in the creative arts are unlikely to be automated, as are new roles that involve managing data and machines.

Within that context, our analysis suggests that digital transformation has the potential to create a significant number of jobs. For example, we found that digitalization could create nearly 6 million jobs in just the electricity and logistics industries by 2025.

But clearly there will be both winners and losers – as the graphic below shows, while the net impact on jobs in the logistics industry could be positive, many sectors will experience job losses.

**Challenges**

Digital technologies fundamentally transform organizations, with the pace of technological change exacerbating the challenge. Organizations must have a coherent strategy that includes a plan to reskill workers. Whereas previous technological revolutions (most notably the industrial revolution) played out over a relatively long period of time, the speed of digital transformation is such that businesses need to move quickly.

# CHAPTER 4 PROPOSED WORK

**UML – USE CASE DIAGRAM:**

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system.

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

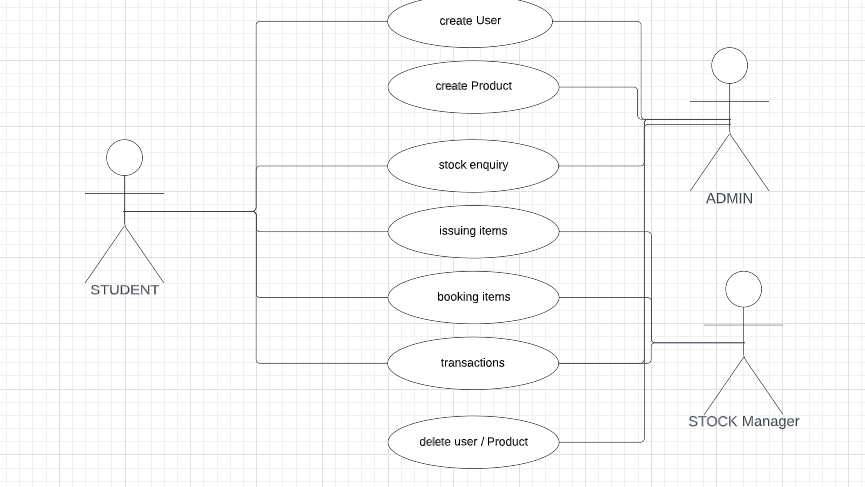
Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system. You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the development process.

Use-case diagrams are helpful in the following situations:

* Before starting a project, you can create use-case diagrams to model a business so that all participants in the project share an understanding of the workers, customers, and activities of the business.
* While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to others what the system should do.
* During the analysis and design phases, you can use the use cases and actors from your use-case diagrams to identify the classes that the system requires.
* During the testing phase, you can use use-case diagrams to identify tests for the system.

The following topics describe model elements in use-case diagrams:

* [**Use cases**](https://www.ibm.com/docs/en/SS8PJ7_9.7.0/com.ibm.xtools.modeler.doc/topics/cuc.html)  
  A use case describes a function that a system performs to achieve the user’s goal. A use case must yield an observable result that is of value to the user of the system.
* [**Actors**](https://www.ibm.com/docs/en/SS8PJ7_9.7.0/com.ibm.xtools.modeler.doc/topics/cactor.html)  
  An actor represents a role of a user that interacts with the system that you are modeling. The user can be a human user, an organization, a machine, or another external system.
* [**Subsystems**](https://www.ibm.com/docs/en/SS8PJ7_9.7.0/com.ibm.xtools.modeler.doc/topics/csubsys.html)  
  In UML models, subsystems are a type of stereotyped component that represent independent, behavioral units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modeling.
* [**Relationships in use-case diagrams**](https://www.ibm.com/docs/en/SS8PJ7_9.7.0/com.ibm.xtools.modeler.doc/topics/crelsme_ucd.html)  
  In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between the model elements.



**UML – CLASS DIAGRAMS:**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of objectoriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

## Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as −

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

## How to Draw a Class Diagram?

Class diagrams are the most popular UML diagrams used for construction of software applications. It is very important to learn the drawing procedure of class diagram.

Class diagrams have a lot of properties to consider while drawing but here the diagram will be considered from a top level view.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represent the whole system.

The following points should be remembered while drawing a class diagram −

* The name of the class diagram should be meaningful to describe the aspect of the system.
* Each element and their relationships should be identified in advance.
* Responsibility (attributes and methods) of each class should be clearly identified
* For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
* Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
* Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

The following diagram is an example of an Order System of an application. It describes a particular aspect of the entire application.

* First of all, Order and Customer are identified as the two elements of the system. They have a one-to-many relationship because a customer can have multiple orders.
* Order class is an abstract class and it has two concrete classes (inheritance relationship) SpecialOrder and NormalOrder.
* The two inherited classes have all the properties as the Order class. In addition, they have additional functions like dispatch () and receive ().

The following class diagram has been drawn considering all the points mentioned above.

## Where to Use Class Diagrams?

Class diagram is a static diagram and it is used to model the static view of a system. The static view describes the vocabulary of the system.

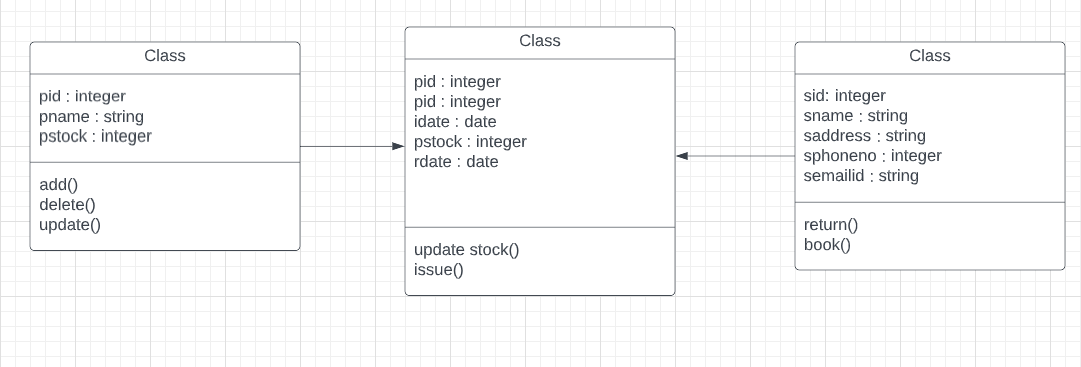
Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.

Generally, UML diagrams are not directly mapped with any object-oriented programming languages but the class diagram is an exception.

Class diagram clearly shows the mapping with object-oriented languages such as Java, C++, etc. From practical experience, class diagram is generally used for construction purpose.

In a nutshell it can be said, class diagrams are used for −

* Describing the static view of the system.
* Showing the collaboration among the elements of the static view.
* Describing the functionalities performed by the system.
* Construction of software applications using object oriented languages.



## ER Diagrams :

**ER Diagram** stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

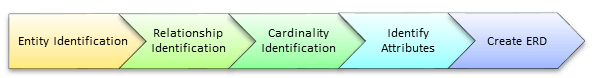
ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

## Why use ER Diagrams?

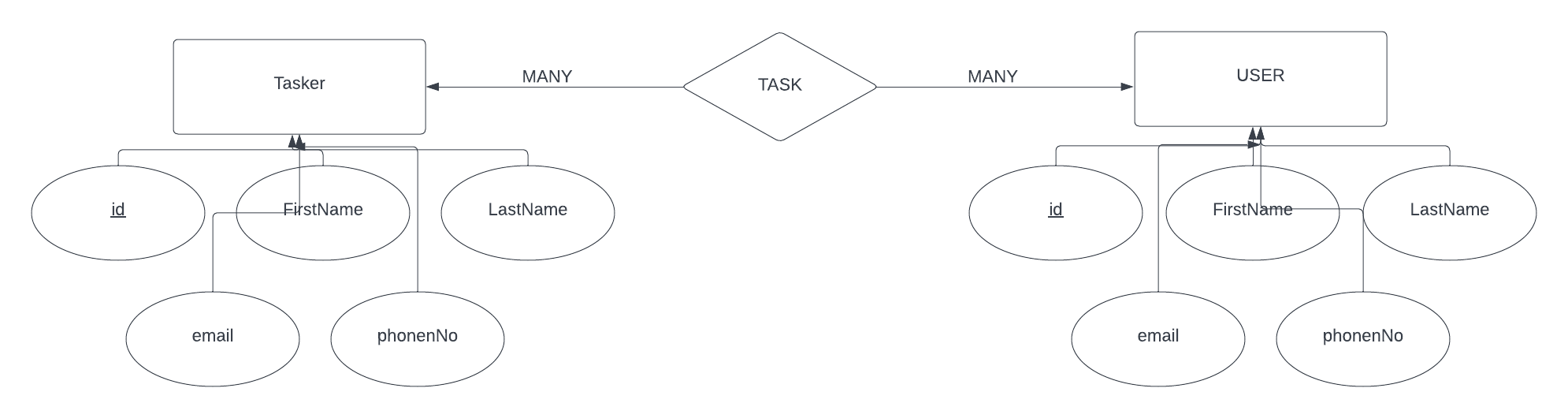
Here, are prime reasons for using the ER Diagram

* Helps you to define terms related to entity relationship modeling
* Provide a preview of how all your tables should connect, what fields are going to be on each table
* Helps to describe entities, attributes, relationships
* ER diagrams are translatable into relational tables which allows you to build databases quickly
* ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
* The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
* ERD Diagram allows you to communicate with the logical structure of the database to users



ER Model in DBMS stands for an Entity-Relationship model

* The ER model is a high-level data model diagram
* ER diagrams are a visual tool which is helpful to represent the ER model
* ER diagrams in DBMS are blueprint of a database
* Entity relationship diagram DBMS displays the relationships of entity set stored in a database
* ER diagrams help you to define terms related to entity relationship modeling
* ER Model in DBMS is based on three basic concepts: Entities, Attributes & Relationships
* An entity can be place, person, object, event or a concept, which stores data in the database (DBMS)
* Relationship is nothing but an association among two or more entities
* A weak entity is a type of entity which doesn’t have its key attribute
* It is a single-valued property of either an entity-type or a relationship-type
* It helps you to defines the numerical attributes of the relationship between two entities or entity sets
* ER- Diagram DBMS is a visual representation of data that describe how data is related to each other
* While Drawing ER diagrams in DBMS, you need to make sure all your entities and relationships are properly labeled.



# CHAPTER 5

**IMPLEMENTATION**

## Requirements:

## FRONTEND –

## React.js :

## Frontend library of java script, it is used to build UIs (User Interfaces).

## BACKEND –

## Node.js : Java script runtime environment used to run java script on a machine rather than browser.

## Express js : it is the backend framework layered on the top of node.js . it mainly used to deal with APIs.

## DATABASE –

## MongoDB : a document oriented , no-SQL database used to store the application data .

## DEPLOYMENT –

## Heroku: it is an hoisting site.

## Final Docs –

## Minor (at GitHub)

## GitLab / GitHub as code Hosting platform

## Link : https://github.com/rohitsingh25/Minor

## OTHER –

## Git, Terminal, VS Code

* 1. **HARDWARE REQUIREMENTS:**

**Laptop / Desktop with :**

**OS – windows / mac / linux**

**RAM – 4 GB minimum**

**CPU – 2 GHz**

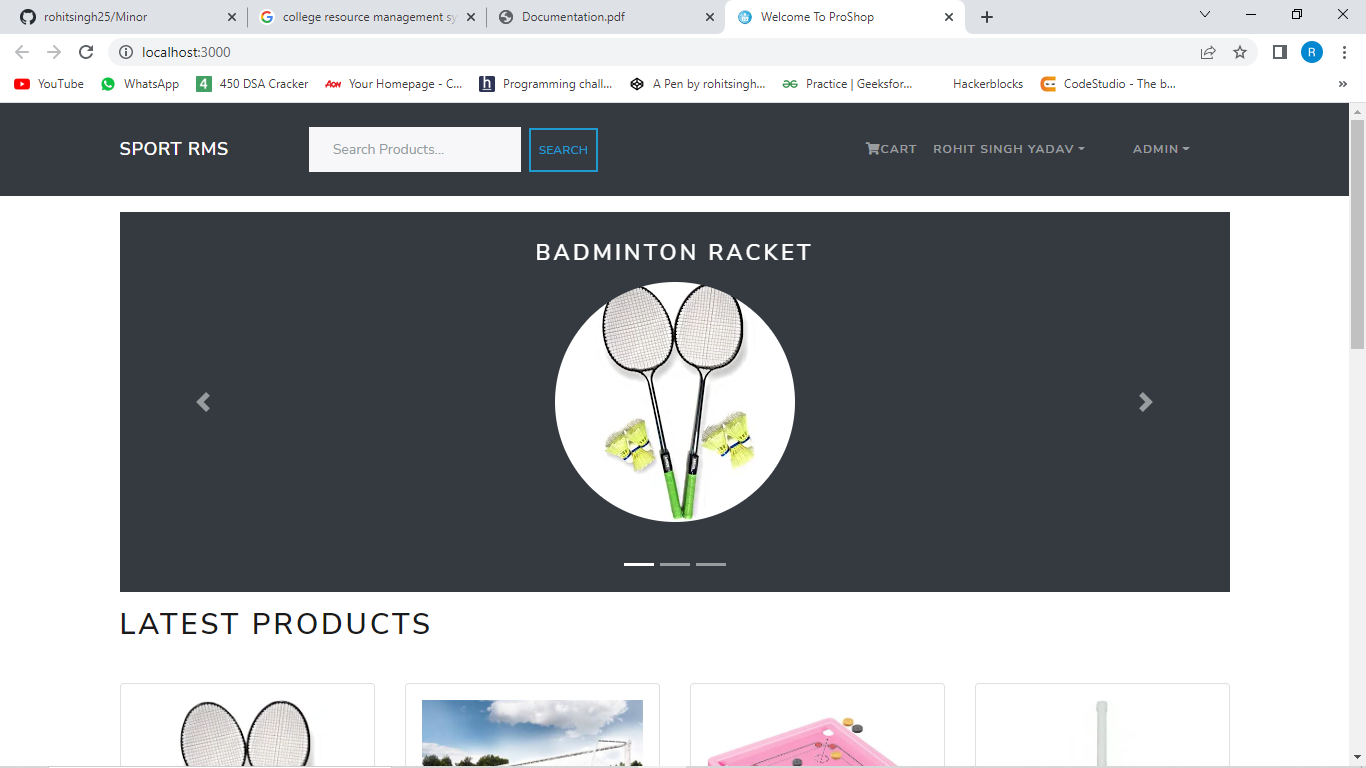
# CHAPTER 6

**Result & Analysis**

## HOME PAGE :

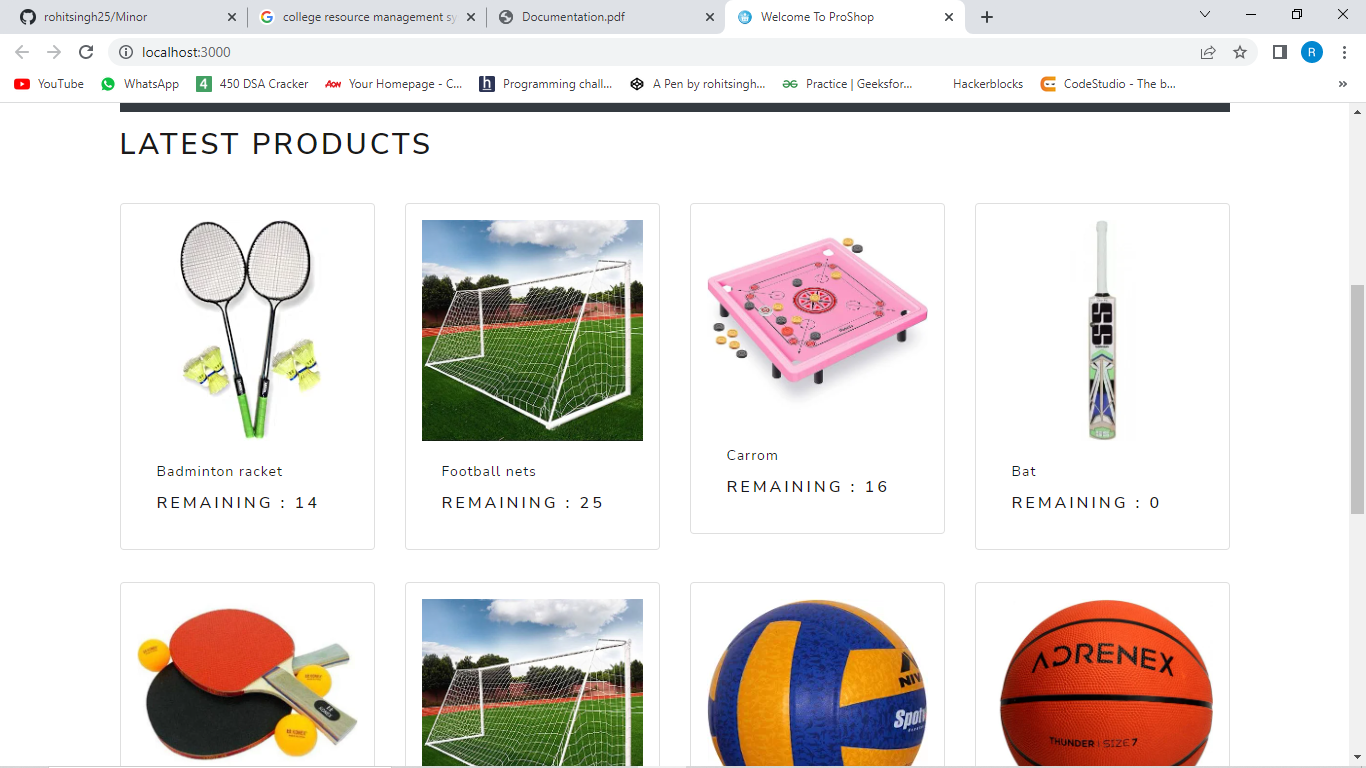
**A student and admin can login to this application with his Email id and Password. This shows the details of the resources present in the sports block .**

## 



## LATEST PRODUCTS :

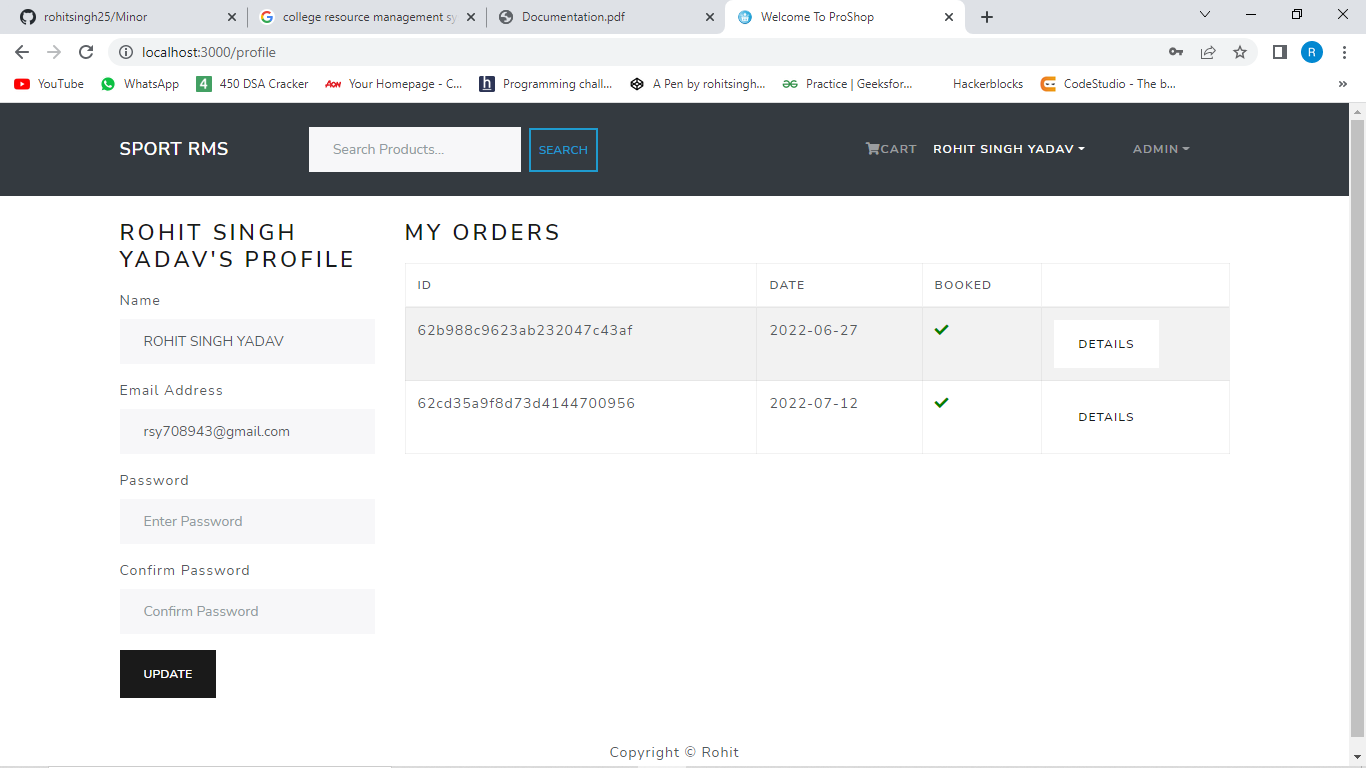
## The user can book a resource if the resource is available and present in stock . displayed over the screen .



## ADMIN PROFILE PAGE :

## This shows the details of the Admin and has an option to change password and logout.

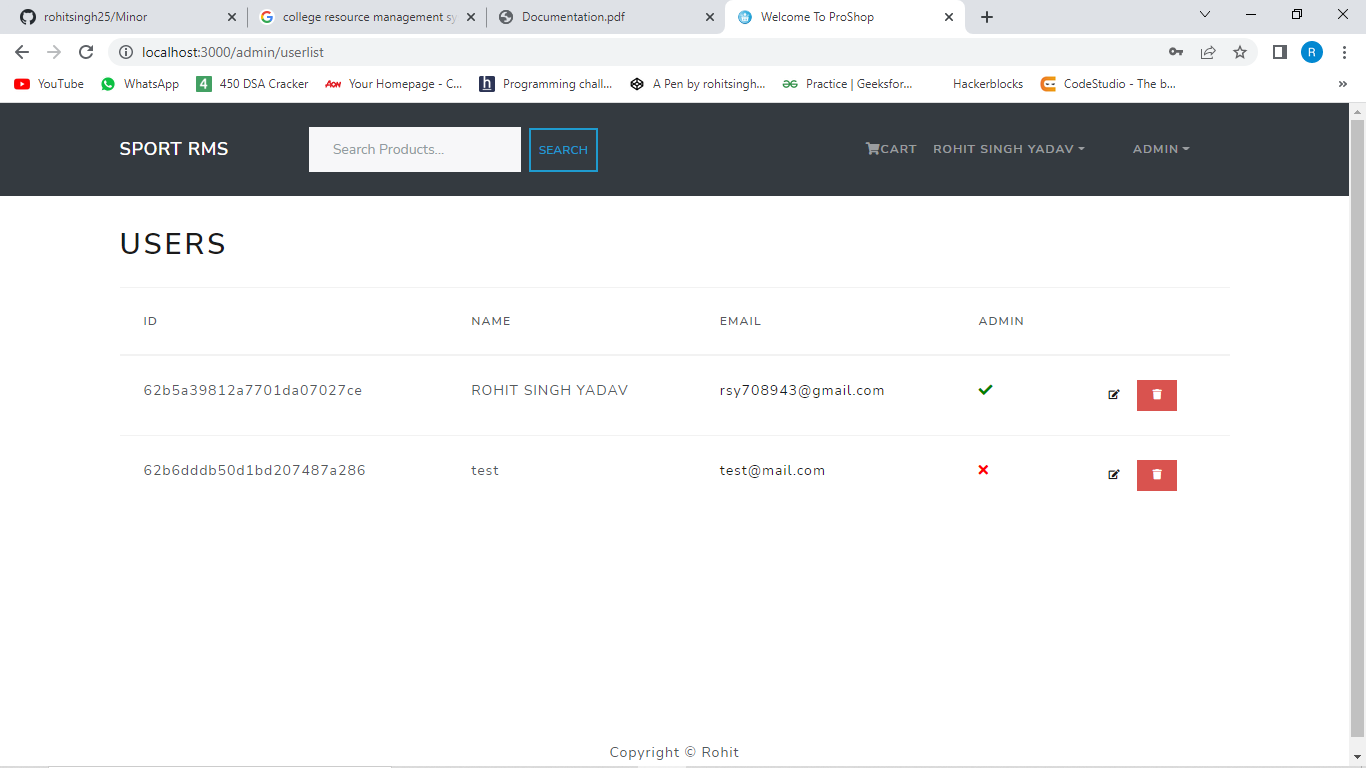
## 



## USER LIST :

## Lists all the users using this application.

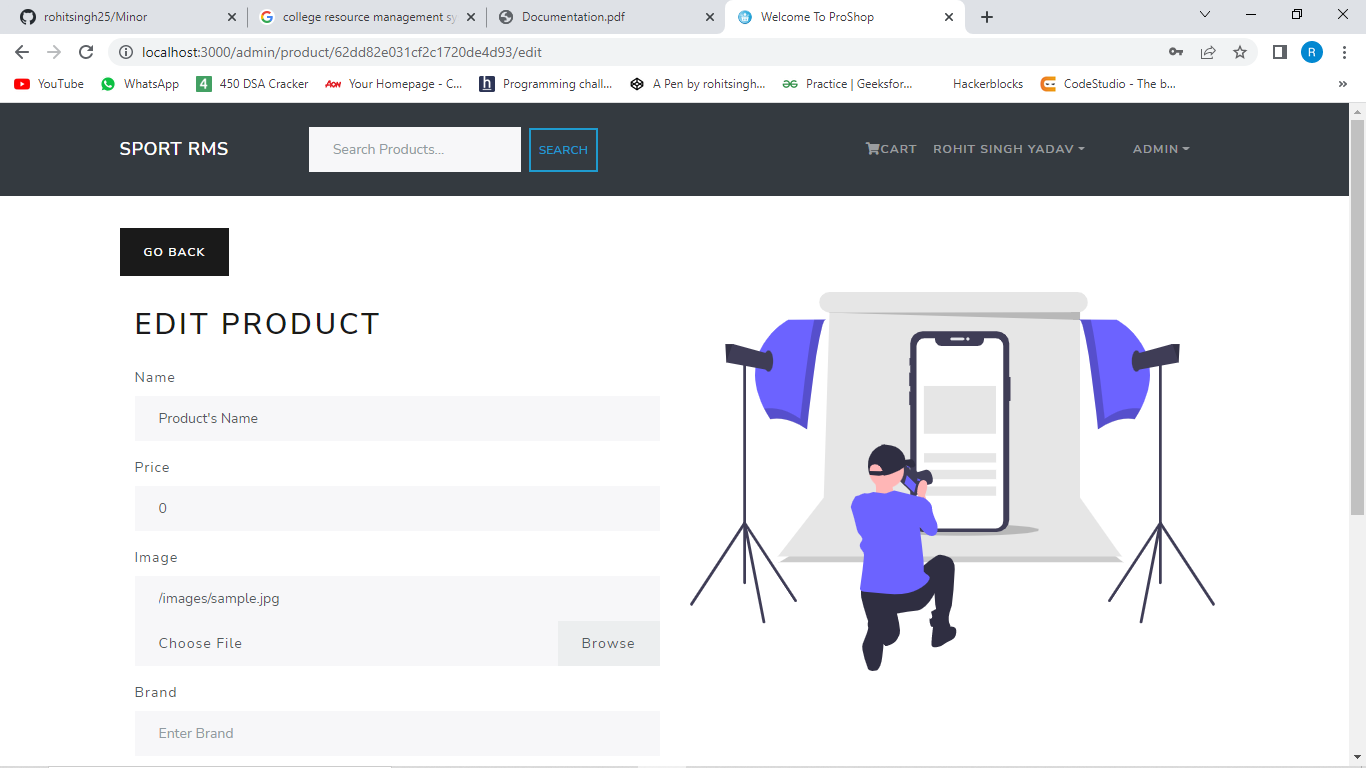
## 



## CREATE PRODUCT:

## UI used to add a product into the database and updating its stock .

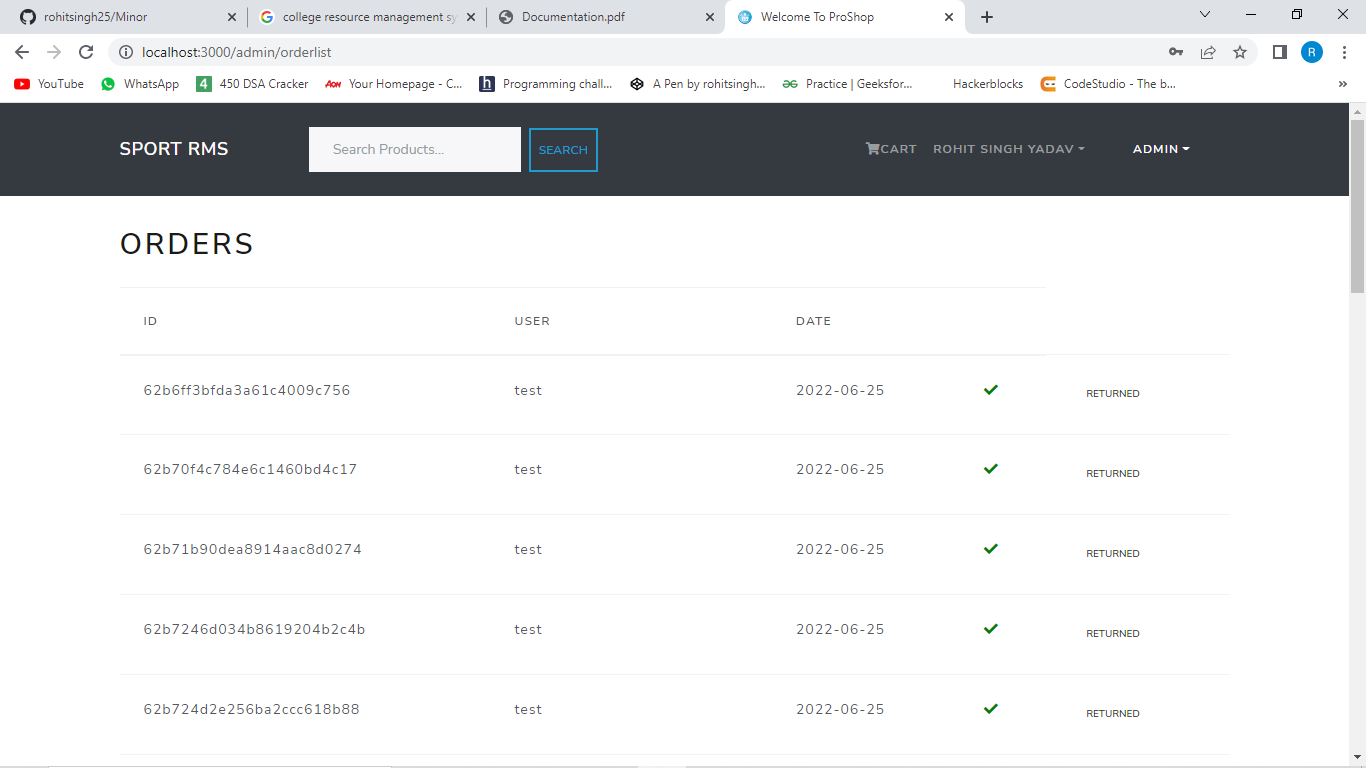
## 



## ORDER HISTORY:

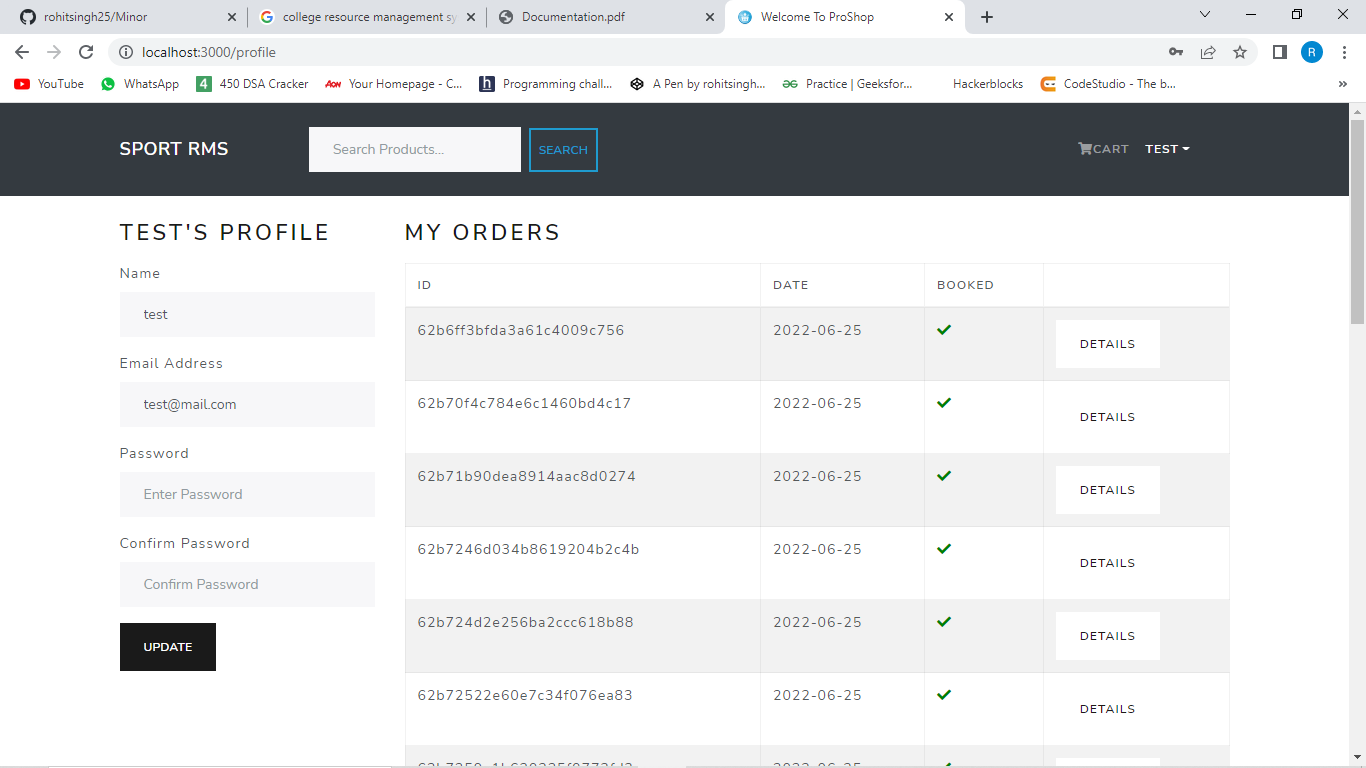
## Lists all the booking made by the user which are returned, not returned, not collected.

## 



## USERS PROFILE :

## This shows the details of the user and has an option to change password and logout.



**Conclusion:**

# CHAPTER 7

**Conclusion & Future Scope**

This project is for computerizing the work employed in order to borrow a sports resource. It is a great improvement over the manual system. The computerization 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

## 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.

# CHAPTER 8

**Bibliography**

**REFERENCES**

* 1. [**https://reactjs.org/**](https://reactjs.org/)
  2. [**https://nodejs.org/en/**](https://nodejs.org/en/)
  3. [**https://expressjs.com/**](https://expressjs.com/)
  4. [**https://www.mongodb.com/**](https://www.mongodb.com/)
  5. [**https://www.xero.com/uk/glossary/stock-management/#:~:text=Stock%20management%20is%20the%20process,in%20your%20inventory%20over%20time**](https://www.xero.com/uk/glossary/stock-management/#:~:text=Stock%20management%20is%20the%20process,in%20your%20inventory%20over%20time)**.**
  6. [**https://www.quora.com**](https://www.quora.com)
  7. [**https://www.wikipedia.com**](https://www.wikipedia.com)
  8. [**https://www.w3schools.com**](https://www.w3schools.com)
  9. [**https://www.tutorialspoint.com**](https://www.tutorialspoint.com)
  10. [**https://www.heroku.com/**](https://www.heroku.com/)