



# CS 415 Project Release 2 Technical Document

Team Name	Student ID	First Name	Last Name	%Contribution	Signature
Group 4	S11133165	Sarvesh	Chand	100	S.C
	S11184309	Hinrich	Palaroan	100	H.P

## Contents

Introduction .....	1
Business Case .....	2
Executive Summary .....	2
Objectives .....	2
Projected Benefits .....	2
Business Rules .....	3
User Roles and Access Control .....	6
System Features .....	6
a. Policy Management: .....	6
Functional Requirements .....	6
Entity Relationship Diagram .....	7
API End Points .....	8
Workflow .....	8
User Interface .....	10
b. Premium Management: .....	10
Functional Requirements .....	10
Entity Relationship Diagram .....	11
API End Points .....	12
Workflow .....	13
User Interface .....	14
c. Customer Management: .....	14
Functional Requirements .....	14
Entity Relationship Diagram .....	15
API End Points .....	16
Workflow .....	17
User Interface .....	18
d. Reporting: .....	20
Functional Requirements .....	20
Workflow .....	20
User Interface .....	21
e. Payment Management (This section will be updated in Release 3) .....	22

Function Requirement.....	22
Entity Relationship Diagram.....	23
API End Points .....	24
Workflow .....	24
User Interface .....	24
f. Customer Requests Upgrade.....	24
Function Requirement.....	24
Entity Relationship Diagram.....	26
API End Points .....	27
Work Flow .....	28
User Interface .....	29
g. Claim Management .....	32
Function Requirement.....	32
Entity Relationship Diagram.....	34
API End Points .....	34
Work Flow .....	35
User Interface .....	36
System Functionalities .....	40
Premium Calculation.....	40
Claim Management Process .....	41
Handling Customer Request .....	41
Security.....	43
Data Encryption.....	43
Authentication .....	43
Authorization .....	43
Testing.....	44
Functional Testing .....	44
Deployment .....	46
System Requirements .....	46
Hardware Requirements .....	46
Software Requirements .....	46
Installation and Configuration .....	46
Training .....	47

Maintenance and Support .....	47
Change Management and Requests.....	48
Introduction .....	48
Change request process.....	48
Submitting a change request .....	49
Review and evaluation.....	49
Approval or rejection.....	50
Implementation.....	51
Communication .....	51
Change request log.....	52
Impact on project scope and timeline.....	52
Change request templates and tools. ....	54
Appendix.....	55
Conclusion .....	57
Reference .....	58

# Introduction

The Wheel Wise IMS has been designed to provide a comprehensive solution for managing and administering car insurance policies. The system is web-based, which means that it can be accessed from any location with an internet connection. The user-friendly interface makes it easy for users to navigate through the system and perform various tasks.

The policy management feature of the IMS allows insurers to create, modify, and terminate policies. This feature also allows insurers to view policy details, including coverage type, policy limits, and deductibles. The claim management feature allows insurers to manage claims efficiently by allowing users to file claims, view claims details, and track the progress of claims.

The premium management feature allows insurers to manage premiums and billing efficiently. This feature allows insurers to calculate premiums based on policy details, view payment history, and track payment status. The payment management feature allows insurers to receive and process payments from customers easily.

The customer management feature allows insurers to manage customer information, including contact details, policy information, and claims history. This feature also allows insurers to view customer interactions with the company, including phone calls, emails, and chat sessions.

The reporting capabilities of the IMS allow insurers to generate various reports, including policy reports, claim reports, premium reports, and customer reports. These reports provide valuable insights into the company's performance and help to identify areas for improvement.

The non-functional requirements of the system, such as performance, security, and compatibility, have been carefully considered during the design and development of the IMS. The system has been designed to ensure optimal performance, with minimal downtime and fast response times. The system is also designed to be secure, with robust encryption and access controls. The system is compatible with various browsers and devices, making it accessible to a wide range of users.

The Wheel Wise IMS is a comprehensive solution for managing and administering car insurance policies. The system offers numerous features and benefits, including policy management, claim management, premium management, payment management, customer management, and reporting capabilities. The system is designed to be user-friendly, efficient, and compliant with relevant regulations. This technical document provides a detailed overview of the system's functionalities, design, and benefits, serving as a valuable reference for stakeholders involved in the design, development, and implementation of the Wheel Wise IMS.

# Business Case

## Executive Summary

The Wheel Wise Insurance Management System is a comprehensive web-based solution designed to streamline the management and administration of car insurance policies, focusing on three main coverage types: basic liability coverage, comprehensive coverage, and collision coverage. The IMS aims to improve the efficiency and effectiveness of insurers, claims staff, and customers by automating various processes, providing a user-friendly interface, and ensuring regulatory compliance and a high level of customer satisfaction.

## Objectives

Simplify policy management for insurers by enabling them to create, view, update, and delete car insurance policies with different coverage types.

Enhance claim management by allowing insurers and claims staff to manage claims related to the three main coverage types.

Provide customers with an intuitive platform to view and manage their car insurance policies, file claims, and track claim status.

Improve premium management by offering insurers a dashboard to manage and update premium amounts for various car insurance policies.

Ensure seamless integration with payment gateways for processing premium payments and claim settlements.

Enhance customer management by allowing authorized users to create, modify, and search customer records.

Offer reporting capabilities to generate policy, claims, and customer reports in various formats, such as PDF.

## Projected Benefits

**Increased Efficiency:** By automating various processes and reducing manual tasks for insurers, claims staff, and customers, the IMS will improve efficiency and productivity.

**Enhanced Customer Satisfaction:** A user-friendly interface and streamlined processes will result in a seamless experience for customers managing their car insurance policies and claims, leading to higher satisfaction levels.

**Improved Decision Making:** Reporting capabilities will provide valuable insights into policy and claims data, enabling insurers and claims staff to make more informed decisions.

**Reduced Operational Costs:** The IMS will help insurers and claims staff manage car insurance policies and claims more effectively, leading to reduced operational costs and improved profitability.

**Regulatory Compliance:** Adhering to relevant regulations, such as IRDA and PCI DSS, the IMS will ensure that insurers remain compliant with industry standards, minimizing the risk of fines or penalties.

The Wheel Wise Insurance Management System offers a tailored and efficient solution for managing car insurance policies, claims, and customer interactions, ultimately resulting in a better overall experience for insurers and policyholders. By investing in the development and implementation of the IMS, insurers can expect to see significant improvements in operational efficiency, customer satisfaction, and regulatory compliance.

## Business Rules

Based on the provided Software Requirements Specification, here are the business rules for the Wheel Wise Insurance System:

### 1. Policy Management

- a) A new policy must have a policy type, policy name, coverage period, and coverage details
- b) Policies can be searched by policy name.
- c) Policy updates require the new policy details.
- d) Policies can be deleted by providing the policy number.
- e) A new policy can only be added by a authorized user (Administrator or Insurer).

### 2. Premium Management

- a. Premium details can be viewed by searching for the customer name or by the policy name.
- b. The premium amount can be updated by providing the new premium amount.

### 3. Customer Management

- a. Authorized users can create new customer records.
- b. Authorized users can modify existing customer records.
- c. Authorized users can search customer records based on various customer criteria such as customer name, email, driving record.

### 4. Reporting

- a. Authorized users can generate policy reports and customer reports.
- b. Reports must support PDF format.

# System Architecture

The overall system architecture for the Wheel Wise Insurance Management System (IMS) can be divided into three main layers: Presentation Layer, Business Logic Layer, and Data Access Layer. These layers interact with each other to provide the desired functionality and ensure a modular and maintainable design.

## 1. Presentation Layer:

This layer is responsible for providing the user interface (UI) through which users interact with the system. The UI is developed using React JS using the Material UI Library and is compatible with modern web browsers. It communicates with the Business Logic Layer through APIs to request and receive data, which is then displayed to the user.

## 2. Business Logic Layer:

The Business Logic Layer, also known as the Application Layer, contains the core logic and processing of the IMS. It is built using a server-side language such as ASP.NET. This layer is responsible for processing user requests received from the Presentation Layer and interacting with the Data Access Layer to retrieve or store data. It also ensures that the appropriate business rules, validations, and security measures are in place.

## 3. Data Access Layer:

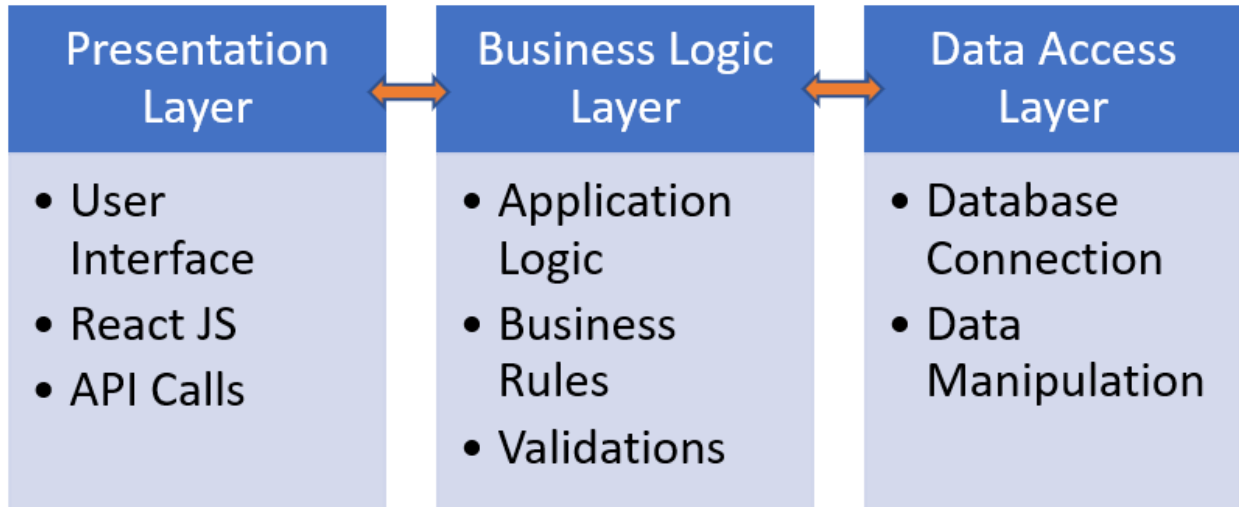
The Data Access Layer is responsible for managing the connection and interactions with the database. It handles data storage, retrieval, and manipulation using a database management system (DBMS) like Microsoft SQL Server. This layer abstracts the database operations from the Business Logic Layer, allowing for easy updates or changes to the database without affecting the application logic.

The relationships between these layers can be described as follows:

- The Presentation Layer sends requests for data or actions to the Business Logic Layer through APIs.
- The Business Logic Layer processes these requests, applying any necessary business rules, validations, or security measures, and then communicates with the Data Access Layer to retrieve or store the required data.
- The Data Access Layer interacts with the database to perform the requested operations and returns the data or results to the Business Logic Layer.
- The Business Logic Layer then sends the processed data or results back to the Presentation Layer, which displays the information to the user.

The diagram below illustrates the overall system architecture, including the components, layers, and their relationships:





This modular architecture ensures that each layer has a distinct responsibility, promoting maintainability, scalability, and flexibility within the Wheel Wise Insurance Management System.

Apart from the above layered architecture, the Wheel Wise Insurance Management System utilizes a combination of the following software architecture styles:

**Client-Server Architecture:** The IMS follows a client-server model, where the client (web browser) interacts with the server-side application through API calls. The server processes the requests, communicates with the database, and returns the results to the client.

**Service-Oriented Architecture (SOA):** The system exposes its functionalities through APIs, allowing different components of the system to communicate with each other using standardized protocols. This approach enables flexibility, reusability, and scalability of the system components.

While the Wheel Wise Insurance Management System primarily employs a combination of Layered, Client-Server, and Service-Oriented Architectures, it can be extended or adapted to incorporate other architecture styles, such as Microservices or Event-Driven, based on the specific requirements and scalability needs of the system.

In the Wheel Wise Insurance Management System, Aspect-Oriented Software Development (AOSD) is employed to enhance modularity and maintainability by effectively separating cross-cutting concerns from the core functionality. AOSD allows us to encapsulate these concerns, such as logging, security, and performance monitoring, into separate aspects that can be woven into the main system's codebase at specific points. This approach reduces code redundancy and complexity, resulting in a cleaner and more manageable system architecture. By utilizing AOSD, the IMS is better equipped to handle changes in requirements or functionality, enabling developers to focus on the core business logic without being hindered by the intricacies of cross-cutting concerns.

# User Roles and Access Control

The Wheel Wise Insurance Management System implements role-based access control to ensure that users can only access the functionalities relevant to their role. The system defines the following user roles:

1. Administrator: Responsible for managing the overall system, including user management, system configuration, and reporting.
2. Insurer: Manages car insurance policies, premiums, customers, and claims.
3. Claims Staff: Handles claim processing, including approving or rejecting claims.
4. Customer: Accesses and manages their car insurance policies, files claims, and tracks claim status.

## System Features

### a. Policy Management:

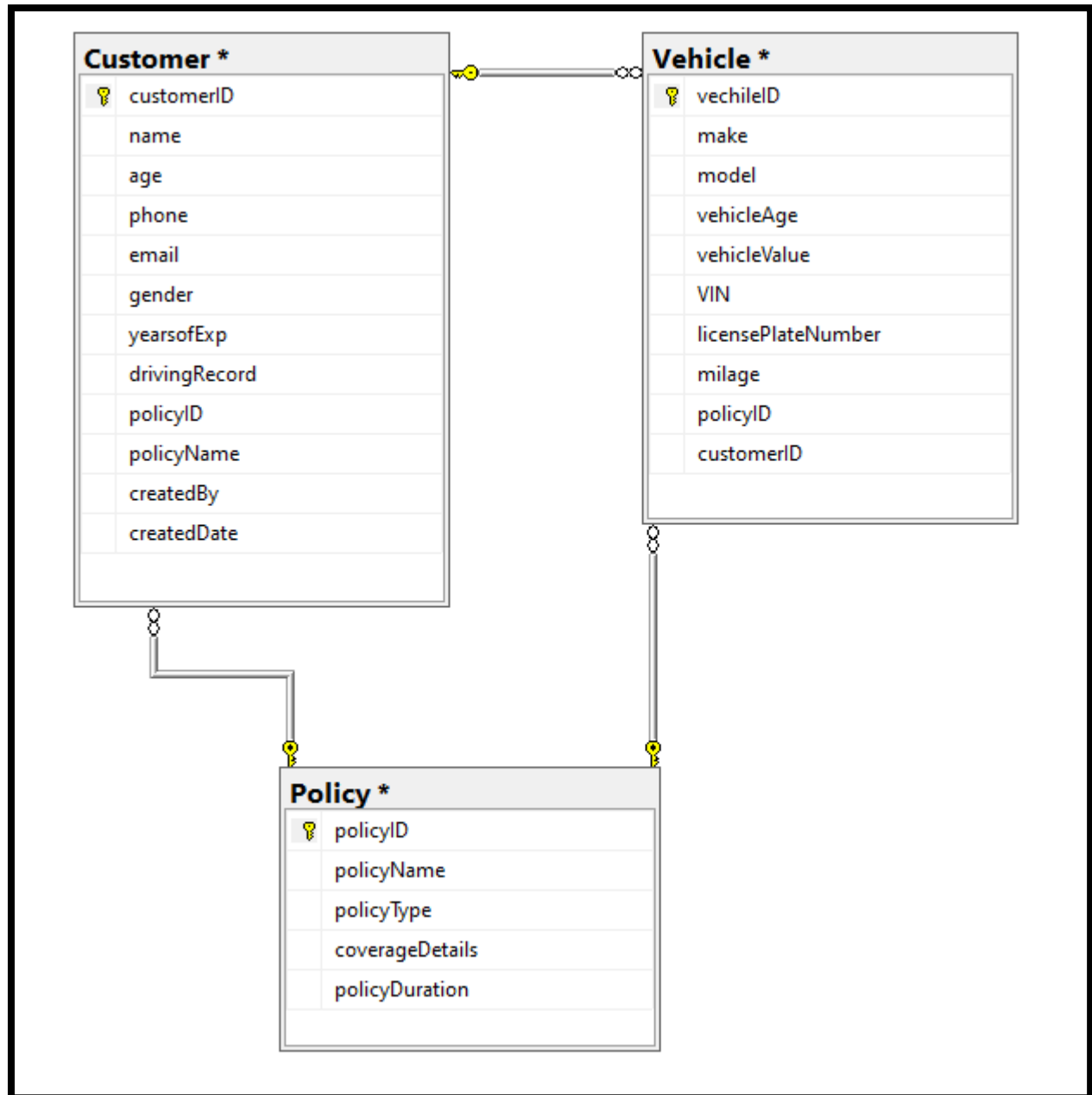
#### Functional Requirements

The functional requirements of the Policy Management feature in the Wheel Wise Insurance Management System are as follows:

1. Create Policy: The system should allow authorized users (e.g., insurers) to create new car insurance policies with different coverage types, such as basic liability, comprehensive, and collision coverage. Users should be able to input necessary policy details, including Customer information, vehicle details, coverage selections, policy start and end dates, and premium amounts.
2. View Policy: Authorized users should be able to view the details of existing car insurance policies, including policy number, Customer information, vehicle details, coverage information, policy duration, and premium amounts.
3. Update Policy: The system should enable authorized users to update the details of existing car insurance policies, such as modifying coverage types, adjusting policy durations, or updating Customer and vehicle information. Any changes made to the policy should be recorded in the system for tracking and auditing purposes.
4. Delete Policy: Authorized users should be able to delete existing car insurance policies when necessary, such as in cases of policy cancellation or erroneous policy creation. Deletion should be handled with appropriate safeguards to prevent accidental data loss.
5. Search and Filter Policies: The system should provide authorized users with the ability to search for specific policies using various criteria, such as policy number, Customer name, vehicle information, or coverage type. Additionally, users should be able to filter and sort the list of policies based on different attributes, such as policy creation date, coverage type, or policy status.

The data model for policies in the Wheel Wise Insurance Management System consists of several entities, their attributes, and the relationships between them. The main entities in this data model are Policy, Customer, and Vehicle.

### Entity Relationship Diagram



## API End Points

Below are the API endpoints related to policy management, including their expected input parameters, output, and error handling:

API Function	Method	API Call	Input Parameters	Output	Error Handling
1. Get all policies	GET	http://localhost:5179/api/Policy	None	A list of policy objects	If no policies are found, return an empty list
2. Create a new policy	POST	http://localhost:5179/api/Policy	Policy object (JSON format)	The created policy object with a new ID	If input data is invalid or incomplete, return a 400 Bad Request status with an error message
3. Get a specific policy by ID	GET	http://localhost:5179/api/Policy/{id}	Policy ID (integer)	The policy object with the specified ID	If the policy ID is not found, return a 404 Not Found status with an error message
4. Update a policy by ID	PUT	http://localhost:5179/api/Policy/{id}	Policy ID (integer), updated policy object (JSON format)	The updated policy object	If the policy ID is not found, return a 404 Not Found status with an error message. If input data is invalid or incomplete, return a 400 Bad Request status with an error message.
5. Delete a policy by ID	DELETE	http://localhost:5179/api/Policy/{id}	Policy ID (integer)	A confirmation message indicating successful deletion	If the policy ID is not found, return a 404 Not Found status with an error message

## Workflow

The typical workflow for policy management in the Wheel Wise Insurance Management System involves several user roles, each with specific permissions, to ensure a secure and efficient process. The main user roles include Insurers, Claims Staff, and Customer.

### 1. Insurer:

Permissions:

- Create, view, update, and delete policies.
- Manage policy details, including coverage types, policy duration, and premium amounts.
- Search and filter policies.

Workflow:

- a. The insurer logs in to the system with their credentials.

b. The insurer creates a new policy by entering customer information, vehicle details, coverage selections, policy duration, and premium amounts.

c. The insurer can view existing policies, modify policy details, or delete policies as needed.

### 2. Claims Staff:

Permissions:

- View policy details.
- Search and filter policies.

Workflow:

- a. The claims staff logs in to the system with their credentials.
- b. The claims staff can view policy details when processing claims to verify coverage types, policy duration, and other relevant information.
- c. The claims staff can search for specific policies using various criteria and filter or sort the list of policies based on different attributes.

### 3. Customer:

Permissions:

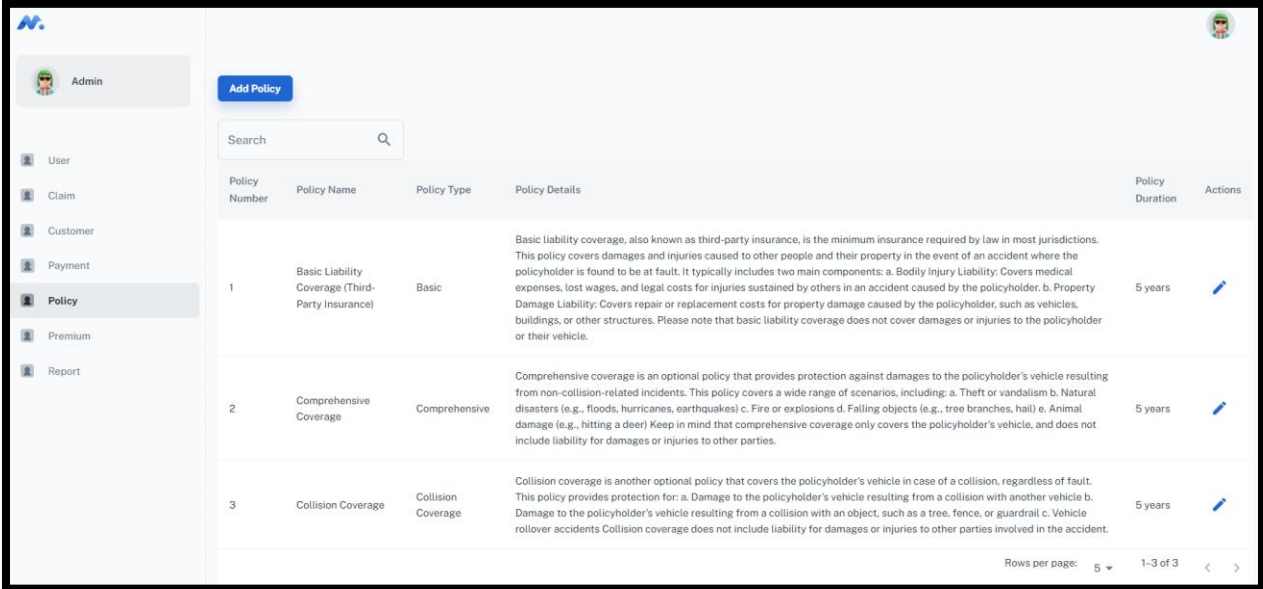
- View their own policies.
- File claims.
- Track claim status.

Workflow:

- a. The customer logs in to the system with their credentials.
- b. The customer can view the details of their own car insurance policies, including policy number, coverage information, policy duration, and premium amounts.
- c. The customer can file claims related to their policies and track the status of their claims.

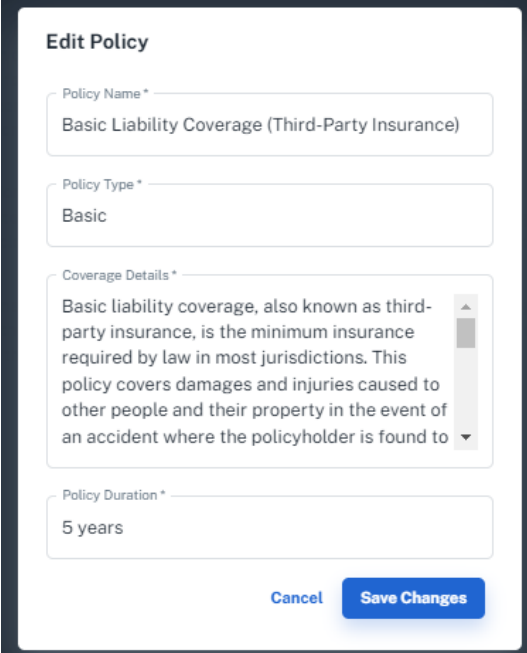
This typical workflow for policy management ensures that each user role has the appropriate permissions to perform their specific tasks, providing a secure and streamlined process for managing car insurance policies in the Wheel Wise Insurance Management System.

## User Interface



The screenshot shows the 'Policy' management interface. On the left is a sidebar with navigation links: Admin, User, Claim, Customer, Payment, Policy (selected), Premium, and Report. The main area features a table of policies with columns: Policy Number, Policy Name, Policy Type, Policy Details, Policy Duration, and Actions. There are three policies listed: Basic Liability Coverage (Third-Party Insurance), Comprehensive Coverage, and Collision Coverage. Each policy has a duration of 5 years and an edit icon. A search bar and an 'Add Policy' button are at the top. The bottom right shows 'Rows per page: 5' and '1-3 of 3'.

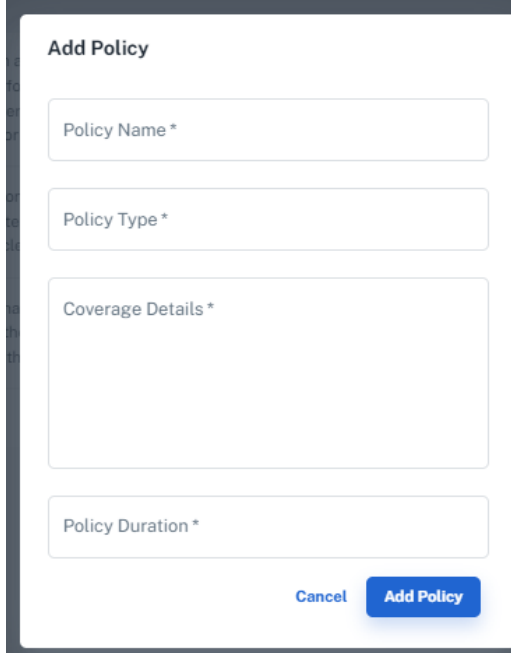
Policy Number	Policy Name	Policy Type	Policy Details	Policy Duration	Actions
1	Basic Liability Coverage (Third-Party Insurance)	Basic	Basic liability coverage, also known as third-party insurance, is the minimum insurance required by law in most jurisdictions. This policy covers damages and injuries caused to other people and their property in the event of an accident where the policyholder is found to be at fault. It typically includes two main components: a. Bodily Injury Liability; Covers medical expenses, lost wages, and legal costs for injuries sustained by others in an accident caused by the policyholder. b. Property Damage Liability; Covers repair or replacement costs for property damage caused by the policyholder, such as vehicles, buildings, or other structures. Please note that basic liability coverage does not cover damages or injuries to the policyholder or their vehicle.	5 years	
2	Comprehensive Coverage	Comprehensive	Comprehensive coverage is an optional policy that provides protection against damages to the policyholder's vehicle resulting from non-collision-related incidents. This policy covers a wide range of scenarios, including: a. Theft or vandalism b. Natural disasters (e.g., floods, hurricanes, earthquakes) c. Fire or explosions d. Falling objects (e.g., tree branches, hail) e. Animal damage (e.g., hitting a deer) Keep in mind that comprehensive coverage only covers the policyholder's vehicle, and does not include liability for damages or injuries to other parties.	5 years	
3	Collision Coverage	Collision Coverage	Collision coverage is another optional policy that covers the policyholder's vehicle in case of a collision, regardless of fault. This policy provides protection for: a. Damage to the policyholder's vehicle resulting from a collision with another vehicle b. Damage to the policyholder's vehicle resulting from a collision with an object, such as a tree, fence, or guardrail c. Vehicle rollover accidents Collision coverage does not include liability for damages or injuries to other parties involved in the accident.	5 years	



The 'Edit Policy' form contains the following fields:

- Policy Name \***: Basic Liability Coverage (Third-Party Insurance)
- Policy Type \***: Basic
- Coverage Details \***: Basic liability coverage, also known as third-party insurance, is the minimum insurance required by law in most jurisdictions. This policy covers damages and injuries caused to other people and their property in the event of an accident where the policyholder is found to
- Policy Duration \***: 5 years

Buttons: Cancel, Save Changes



The 'Add Policy' form contains the following fields:

- Policy Name \***
- Policy Type \***
- Coverage Details \***
- Policy Duration \***

Buttons: Cancel, Add Policy

## b. Premium Management:

### Functional Requirements

The functional requirements of the Premium Management feature in the Wheel Wise Insurance Management System are as follows:

**Calculate Premiums:** The system should automatically calculate insurance premiums for various car insurance policies based on factors such as vehicle information and customers driving history.

**Update Premium Amounts:** Authorized users (e.g., insurers) should be able to view and update premium amounts for various car insurance policies as needed, such as when customer risk factors change.

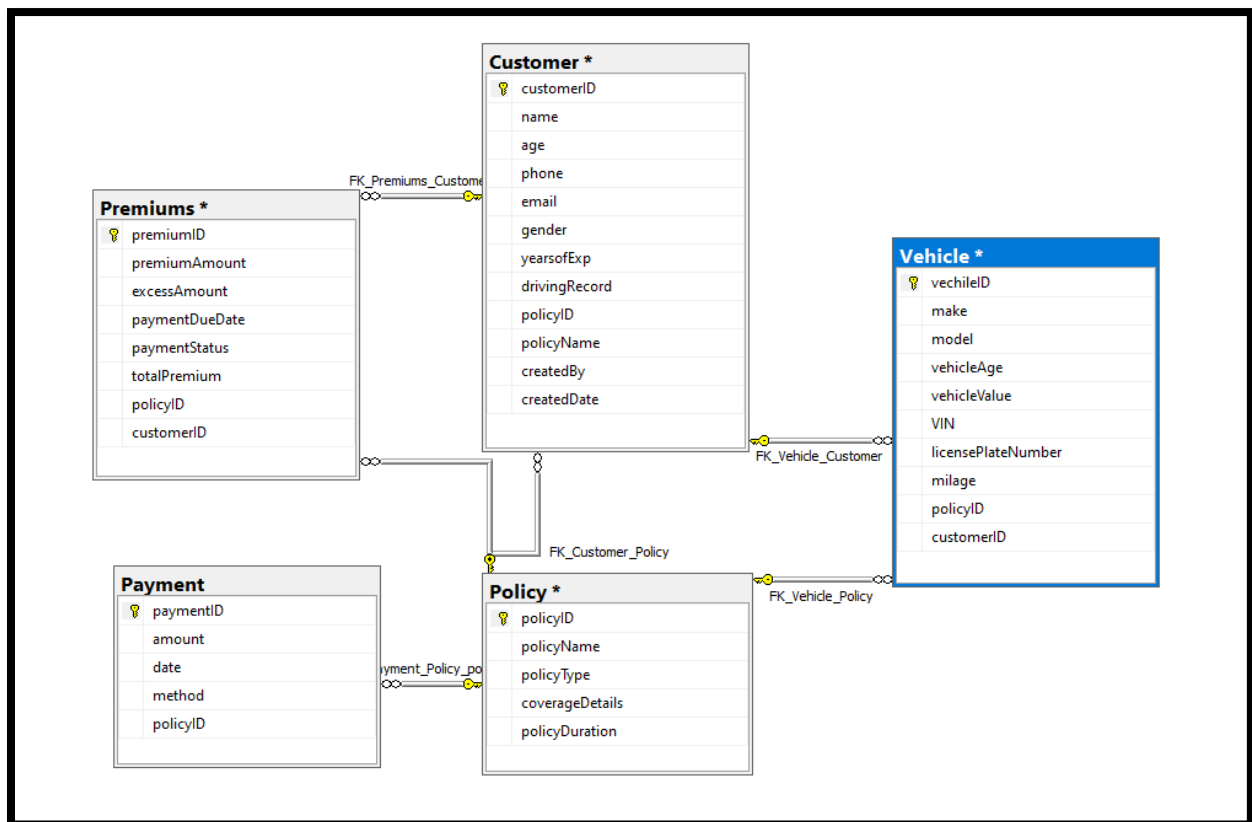
**Process Premium Payments:** The system should facilitate the processing of premium payments by integrating with payment gateways to support various payment methods, such as credit cards, debit cards, and electronic funds transfers (EFT).

**Manage Payment Schedules:** Authorized users should be able to manage payment schedules for customers, including setting up recurring payment plans and sending payment reminders or notifications.

**Record Payment Transactions:** The system should record all premium payment transactions, including payment dates, amounts, methods, and confirmation numbers, to maintain an accurate payment history for each customer.

**Handle Payment Failures:** The system should provide mechanisms to handle payment failures, such as sending payment failure notifications to customers, retrying failed payments, or applying late payment fees as per the policy terms.

### Entity Relationship Diagram



## API End Points

Below are the API endpoints related to premium management, including their expected input parameters, output, and error handling:

API Function	Method	API Call	Input Parameters	Output	Error Handling
1. Get all premiums	GET	http://localhost:5179/api/Premiums	None	A list of premium objects	If no premiums are found, return an empty list
2. Create a new premium	POST	http://localhost:5179/api/Premiums	Premium object (JSON format)	The created premium object with a new ID	If input data is invalid or incomplete, return a 400 Bad Request status with an error message
3. Get a specific premium by ID	GET	http://localhost:5179/api/Premiums/{id}	Premium ID (integer)	The premium object with the specified ID	If the premium ID is not found, return a 404 Not Found status with an error message
4. Update a premium by ID	PUT	http://localhost:5179/api/Premiums/{id}	Premium ID (integer), updated premium object (JSON format)	The updated premium object	If the premium ID is not found, return a 404 Not Found status with an error message. If input data is invalid or incomplete, return a 400 Bad Request status with



					an error message.
5. Delete a premium by ID	DELETE	http://localhost:5179/api/Premiums/{id}	Premium ID (integer)	A confirmation message indicating successful deletion	If the premium ID is not found, return a 404 Not Found status with an error message

## Workflow

The typical workflow for premium management in the Wheel Wise Insurance Management System involves several user roles, each with specific permissions, to ensure a secure and efficient process. The main user roles include Insurers (or Insurance Agents) and customers (Customers).

Insurer:

Permissions:

Calculate and update premium amounts for various car insurance policies.

Manage payment schedules for customers.

Generate premium and payment reports.

Workflow:

- a. The insurer logs in to the system with their credentials.
- b. The system calculates, or insurer updates premium amounts for car insurance policies based on coverage type, limits, deductibles, vehicle information, and the customer's driving history.
- c. The system sets up and manages payment schedules for customers on an annual basis and sends payment reminders or notifications.

Customer:

Permissions:

View premium details and payment schedules.

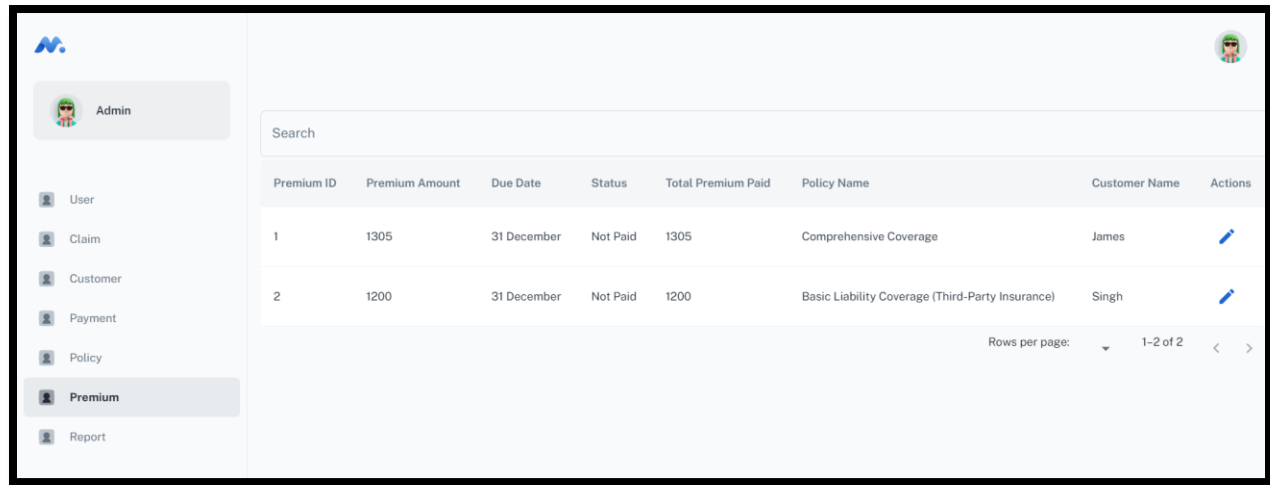
Make premium payments.

Workflow:

- a. The Customer logs in to the system with their credentials.
- b. The Customer views their premium details and payment schedules for their car insurance policy.
- c. The Customer makes premium payments using the integrated payment gateway, which supports various payment methods.

The system records the payment transaction details, updates the next payment due date, and sends a payment confirmation to the customer.

## User Interface



The 'Edit Premium' form is titled 'Edit Premium' and includes the instruction 'Update the premium details below.' It contains five input fields: 'Premium Amount' (1305), 'Due Date' (31 December), 'Status' (Not Paid), and 'Total Premium Paid' (1305). At the bottom right of the form are 'Cancel' and 'Save' buttons.

### c. Customer Management:

#### Functional Requirements

The functional requirements of the Customer Management feature in the Wheel Wise Insurance Management System are as follows:

**Create Customer Profile:** The system should allow authorized users (e.g., insurers) to create new customer profiles with essential customer details, including name, date of birth, address, phone number, and email address.

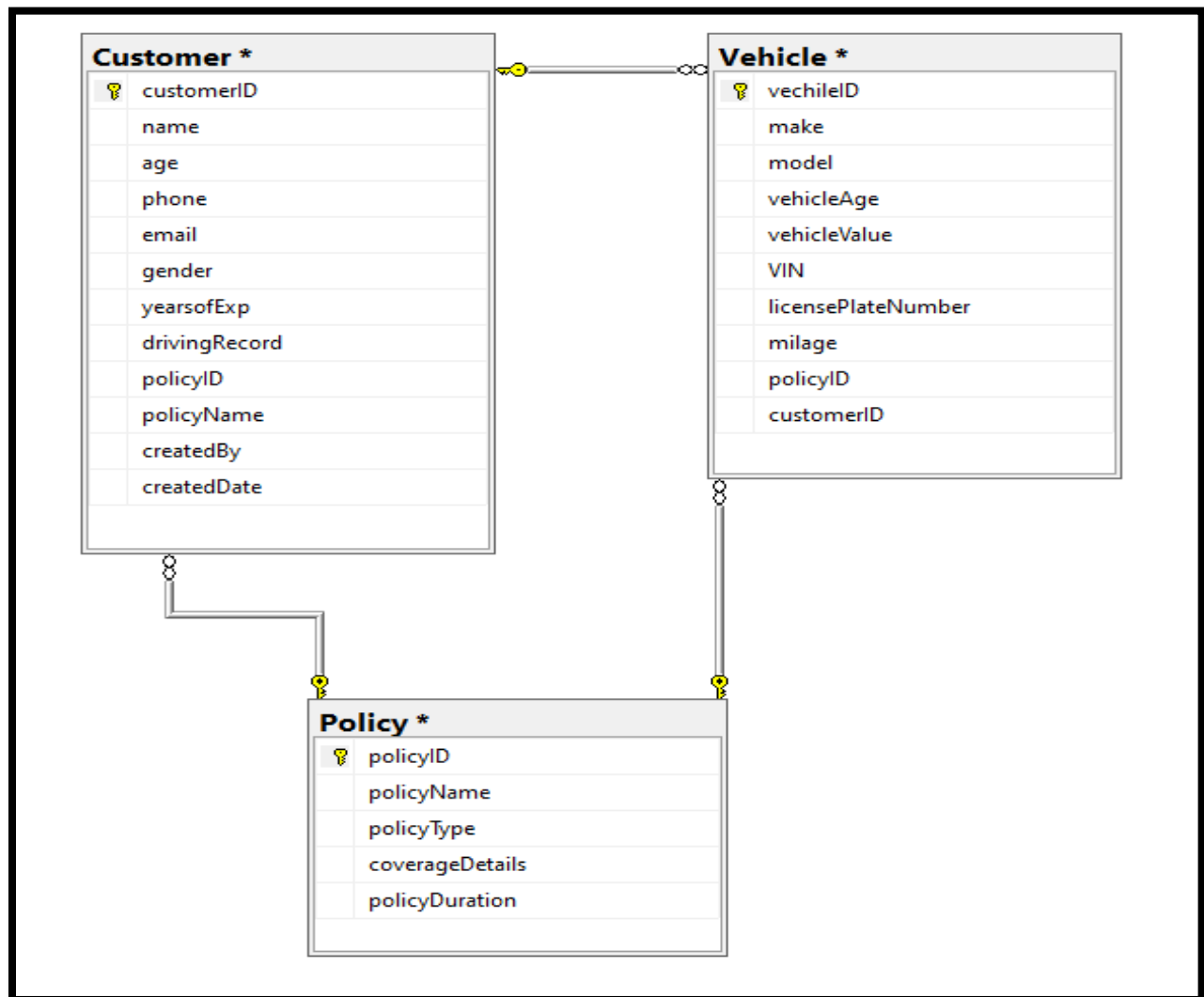
**View Customer Profile:** Authorized users should be able to view the details of existing customer profiles, including customer identification numbers, names, dates of birth, addresses, phone numbers, and email addresses.

**Update Customer Profile:** The system should enable authorized users to update the details of existing customer profiles, such as modifying personal information, updating contact details, or adding secondary contact information. Any changes made to the customer profile should be recorded in the system for tracking and auditing purposes.

**Delete Customer Profile:** Authorized users should be able to delete existing customer profiles when necessary, such as in cases of duplicate profiles or erroneous profile creation. Deletion should be handled with appropriate safeguards to prevent accidental data loss.

**Search and Filter Customers:** The system should provide authorized users with the ability to search for specific customers using various criteria, such as name, customer identification number, phone number, or email address. Additionally, users should be able to filter and sort the list of customers based on different attributes, such as account creation date or the number of active policies.

### Entity Relationship Diagram



## API End Points

Below are the API endpoints related to customer management, including their expected input parameters, output, and error handling:

API Function	Method	API Call	Input Parameters	Output	Error Handling
1. Get all customers	GET	http://localhost:5179/api/Customer	None	A list of customer objects	If no customers are found, return an empty list
2. Create a new customer	POST	http://localhost:5179/api/Customer	Customer object (JSON format)	The created customer object with a new ID	If input data is invalid or incomplete, return a 400 Bad Request status with an error message
3. Get a specific customer by ID	GET	http://localhost:5179/api/Customer/{id}	Customer ID (integer)	The customer object with the specified ID	If the customer ID is not found, return a 404 Not Found status with an error message
4. Update a customer by ID	PUT	http://localhost:5179/api/Customer/{id}	Customer ID (integer), updated customer object (JSON format)	The updated customer object	If the customer ID is not found, return a 404 Not Found status with an error message. If input data is invalid or incomplete, return a

					400 Bad Request status with an error message.
5. Delete a customer by ID	DELETE	http://localhost:5179/api/Customer/{id}	Customer ID (integer)	A confirmation message indicating successful deletion	If the customer ID is not found, return a 404 Not Found status with an error message

## Workflow

The typical workflow for customer management in the Wheel Wise Insurance Management System involves several user roles, each with specific permissions, to ensure a secure and efficient process. The main user roles include Insurers, Claims Staff, and Customers.

### 1. Insurer:

#### Permissions:

- Create, view, update, and delete customer profiles.
- Manage customer details, including personal information and contact details.
- Search and filter customer profiles.

#### Workflow:

- a. The insurer logs in to the system with their credentials.
- b. The insurer creates a new customer profile by entering essential customer details.
- c. The insurer can view existing customer profiles, modify customer details, or delete customer profiles as needed.
- d. The insurer can search for specific customers using various criteria and filter or sort the list of customers based on different attributes.

### 2. Claims Staff:

#### Permissions:

- View customer profiles.
- Search and filter customer profiles.

#### Workflow:

- a. The claims staff logs in to the system with their credentials.
- b. The claims staff can view customer profiles when processing claims to verify customer information and contact details.
- c. The claims staff can search for specific customers using various criteria and filter or sort the list of customers based on different attributes.

### 3. Customer:

Permissions:

- View and update their own customer profile.

Workflow:

- The customer logs in to the system with their credentials.
- The customer can view their own customer profile, including personal information and contact details.

## User Interface

Admin

User

Claim

Customer

Payment

Policy

Premium

Report

Add Customer

Search by Name or Email

Customer ID	Name	Age	Gender	Email	Phone	Year of Experience	Driving Record	Policy	Actions
1	James	28	male	james@gmail.com	9115687	3	clean	2	<div><div></div><div></div></div>
2	Singh	25	male	sarvesh.chand12@gmail.com	9149658	2	clean	1	<div><div></div><div></div></div>
3	Singh	25	male	sarvesh.chand12@gmail.com	9149658	2	clean	1	<div><div></div><div></div></div>
4	string	string	string	sarvesh.chand12@gmail.com	string	0	string	1	<div><div></div><div></div></div>
5	Singh	25	male	sarvesh.chand12@gmail.com	9149658	2	clean	1	<div><div></div><div></div></div>

Rows per page: 51-5 of 5

Vehicle Table

Vehicle ID	Make	Model	Vehicle Age	Vehicle Value	VIN	License Number	Milage	Policy Name	Customer Name
1	Nissan	G32	2	45000		ABC123	14998	Comprehensive Coverage	James
2	Test	string	0	string		string	0		

Rows per page: 10Null Null of 2

**Add Customer**

Name \*

Email \*

Phone \*

Age \*

Gender \*

Driving Experience \*

Driving Record \*

clean

Make \*

Model \*

Vehicle Age \*

Vehicle VIN \*

Gender \*

Driving Experience \*

Driving Record \*

clean

Make \*

Model \*

Vehicle Age \*

Vehicle VIN \*

License Plate Number \*

Milage \*

Vehicle Value \*

Premium Amount: \$

Cancel Add Customer

**Edit Customer**

Selected Policy :2

Name

James

Age

28

Gender

Email

james@gmail.com

Phone

9115687

Year of Experience

3

Driving Record

clean

Cancel Update

### d. Reporting:

#### Functional Requirements

The functional requirements of the Reporting feature in the Wheel Wise Insurance Management System are as follows:

1. **Generate Reports:** The system should allow authorized users to generate various reports related to policies, claims, and customers, providing valuable insights into business operations and performance.
2. **Report Types:** The system should support multiple types of reports, including:
  - **Policy Reports:** Reports on the types policies and how much customers and premiums each policy has.
  - **Claims Reports:** Reports on open claims, closed claims, approved claims, rejected claims, and claim processing time.
  - **Customer Reports:** Reports on new customers, customer demographics, and customer policy distribution.
- 3.. **Export Report Data:** The system should allow authorized users to export report data in PDF format.

The reporting feature utilizes the API's of other features such as policy and customer to generate the reports.

#### Workflow

The typical workflow for generating and viewing reports in the Wheel Wise Insurance Management System involves several user roles, each with specific permissions, to ensure a secure and efficient process. The main user roles include Admins, Insurers and Claims Staff.

##### 1. Admins:

Permissions:

- Generate policy and customer reports.
- View and export generated reports in PDF format.

Workflow:

- a. The admins logs in to the system with their credentials.
- b. The admin selects the desired report type (policy or customer) and specifies any required filters or parameters.
- c. The system generates the report based on the provided parameters and displays it to the admin.
- d. The insurer can view the report and export it in PDF format for further analysis or sharing with external parties.

##### 1. Insurer:

Permissions:

- Generate policy and customer reports.
- View and export generated reports in PDF format.

Workflow:

- a. The insurer logs in to the system with their credentials.



- b. The insurer selects the desired report type (policy or customer) and specifies any required filters or parameters.
- c. The system generates the report based on the provided parameters and displays it to the insurer.
- d. The insurer can view the report and export it in PDF format for further analysis or sharing with external parties.

### 2. Claims Staff:

#### Permissions:

- Generate claims reports.
- View and export generated reports in PDF format.

#### Workflow:

- a. The claims staff logs in to the system with their credentials.
- b. The claims staff selects the claims report type and specifies any required filters or parameters.
- c. The system generates the report based on the provided parameters and displays it to the claims staff.
- d. The claims staff can view the report and export it in PDF format for further analysis or sharing with external parties.

## User Interface



### e. Payment Management (**This section will be updated in Release 3**)

#### Function Requirement

The functional requirements of the Payment feature in the Wheel Wise Insurance Management System are as follows:

##### A Payment Receipt:

The system should integrate with one or more payment gateways to facilitate the receiving of payments from customers. This feature will allow customers to make payments for their policy premiums and any other charges directly from their dashboard.

Upon successful payment, the system should automatically update the status of the corresponding policy or claim. For policy premiums, the status may be updated to "Paid," and for claims, the status could be updated to "Settled".

The system should also generate a digital receipt for each payment, which can be sent to the customer via email or can be made available for download from the customer's dashboard.

##### B. Payment Refunds:

In situations where a refund is necessary, such as when a policy is cancelled or a claim is rejected, the system should have the functionality to facilitate the refund process.

When a refund is initiated, the system should automatically update the status of the related policy or claim to reflect the initiation of the refund process. Once the refund has been successfully processed through the payment gateway, the status should then be updated to "Refunded".

##### C. Payment Tracking and History:

The system should keep track of all payment activities, including payments received and refunds issued. This information should be stored in a way that it can easily be retrieved for review or audit purposes.

##### D. Secure Payment Processing:

The system should ensure that all payment information is processed securely. This means using encryption to protect sensitive data, such as credit card numbers, and ensuring that the system is compliant with all relevant payment card industry (PCI) standards.

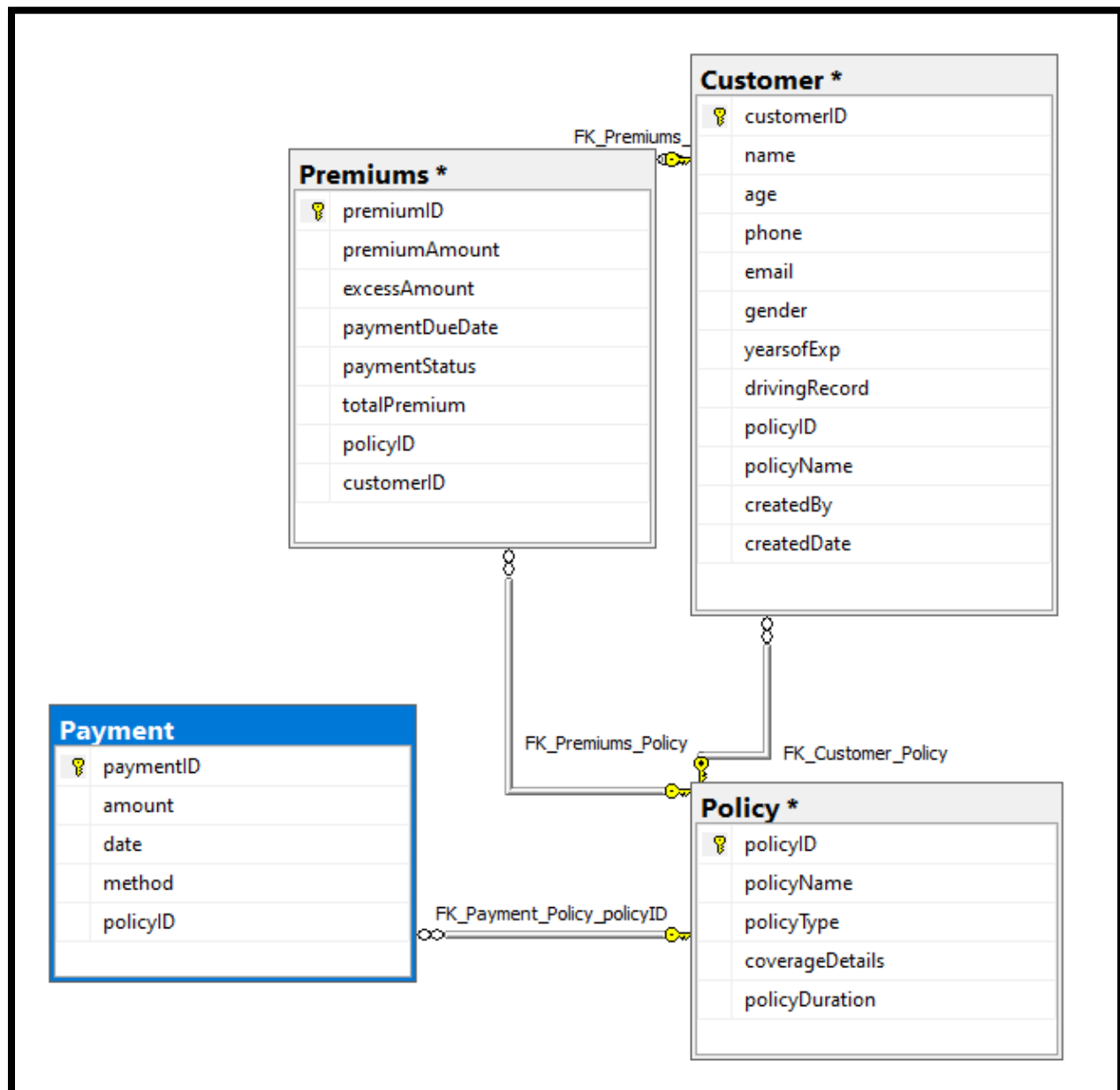
##### E. Payment Notifications:

Whenever a payment is made or a refund is issued, the system should automatically generate a notification. This notification should be sent to the customer, and it should provide details about the transaction, including the amount, the date, and the associated policy or claim.

## F. Payment Failures:

In cases where a payment fails due to insufficient funds, expired credit card, or any other reason, the system should notify the customer about the failed transaction and provide suggestions for resolving the issue.

## Entity Relationship Diagram



API End Points

Workflow

User Interface

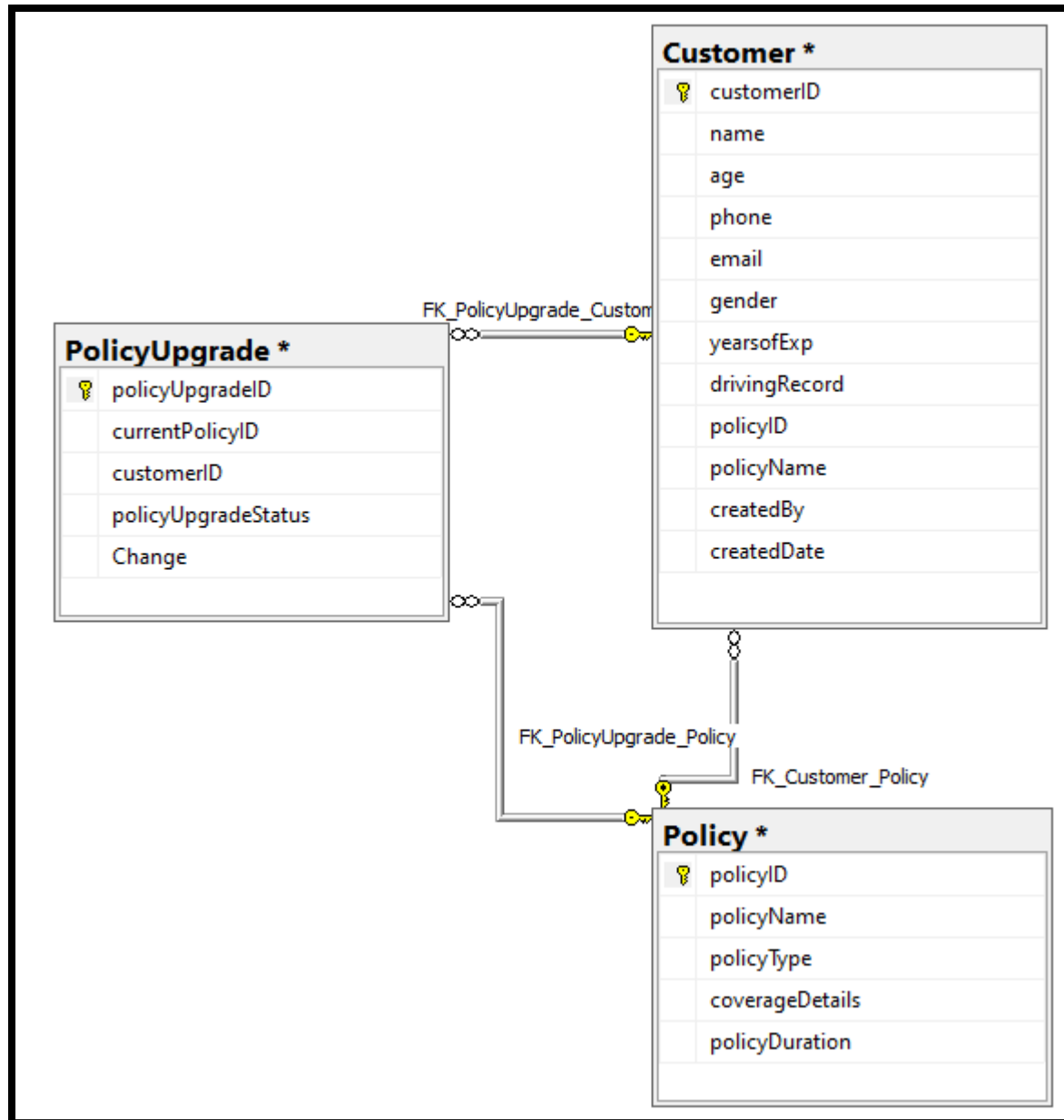
### f. Customer Requests Upgrade

#### Function Requirement

1. The system should provide a "Request Upgrade" option in the browser interface for customers to initiate an upgrade of their existing policies.
2. The system should determine the available upgrade options based on the insurer's policies and present them to the customer for selection.
3. In Scenario 1, the system should allow customers to edit their existing policies by upgrading from a Standard (Basic) cover plan to an Executive (Advanced) cover plan. This upgrade should include any additional features or benefits associated with the Executive cover plan.
4. In Scenario 2, the system should enable customers to edit their existing policies by increasing or decreasing the sum insured and excess amount. These changes should be reflected in the premiums charged for the policy.
5. In Scenario 3, the system should support the creation of multiple policies for a customer, with a group discount applied to the premiums. The system should calculate the discounted premiums based on the number of policies associated with the customer's account.
6. The system should validate the customer's eligibility for each upgrade option based on the insurer's policies and business rules.
7. The system should ensure that all upgrades and changes to policies are accurately recorded and reflected in the customer's account and relevant policy documents.
8. The system should generate notifications or alerts to inform the insurer's staff about customer requests for policy upgrades, enabling them to review and process the requests in a timely manner.
9. The system should provide appropriate error handling and validation to prevent invalid or conflicting upgrade requests.

10. The system should maintain an audit trail of all upgrade requests, including the details of the original policy, requested changes, approval status, and any additional information related to the upgrade process.
11. The system should generate updated policy documents reflecting the changes made through the upgrade process and provide them to the customer.
12. The system should ensure the security and privacy of customer data throughout the upgrade process, adhering to relevant data protection regulations.
13. The system should provide a user-friendly interface that guides customers through the upgrade process and provides clear instructions and feedback at each step.
14. The system should be scalable and capable of handling a high volume of upgrade requests simultaneously to ensure efficient processing and minimal downtime.
15. The system should integrate with relevant internal systems and databases to retrieve and update policy information accurately and in real-time during the upgrade process.

## Entity Relationship Diagram



## API End Points

API Function	Method	API Call	Input Parameters	Output	Error Handling
Get all policy upgrades	GET	<a href="http://localhost:5179/api/PolicyUpgrade">http://localhost:5179/api/PolicyUpgrade</a>	None	A list of policy upgrade objects	If no policy upgrades are found, return an empty list
Create a new policy upgrade	POST	<a href="http://localhost:5179/api/PolicyUpgrade">http://localhost:5179/api/PolicyUpgrade</a>	Policy upgrade object (JSON format)	The created policy upgrade object with a new ID	If input data is invalid or incomplete, return a 400 Bad Request status with an error message
Get a specific policy upgrade by ID	GET	<a href="http://localhost:5179/api/PolicyUpgrade/{id}">http://localhost:5179/api/PolicyUpgrade/{id}</a>	Policy upgrade ID (integer)	The policy upgrade object with the specified ID	If the policy upgrade ID is not found, return a 404 Not Found status with an error message
Update a policy upgrade by ID	PUT	<a href="http://localhost:5179/api/PolicyUpgrade/{id}">http://localhost:5179/api/PolicyUpgrade/{id}</a>	Policy upgrade ID (integer), updated policy upgrade object (JSON format)	The updated policy upgrade object	If the policy upgrade ID is not found, return a 404 Not Found status with an error message. If input data is invalid or incomplete, return a 400 Bad Request status with

					an error message
Delete a policy upgrade by ID	DELETE	<a href="http://localhost:5179/api/PolicyUpgrade/{id}">http://localhost:5179/api/PolicyUpgrade/{id}</a>	Policy upgrade ID (integer)	A confirmation message indicating successful deletion	If the policy upgrade ID is not found, return a 404 Not Found status with an error message

## Work Flow

### A. Request Upgrade:

1. On the customer's policy page, there should be a "Request Upgrade" button. This will direct the customer to an Upgrade Request form.
2. The form should show the details of the customer's current policy and the possible upgrade options. These options will depend on the existing policy's coverage and limits.

### B. Upgrade Options:

The form should allow for the following scenarios:

#### i. Scenario 1 - Upgrade Cover Plan:

1. If the current policy is a Standard (Basic) cover plan, the form should include an option to upgrade to an Executive (Advanced) cover plan.
2. On selection, the system should calculate the new premium for the Executive cover plan and display it to the customer.
3. If the customer accepts the new premium, they can submit the form to request the upgrade.

#### ii. Scenario 2 - Adjust Sum Insured and Excess Amount:

1. The form should include fields to adjust the sum insured and excess amount, along with an explanation of how these changes would affect the premium.
2. As the customer adjusts these values, the system should dynamically calculate and display the updated premium.



3. If the customer is satisfied with the changes, they can submit the form to request the upgrade.

### iii. Scenario 3 - Multiple Policies with Group Discount:


1. If the customer has multiple policies, the form should include an option to apply a group discount to the premiums.
2. On selection, the system should calculate the total premium with the group discount and display it to the customer.
3. If the customer accepts the new total premium, they can submit the form to request the upgrade.


### C. Submission and Processing:


1. Once the customer submits the form, the system should send the upgrade request to the insurer for approval.
2. The system should then update the customer's policy details as per the upgrade request and recalculate the premium accordingly.
3. The customer should be notified of the status of their upgrade request and any changes to their policy and premium.

## User Interface

### Customer View



 sarvesh

 Dashboard

Phone

243235

Excess Amount

\$ 300

#### Vehicle Details

Vehicle Make	Lexus	Vehicle Model	24
Vehicle Age	2	Vehicle Value	\$212119
VIN	213123	License Plate Number	21321
Mileage	212		

#### Premium Details

Premium Amount	\$1850	Premium Status	Not Paid
Premium Due Date	31 December	Insured Amount/ Sum Insured	\$212119

#### Policy Details

Basic Liability Coverage (Third-Party Insurance)

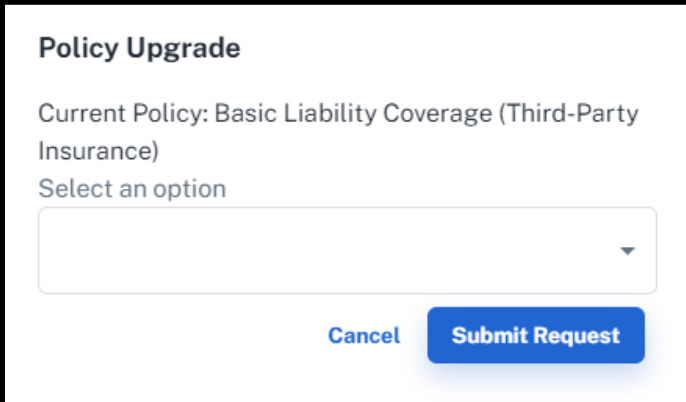
Basic liability coverage, also known as third-party insurance, is the minimum insurance required by law in most jurisdictions. This policy covers damages and injuries caused to other people and their property in the event of an accident where the policyholder is found to be at fault. It typically includes two main components: a. Bodily Injury Liability: Covers medical expenses, lost wages, and legal costs for injuries sustained by others in an accident caused by the policyholder. b. Property Damage Liability: Covers repair or replacement costs for property damage caused by the policyholder, such as vehicles, buildings, or other structures. Please note that basic liability coverage does not cover damages or injuries to the policyholder or their vehicle.

[Request Upgrade](#)
[Add Claim](#)
[Make Payment](#)
[Download PDF](#)

#### Claim Details

No claims data available for this customer.

To Request Upgrade



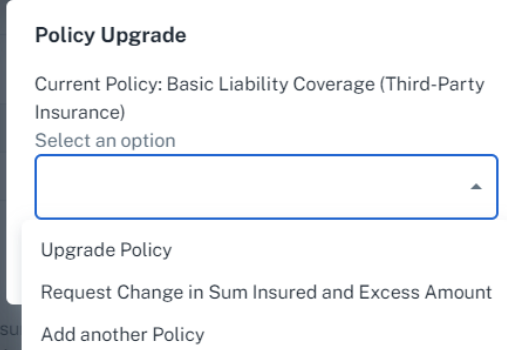
**Policy Upgrade**

Current Policy: Basic Liability Coverage (Third-Party Insurance)

Select an option

[Cancel](#) [Submit Request](#)

Select Scenario



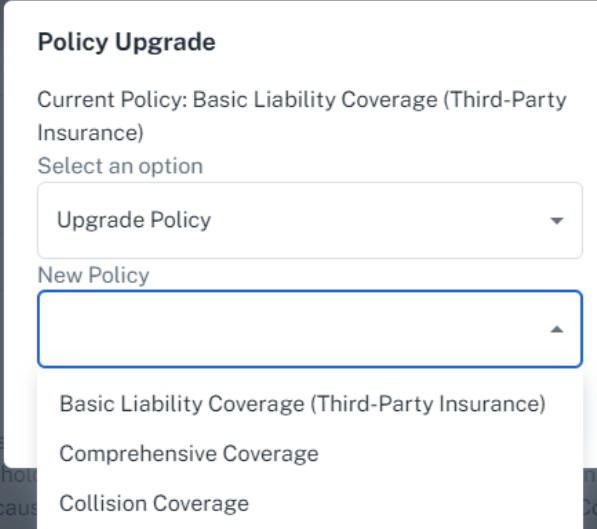
**Policy Upgrade**

Current Policy: Basic Liability Coverage (Third-Party Insurance)

Select an option

- Upgrade Policy
- Request Change in Sum Insured and Excess Amount
- Add another Policy

Senario 1



**Policy Upgrade**

Current Policy: Basic Liability Coverage (Third-Party Insurance)

Select an option

New Policy

- Basic Liability Coverage (Third-Party Insurance)
- Comprehensive Coverage
- Collision Coverage

### Senario 2

**Policy Upgrade**  
Current Policy: Basic Liability Coverage (Third-Party Insurance)  
Select an option  

Request Change in Sum Insured and Excess ... ▼

Change in Sum Insured \*

Change in Excess \*

[Cancel](#) [Submit Request](#)

### Senario 3



**Policy Upgrade**  
Current Policy: Basic Liability Coverage (Third-Party Insurance)  
Select an option  

Add another Policy ▼

New Policy  
Basic Liability Coverage (Third-Party Insuran... ▼

[Cancel](#) [Submit Request](#)

### Insurer View

Change Request Management					
Policy Upgrade ID	Current Policy Name	Customer Name	Status	Requested Change	Actions
1	Basic Liability Coverage (Third-Party Insurance)	sarvesh	Pending	The customer has request a policy upgrade. The policy is policy number: 2	 

Rows per page: 5 1-1 of 1 < >

After the change has been implemented

### Edit Change Request Status

Status

Pending

Cancel Save

## g. Claim Management

### Function Requirement

#### A. Claim Creation:

The system should provide a feature that allows insurers to create a new claim on behalf of a policyholder. This should involve the input of claim details such as the policy number, claim amount, claim type, and claim date.

Upon successful claim creation, the system should assign a unique claim number to the new claim and update the claim status to "Pending". The policyholder should also be notified of the claim's creation, and the claim details should be made available for them to view from their dashboard.

#### B. Claim Viewing:

The system should offer a search feature that allows insurers to locate and view a claim's details using a claim number, policy number, or customer name.

The claim's details, including its current status and any associated documents or evidence, should be presented in a clear and organized manner. The claim's history, including any status changes and notes added by claims staff, should also be viewable.

### C. Claim Updating:

Insurers should be able to update a claim's details, including its status. This could involve approving or rejecting the claim based on the provided evidence and the terms of the policyholder's insurance policy.

Whenever a claim is updated, the system should automatically notify the policyholder of the changes. If a claim is approved, the system might also initiate the payment process, depending on the insurer's policies and procedures.

### D. Claim Deletion:

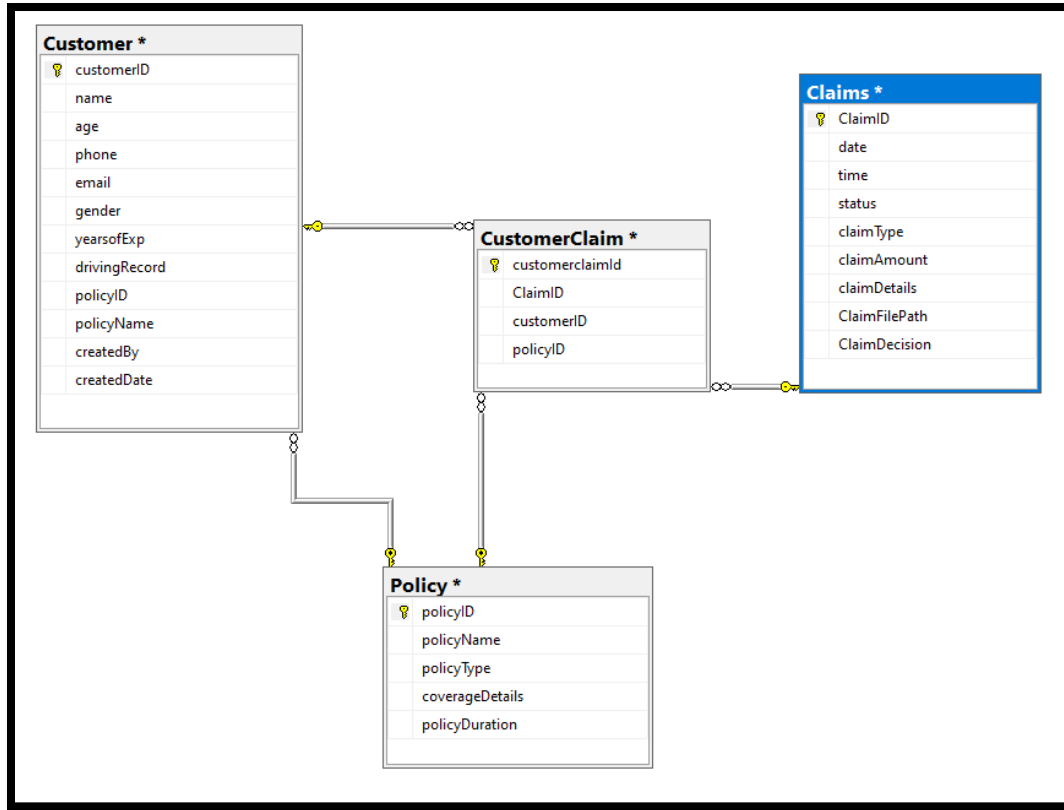
In situations where a claim is found to be invalid or is created in error, insurers should have the ability to delete the claim from the system. This should involve providing the claim number and possibly a reason for the deletion.

The system should ensure that all associated data is removed and that the policyholder is notified of the claim's deletion. However, for audit purposes, the system might keep a record of the deletion, including the claim number, the deletion date, and the reason for deletion.

### E. Claim History and Auditing:

The system should keep a detailed record of all claim activities, including claim creation, updates, and deletion. This claim history should be viewable by insurers and should be designed in a way that it can be used for auditing purposes if needed.

## Entity Relationship Diagram



## API End Points

API Function	Method	API Call	Input Parameters	Output	Error Handling
Get all claims	GET	http://localhost:5179/api/Claims	None	A list of claim objects	If no claims are found, return an empty list
Create a new claim	POST	http://localhost:5179/api/Claims	Claim object (JSON format)	The created claim object with a new ID	If input data is invalid or incomplete, return a 400 Bad Request status with an error message

Get a specific claim by ID	GET	http://localhost:5179/api/Claims/{id}	Claim ID (integer)	The claim object with the specified ID	If the claim ID is not found, return a 404 Not Found status with an error message
Update a claim by ID	PUT	http://localhost:5179/api/Claims/{id}	Claim ID (integer), updated claim object (JSON format)	The updated claim object	If the claim ID is not found, return a 404 Not Found status with an error message. If input data is invalid or incomplete, return a 400 Bad Request status with an error message
Delete a claim by ID	DELETE	http://localhost:5179/api/Claims/{id}	Claim ID (integer)	A confirmation message indicating successful deletion	If the claim ID is not found, return a 404 Not Found status with an error message

## Work Flow

### 1. Claim Creation:

An insurer logs into the Insurance Management System (IMS).

The insurer navigates to the "Create Claim" section.

The insurer inputs the claim details (policy number, claim amount, claim type, claim date) into the designated fields.

The system validates the provided details and checks the policy number against the database.

Upon validation, the system generates a unique claim number, assigns it to the new claim, and sets the claim status as "Pending".

The insurer submits the claim, and the system saves it into the database.

The system sends a notification to the policyholder regarding the creation of the claim and updates the claim details on the policyholder's dashboard.

### 2. Claim Viewing:

The insurer navigates to the "View Claim" section.

The insurer inputs the claim number, policy number, or customer name into the search field.

The system retrieves the corresponding claim details from the database.

The system displays the claim details, including status and any associated documents or evidence, in a user-friendly format. The claim history, including any status changes and notes, are also displayed.

### 3. Claim Updating:

The insurer navigates to the "Update Claim" section.

The insurer searches for the claim using the claim number, policy number, or customer name.

The system displays the claim details. The insurer can edit the claim details and change the claim status (approve/reject).

The insurer submits the updated claim details, and the system saves the changes into the database.

The system sends a notification to the policyholder about the changes made to their claim. If approved, the system initiates the payment process.

### 4. Claim Deletion:

The insurer navigates to the "Delete Claim" section.

The insurer inputs the claim number and provides a reason for deletion.

The system validates the claim number and deletion reason.

Upon validation, the system removes the claim from the active claims database and sends a notification to the policyholder about the claim deletion.

The system keeps a record of the deletion, including the claim number, deletion date, and reason for deletion, for auditing purposes.

## User Interface

### Customer View



sarvesh
 

Dashboard

Phone

243235

Excess Amount

\$ 300

### Vehicle Details

Vehicle Make	Lexus	Vehicle Model	24
Vehicle Age	2	Vehicle Value	\$212119
VIN	213123	License Plate Number	21321
Mileage	212		

### Premium Details

Premium Amount	\$1850	Premium Status	Not Paid
Premium Due Date	31 December	Insured Amount/ Sum Insured	\$212119

### Policy Details

Basic Liability Coverage (Third-Party Insurance)

Basic liability coverage, also known as third-party insurance, is the minimum insurance required by law in most jurisdictions. This policy covers damages and injuries caused to other people and their property in the event of an accident where the policyholder is found to be at fault. It typically includes two main components: a. Bodily Injury Liability: Covers medical expenses, lost wages, and legal costs for injuries sustained by others in an accident caused by the policyholder. b. Property Damage Liability: Covers repair or replacement costs for property damage caused by the policyholder, such as vehicles, buildings, or other structures. Please note that basic liability coverage does not cover damages or injuries to the policyholder or their vehicle.

[Request Upgrade](#)
[Add Claim](#)
[Make Payment](#)
[Download PDF](#)

### Claim Details

No claims data available for this customer.

## Add Claim

### Add Claim

Customer Name: sarvesh

Policy Name: Basic Liability Coverage (Third-Party Insurance)

Date: 5/18/2023

Time: 6:04:32 PM

Status: Pending

Claim Type \*

Claim Details \*

Upload Supporting Documents

Choose File

No file chosen

Cancel

Add Claim

After Claim is filed

[Request Upgrade](#)
[Add Claim](#)
[Make Payment](#)
[Download PDF](#)

### Claim Details

Claim ID	Policy Name	Claim Type	Claim Amount	Status	Claim File
1	Basic Liability Coverage (Third-Party Insurance)	Property Damage Liability	0	Pending	<a href="#">Download File</a>

Insurer View

Claim ID	Date	Status	Type	Amount	Details	Customer Name	Policy Name	Claim Decision	File	Action
1	5/18/2023	Pending	Property Damage Liability	0	Car destroyed by File	sarvesh	Basic Liability Coverage (Third-Party Insurance)		<a href="#">Download</a>	<a href="#">Edit</a> <a href="#">Delete</a>

Rows per page: 5 1-1 of 1

To View Claim Insurer will click on edit claim.

### Edit Claim

#### Update Claim Details

Status

Amount

Details

Type

Reason \*

[Download File](#)

[Cancel](#)
[Update](#)

Insurer can update claim status.

### Edit Claim

Update Claim Details

Status

Approved

Amount

1000

Details

Car destroyed by Fire

Type

Property Damage Liability

Reason \*

Good Claim

Download File

Cancel Update

Claim ID	Date	Status	Type	Amount	Details	Customer Name	Policy Name	Claim Decision	File	Action
1	5/18/2023	Approved	Property Damage Liability	1000	Car destroyed by Fire	sarvesh	Basic Liability Coverage (Third-Party Insurance)	Good Claim	<a href="#">Download File</a>	<a href="#">Edit</a> <a href="#">Delete</a>

The customer is also able to view the updated status.

Claim ID	Policy Name	Claim Type	Claim Amount	Status	Claim File
1	Basic Liability Coverage (Third-Party Insurance)	Property Damage Liability	1000	Approved	<a href="#">Download File</a>

# System Functionalities

## Premium Calculation

The `calculatePremium`` function helps determines the premium amount for an insurance policy based on a number of factors. Here's how it works:

1. **Base Premium:** The base premium is initially set at 200.
2. **Age Factor:** If the age of the customer is less than 25, an additional 300 is added to the premium. If the customer is 25 or older, an additional 200 is added.
3. **Experience Factor:** If the customer has less than 5 years of driving experience, the premium is increased by 250.
4. **Driving Record Factor:** Depending on the driving record of the customer, an additional amount is added to the premium. If the record is "fine", 400 is added. If the record shows an "accident", 200 is added. If the record is neither "fine" nor "accident", no amount is added.
5. **Vehicle Value Factor:** If the value of the vehicle is less than 30,000, an additional 200 is added to the premium. If the vehicle's value is 30,000 or more, an additional 400 is added.
6. **Mileage Factor:** If the mileage of the vehicle is less than 50,000, an additional 200 is added to the premium. If the mileage is 50,000 or more, an additional 400 is added.
7. **Vehicle Make Factor:** If the vehicle make is one of the specified luxury brands ("BMW", "Mercedes-Benz", "Audi", "Lexus", "Porsche", "Tesla"), an additional 500 is added to the premium.

After all the factors are considered and the premium is calculated, the function uses the `setPremiumAmount` function to store the premium amount, and the `setexcessAmount` function to set the excess amount (which is 20% of the calculated premium).

### Claim Management Process

#### **A. Load a Claim on Behalf of the Customer:**

The Insurance Management System (IMS) should be equipped with a user-friendly interface for claims staff. This interface includes options to input necessary information regarding the claim. It also include an option to upload files, functioning as proof of loss, through a file picker mechanism.

When a claim is loaded, the system should automatically assign a status of "Pending" to the claim and save all information related to the claim in the database. The system also ensures that the uploaded files are securely stored and easily accessible for review.

#### **B. Update the Existing Claim Status:**

The system allows claims staff to update the status of a claim, with options including "Pending", "Approved", or "Rejected". Upon updating the status, the system should automatically save the change in the database.

A claim is approved or rejected, by the Claims Manager, a Insurer with that job role.

#### **C. Customer Dashboard:**

Customers has the ability to create claims directly from their dashboard. The dashboard includes a user-friendly interface where customers can input necessary information and upload files as proof of loss.

When a customer submits a claim, the system automatically assigns a status of "Pending" to the claim, save all information related to the claim in the database.

The system also ensures customers are kept informed about the status of their claim.

### Handling Customer Request

The system functionality for handling customer request can be described as follows:

#### **1. Upgrade Request:**

The system provides an option for customers to request an upgrade of their existing policies. This could be achieved through a button or a link labelled "Request Upgrade" on the customer's dashboard or in the policy details section.

#### **2. Upgrade Options:**

After selecting the "Request Upgrade" option, customers are presented with different upgrade options. These options will depend on the specific policy the customer currently holds and the available upgrade paths.

### **3. Scenario 1 - Cover Plan Upgrade:**

Customers is be able to upgrade from a Basic cover plan to an Advanced cover plan. The system should update the policy details to reflect the new cover plan and adjust the premium according to the rates defined for the Advanced cover plan.

### **4. Scenario 2 - Adjust Sum Insured and Excess Amount:**

Customers has the option to increase or decrease the sum insured and the excess amount. Upon making these changes, the Insurer will recalculate the premium, reflecting the updated sum insured and excess amount.

### **5. Scenario 3 - Multiple Policies with Group Discount:**

The system allows customers to adopt multiple policies and get a group discount on premiums. When a customer adds a new policy, the system will check if the customer is eligible for a group discount (based on the number of policies or specific policy combinations) and, if so, automatically apply the discount to the premium calculations.

### **6. Policy Update:**

After the customer selects their desired upgrade option(s) and confirms their choice, the system will send a request to update the policy details in the database. The customer will then receive a notification confirming the successful upgrade of their policy.

### **7. Insurer Approval:**

Upgrades require approval from the insurer.

### Security

The Wheel Wise Insurance Management System implements various security measures to ensure the confidentiality, integrity, and availability of data within the system. These measures include data encryption, authentication, authorization, and adherence to specific compliance requirements.

### Data Encryption

Data encryption is employed to protect sensitive information, such as customer personal details, policy data, and payment information, both at rest and in transit.

For data in transit, the system uses HTTPS with SSL/TLS encryption to secure communication between the client and the server.

For data at rest, encryption techniques are used to secure sensitive information stored in the database, such as passwords and payment details.

### Authentication

The system requires users to provide valid login credentials, such as a username and password, to access the application.

Passwords are stored securely using a combination of hashing and salting techniques to protect against unauthorized access and potential attacks.

### Authorization

Role-based access control (RBAC) is used to manage user permissions, ensuring that users can only access and perform actions within the system according to their assigned roles.

Roles are defined based on user responsibilities, such as insurers, claims staff, and customers, with specific permissions granted to each role to control access to system resources.

# Testing

## Functional Testing

The Web API's were tested using a tool called Swagger and the results of the test have been displayed below.

User API End Point		View Image
GET /api/Users	Passed	<a href="#">Click to view</a>
POST /api/Users	Passed	<a href="#">Click to view</a>
GET /api/Users/{id}	Passed	<a href="#">Click to view</a>
PUT /api/Users/{id}	Passed	<a href="#">Click to view</a>
DELETE /api/Users/{id}	Passed	<a href="#">Click to view</a>
POST /api/Users/login	Passed	<a href="#">Click to view</a>

Policy API End Point		View Image
GET /api/Policy	Passed	<a href="#">Click to view</a>
POST /api/Policy	Passed	<a href="#">Click to view</a>
GET /api/Policy/{id}	Passed	<a href="#">Click to view</a>
PUT /api/Policy/{id}	Passed	<a href="#">Click to view</a>
DELETE /api/Policy/{id}	Passed	<a href="#">Click to view</a>

Customer API End Point		View Image
GET /api/Customer	Passed	<a href="#">Click to view</a>
POST /api/Customer	Passed	<a href="#">Click to view</a>
GET /api/Customer/{id}	Passed	<a href="#">Click to view</a>
PUT /api/Customer/{id}	Passed	<a href="#">Click to view</a>
DELETE /api/Customer/{id}	Passed	<a href="#">Click to view</a>

Vehicle API End Point		View Image
GET /api/Vehicle	Passed	<a href="#">Click to view</a>
POST /api/Vehicle	Passed	<a href="#">Click to view</a>
GET /api/Vehicle/{id}	Passed	<a href="#">Click to view</a>
PUT /api/Vehicle/{id}	Passed	<a href="#">Click to view</a>
DELETE /api/Vehicle/{id}	Passed	<a href="#">Click to view</a>

Premiums API End Point		View Image
GET /api/Premiums	Passed	<a href="#">Click to view</a>
POST /api/Premiums	Passed	<a href="#">Click to view</a>
GET /api/Premiums/{id}	Passed	<a href="#">Click to view</a>
PUT /api/Premiums/{id}	Passed	<a href="#">Click to view</a>
DELETE /api/Premiums/{id}	Passed	<a href="#">Click to view</a>



Claims API End Point		View Image
GET /api/ Claims	Passed	<a href="#">Click to view</a>
POST /api/ Claims	Passed	<a href="#">Click to view</a>
GET /api/ Claims /{id}	Passed	<a href="#">Click to view</a>
PUT /api/ Claims /{id}	Passed	<a href="#">Click to view</a>
DELETE /api/ Claims /{id}	Passed	<a href="#">Click to view</a>

PolicyUpgrade API End Point		View Image
GET /api/PolicyUpgrade	Passed	<a href="#">Click to view</a>
POST /api/PolicyUpgrade	Passed	<a href="#">Click to view</a>
GET /api/PolicyUpgrade /{id}	Passed	<a href="#">Click to view</a>
PUT /api/PolicyUpgrade /{id}	Passed	<a href="#">Click to view</a>
DELETE /api/ PolicyUpgrade /{id}	Passed	<a href="#">Click to view</a>

Payment API End Point		View Image
GET /api/Payment	Passed	<a href="#">Click to view</a>
POST /api/Payment	Passed	<a href="#">Click to view</a>
GET /api/Payment /{id}	Passed	<a href="#">Click to view</a>
PUT /api/Payment /{id}	Passed	<a href="#">Click to view</a>
DELETE /api/ Payment /{id}	Passed	<a href="#">Click to view</a>

React UI Test case were tested manually. The table lists the cases that were tested and their results.

• Test case 1	Create policy	Passed
• Test case 2	View/Search policy	Passed
• Test case 3	Update policy	Passed
• Test case 4	Delete policy	Passed
• Test case 5	Create claim	Passed
• Test case 6	View/Search claim	Passed
• Test case 7	Update claim	Passed
• Test case 8	Delete claim	Passed
• Test case 9	View premium	Passed
• Test case 10	Update premium	Passed
• Test case 11	Receive payment	Not Yet Completed
• Test case 12	Refund payment	Not Yet Completed
• Test case 13	Create user	Passed
• Test case 14	Update user	Passed
• Test case 15	Delete user	Passed

## Deployment

### System Requirements

#### Hardware Requirements

The following hardware requirements are recommended for optimal performance of the Wheel Wise Insurance Management System:

- Server: Quad-core processor, 16 GB RAM, 1 TB HDD or SSD storage
- Client: Modern computer or mobile device with a compatible web browser

#### Software Requirements

The software requirements for the IMS are as follows:

- Server: Windows Server
- Web server: Microsoft's Internet Information Services (IIS)
- Database: Microsoft SQL Server
- Backend: .NET Framework
- Frontend: HTML, CSS, JavaScript, and a modern web browser (e.g., Google Chrome, Mozilla Firefox, Microsoft Edge), node.js

### Installation and Configuration

To set up the Wheel Wise Insurance Management System, follow these steps:

1. Install the required server operating system, web server, database, and backend language on the server.
2. Download and extract the IMS source code to the appropriate directory on the server.
  - [Download Web API code.](#)
  - [Download React UI code.](#)
  - User Manual
  - 3. Configure the web server and the database according to the User Manual.
4. Create the required database tables and populate them with initial data, as specified in the IMS User Manual.
5. Configure the application's settings, such as database connection details, payment gateway integration, and email server settings.
6. Perform a test run of the IMS to ensure proper installation and configuration.

### Training

To ensure a smooth adoption of the Wheel Wise Insurance Management System, a comprehensive training program and documentation will be provided to users. Training sessions will be conducted for each user role, focusing on the specific functionalities and responsibilities associated with their role. In addition, detailed user guides, help documentation, and technical support will be available to assist users in navigating and utilizing the system effectively.

### Maintenance and Support

The Wheel Wise Insurance Management System will be regularly maintained and updated to address any issues, incorporate new features, and ensure compatibility with the latest technology standards. Technical support will be available to assist users with any system-related issues or concerns, providing prompt and efficient resolution to maintain the optimal performance of the IMS.

# Change Management and Requests

## Introduction

The purpose of this section is to outline the importance of a structured approach for managing change requests throughout the project lifecycle. Change management is an essential aspect of any software development project, as it ensures that all modifications, enhancements, or adjustments to the project's scope, requirements, or design are thoroughly analyzed, evaluated, and implemented in a controlled and consistent manner.

A structured approach to change management helps maintain the project's stability, minimize risks, and prevent unintended consequences that may arise from unplanned or uncoordinated changes. By following a well-defined process, the project team can effectively prioritize, review, and approve change requests, ensuring that all stakeholders are informed and involved in the decision-making process. This ultimately leads to a more successful project outcome, as it promotes better communication, collaboration, and transparency among team members and stakeholders, while preserving the project's integrity and alignment with its objectives.

## Change request process.

The process for submitting, reviewing, approving, and implementing change requests typically involves the following steps:

**Request submission:** A change request is submitted by an individual or team, typically through a designated change management system. The request should include details such as the reason for the change, the impact of the change, and any proposed solutions.

**Change request review:** The change request is reviewed by the change management team to determine its feasibility and impact. This step involves analyzing the potential risks and benefits of the proposed change and assessing whether it aligns with the organization's overall objectives.

**Change request approval:** Once the change request is deemed feasible and beneficial, it is then approved by the relevant stakeholders. This step may involve seeking approval from different levels of management, depending on the scope and impact of the change.

**Change implementation planning:** After approval, a detailed plan is developed for implementing the change. This plan should outline the timeline, resources required, and potential risks associated with the change.

**Change implementation:** The change is then implemented according to the plan. This may involve testing the change in a controlled environment before rolling it out to the broader organization.

**Change review and evaluation:** Once the change has been implemented, it is then reviewed to assess its effectiveness and ensure that it is meeting the desired outcomes. This step involves gathering feedback from stakeholders and analyzing data to measure the impact of the change.

**Change closure:** Finally, the change is closed out, and all documentation related to the change is updated to reflect the new state of the system or process. This step involves updating any relevant policies, procedures, or training materials to ensure that everyone is aware of the new change.

### Submitting a change request

Stakeholders can submit change requests through various channels, such as email, a web-based system, or a designated change request form. Regardless of the method, the change request should include the following information:

**Description of the change:** The change request should clearly describe the proposed change. This includes what is changing, why the change is necessary, and what is the expected outcome.

**Rationale:** The change request should provide a clear rationale for why the change is necessary. This should include the business or operational problem that the change is trying to address and why the proposed solution is the best approach.

**Priority:** Stakeholders should also indicate the priority of the change. This helps to ensure that the most critical changes are addressed first. The priority can be based on factors such as the potential impact on operations or the business, the urgency of the change, and the resources required for implementation.

**Potential impact:** The change request should also include an assessment of the potential impact of the change. This includes identifying any potential risks or negative consequences of the change, as well as any potential benefits.

**Supporting documentation:** Depending on the scope and complexity of the change, stakeholders may also need to provide supporting documentation. This may include technical specifications, project plans, and risk assessments.

**Approval requirements:** The change request should specify any approvals required for the change. This may include approvals from senior management, regulatory bodies, or other stakeholders.

### Review and evaluation

The process for reviewing and evaluating change requests typically involves the following steps:

**Initial review:** When a change request is received, it should be reviewed by the project manager and development team to assess its feasibility and potential impact. This step involves analyzing the potential risks and benefits of the proposed change and determining whether it aligns with the organization's overall objectives.

**Impact assessment:** After the initial review, the change management team conducts an impact assessment to evaluate the potential impact of the change on the organization. This step involves identifying all the areas of the organization that could be affected by the change, as well as any potential risks or negative consequences.

**Cost-benefit analysis:** The change management team then conducts a cost-benefit analysis to determine whether the benefits of the change outweigh the costs. This analysis should take into account the potential impact on operations, resources required for implementation, and the expected return on investment.

**Change request prioritization:** If multiple change requests are received, the change management team will prioritize them based on their potential impact and benefits. This helps to ensure that the most critical changes are addressed first.

**Review by stakeholders:** Once the impact assessment and cost-benefit analysis are completed, the change request should be reviewed by other stakeholders, such as senior management, regulatory bodies, or customers. This step helps to ensure that the proposed change is aligned with the organization's strategic objectives and meets the needs of all stakeholders.

**Approval:** After the change request has been reviewed and evaluated, it must be approved before implementation. This may involve seeking approval from different levels of management, depending on the scope and impact of the change.

**Documentation:** Finally, all documentation related to the change request should be updated to reflect the new state of the system or process. This step involves updating any relevant policies, procedures, or training materials to ensure that everyone is aware of the new change.

### Approval or rejection

The criteria for approving or rejecting change requests can vary depending on the organization, the nature of the change, and the impact on operations. Generally, the criteria for approving or rejecting change requests should be based on the following factors:

**Feasibility:** The proposed change should be feasible, meaning that it is technically possible to implement, and the necessary resources (such as personnel, budget, and time) are available to make the change.

**Impact:** The change should have a positive impact on operations, such as improving efficiency, increasing productivity, or enhancing customer satisfaction.

**Risk:** The potential risks associated with the change should be identified and mitigated, and the level of risk should be acceptable to the organization.

**Cost-effectiveness:** The benefits of the change should outweigh the costs of implementing it.

**Alignment with strategy:** The proposed change should be aligned with the organization's overall strategy and goals.

In terms of who has the authority to approve or reject change requests, this can vary depending on the organization's structure and policies. In general, the decision-making authority should be based on the level of impact and risk associated with the change. For example:

**Low-impact changes:** These changes can be approved by lower-level managers or supervisors.

**Medium-impact changes:** These changes may require approval from senior managers or a change control board.

**High-impact changes:** These changes typically require approval from top-level executives or a steering committee.

Ultimately, the decision-making authority for change requests should be clearly defined in the organization's change management policies and procedures to ensure consistency and accountability.

### Implementation

If a change request is approved, the following steps are typically involved in implementing the change:

**Update the project plan:** The project plan should be updated to reflect the approved change. This includes revising the timeline, budget, and resource allocation to ensure that the change can be implemented effectively.

**Communicate the change:** The change should be communicated to all stakeholders who will be affected by it, such as employees, customers, and suppliers. This helps to ensure that everyone is aware of the change and can prepare accordingly.

**Revise technical documentation:** Technical documentation, such as user manuals or training materials, should be updated to reflect the change. This ensures that everyone has accurate and up-to-date information about the system or process.

**Allocate resources:** The necessary resources, such as personnel, equipment, and budget, should be allocated to implement the change. This ensures that the change can be implemented effectively and efficiently.

**Test the change:** The change should be tested in a controlled environment before rolling it out to the broader organization. This helps to identify any issues or problems that may arise during implementation.

**Implement the change:** Once the change has been tested and any issues have been addressed, it can be implemented according to the revised project plan.

**Monitor and evaluate the change:** After implementation, the change should be monitored and evaluated to ensure that it is meeting its intended objectives and that there are no unexpected negative consequences. This helps to ensure that the change is effective and sustainable over the long term.

### Communication

Effective communication is critical when implementing any change in an organization. This is because changes can affect various stakeholders in different ways, and communication ensures that everyone is aware of the new requirements and how they are affected. Here are some reasons why communication is essential when implementing change:

**Builds trust:** Communication builds trust between the organization and its stakeholders. By being transparent about the change and how it will affect stakeholders, the organization can show that it values their input and is committed to their well-being.

**Reduces resistance:** When stakeholders are informed about the change, they are more likely to accept it. By understanding the reasons for the change and how it will benefit the organization, stakeholders are more likely to support the change.

**Facilitates implementation:** Effective communication ensures that all stakeholders are aware of the new requirements and are prepared to implement them. This can help to ensure that the change is implemented smoothly and efficiently.

**Avoids confusion:** Without effective communication, stakeholders may become confused about the new requirements, which can lead to errors, delays, and other problems. Clear communication can help to avoid these issues and ensure that everyone is on the same page.

**Enables feedback:** Communication allows stakeholders to provide feedback on the change, which can help to identify any issues or concerns that need to be addressed. This feedback can then be used to improve the change and ensure that it is effective and sustainable over the long term.

### Change request log.

Maintaining a log of all submitted change requests is essential to ensure that the change management process is transparent, efficient, and well-documented. The log should include the following information:

**Requestor's name:** The name of the person or team who submitted the change request.

**Date submitted:** The date when the change request was submitted.

**Description:** A brief description of the change request, including the proposed change and the rationale behind it.

**Priority:** The priority assigned to the change request, such as high, medium, or low.

**Status:** The current status of the change request, such as pending, approved, rejected, or implemented.

**Comments:** Any relevant comments or feedback related to the change request, such as reasons for rejection or implementation notes.

This log can be maintained in a separate document or tool, such as a spreadsheet, a project management tool, or a change management system. It should be accessible to all stakeholders involved in the change management process, including project managers, development teams, and senior management. The log should also be regularly updated to reflect the current status of each change request.

### Impact on project scope and timeline

Change requests can have a significant impact on project scope, timeline, and resources. Here are some ways that change requests can impact these areas:

**Project scope:** Change requests can expand or narrow the project scope. For example, if a change request is approved to add new features to a software product, this could increase the project scope. Conversely, if a change request is approved to remove features, this could narrow the project scope. It is important to ensure that any changes to the project scope are documented and communicated to all stakeholders.

**Project timeline:** Change requests can also impact the project timeline. If a change request is approved, it may require additional time to implement, test, and roll out. Conversely, if a change request is rejected, it may result in time savings. In either case, it is important to update the project plan to reflect any changes to the timeline.



Project resources: Change requests can also impact project resources. If a change request is approved, it may require additional personnel, budget, or equipment to implement. Conversely, if a change request is rejected, it may free up resources that can be allocated elsewhere. It is important to ensure that any changes to project resources are documented and communicated to all stakeholders.

If a change request leads to significant alterations, it is important to update the project plan and other relevant documentation accordingly. This helps to ensure that everyone is aware of the changes and that the project remains on track. For example, if a change request is approved that adds new features to a software product, the project plan should be updated to reflect the additional work required. This could include updating the timeline, budget, and resource allocation to ensure that the project can still be completed on time and within budget.

### Change request templates and tools.

Change Request Tracking Tool: A dedicated change request tracking tool (Trello) will be used to manage and track change requests. This tool should allow stakeholders to submit change requests, track the status of the requests, and receive notifications when a request is approved or rejected.

Requestor's Name:	
Date:	
Description:	
Rationale:	
Priority:	High / Medium / Low
Potential Impact:	
Supporting	
Documentation:	
Approval	
Requirements:	

## Appendix

Requestor's Name:	Jashvir
Date:	28 <sup>th</sup> of April
Description:	<p>We have identified potential security vulnerabilities in our IMS system that require additional security measures. We need to implement additional security measures to ensure that the system is secure and that our sensitive data is protected. The additional security measures include:</p> <ul style="list-style-type: none"> <li>• Implementing multi-factor authentication for all users who access IMS</li> <li>• Regularly performing vulnerability assessments and penetration testing to identify and mitigate potential security threats</li> </ul>
Rationale:	It is a vital part of the project and needs to be implemented for the security of the site.
Priority:	Medium
Potential Impact:	<p>Implementing these additional security measures will require some downtime for the IMS system while the changes are being made. During this time, users will not be able to access the system. We will need to schedule the changes during a time when the least number of users are affected, and we will provide advance notice to all users.</p>

Documentation:	<a href="https://elearn.usp.ac.fj/mod/resource/view.php?id=121325">https://elearn.usp.ac.fj/mod/resource/view.php?id=121325</a>
Approval	Approved by Team
Requirements:	<ul style="list-style-type: none"><li>• Implementing multi-factor authentication for all users who access IMS</li><li>• Regularly performing vulnerability assessments and penetration testing to identify and mitigate potential security threats.</li></ul>

# Conclusion

In conclusion, the Wheel Wise Insurance Management System (IMS) provides a robust, comprehensive, and user-friendly solution for managing and administering car insurance policies. Through its advanced features, such as policy management, claim management, premium management, payment management, customer management, and reporting capabilities, the IMS streamlines the insurance process, reducing operational costs and increasing efficiency.

The web-based nature of the system ensures that it is accessible from any location with an internet connection, catering to the needs of a modern, connected workforce. The non-functional requirements, including performance, security, and compatibility, have been meticulously addressed to provide a reliable, secure, and accessible system for its users.

The reporting capabilities offer valuable insights into the company's performance, enabling stakeholders to identify areas for improvement and implement strategic changes. By providing a centralized platform for managing all aspects of car insurance policies, the Wheel Wise IMS not only simplifies the day-to-day tasks of insurers but also enhances the customer experience, leading to increased customer satisfaction and retention.

This technical report has provided an in-depth analysis of the Wheel Wise IMS, its features, design, and benefits, serving as a comprehensive reference for stakeholders involved in the project. The successful implementation of the Wheel Wise IMS will undoubtedly contribute to the growth and success of the company, positioning it as a leader in the car insurance industry.

### Reference

#### 1. React JS

- Facebook, 2014. React: A JavaScript library for building user interfaces. [online] Available at: <https://reactjs.org> [Accessed 20 April 2023].

#### 2. ASP.NET Web APIs

- Microsoft, 2012. ASP.NET Web API: A framework for building HTTP services. [online] Available at: <https://docs.microsoft.com/en-us/aspnet/web-api/> [Accessed 20 April 2023].

#### 3. Vehicle Insurance Management

- Harrington, S.E. and Niehaus, G.R., 2013. Risk Management and Insurance. 3rd ed. Boston, MA: McGraw-Hill/Irwin.

#### 4. Agile Software Development

- Schwaber, K. and Sutherland, J., 2017. The Scrum Guide: The Definitive Guide to Scrum. [online] Available at: <https://www.scrumguides.org/scrum-guide.html> [Accessed 20 April 2023].

- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D., 2001. Manifesto for Agile Software Development. [online] Available at: <https://agilemanifesto.org/> [Accessed 20 April 2023].

#### 5. Change Request Process

- Project Management Institute, 2017. A Guide to the Project Management Body of Knowledge (PMBOK Guide), 6th ed. Newtown Square, PA: Project Management Institute.