

# INSURANCE POLICY MANAGEMENT SYSTEM

## OIMS

### TEAM MEMBERS

1. Buddareddygaru Bharat Kumar Reddy
2. Mrunalini Basavaraj Bulbule
3. Mukkala Naresh Reddy
4. Pooja TG
5. Shaik Dimple Gousiya
6. Shidhi Vinayak
7. Anand Kumar Yadav
8. Chillathoti Keerthi
9. Morumpalli Sai Kumar Reddy
10. Thejaswi P

Project code	4
Project Name	Insurance Policy Management System.

## Table of Content

1. INTRODUCTION
2. SYSTEM OVERVIEW
3. SUB-SYSTEM DETAILS
4. DATA ORGANIZATION
5. REST APIs to be Built
6. ASSUMPTIONS
7. ADVANTAGES
8. OUTPUT SCREENSHOTS OF OIMS

## 1. Introduction

The project Insurance Policy Management System is a web-based application that allows the administrator to handle all the activities online quickly and safely. Using Interactive GUI anyone can quickly learn to use the complete system. Using this, the administrator doesn't have to sit and manage the entire activities on paper, and at the same time, the head will feel comfortable to keep check of the whole system. This system will give him power and flexibility to manage the entire system from a single online portal.

### Scope and Overview:

The scope of the “**Insurance Policy Management System**” will be to provide the functionality as described below. The system will be developed on a Windows operating system using Java/J2EE, Hibernate, Spring.

## 2. System Overview

The “**Insurance Policy Management System**” supports basic functionalities (explained in section 2.1) for all below listed users.

- Administrator (A)
- Organizer
- Customer (C)

### 2.1 Authentication & Authorization

#### 2.1.1 Authentication:

1. Any Customer or End-user may login using a unique **User id** and **Password**.
2. Admin can also do login using his/her unique **User id** and **Password**.

### **2.1.2 Authorization**

The below listed Operations can be done by both Administrator and Customer.

#### **Administrator and Policy Provider:**

1. Able to log in as admin or Organiser in Insurance Policy Management System with his/her Uniqueuser id and Password.
2. Able to update or Remove Policies Price, Name, Types, Description and its Status.
3. Able to view Customer details and Booked policies.

#### **Customer:**

1. Able to login as Customer in Insurance Policy Management System with his/her unique user id and Password.
2. Able to Register in to System using his/her Name, Email, Phone Number, Address and Password.
3. Able to Apply for the policy and purchase using Card Payment method.
4. Able to view the list of Categories, Sub-Categories and Policies he/she holds.

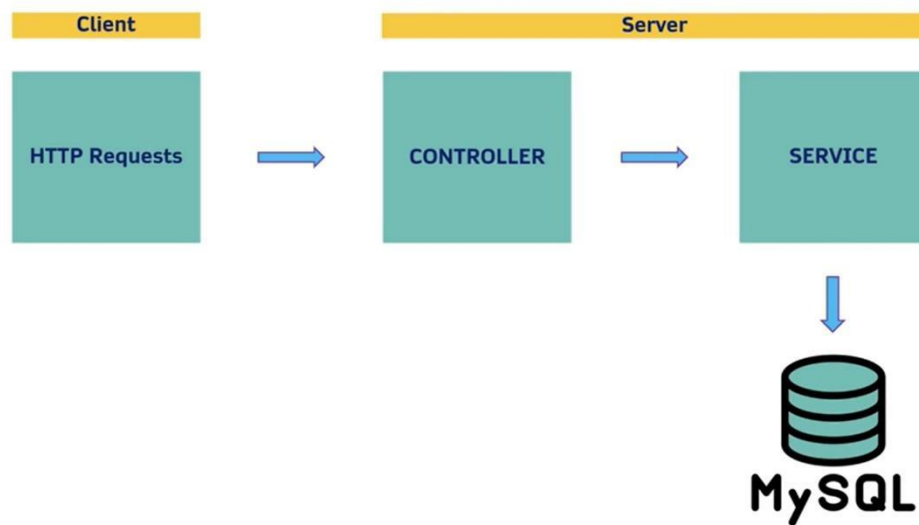
## 2.2 Functional Flow

The functional flow of the messages across different application components is shown below.

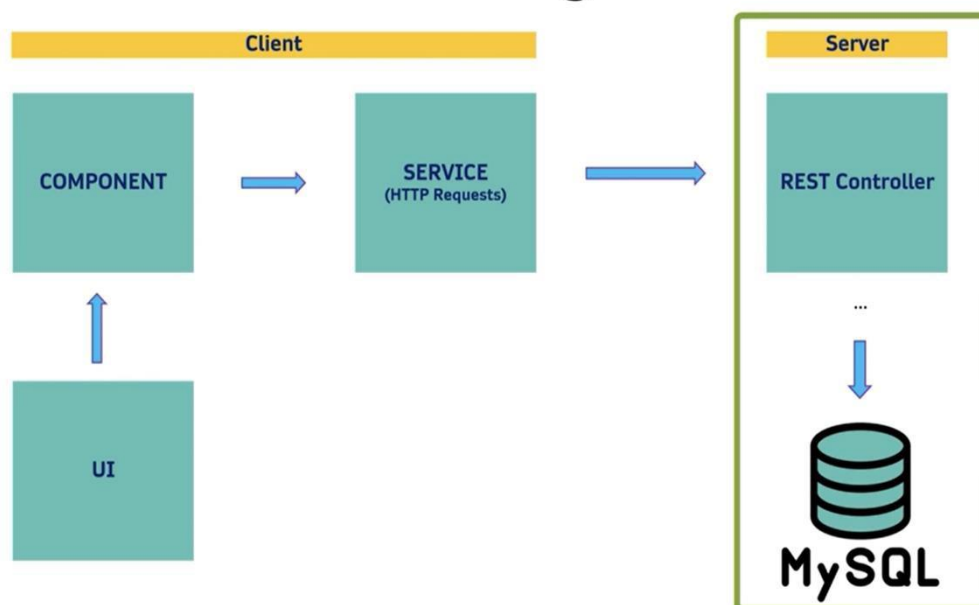
Ex. – Working Flow of a web application



### API Design



### API Design



## 2.3 Environment:

### A. Hardware Requirements:

1. Intel hardware machine (PC with Pentium 1 and above, PC with 2 GB RAM, PC with 250 GB HDD or more).
2. Preferable Operating System -Windows.

### B. Software Requirements:

Number	Description	Type
1	Front-End	Angular Framework
2	Back-End	Spring Boot
3	Database	MySQL
4	Server	Apache Tomcat
5	IDE	Eclipse and Visual Studio Code
6	Testing Tools	Postman and Junit Framework

## Front-End:



Angular is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

## Back-End:



Spring boot is a very convenient java-based framework comparing with others to develop a REST API for the backend. We don't need to waste our time doing a lot of coding for the configuration since spring boot has the feature of autoconfiguration and an embedded inbuilt Tomcat server.

## Database:



MySQL is an open-source relational database management system to manage databases efficiently. It is a very flexible DBMS that provides advanced features and reliability.

## Unit Testing:



We used Postman to initially test the backend REST API endpoints. Postman provides a sleek user interface with which to make HTML requests, without the difficulty of writing a bunch of code to test an API's functionality.

JUnit5 is used for the unit testing of the backend. It is a unit testing framework for java that helps to define the flow of execution of our code using different annotations.



### 3. Sub-system Details

The Insurance Policy Management System is defined, wherein all users need to login successfully before performing any of their respective operations.

Find below (section 3.1 & 3.2 & 3.3) tables that provides functionality descriptions for each type of user /Customer.

#### 3.1 Administrator:

The **Administrator** as a user is defined to perform below listed operations after successful login.

ID	Objects	Operations	Data to include	Remarks
ID-001 To 012	<i>Policy</i>	View	Policy Id, Policy Name, Policy Price, Description.	
ID-005 To ID-0010	<i>Customer</i>	View	Username, email, phone number, Address.	
ID- 001 to ID-005	<i>Booking status</i>	View	Policy Price, Policy Type, Payment Status	
ID- 002 To 004	<i>Organizers</i>	View	Username, Email, Phone number, Policies approved.	

### 3.2 Organizer:

The Organizer as a user is defined to perform below listed operations after successful login.

ID	Objects	Operations	Data to include	Remarks
AD-001 To AD-005	<i>Policy</i>	Add  View  Delete  Modify	Policy Id, Policy Name, Policy Price, Description.	
AD-005 To AD-0010	<i>Customer</i>	View	Username, email, phone number, Address.	
AD – 0011 to AD - 0013	<i>Booking status</i>	View	Policy Id, User Id, Policy Price, Policy Type, Payment Status	

### 3.3 Customer

The customer as a user is defined to perform below listed operations after successful login.

ID	Objects	Operations	Data to include	Remarks
US-001	<i>User</i>	Register	User Id, Username, Password, Email, Phone Number, Password	
US-002	<i>Policy</i>	Apply and delete Policy	Policy Type, Policy Name, Price, Quantity.	

### 3.4 Login | Logout

#### [Web Application - J2EE, Hibernate, Spring]

- Go to Registration screen when you click on Register link.
- Go to Success screen when you login successfully after entering valid username & password fetched from the database.
- Redirect back to same login screen if username & password are not matching.

## 4. Data Organization

This section explains the data storage requirements of the Policy Booking Entry System and **indicative** data description along with suggested table (database) structure. The following section explains few of the tables (fields) with description. However, in similar approach need to be considered for all other tables.

### 4.1 Table: Registration (Database table name : member)

The user specific details such as username, email, phone etc. Authentication, and authorization / privileges should be kept in one or more tables, as necessary and applicable.

Field Name	Description
<i>Member_id</i>	User ID is auto generated after registration(Primary Key)
<i>first_name</i>	First Name of the Customer
<i>last_name</i>	Sur/Last Name of Customer
<i>Email</i>	Customer Email Id
<i>Phone Number</i>	10-digit contact number of users
<i>Password</i>	User Password

#### 4.2 Table: Policy\_Details (Database table name :policy)

This table contains information related to a policy.

Field	Description
<i>Policy_id</i>	Unique Policy Id which is the Primary key to identify the policy.
<i>policy_name</i>	Name of the Policy {Eg;Jeevan policy, Vaahan policy }
<i>policy_place</i>	Availability of the Policy
<i>policy_contact</i>	Contact number
<i>Member_id</i>	Organizer Id (Foreign Key)

#### 4.3 Table: Booking\_Details (Database table name : booking)

This table contains information related to policy booking details.

Field Name	Description
<i>booking_Id</i>	Unique Order ID Auto Generated(Primary Key)
<i>Policytype_name</i>	User Id corresponding to logged in user.
<i>type_id</i>	Policy type Id of the Selected policy(Foreign Key)
<i>member_id</i>	Organizer's Id of the Selected Policy(Foreign Key)
<i>Policy_count</i>	Quantity of the policy (E.g., 2 or 3 number of policy )
<i>Date</i>	Date when the policy has been booked.
<i>Policytype_cost</i>	Price of the policy.
<i>Total_cost</i>	Total price of the policy
<i>Payment_status</i>	Customer's payment status (Eg: pending or processed).

#### 4.4 Table: PolicyType\_Details (Database table name: policytype)

This table contains information related to a policy type details.

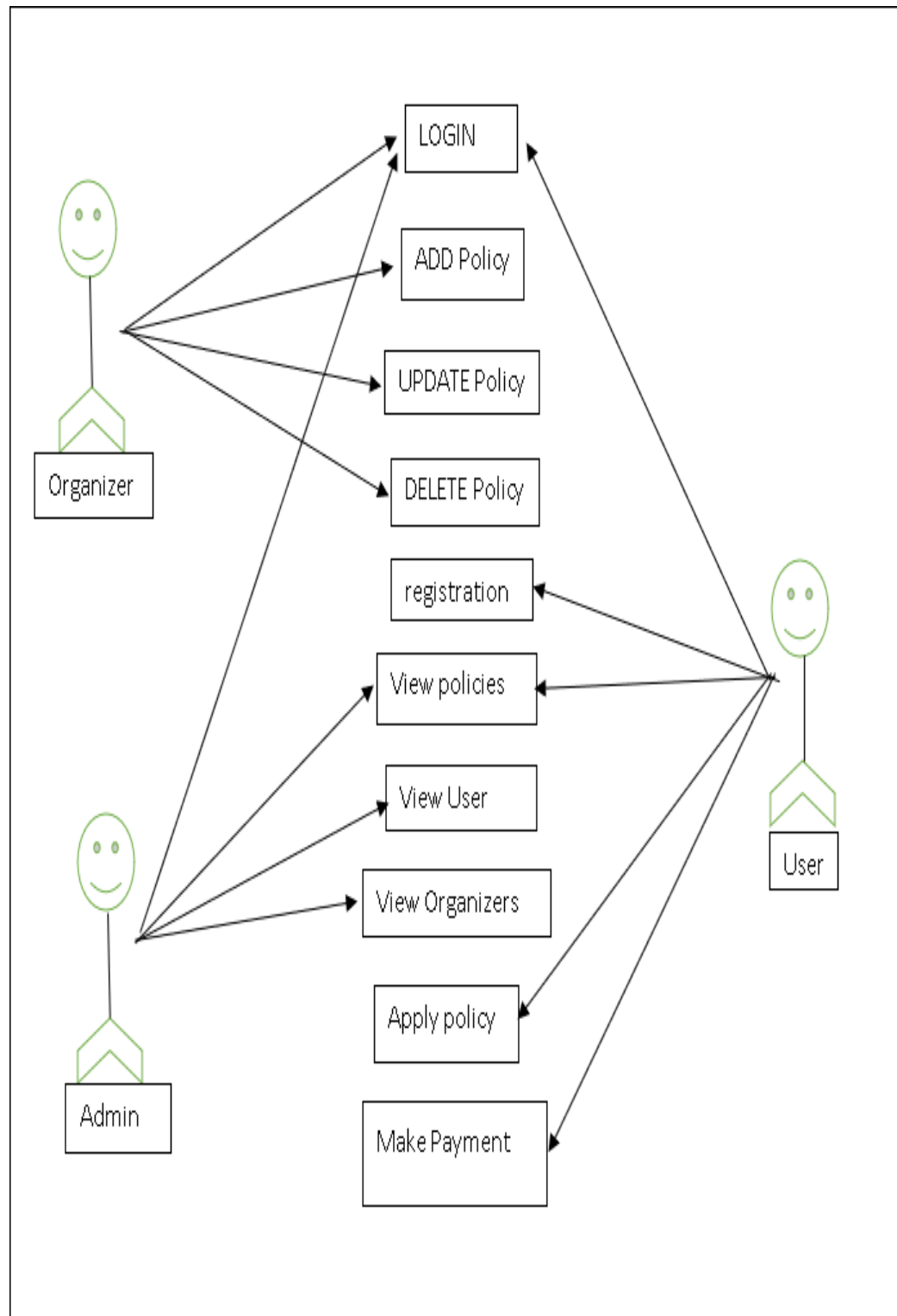
Field	Description
<i>policytype_id</i>	Unique Policy Id which is the Primary key to identify the policy.
<i>policytype_name</i>	Name of the Policy Type {Eg; Life, Health}
<i>policytype_cost</i>	Price of the policy
<i>organizer_contact</i>	Organizer Contact number
<i>Member_id</i>	Organizer Id(ForeignKey)

#### 4.5 Table: Notification\_Details (Database table name : notifications)

This table contains information related to a policy.

Field	Description
<i>notification_id</i>	Unique Notification Id which is the Primary key.
<i>message</i>	Get notification for every action to the User/Organizer/admin.
<i>Member_id</i>	Member Id who's received the notification(Foreign Key)
<i>date</i>	Date when notification was received
<i>time</i>	Date when notification was received.

## 4.5 UML Diagram



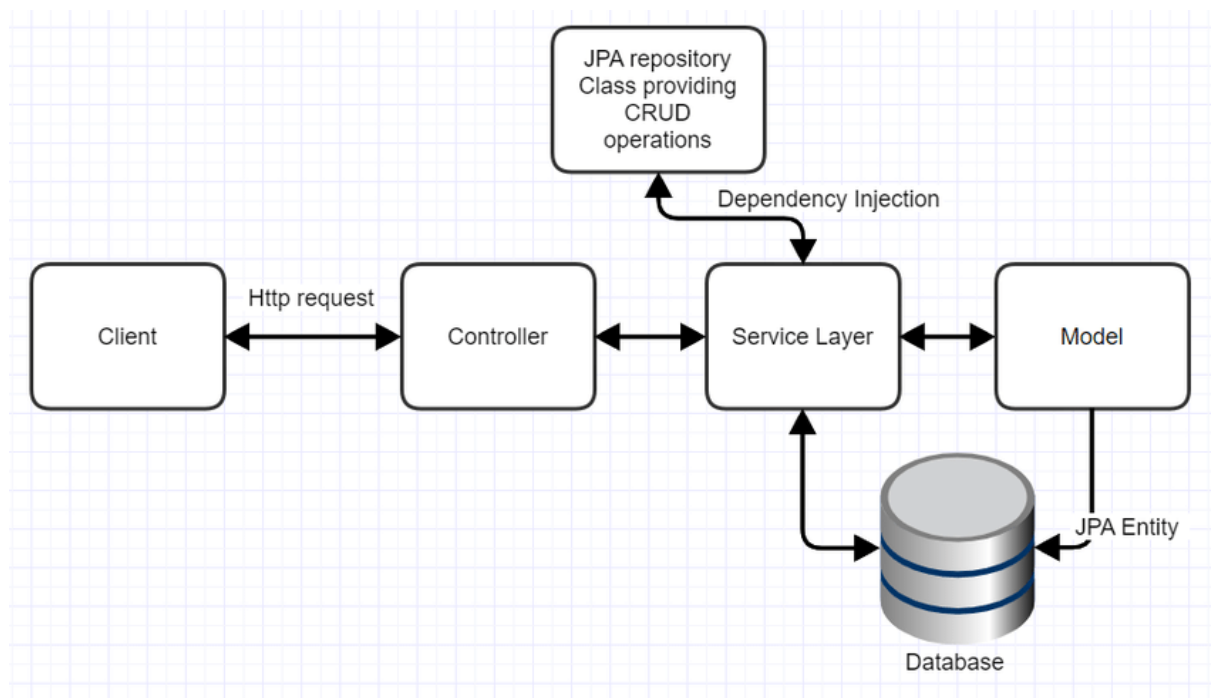
## 5. REST APIs to be Built.

### Technology stack:

- Spring Boot
- Spring REST
- Spring Data JPA

### 5.1. Steps for creating a project in **Spring Boot**:

- In Eclipse IDE, we have to create a new Maven Project using required Group Id , Artifact Id ,Packaging and version.
- In POM.xml we need dependencies such as Spring Data JPA, Spring Boot Devtools , Mysql Driver and Spring web to support Spring application.
- Later we have to configure application.properties to connect withMysql database.
- By creating Model, Service, Repository, Controller Packages we can perform the required Business Logics.



**Fig A : Representation of Spring MVC pattern**



## 5.2 Steps to create entities to perform Business logics

### 1. Creating *Member* Entity:

Here will have multiple layers into the application:

1. In Eclipse IDE, We need to create an (Model class)Entity: *Member*.
2. By Creating a MemberRepository interface and will make use of Spring Data JPA.
  - a. Will have a query to validate member.
  - b. Add the member details by extending JPA Repository.
3. By Creating a MemberService class we can perform the required Business logics and exposes all Services.
4. Finally, by creating a MemberController class will handle all http requests and also will have following URI 'S:

URI	METHODS	Description	Format
<u>/member</u> <u>/getMember/{memberId}</u>	GET	Give a single user description searched based on username	JSON
/member /add	POST	Add the user details	JSON
/member /updateMember /{member}	PUT	Update the user details	JSON
/member /deleteMember/ {memberId}	DELETE	Delete user by id	String
/member /getAllUsers	GET	Give all Users description	JSON
/member /getAllOrganizers	GET	Give all Organizers description.	JSON

## 2. Creating *Booking* Entity:

Build a RESTful resource for **Booking** manipulations, where following operations to be carried out. Here will have multiple layers into the application:

1. In Eclipse IDE, We need to create an (Model class)Entity: *Booking*.
2. By Creating a BookingRepository interface and will make use of Spring Data JPA.
  - a. Will have deleteBookingById method to remove booking with specific policy Id.
  - b. Will have update method for updating the policy and its Policy Count.
  - c. Will have List method to see the list of Bookings made.
  - d. Add the member details by extending JPA Repository.
3. By Creating a BookingService class we can perform the required Business logics and exposes all Services.
4. Finally, by creating a BookingController class will handle all http requests and also will have following URI 'S:

URI	METHODS	Description	Format
<u>/booking/getBooking/{bookingId}</u>	GET	Give a single user description searched based on username	JSON
/booking/add	POST	Add the user details	JSON
/booking/deleteBooking/{bookingId}	DELETE	Delete booking by Id	String
/booking/getAllBookings	GET	Give all Users description	JSON
/booking/bookingsByUserId/{userId}	GET	Give all Organizers description.	JSON

/booking/bookingDetail/{bookingId}	GET	Give details of the booked policy	JSON
/booking/doPayment/{bookingId}	GET	Make Payment	JSON
/booking/upcomingBookings/{orgId}	GET	Give List of Upcoming Bookings only	JSON
/booking/previousBookings/{orgId}	GET	Give previous booking	JSON
/booking/getDates/{policyId}	GET	Give dates of bookings for the particular policy	JSON
/booking/checkActiveBookings/{policyId}	GET	Give active booking on policy	JSON

### 3. Creating *Policy* Entity:

Build a RESTful resource for **Policy** manipulations, where CRUD operations to be carried out. Here will have multiple layers into the application:

- (1) In Eclipse IDE, We need to create an (Model class)Entity: *Policy*.
- (2) By Creating a PolicyRepository interface and will make use ofSpring Data JPA.
  - (a) Will have findByPolicyName method.
  - (b) Add the Policy details method.
  - (c) Will have deletePolicyById method.
  - (d) Will have findAllPolicies method.
  - (e) Will have list types to view all policies.
- (3) By Creating a PolicyService class we can perform the required Business logics and exposes all Services.
- (4) Finally, by creating a PolicyController class will handle all http requests and also will have following URI 'S:

URI	METHODS	Description	Format
<u>/policy/organizer/{orgId}</u>	GET	Give all policys by organizer Id	JSON
/policy /places	GET	Give all distint places	JSON
/policy/ /getPolicys/{place}	GET	Give policy for Selected Places	JSON
/policy /add	POST	Add Policy	JSON
/policy /getPolicy/{policyId}	GET	Give policy by policy Id	JSON
/policy /getPolicys	GET	Give List of the policy	JSON
/policy/updatePolicy	PUT	Update the Policy	JSON
/policy//deletePolicy/{policyId}	DELETE	Delete the Policy	String

#### 4. Creating *PolicyType* Entity:

Here will have multiple layers into the application:

1. In Eclipse IDE, We need to create an (Model class)Entity: *PolicyType*.
2. By Creating a PolicyTypeRepository interface and will make use of Spring Data JPA.
  - a. Will have save method helps to save PolicyType in portal.
  - b. Will have deletePolicyTypeById method to remove policy type with specific policy Id.
  - c. Will have update method for updating the policy Type and its Quantities.
  - d. Will have List method to see the list of policytypes.

3. By Creating a PolicyTypeService class we can perform the required Business logics and exposes all Services.

4. Finally, by creating a PolicyTypeController class will handle all http requests and also will have following URI 'S:

URI	METHODS	Description	Format
<u>/policytype/getPolicyTypes/{policyId}</u>	GET	Give all policytypes by policy Id	JSON
/policytype/getOne/{policytypeName}/{policyId}	GET	Give policyType by Name and PolicyId	JSON
/policytype/getPolicyType/{policytypeId}	GET	Give PolicyType	JSON
/policytype/add	POST	Add PolicyType	JSON
/policytype/updatePolicyType	PUT	Update the PolicyType	JSON
/policytype/deletePolicyType/{policytypeId}	DELETE	Delete the PolicyType	String

## 5. Creating *Notification* Entity:

Here will have multiple layers into the application:

1. In Eclipse IDE, We need to create an (Model class)Entity: *Notification*.
2. By Creating a NotificationRepository interface and will make use ofSpring Data JPA.
  - a. Will have List method to see the list of notifications.
  - b. Will have deletenotificationById method to remove Notifications.
- 3.By Creating a NotificationService class we can perform the requiredBusiness logics and exposes all Services.

4. Finally, by creating a NotificationController class will handle all httprequests and also will have following URI 'S:

URI	METHODS	Description	Format
<u>/notification/getNotification/{memberId}</u>	GET	Give Notifications	JSON
/notification/deleteNotification/{notificationId}	DELETE	Delete the Notification	String

## 6. Assumptions :

- Each user(Admin,Organizer,Customer) must have a valid user id and password
- Server must be running for the system to function
- Users must log in to the system to access any record.
- Only the Administrator can View both Organizer's and User's records. Only admin can add,delete,update,view Policies and its Types.

## 7.Advantages:

1. Financial Protection
2. Distribution of Risk/Spreading of Risk.
3. Stability of Living Standard.
4. Encouragement to Savings.
5. Job Opportunities.
6. Promotes foreign/international trades.

### 1. Financial Protection:

In life, there is no such thing as a guarantee. There may be a loss of life, as well as some business accidents. The loss is difficult to bear in both of these cases. As a result, insurance provides financial protection against such a sudden loss.

### 2. Distribution of Risk/Spreading of Risk:

The underlying concept of insurance is to spread the risk across a large number of people. People pay a certain amount to an insurance company up to a certain time or lifetime and receive a refund if a loss occurs. Risk in life or business cannot

be eliminated, but it can be reduced, distributed, or shared. So, in this case, insurance companies bear risks in order to redistribute business and individual risks among insurance companies.

### **3. Stability of Living Standard:**

Insurance provides financial support to ensure that people can sustain and maintain stability in living standards against an unforeseen risk of losses.

### **4. Encouragement to Savings:**

In insurance, people pay a certain amount of money for a fixed time or lifetime based on an agreement and this helps to develop a habit of saving money. Knowing the importance of saving, people start doing saving in various fields.

### **5. Job Opportunities:**

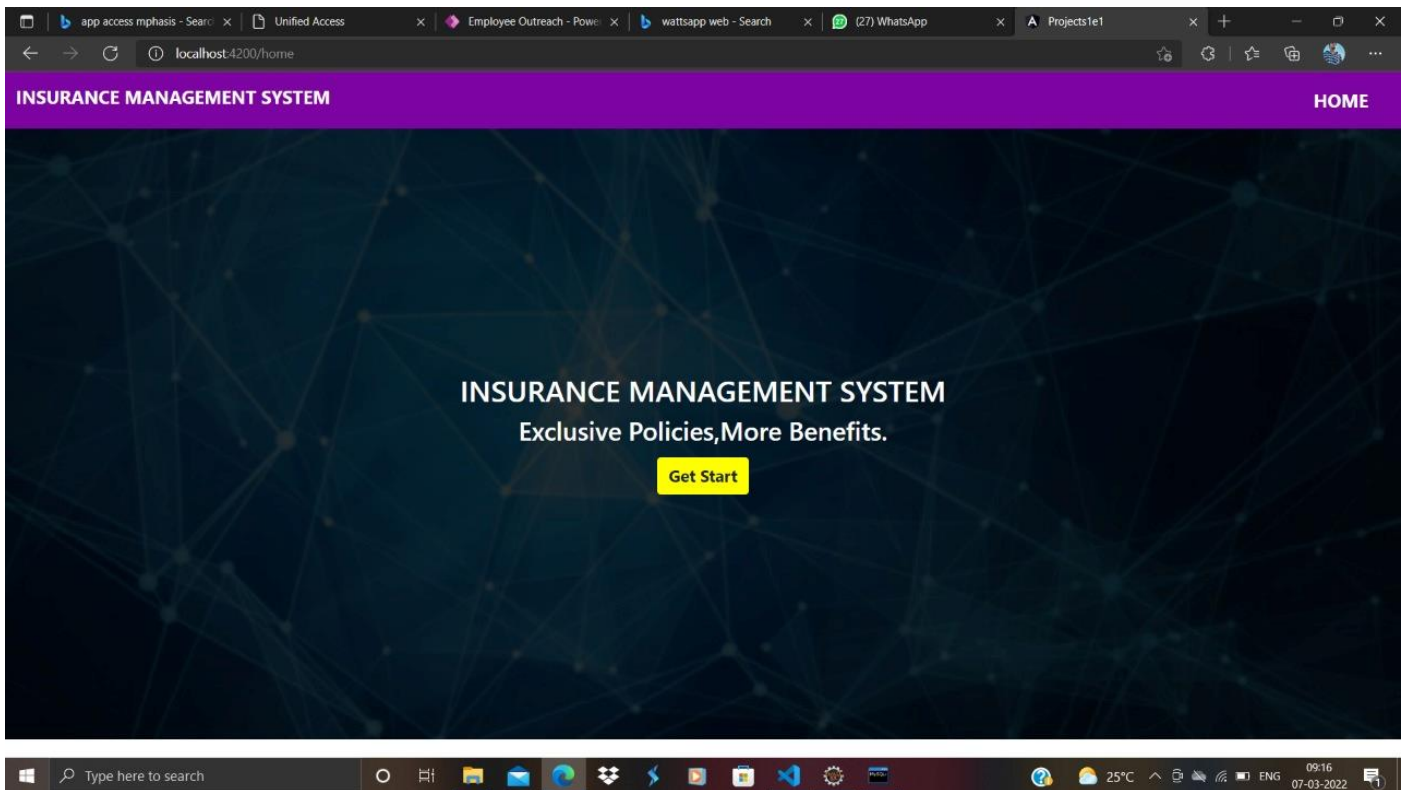
Insurance, like any other business, is a successful business model. It targets many entrepreneurs and business owners. As a result, there is a lot of cash flow in the business. They need employees to handle and maintain cash flow and run the business, so they open vacancies in various positions based on qualifications and provide job opportunities.

### **6. Promotes foreign/international trades:**

Many years ago, people were afraid to engage in international trade because of the possibility of accidents while transporting goods via ships, roads, or other modes of transportation. However, in today's competitive global economy, insurance companies bear all of those risks and compensate for losses. They also protect an exporter of goods and services from nonpayment by a foreign buyer.

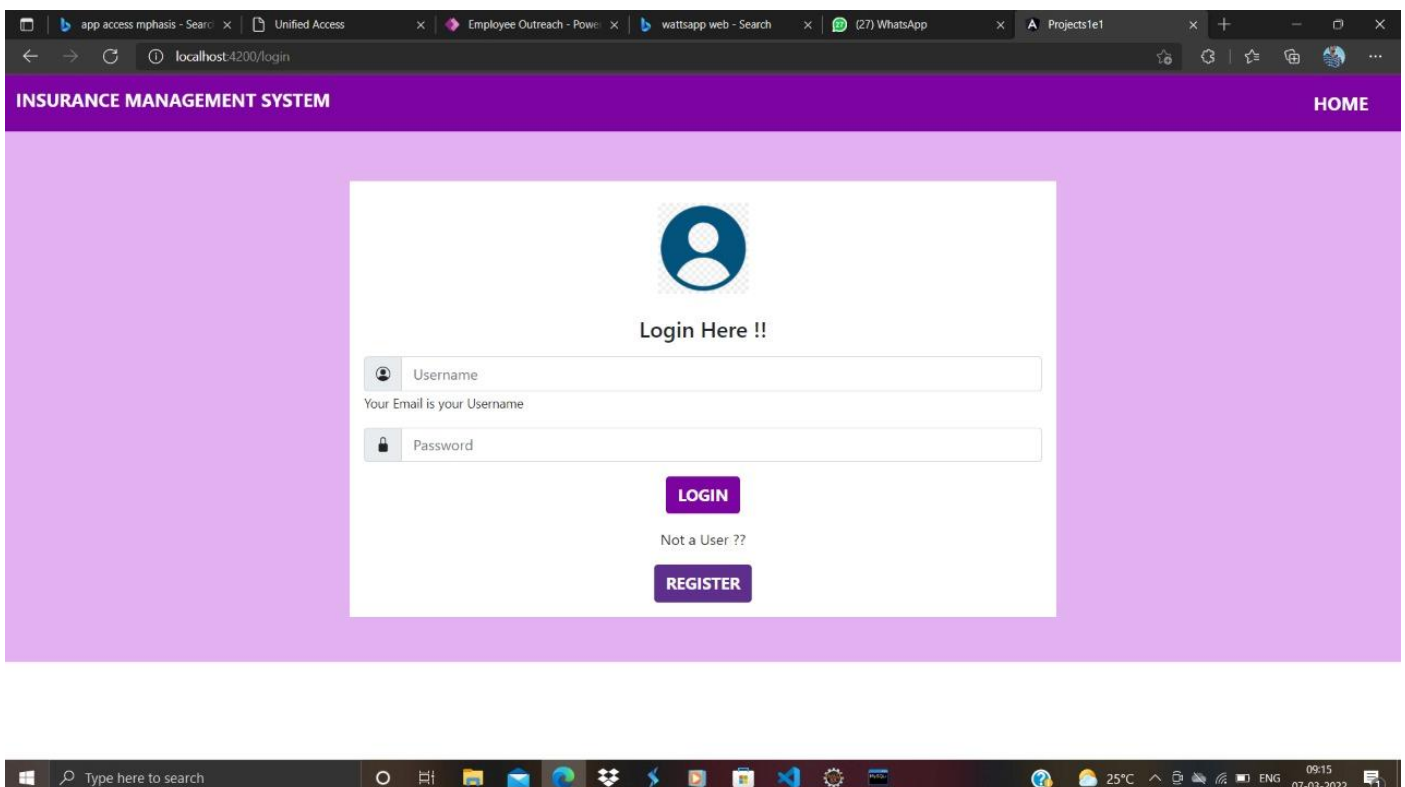
## 8. Output Screenshots of OIMS:

### WELCOME PAGE



### LOGIN PAGE

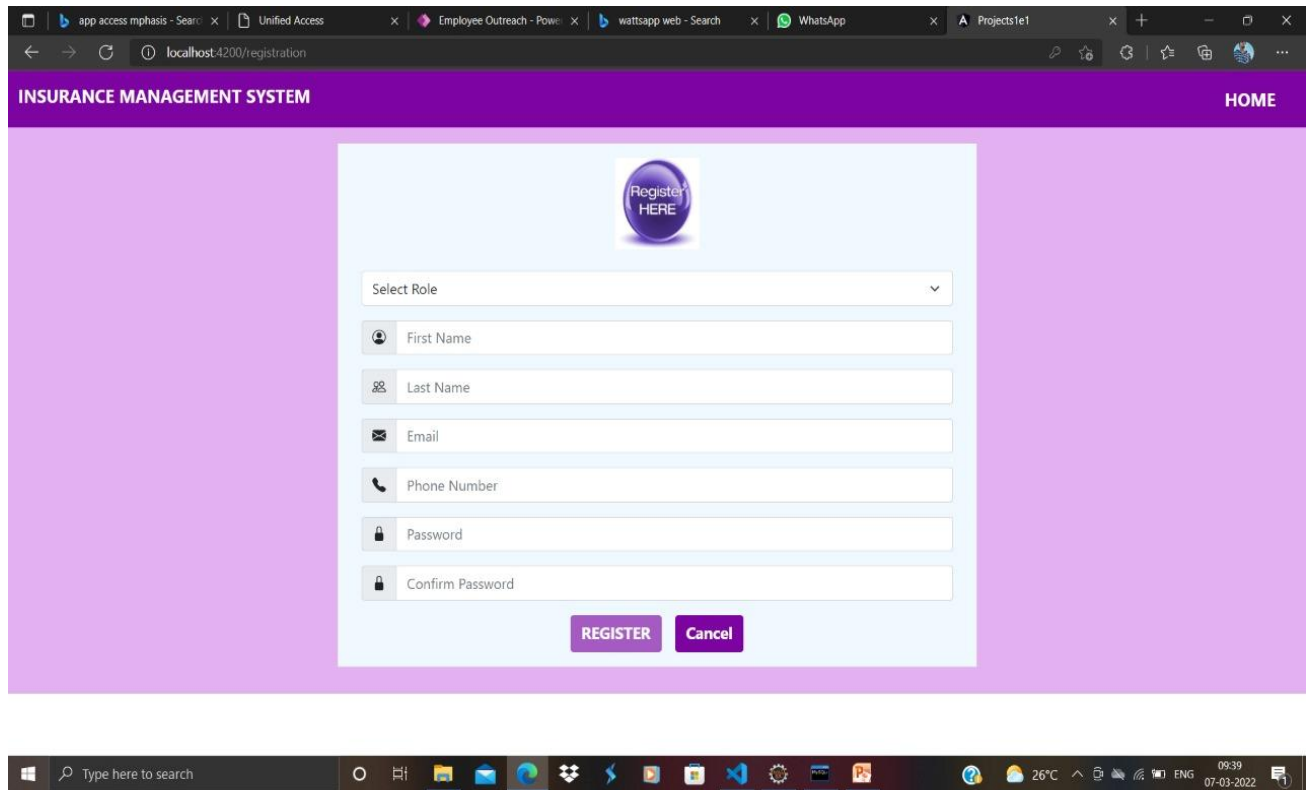
This is the login page for Admin, User and Organizer.





# REGISTRATION PAGE FOR USER AND ORGANIZER

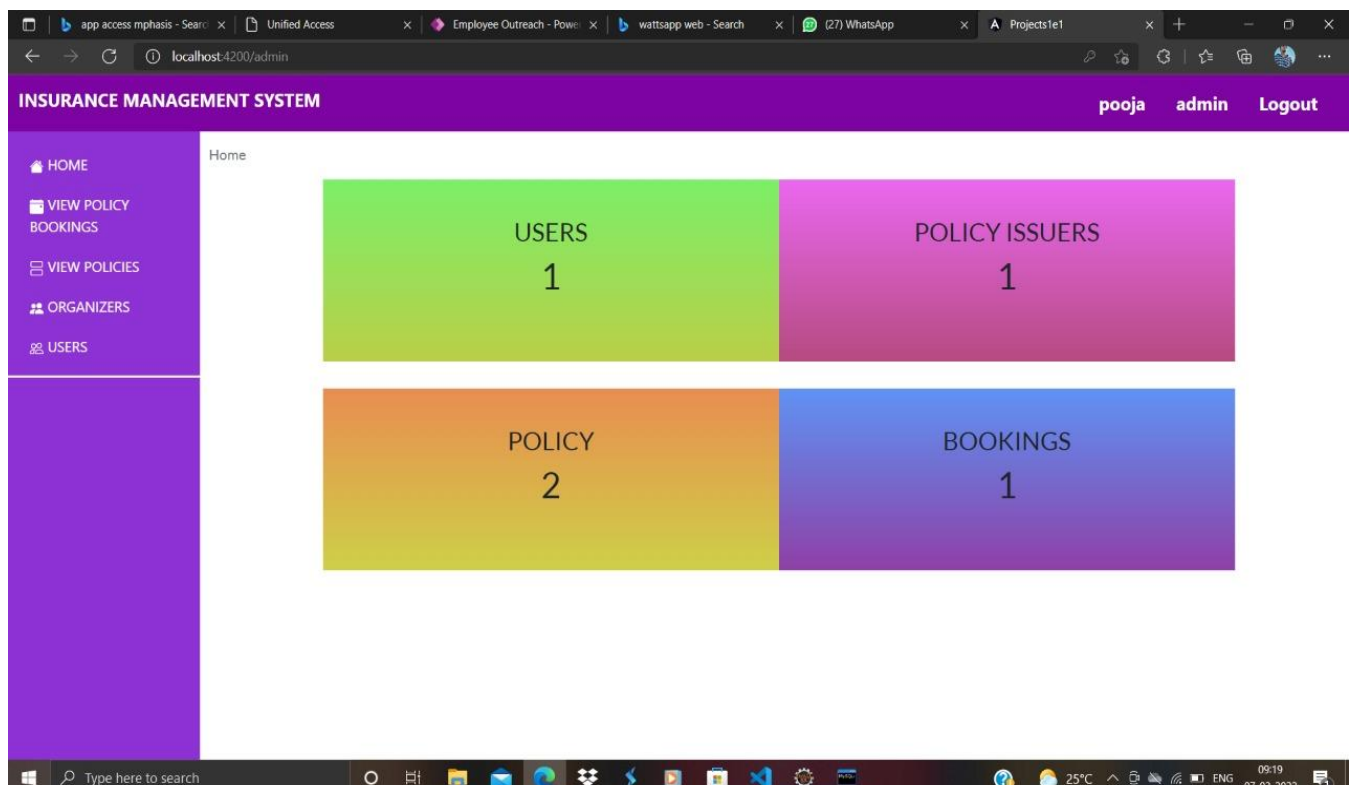
This page is used for new users and organizer to register into portal.



The screenshot shows a web browser window with the URL `localhost:4200/registration`. The page has a purple header with the text "INSURANCE MANAGEMENT SYSTEM" on the left and "HOME" on the right. The main content area is light blue and contains a registration form. At the top of the form is a purple button with a white "Register HERE" icon. Below this is a "Select Role" dropdown menu. The form includes input fields for "First Name", "Last Name", "Email", "Phone Number", "Password", and "Confirm Password". At the bottom of the form are two buttons: "REGISTER" (purple) and "Cancel" (dark blue). The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right shows the date and time as "09:39 07-03-2022".

## ADMIN: HOME

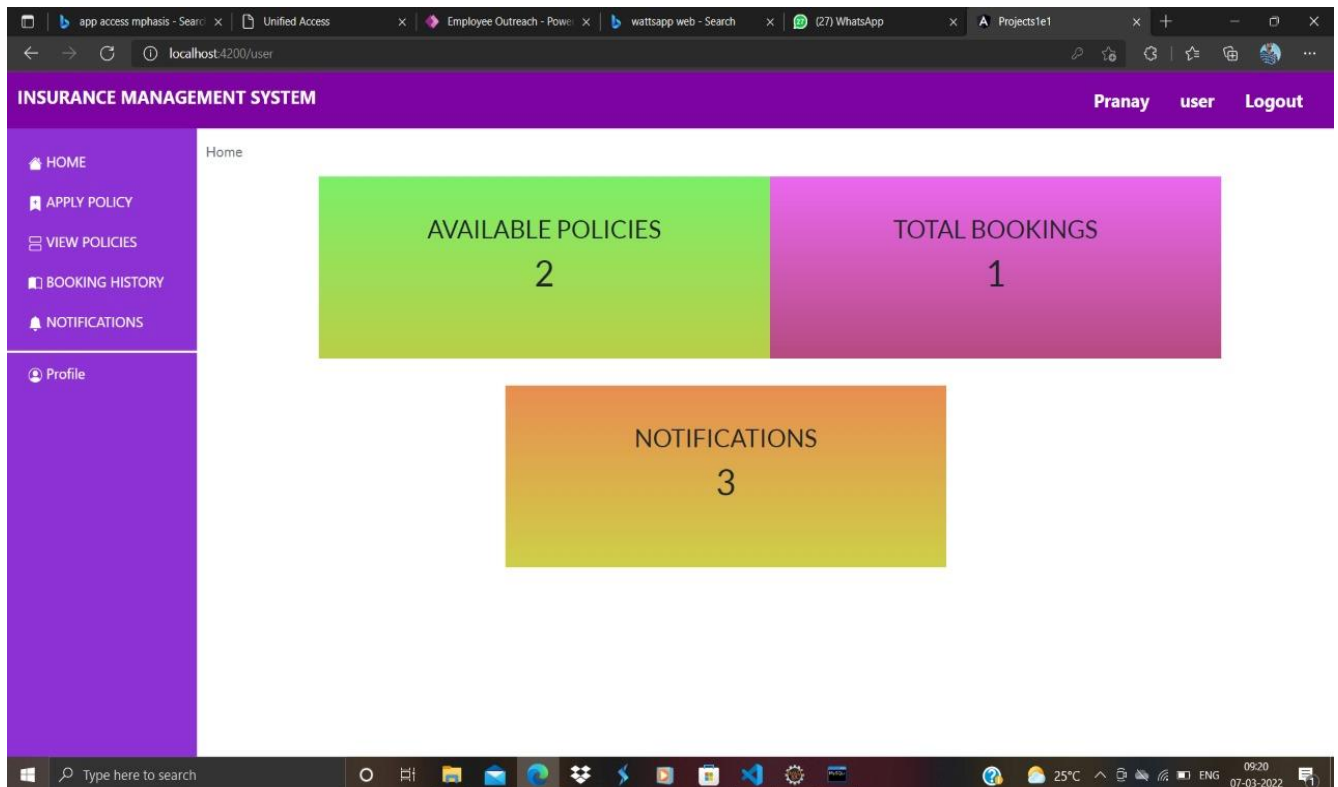
Once after successful admin login, we will get homepage. Where we can view policy booking, view policies, organizers and users.



The screenshot shows a web browser window with the URL `localhost:4200/admin`. The page has a purple header with the text "INSURANCE MANAGEMENT SYSTEM" on the left and "pooja admin Logout" on the right. The main content area is white and contains a dashboard. On the left is a purple sidebar with a list of menu items: "HOME", "VIEW POLICY BOOKINGS", "VIEW POLICIES", "ORGANIZERS", and "USERS". The main content area is divided into four colored boxes: a green box labeled "USERS 1", a pink box labeled "POLICY ISSUERS 1", an orange box labeled "POLICY 2", and a blue box labeled "BOOKINGS 1". The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right shows the date and time as "09:19 07-03-2022".

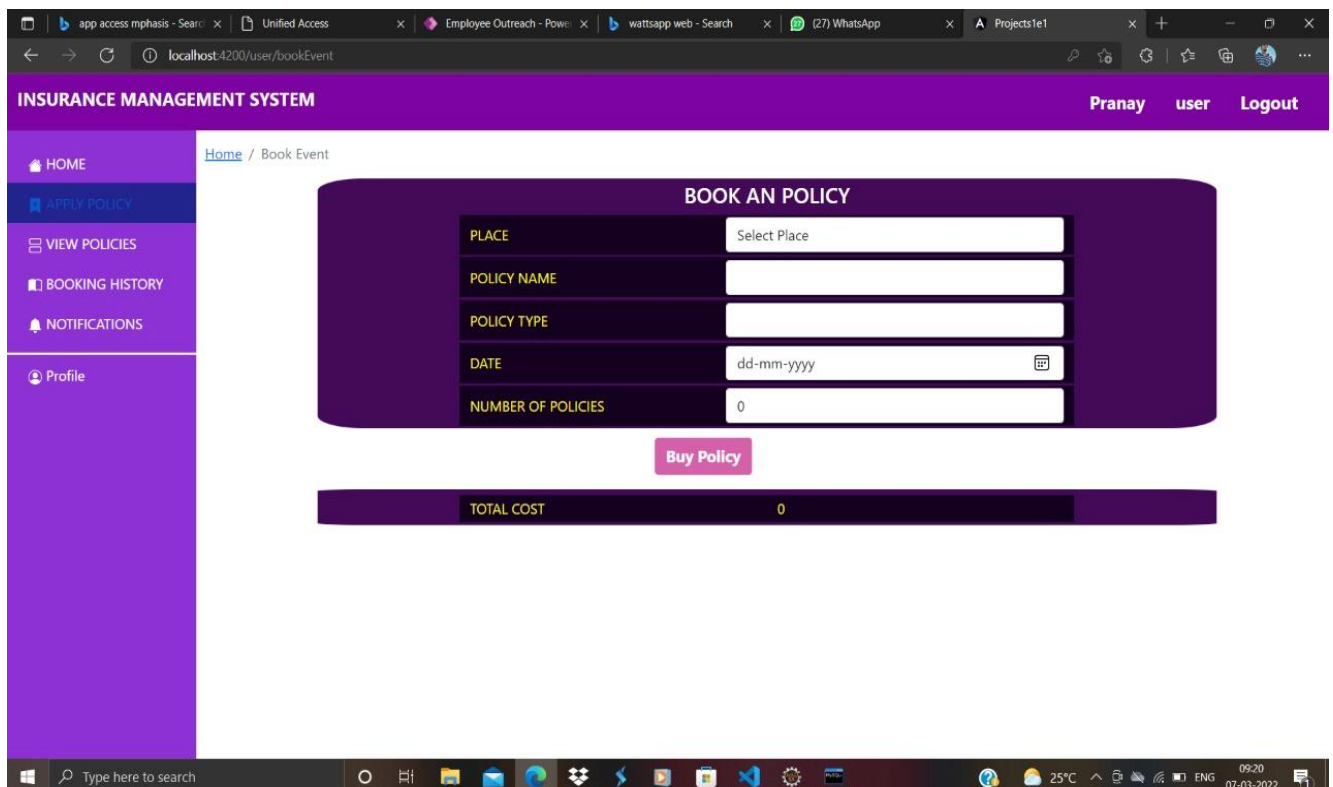
## USER: HOME

After successful registration of user, we will get homepage. Where we can Apply policy, View booking history, policies, Notifications and profiles.



## USER: POLICY BOOKING PAGE

From this page user can policy and buy policy.



## USER: VIEW AVAILABLE POLICIES

From this page user can view all the policies which are available in the platform.

The screenshot shows a web browser window with the URL `localhost:4200/user/viewVenues`. The page title is "INSURANCE MANAGEMENT SYSTEM". The user is logged in as "Pranay user" and can click "Logout". The left sidebar contains links: HOME, APPLY POLICY, VIEW POLICIES (highlighted), BOOKING HISTORY, NOTIFICATIONS, and Profile. The main content area shows a table of available policies:

Name	Place	Contact	Action
LIC Policy	Chhattisgarh	9111111111	<a href="#">Details</a> <a href="#">Availability</a>
Super Health Policy	Delhi	9111111111	<a href="#">Details</a> <a href="#">Availability</a>

## ORGANIZER: HOME

Once after successful organizer login, we will get homepage. Where we can view manage policy, view booking, booking history, notifications and profile.

The screenshot shows a web browser window with the URL `localhost:4200/organizer`. The page title is "INSURANCE MANAGEMENT SYSTEM". The user is logged in as "Vijaya organizer" and can click "Logout". The left sidebar contains links: HOME, MANAGE POLICY, VIEW BOOKINGS, BOOKING HISTORY, NOTIFICATIONS, and Profile. The main content area shows a dashboard with four cards:

- POLICY 2
- UPCOMING BOOKINGS 0
- NOTIFICATIONS 3
- PAST BOOKINGS 1

## ORGANIZER: MANAGE POLICY PAGE

From this page we can add policies and view all the existing policies.

The screenshot shows a web browser window with the URL `localhost:4200/organizer/manageVenue`. The page title is "INSURANCE MANAGEMENT SYSTEM". The user is logged in as "Vijaya" and is in the "organizer" role. The page has a sidebar with navigation links: HOME, MANAGE POLICY, VIEW BOOKINGS, BOOKING HISTORY, NOTIFICATIONS, and Profile. The main content area is titled "Manage Your Policies" and contains an "ADD Policy" button. Below the button is a table with the following data:

ID	Policy Name	Place	Contact	Action
2	LIC Policy	Chhattisgarh	9111111111	<a href="#">Details</a> <a href="#">Delete</a>
3	Super Health Policy	Delhi	9111111111	<a href="#">Details</a> <a href="#">Delete</a>

## ORGANIZER: ADD/UPDATE/DELETE POLICY

From this page the organizer can add, update and delete policy.

The screenshot shows a web browser window with the URL `localhost:4200/organizer/venueDetails/4/showEvents/4`. The page title is "INSURANCE MANAGEMENT SYSTEM". The user is logged in as "Vijaya" and is in the "organizer" role. The page has a sidebar with navigation links: HOME, MANAGE POLICY, VIEW BOOKINGS, BOOKING HISTORY, NOTIFICATIONS, and Profile. The main content area is titled "Policy Details" and contains an "UPDATE" button. Below the button is a form with the following data:

Policy Name	HDFC Property Insurance
Policy Place	Delhi
Policy Contact	9222222222

Below the form are two buttons: "Add Type and Price" and "Add Policy". Below these buttons is a table with the following data:

Property insurance	125000	<a href="#">Update</a>	<a href="#">Delete</a>
--------------------	--------	------------------------	------------------------

## **8. CONCLUSION:**

- ❖ Insurance management system is all about the modernizing a insurance company through use of technology.
- ❖ Computers helps in it and take over the manual system for quick and easy functioning. This insurance management system is a quite the reliable and is proven on many stages.
- ❖ All the basic requirements of the insurance company are provided here in order to manage it perfectly and large amount of data can also be stored.
- ❖ It gives many facilities like searching for the detail of policies, available policies as well as buying facilities . So its a important system for modern days.