Vehicle Insurance System

Problem statement:

Imagine you are working for an insurance company that wants to modernize its operations by developing a comprehensive Insurance Management System for automobiles. The goal is to create a web-based application that allows users to manage policies, claims, and customer information efficiently.

Scope:

1. **List available policies**: Home page shows the available vehicle insurance policies and the description of company with user reviews. Shows statistical data of number of policies active and number of claims served.

2. **Proposal Submission:** Users can register and login in the application. Users can submit their vehicle insurance policy proposals for categories of vehicles like truck, motorcycle or camper van.

   Status of policy is proposal submitted.

3. **Proposal review:** Insurance team reviews the proposal form and approves based on guidelines.

   Proposal team can request the applicant for additional details and documents and approve or reject the policy. Status of policy is changed to quote generated.

4. **Sending Quotes:** Once a policy is submitted quote will be emailed with details of premium to be paid. Quote should be based on the policy package selected and add-ons selected based on defined business logic and appropriate calculations should be incorporated in the business logic of premium calculations.
5. **Premium payment:** Users can complete the payment based on quote. Once the payment is done, officers can change policy status to active .User gets a copy of policy document through email.
6. **Track policy status:** There are status like proposal submitted, quote generated, active, expired

7. **Premium reminder email:** Application sends out an email to the policy holder one week before policy is about to expire.

Technologies:

- Frontend: React.js / Angular Js.
- Backend: Java, Spring Boot/C#, .Net / Python Djnago for API development.
- Database: MySql / Sql Server.
- Authentication: JSON Web Tokens (JWT) for secure user authentication.

Reference Link: Tricentis Vehicle Insurance

Car Insurance - Buy Four Wheeler Policy Online @Best Price (bajajallianz.com)

Use case Diagram:

Actor: User

- Use Case: Register Account
- Use Case: Log In

- Use Case: **Proposal Submission**
    - Create new Proposal for policy of vehicle insurance.
    - View the status of proposal.
    - Complete the payment when quote is submitted.
- Use Case: Check status of policy and Download policy document
- Use Case: Log Out

Actor: Officer

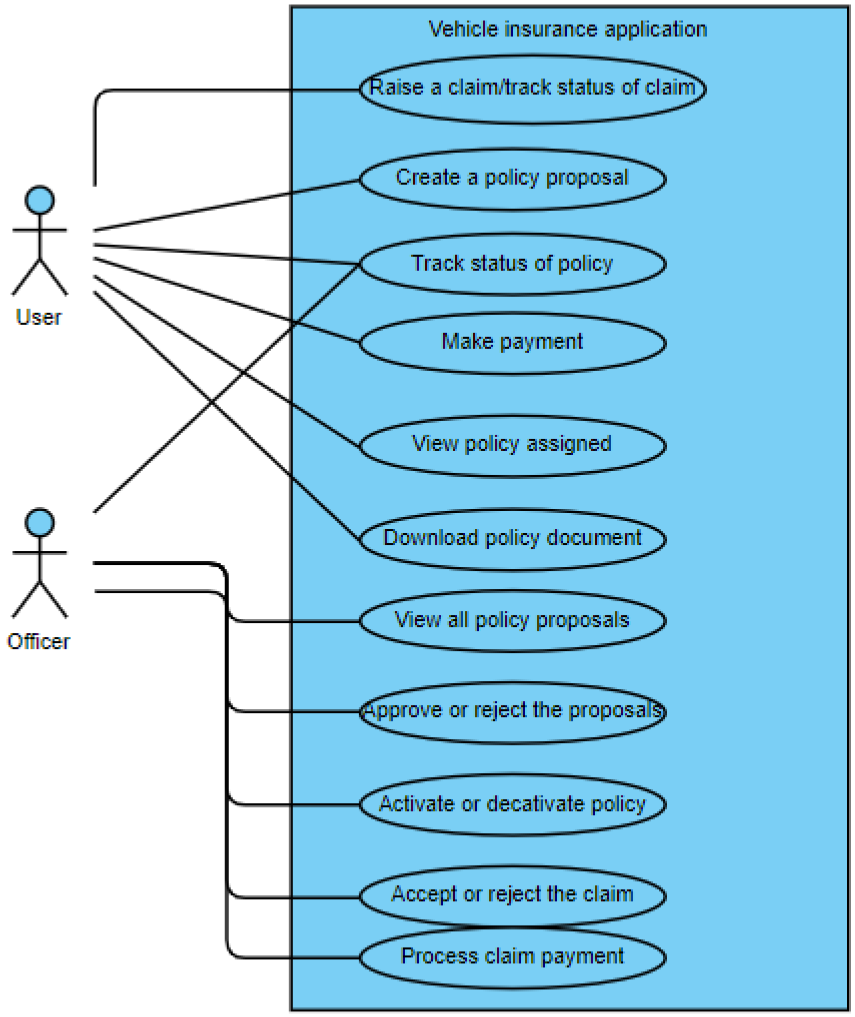
- Use Case: Log In
- Use Case: Manage proposal documents.
    - Request for additional information/document from user.
    - Approve the proposal form and make the policy status as active.
- Use Case: Log Out

System: Security and Authentication

- Use Case: Authenticate User

Associations:

- User creates proposal (Create incident, Check status).
- Officer can approve or reject the policy proposal
- Officer can verify the submitted documents and request for payment.
- Officer can approve the payment and change the status to verified
- Officer can approve or reject the proposal.
- Officer can add new policy with features and add ons


Vehicle insurance application
Raise a claim/track status of claim
Create a policy proposal
Track status of policy
Make payment
View policy assigned
Download policy document
View all policy proposals
Approve or reject the proposals
Activate or decativate policy
Accept or reject the claim
Process claim payment
User
Officer

Development Process:

1. **User Registration and Login:**
   - Users can create accounts, providing personal details (Name, Address, Date of birth, aadhaar number, age should be calculated from DOB, PAN number)
   - The system validates the information and creates user profiles.
   - Users log in using their credentials (username/email and password).

2. **User Dashboard and Policy proposal:**

   Users can log in to profile and access the proposal management section which has the following functionalities.
   - **Create new proposal for vehicle insurance policy** (car/bike/camper van)
     - Proposal form duly filled will be submitted to officers of insurance company for the review.

- Status of policy is to be displayed to the user and user can track the status through the stages.(like **proposal submitted, quote generated, active, expired**
- **Once the officer receives the proposal form, he can request fir additional data or documents needed if any. If the prososl seems in accordance with the prerequistes of policy, Officer can approve it and generate the quote based on the underlying business logic.**
- Each policy should be assigned a unique id and should be emailed to user upon successful payment as policy document which includes the details of policy. Downloadable pdf to be generated upon successful incident creation. Each incident will should be automatically assigned with a status **initiated** upon successful creation.

- **View all policies**: User should be able to track all the policies assigned to the him with highlighted status.(**proposal submitted, quote generated, active, expired**)

- **Premium payment:** Users can complete the payment based on a quote. Once the payment is done, officers can change policy status to active. User gets a copy of policy document through email.

3. Officer Dashboard and Policy Management:

Officer can approve or reject the policy proposal.

Officer can verify the submitted documents and request for payment.

Officer can approve the payment and change the status to verified

Officer can approve or reject the proposal.

Officer can add new policy with features and add ons

Functionalities

User Authentication and Authorization:

Design a secure login system for both customers and administrators.

Implement role-based access control to ensure that only authorized users can perform specific actions.

Customer Management:

Allow users to create and manage their profiles with personal information.

Implement a feature for customers to view and edit their contact details.

Policy Management:

Design a system to create, view, and edit insurance policies by the officer

Include features such as policy renewal and cancellation.

Claim Processing:

Develop a claim management system where users can file new claims.

Implement a workflow for claim processing, including validation, approval, and rejection.

Allow users to track the status of their claims.

Premium Calculation:

Create a module to calculate insurance premiums based on various factors (e.g., age of vehicle, coverage type, risk factors).

Implement a dynamic pricing model that can be adjusted based on real-time data.

Policy Documents and Communication:

Enable users to access and download policy documents.

Implement a communication system to send alerts and notifications to customers about policy updates, renewals, and claims.

Security and Compliance:

- User authentication and authorization are enforced to ensure data privacy.

  1. JWT Authentication:

  JWT authentication involves generating a token upon successful user login and sending it to the client. The client includes this token in subsequent requests to authenticate the user.

  - User Login: Upon successful login (using valid credentials), generate a JWT token on the server.
  - Token Payload: The token typically contains user-related information (e.g., user ID, roles, expiration time).
  - Token Signing: Sign the token using a secret key known only to the server. This ensures that the token hasn't been tampered with.
  - Token Transmission: Send the signed token back to the client as a response to the login request.

- Client Storage: Store the token securely on the client side (e.g., in browser storage or cookies).

2. JWT Authorization:

JWT authorization involves checking the token on protected routes to ensure that the user has the required permissions.

- Protected Routes: Define routes that require authentication and authorization.
- Token Verification:
    1. Extract the token from the request header.
    2. Verify the token's signature using the server's secret key.
- Payload Verification:
    1. Decode the token and extract user information.
    2. Check user roles or permissions to determine access rights.
- Access Control: Grant or deny access based on the user's roles and permissions.

Logout:

- Logging out involves invalidating the JWT token on both the client and the server to prevent further unauthorized requests.

Project Development Guidelines

The project to be developed based on the below design considerations.

1	**Backend Development**	• Use Rest APIs (Springboot/ASP.Net Core WebAPI to develop the services • Use Java/C# latest features. • Use ORM with database. • perform backend data validation. • Use Swagger to invoke APIs. • Implement API Versioning. • Implement security to allow/disallow CRUD operations. • Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant. • Any error message or exception should be logged and should be user-readable (not technical). • Database connections and web service URLs should be configurable. • Implement Unit Test Project for testing the API. • Implement JWT for Security. • Implement Logging.

		• Follow Coding Standards with proper project structure.

Frontend Constraints

1.	**Layout and Structure**	Create a clean and organized layout for your registration and login pages. You can use a responsive grid system (e.g., Bootstrap or Flexbox) to ensure your design looks good on various screen sizes.
2	**Visual Elements**	**Logo:** Place your application's logo at the top of the page to establish brand identity.
		Form Fields: Include input fields for email/username and password for both registration and login. For registration, include additional fields like name and possibly a password confirmation field.
		Buttons: Design attractive and easily distinguishable buttons for "Register," "Login," and "Forgot Password" (if applicable).
		Error Messages: Provide clear error messages for incorrect login attempts or registration errors.
		Background Image: Consider using a relevant background image to add visual appeal.
		Hover Effects: Change the appearance of buttons and links when users hover over them.
		Focus Styles: Apply focus styles to form fields when they are selected
3.	**Color Scheme and Typography**	Choose a color scheme that reflects your brand and creates a visually pleasing experience. Ensure good contrast between text and background colors for readability. Select a legible and consistent typography for headings and body text.
4.	**Registration Page/Add Bank Employee**	**Form Fields:** Include fields for users to enter their name, email, password, and any other relevant information. Use placeholders and labels to guide users.
		Validation: Implement real-time validation for fields (e.g., check email format) and provide immediate feedback for any errors. **Form Validation:** Implement client-side form validation to ensure required fields are filled out correctly before submission.
		Password Strength: Provide real-time feedback on password strength using indicators or text. **Password Requirements**: Clearly indicate password requirements (e.g., minimum length, special characters) to help users create strong passwords.
		Registration Success: Upon successful registration, redirect users to the login page.
5.	**Login Page: Customer/Bank Employee**	**Form Fields:** Provide fields for users to enter their email and password.
		Password Recovery: Include a "Forgot Password?" link that allows users to reset their password.
6.	**Common to React/Angular**	• Use Angular/React to develop the UI. • Implement Forms, data binding, validations, error message in required pages. • Implement Routing and navigations.

		• Use JavaScript to enhance functionalities. • Implement External and Custom JavaScript files. • Implement Typescript for Functions Operators. • Any error message or exception should be logged and should be user-readable (and not technical). • Follow coding standards. • Follow Standard project structure. • Design your pages to be responsive so they adapt well to different screen sizes, including mobile devices and tablets.

Good to have implementation features:

- Generate a SonarQube report and fix the required vulnerability.
- Use the Moq framework as applicable.
- Create a Docker image for the frontend and backend of the application .
- Implement OAuth Security.
- Implement design patterns.
- Deploy the docker image in AWS EC2 or Azure VM.
- Build the application using the AWS/Azure CI/CD pipeline. Trigger a CI/CD pipeline when code is checked-in to GIT. The check-in process should trigger unit tests with mocked dependencies.
- Use AWS RDS or Azure SQL DB to store the data.