Project: Insurance Policy Management

FUNCTIONAL SPECIFICATION

Group:	4
Project Name:	Insurance Policy Management

Table of Contents

1 Introduction

Pvt Ltd is a company which builds a software system which is responsible for adding and processing of a product.

Pvt Ltd plans to develop "Insurance Policy Management" web application, where users can register, login, view the list of available policies and enroll themselves.

Scope and overview
The scope of the “Insurance policy management” will be described below. The system is developed on windows OS using Spring-boot, Angular, MySQL

2 System Overview

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

- Administrator (A)
- Customer (C)

2.1 Authentication & Authorization

2.1.1 Authentication:

Any end-user should be authenticated using a unique user-id and password.

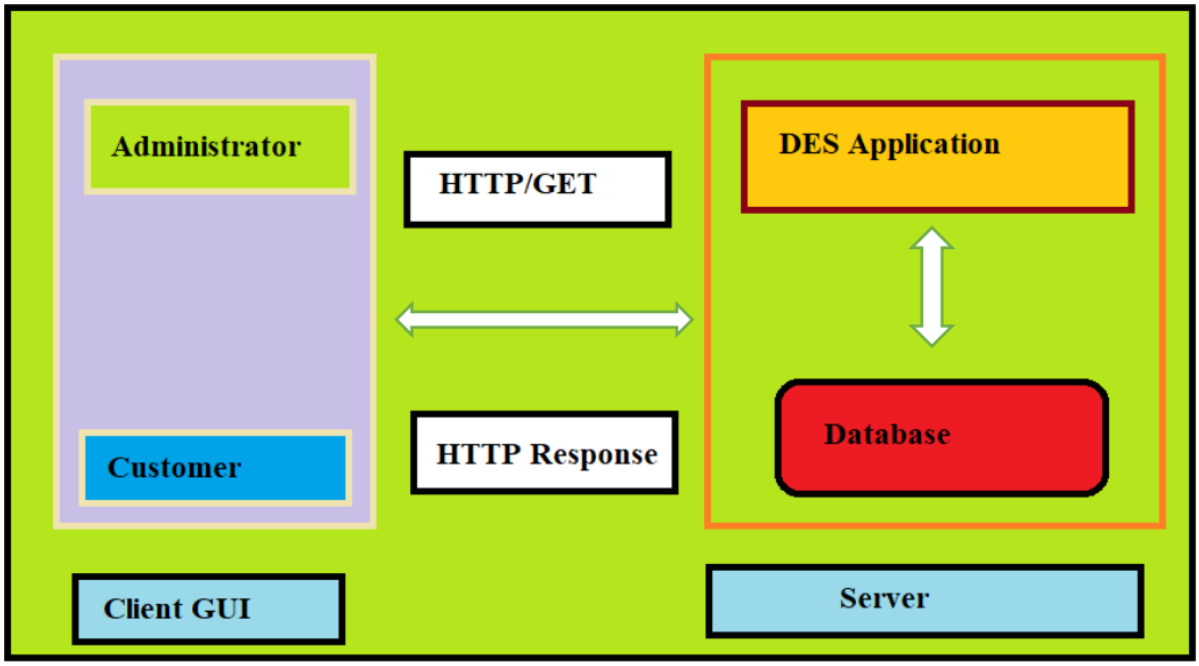
2.1.2 Authorization

The operations supported and allowed would be based on the user type. For example, Administrator has the rights to add policy information and view customer details. He can also view policy details and approve/reject policies

Whereas Customer/users has a right to view policy and apply for policies.

2.2 Functional Flow

The functional flow of the messages across different application components is shown below. Ex. - Web Application.



2.3 Environment:

The system will be developed on any Windows OS machine using J2EE, MySQL, Spring-boot, Angular.

- Intel hardware machine (PC P4-2.26 GHz, 512 MB RAM, 40 GB HDD)
- Server – Apache Tomcat 8 or higher
- Database – MySQL
- JRE 11
- Eclipse IDE or Spring Tool Suite

FUNCTIONAL SPECIFICATION

3. Sub-system Details

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

Find below (section 3.1 & 3.2) tables that provides functionality descriptions for each type of user / sub-system. Against each requirement, indicative data is listed in column 'Data to include'. Further, suggested to add/modify more details wherever required with an approval from customer/faculty.

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
01 To 04	Policy	Add View Delete Modify	Policy Id, Policy Name, Policy Price, Tenure	
	Category	Add View Delete Modify	Category name	
	User	View	User Id, User Name, Password, Type	
11 To 13	Approvals	View modify	User-id,Policy-id ,Policy price, quantity, Status	
	Queries	View modify	Query id, user-id, question, answer	

3.2 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	User Login	Register	User-Id, Username, Password, Email, Phone Number, Address.	
US-002	Policy	View Add	Policy Id, Policy Name, Price, Quantity, Status	
	Queries	View Add	Query id, user-id, question, answer	

FUNCTIONAL SPECIFICATION

3.3 Login | Logout

[Web Application - MySQL, Spring-boot, Angular]

- 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.
- Implement Session tracking for all logged in users before allowing access to application features. Anonymous users should be checked, unless explicitly mentioned.

4.Data Organization

This section explains the data storage requirements of the Inusrance Policy Management 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: User

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
Username	Username of the Customer used as a login id, Here user-name will be Primary Key
User-id	User-id is auto generated
Password	User Password
Email	Customer Email Id
Phone Number	10-digit contact number of users

4.2Table: Admin

The Admin specific details such admin-id, password. Authentication, and authorization / privileges should be kept in one or more tables, as necessary and applicable.

Field Name	Description
Admin-id	Admin id is default ADMIN , Here admin Id will be Primary Key
password	Admin password is default ADMIN

4.3Table:category

Policy categories

Field Name	Description
Category	Policy categories, Here this will be Primary Key

4.1 Table: Policies

This table contains information related to a product.

Field Name	Description
Policy Id	Unique policy Id, Here policy Id will be Primary Key
Policy Name	Name of the Policy e.g., Car insurance etc.
Category	Type of Policy

Amount	Price of the Policy
Tenure	Policy period

4.2 Table: Approvals

This table contains information related to policy requests sent by the customer.

Field Name	Description
User-name	User-name corresponding to logged in user, Here user-name will be Primary Key
Policy-id	Id of the policy
Request-id	Request id is auto generated
Date	Applied date
Status	Policy Availability Status, Example: approved/reject .

4.3 Table: Approved

This table contains information related to Approved requests approved by admin.

Field Name	Description
User-name	User-name corresponding to logged in user, Here user-name will be Primary Key
Policy-id	Id of the policy
Request-id	Request id is auto generated
Date	Approved date
Status	Policy Availability Status, Example: approved/reject .

4.4 Table: Disapproved

This table contains information related to Disapproved requests by admin.

Field Name	Description
User-name	User-name corresponding to logged in user, Here user-name will be Primary Key
Policy-id	Id of the policy
Request-id	Request id is auto generated
Date	Disapproved date
Status	Policy Availability Status, Example: approved/reject .

4.5 Table: queries

This table contains information related to policy requests sent by the customer.

Field Name	Description
User-name	User-name corresponding to logged in user, Here user-name will be Primary Key
Query-id	Query-id is auto generated
question	Question raised by the customer

answer	Answer given by the admin

FUNCTIONAL SPECIFICATION

a. REST APIs to be Built.

Create following REST resources which are required in the application,

i. Creating **User** Entity: Create Spring Boot with Microservices Application with Spring Data JPA

Technology stack:

1. Spring Boot
2. Spring REST
3. Spring Data JPA

Here will have multiple layers into the application:

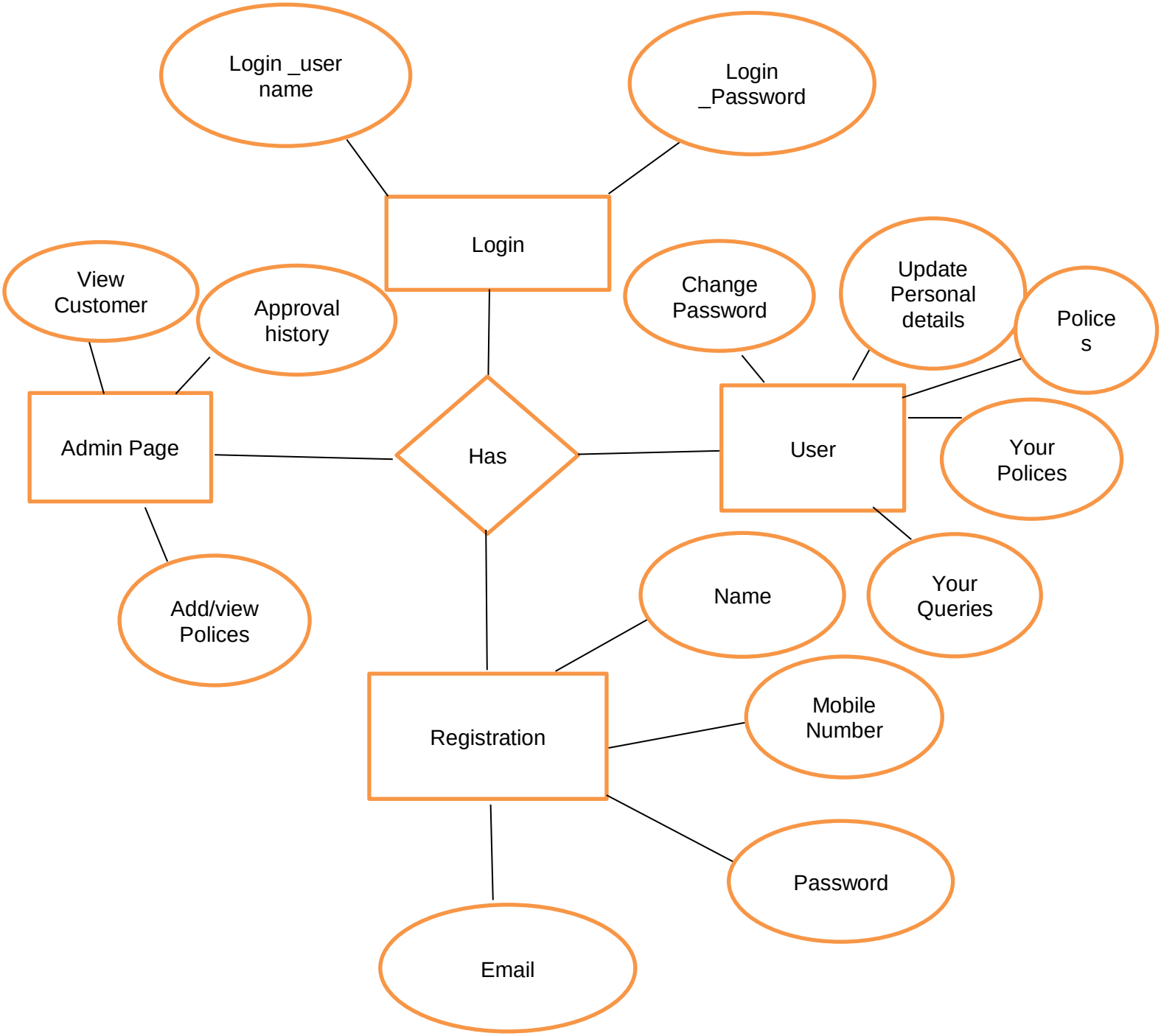
1. Create an Entity: Admin,User,Category,Policy,Approvals, Approved, Disapproved, Queries
2. Create a Repository interfaces and will make use of Spring Data JPA
   a. Will have find ByID method.
   b. Add the details
3. Create a Service classes and will expose all these services.
4. Finally, create a Rest Controller will have the following Urls:

URL	METHODS	Description	Format
/api/vi/admin	POST	Admin login access	string
/api/vi/approvals	GET	Get all the approval requests	JSON
/api/vi/approvals/{user name}	GET	Retrieving approval for a specific record	JSON
api/vi/approvals	POST	Adding approvals	string
api/vi/approved	GET	Retrieving all approved request	JSON
api/vi/approved/{userName}	GET	Retrieving all approved request for a specific record	JSON
api/vi/approved	POST	Adding approved	string
api/vi/categories	GET	Retrieving all categories	JSON
api/vi/categories/{category}	GET	Retrieving specific category	JSON
api/vi/category	POST	Adding category	string
api/vi/category/{category}	DELETE	Deleting the category	String
api/vi/disapproved	GET	Retrieving all disapproved requests	JSON
api/vi /disapproved/{userName}	GET	Retrieving specific disapproved requests	JSON
api/vi /disapproved	POST	Adding disapproved requests	string
api/vi /policies	GET	Retrieving all policies	JSON
api/vi /policy/{policyId}	GET	Retrieving specific policy	JSON
api/vi /policy	POST	Adding the policies	string
api/vi /policy/{policyId}	PUT	Updating the policy	string
api/vi /policy/policyId}	DELETE	Deleting the policy	string
api/vi /query	GET	Retrieving all the queries	JSON
api/vi /query/{userName}	GET	Retrieving specific queries	JSON
api/vi /query	POST	Adding queries	string
api/vi /query/{userName}	PUT	Updating the query	string
api/vi /users	GET	Retrieving all the users	JSON
api/vi /user/{userName}	GET	Retrieving specific users	JSON
api/vi /userdetails/{userName}	PUT	Updating the user detail	string
api/vi /userpassword/{userName}	PUT	Updating the passwords	string
api/vi /users	POST	User login validation	string

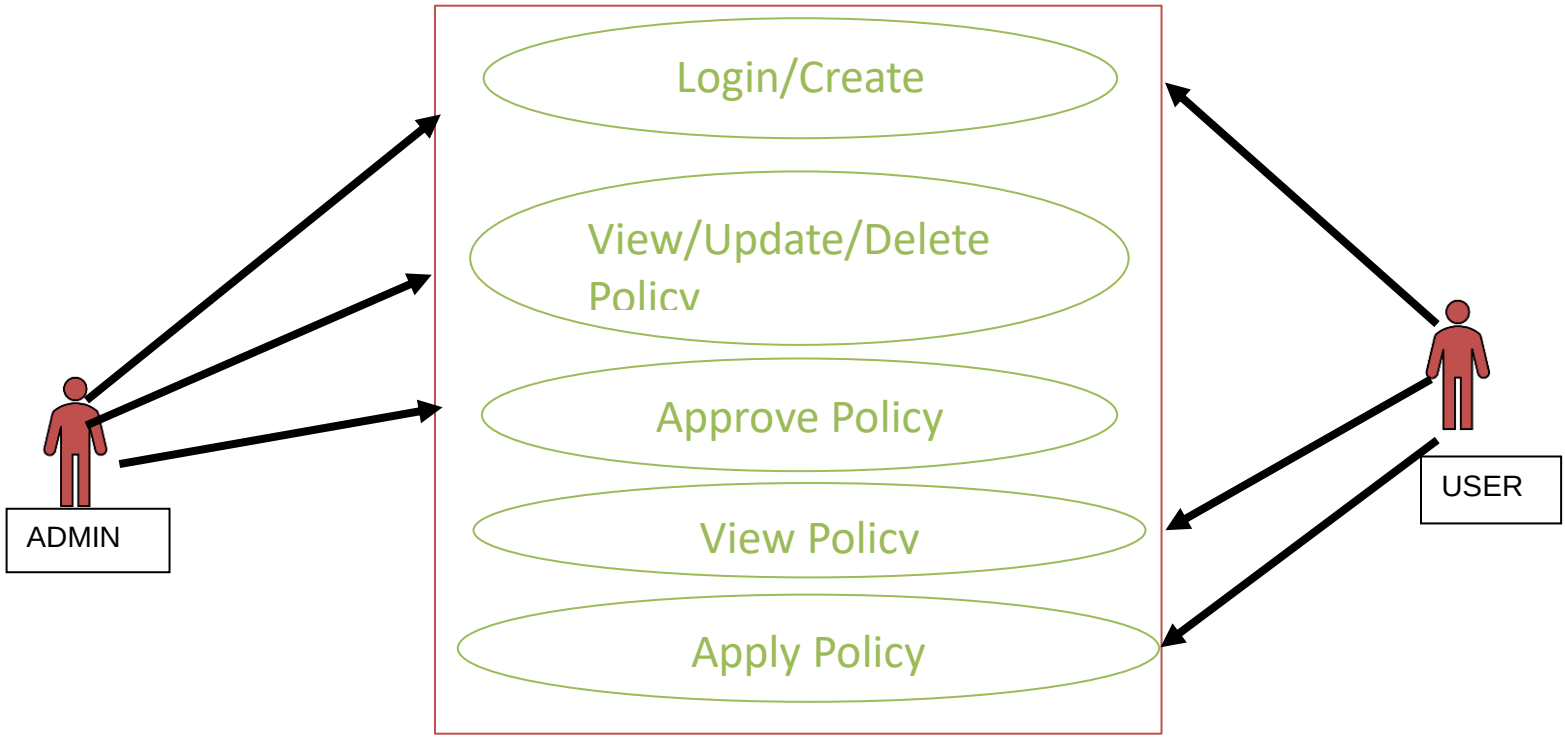
6. Assumptions

- User Interface: The type of client interface (front-end) to be supported - Angular based
- The administrator can add and remove policy into the database on a weekly basis.
- You must not allow user to add same policy twice.

.

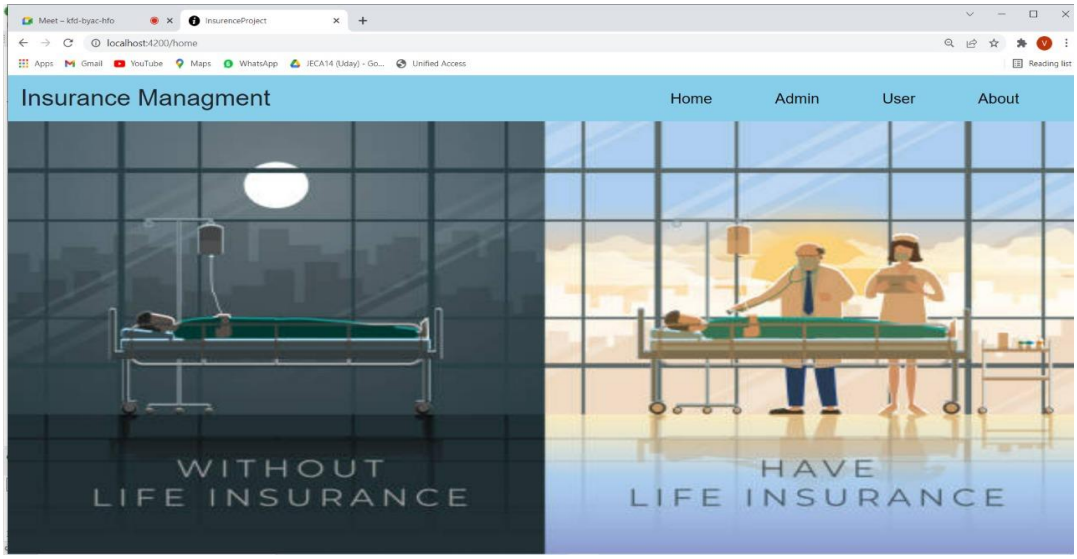

Login _user name
Login _Password
Login
View Customer
Approval history
Change Password
Update Personal details
Police s
Admin Page
Has
User
Your Polices
Add/view Polices
Name
Your Queries
Registration
Mobile Number
Password
Email

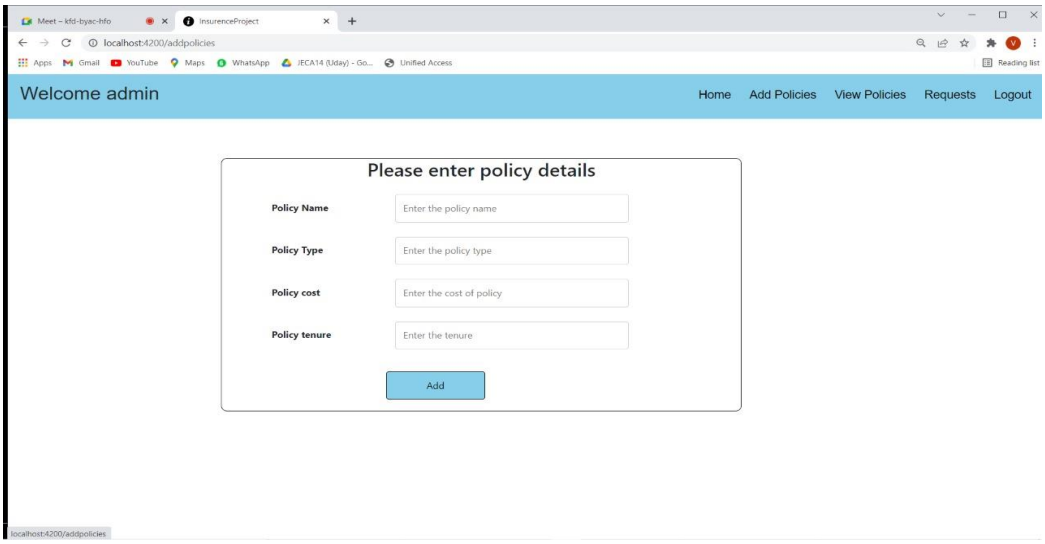
8. Use Case Diagram

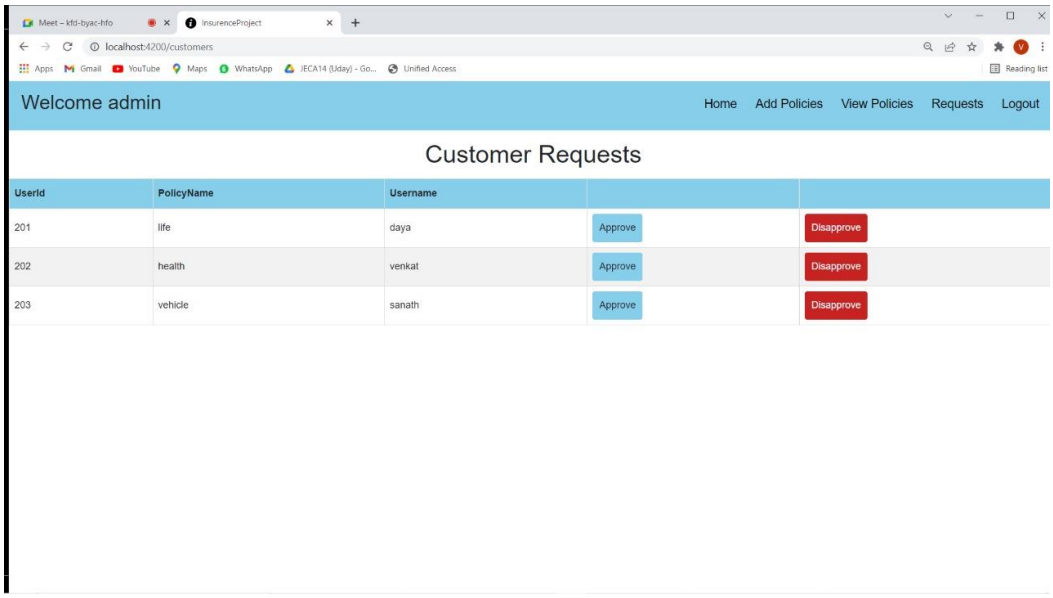


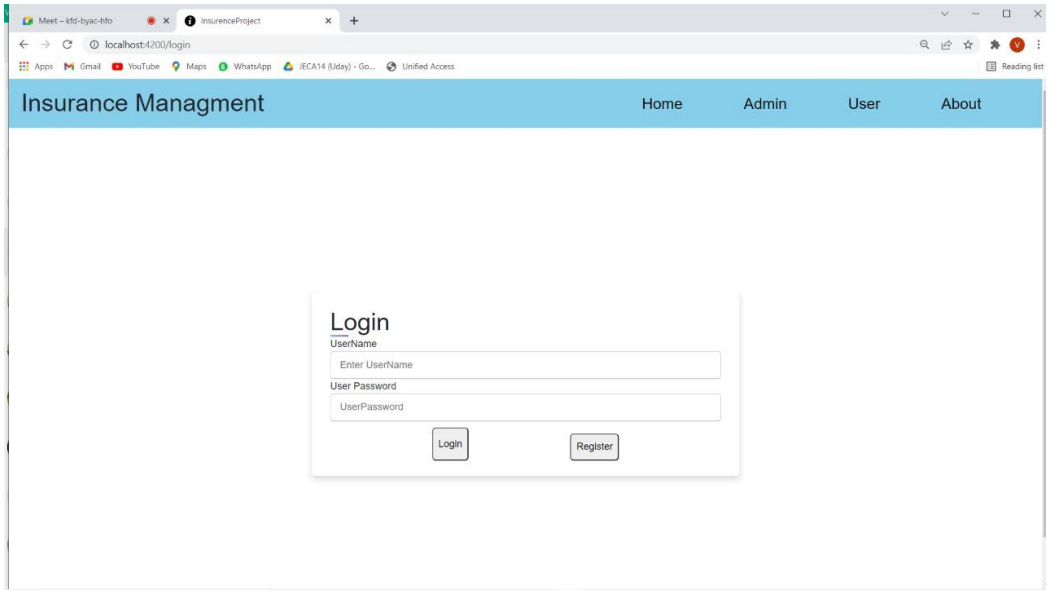
8 Output Screenshots for your Project

HOME page






Meet – kfd-byac-hfo
InsuranceProject
localhost:4200/customers
Apps
Gmail
YouTube
Maps
WhatsApp
JECA14 (Uday) - Go...
Unified Access
Reading list
Welcome admin
Home
Add Policies
View Policies
Requests
Logout
Customer Requests
UserId
PolicyName
Username
201
life
daya
Approve
Disapprove
202
health
venkat
Approve
Disapprove
203
vehicle
sanath
Approve
Disapprove


Meet – kfd-byac-hfo
InsuranceProject
localhost:4200/login
Apps
Gmail
YouTube
Maps
WhatsApp
JECA14 (Uday) - Go...
Unified Access
Reading list
Insurance Managment
Home
Admin
User
About
Login
UserName
Enter UserName
User Password
UserPassword
Login
Register