**Document Status : Approved**

**Revision / Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ver.** | **Date** | **Changed by** | **Modifications** |
| 1.0 | 6/08/2025 | Shiva Prasad P |  |

**List of Abbreviations**

DFD  **D**ata **F**low **D**iagram

ER  **E**ntity **R**elationship

FHD  **F**unction **H**ierarchy **D**iagram

HLD **H**igh **L**evel **D**esign

LLD  **L**ow **L**evel **D**esign

GUI  **G**raphical **U**ser **I**nterface

IEEE  **I**nstitute of **E**lectricaland **E**lectronic **E**ngineers

S/W **S**oft**w**are

SDL **S**pecification **D**escription **L**anguage

StrD **Str**ucture**d**

Table of Contents

1. INTRODUCTION ............................................................................................................................. 2

2. DESIGN SCOPE.............................................................................................................................. 2

3. DESIGN METHODOLOGY ............................................................................................................. 3

4. DESIGN NOTATIONS ..................................................................................................................... 3

5. DESIGN CONSIDERATIONS ......................................................................................................... 3

6.DESIGN OVERVIEW ........................................................................................................................... 3

7.DECOMPOSITION ............................................................................................................................11

DH‑1‑1 Customer ....................................................................................................................................12  
DH‑1‑2 Insurer ........................................................................................................................................12  
DH‑1‑3 Invoice ........................................................................................................................................12  
DH‑1‑4 Payment .....................................................................................................................................12  
DH‑1‑5 Policy ..........................................................................................................................................12  
DH‑1‑6 TaxRate ......................................................................................................................................12  
DH‑1‑7 CustomerController ...................................................................................................................13  
DH‑1‑8 CustomerService ......................................................................................................................13  
DH‑1‑9 CustomerServiceImpl ................................................................................................................13  
DH‑1‑10 CustomerRepository ................................................................................................................13  
DH‑1‑11 InvoiceRepository ...................................................................................................................13  
DH‑1‑12 NotificationRequest ................................................................................................................13  
DH‑1‑13 Payment (Entity) ....................................................................................................................14  
DH‑1‑14 Policy (Entity) .........................................................................................................................14  
DH‑1‑15 InsurerController ......................................................................................................................14  
DH‑1‑16 InsurerService ..........................................................................................................................14  
DH‑1‑17 LoginController ........................................................................................................................15  
DH‑1‑18 LoginService / LoginServiceImpl .............................................................................................15  
DH‑1‑19 OtpService ................................................................................................................................15  
DH‑1‑20 CustomerRepository (Login Module) ......................................................................................15  
DH‑1‑21 InsurerRepository (Login Module) ..........................................................................................15  
DH‑1‑22 NotificationController .............................................................................................................16  
DH‑1‑23 NotificationService ..................................................................................................................16

8.INTERFACE DESIGN........................................................................................................................19

8.1 User Interface ............................................................................................................................19

9.DATA DESIGN ..................................................................................................................................19

9.2 Data structure (data types, arrays, and structures) ..............................................................21

10.REUSABILITY .................................................................................................................................21

11. DESIGN ALTERNATIVES .............................................................................................................21

12. DESIGN FEASIBILITY ...................................................................................................................21

13. ADDITIONAL HARDWARE AND SOFTWARE REQUIRED .......................................................21

14. TESTING STRATEGY ....................................................................................................................21

15. TRACEABILITY MATRIX...............................................................................................................21

16. REFERENCES................................................................................................................................22

# Introduction

The purpose of the Billing & Premium Payment Portal project is to provide a secure, user-friendly online system that enables insurance members to view, manage, and pay their monthly premiums conveniently and safely. This portal is designed to streamline the entire premium payment process, support auto-pay features and payment history, and integrate with major payment gateways like Razorpay for secure transactions.

The portal automates reminders for payments, invoices, auto-pay changes, and policy expiries via email/SMS, ensuring members never miss due dates.

It offers real-time access to payment and invoice histories, supports multiple payment options (including cash), and lets users manage preferences easily.

# Design Scope

The scope of this project covers the end-to-end experience and supporting infrastructure for insurance members and administrators to manage premium payments securely and efficiently through a web portal.

Customer Functions: To login,view invoices , pay premiums for invoices, autopay settings, payment history, receive notifications

Insurers : Issue Invoices, Payment History of all customers, Send Remainders like Policy expiry remainders, Pay By Cash, invoice history of all customers.

# Design Methodology

Object Oriented Analysis and Design (OOAD) methodology has been used for breaking down the specification into functionally independent units.

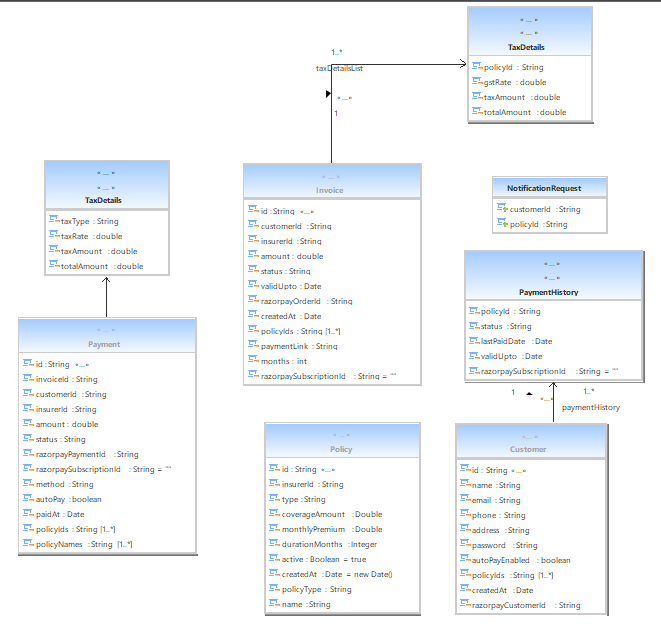
# Design Notations

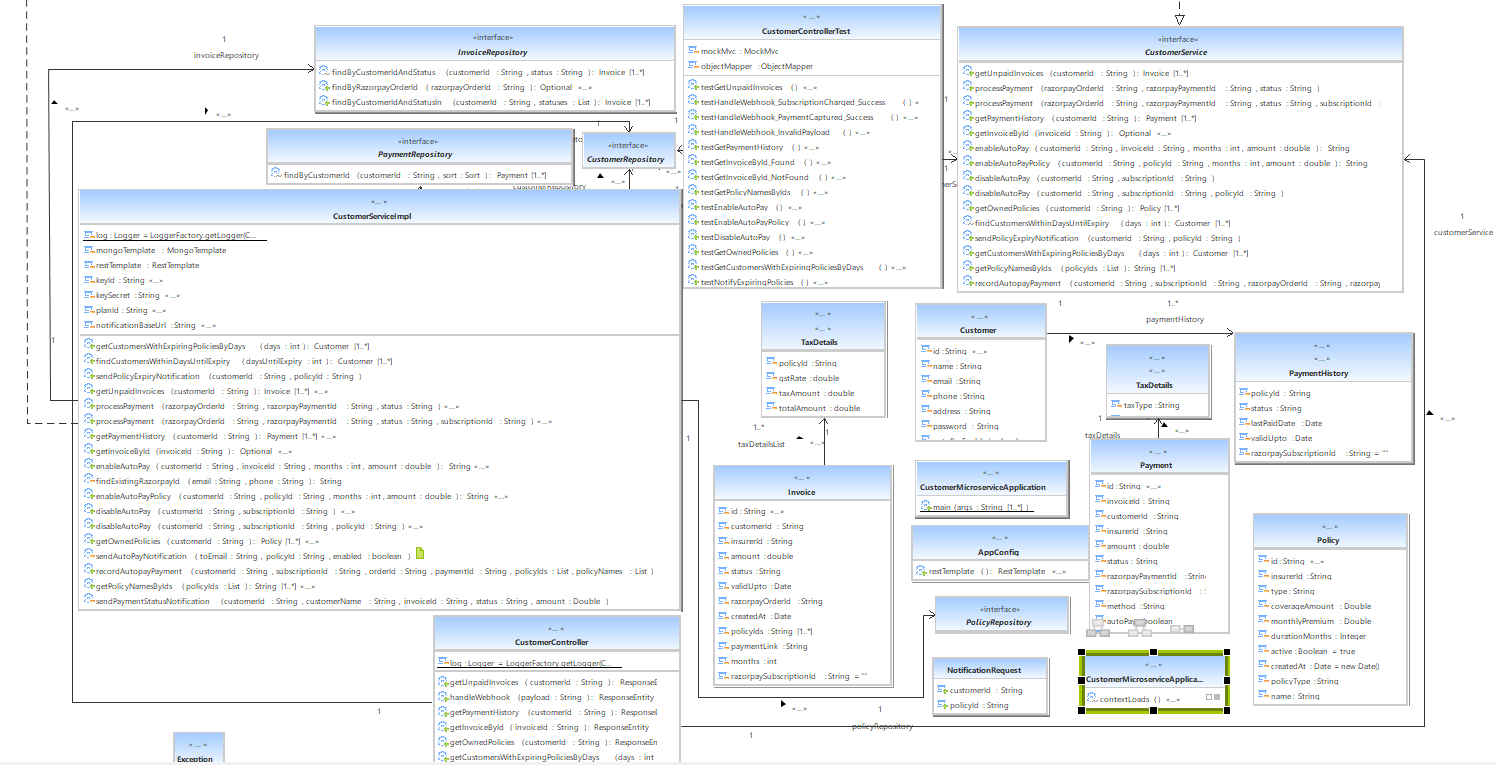
The naming conventions followed conform to Unified Modelling Language (UML) as Object Oriented Analysis and Design (OOAD) is followed.

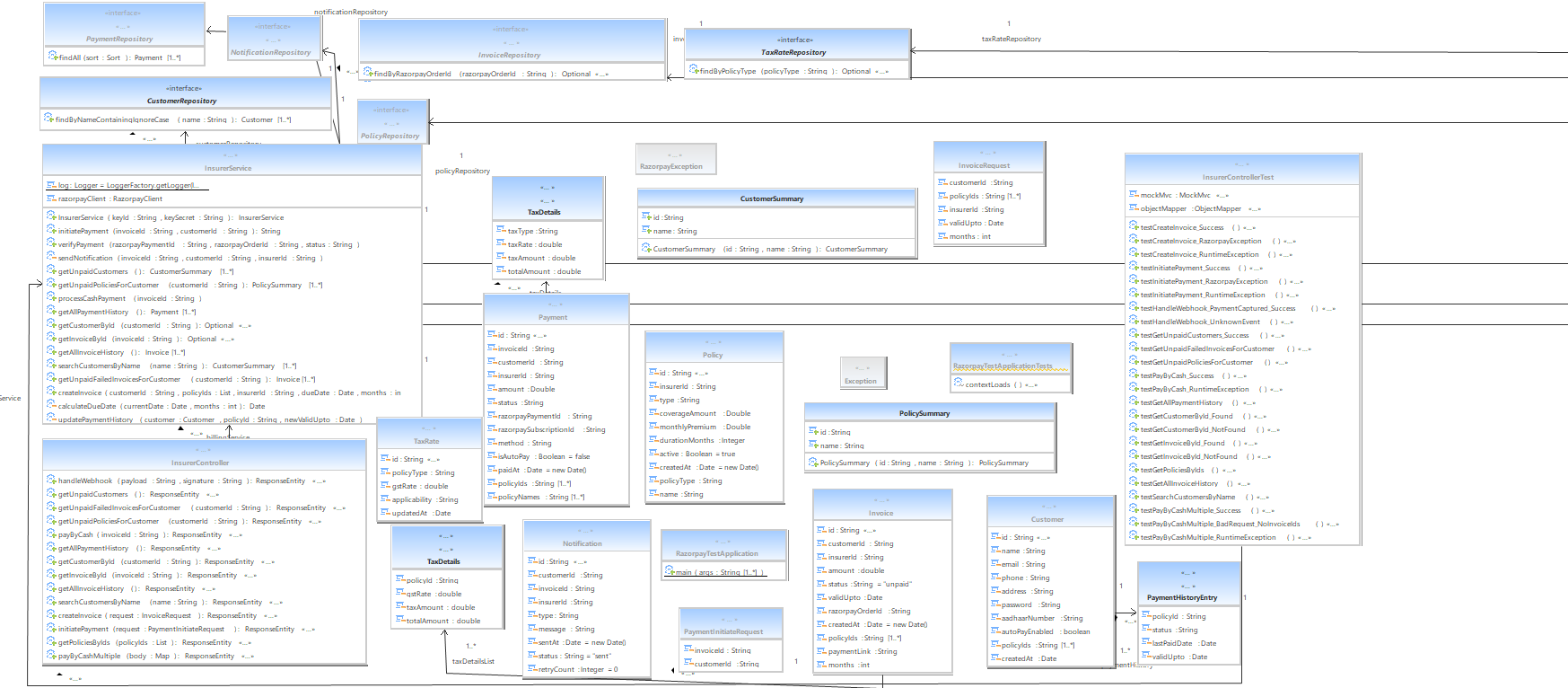
# Design Considerations

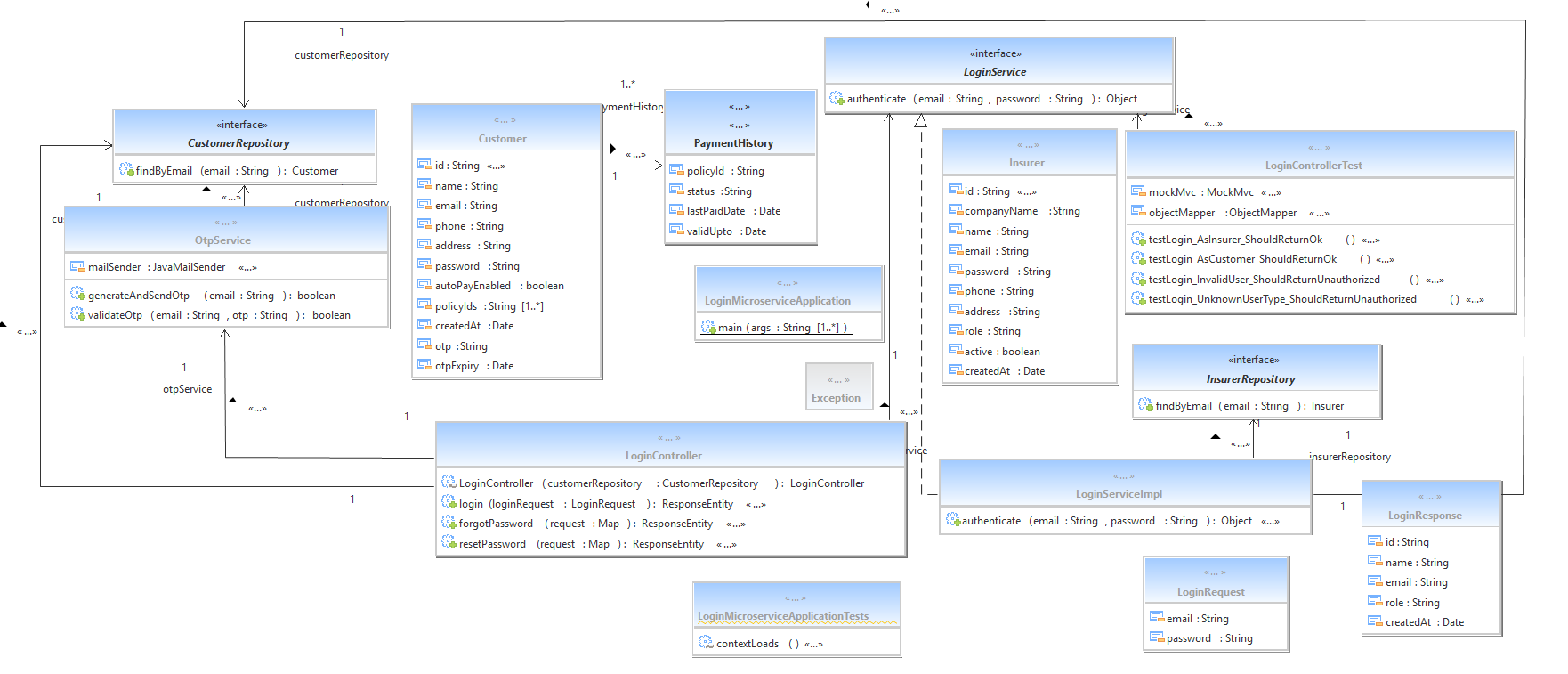
Not Applicable.

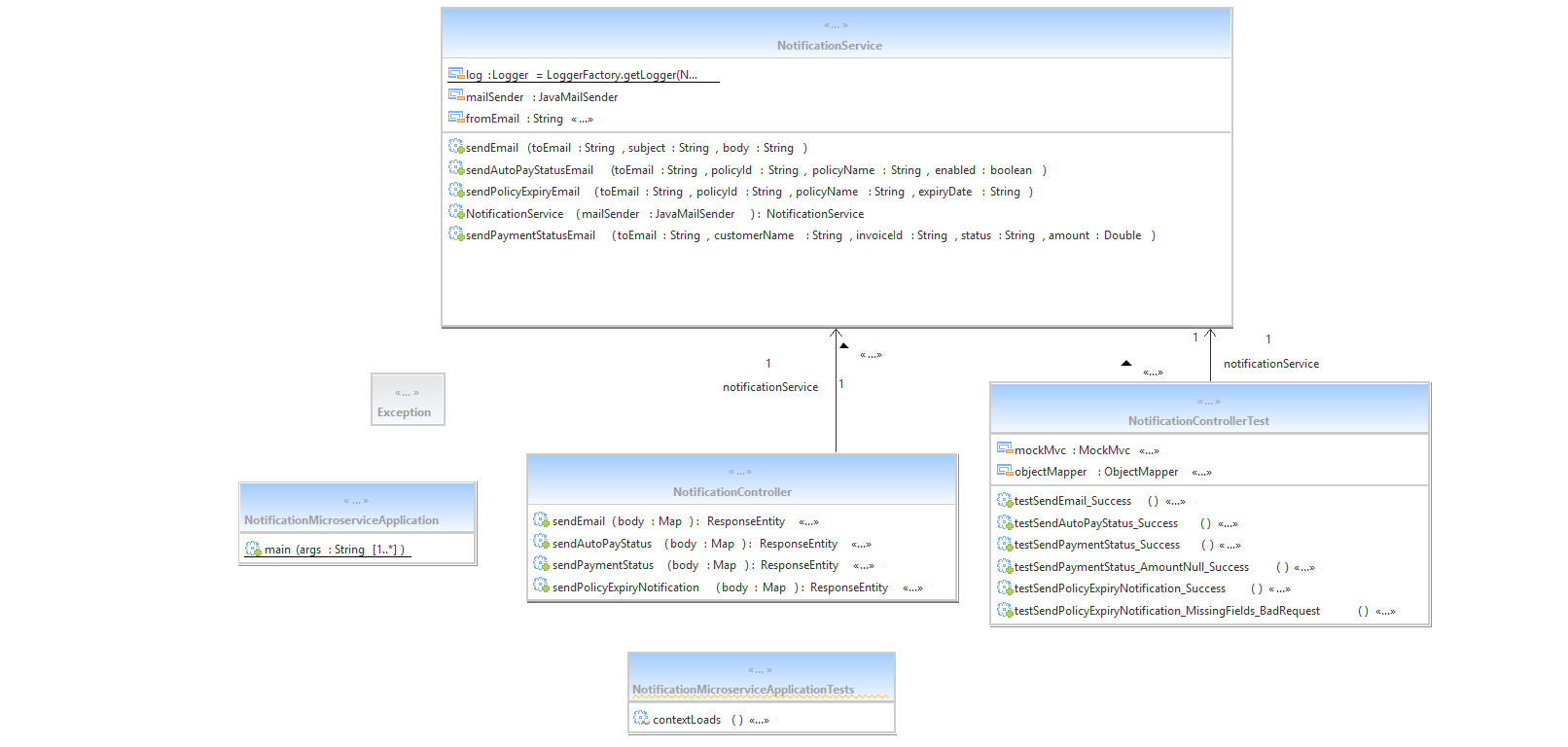
# 6. Design Overview









# 7. Decomposition

**DH-1-1 Customer**

**Inputs**: Sets the customer details (name, email, phone, address, password, auto-pay status, linked policy IDs, payment history, creation date, Razorpay customer ID).

**Outputs**: Gets the customer details from the database.

**Scope**: Specific

**Description**:

This class (training.iqgateway.entity.Customer) is a model class for the MongoDB collection customers, representing the insurance customer profile along with linked policies and payment history.

**DH-1-2 Insurer**

**Inputs**: Sets the insurer details (name, email, phone, address, password, creation date, and other insurer-specific fields).

**Outputs**: Gets the insurer details from the database.

**Scope**: Specific

**Description**:

This class (training.iqgateway.entity.Insurer) is a model class for the MongoDB collection insurers, representing insurance providers who manage policies, invoices, and payments.

**DH-1-3 Invoice**

**Inputs**: Sets the invoice details (customer ID, insurer ID, policy IDs, amount, tax details list, status, valid up-to date, Razorpay order ID, payment link, months, creation date, subscription ID).

**Outputs**: Gets the invoice details from the database.

**Scope**: Specific

**Description**:

This class (training.iqgateway.entity.Invoice) is a model class for the MongoDB collection invoices, representing billing records generated by insurers for customers' policy premiums, including payment gateway metadata.

**DH-1-4 Payment**

**Inputs**: Sets the payment details (invoice ID, customer ID, insurer ID, amount, status, Razorpay payment/subscription IDs, payment method, auto-pay flag, tax details, paid date, list of policy IDs/names).

**Outputs**: Gets the payment details from the database.

**Scope**: Specific

**Description**:

This class (training.iqgateway.entity.Payment) is a model class for the MongoDB collection payments, storing records of completed payments (online and cash), including associated tax and policy data.

**DH-1-5 Policy**

**Inputs**: Sets the policy details (name, coverage amount, monthly premium, duration, active status, policy type reference, creation date).

**Outputs**: Gets the policy details from the database.

**Scope**: Specific

**Description**:

This class (training.iqgateway.entity.Policy) is a model class for the MongoDB collection policies, defining the insurance products offered by insurers and linked to customer accounts.

**DH-1-6 TaxRate**

**Inputs**: Sets the tax rate details (policy type, GST rate, applicability, updated date).

**Outputs**: Gets the tax rate details from the database.

**Scope**: Specific

**Description**:

This class (training.iqgateway.entity.TaxRate) is a model class for the MongoDB collection taxRates, defining tax configurations for each type of insurance policy for invoice calculations.

**DH‑1‑7 CustomerController**

**Inputs**: HTTP requests from customers/clients (invoice ID, customer ID, policy IDs, payment data, webhook payloads, notification requests, etc.).

**Outputs**: HTTP responses containing invoices, payments, policies, customer details, success/failure messages.

**Scope**: Specific

**Description**: This Spring REST controller handles all customer‑facing operations such as viewing unpaid invoices, retrieving payment history, processing webhooks from Razorpay, managing auto‑pay, fetching owned policies, and triggering policy expiry notifications.

**DH‑1‑8 CustomerService**

**Inputs**:  
Method calls from controllers or other services, providing customer IDs, invoice IDs, policy IDs, payment/order/subscription IDs, autopay configuration parameters, and date ranges for expiry checks.

**Outputs**:  
Data returned to the caller, including lists of unpaid invoices, payment history, owned policies, subscription IDs, and operation acknowledgements for payment processing, autopay management, and notifications.

**Scope**:  
Specific

**Description**:  
A service interface in the business layer that defines all customer‑related operations — invoice retrieval, payment processing, autopay enable/disable, policy management, and notification triggers — without exposing implementation details.

**DH‑1‑9 CustomerServiceImpl**

**Inputs**:  
Actual runtime parameters passed from the controller layer — customer IDs, invoice IDs, policy IDs, amounts, months, webhook payload data, and payment gateway identifiers (order ID, payment ID, subscription ID).

**Outputs**:  
Persisted changes to MongoDB collections (customers, invoices, payments, policies), returned datasets to controllers, subscription IDs from Razorpay, and HTTP calls to the Notification Microservice.

**Scope**:  
Specific

**Description**:  
The concrete implementation of CustomerService containing the business logic for handling customer operations — integrating with MongoDB repositories, Razorpay APIs for one‑time and recurring payments, filtering expiring policies, updating payment histories, and sending notification requests over HTTP to the Notification Microservice.

**DH‑1‑10 CustomerRepository**

**Inputs**:  
Customer ID or other query filters passed in via Spring Data MongoDB operations.

**Outputs**:  
Customer entity objects retrieved from or persisted to the customers collection.

**Scope**:  
Specific

**Description**:  
A Spring Data MongoRepository for performing CRUD operations on the Customer collection, providing built‑in methods for saving, finding, and deleting customer records.

**DH‑1‑11 InvoiceRepository**

**Inputs**:  
Customer ID, invoice status, list of statuses, or Razorpay order ID parameters for query methods.

**Outputs**:  
Invoice entity objects matching the criteria from the invoices collection.

**Scope**:  
Specific

**Description**:  
A Spring Data MongoRepository for the Invoice collection with custom query methods to find invoices by customer and status or by Razorpay order ID.

**DH‑1‑12 NotificationRequest**

**Inputs**:  
Customer ID and Policy ID for which a notification is to be generated.

**Outputs**:  
Simple DTO carrying these two values to be used by services/controllers.

**Scope**:  
Specific

**Description**:  
A plain Java object (POJO) used to pass notification trigger information (customerId, policyId) between layers.

**DH‑1‑13 Payment**

**Inputs**:  
Payment details such as invoice ID, customer ID, insurer ID, amount, payment status, Razorpay IDs, method, autopay flag, tax breakdown, policy IDs, and payment date.

**Outputs**:  
Payment entity records retrieved from or stored into the payments collection.

**Scope**:  
Specific

**Description**:  
MongoDB entity for payments capturing both online and cash payments and their tax/policy details.

**DH‑1‑14 Policy**

**Inputs**:  
Policy attributes such as insurer ID, type, coverage amount, monthly premium, active status, policy type, name, and creation date.

**Outputs**:  
Policy entity records retrieved from or stored into the policies collection.

**Scope**:  
Specific

**Description**:  
MongoDB entity representing insurance product details offered by an insurer, linked to customers via policy IDs.

**DH‑1‑15 InsurerController**

**Inputs**:  
HTTP requests from insurer clients with parameters and payloads such as customer IDs, invoice IDs, policy IDs, search strings, invoice creation details, payment initiation details, webhook payloads from Razorpay, and cash payment request bodies.

**Outputs**:  
HTTP responses containing created invoices, lists of customers, invoices, policies, payments, search results, and plain text success/failure acknowledgements for processed operations.

**Scope**:  
Specific

**Description**:  
This Spring REST controller handles all insurer‑side operations including invoice creation, initiating payments via Razorpay, receiving and processing Razorpay webhooks, retrieving unpaid customers/policies/invoices, processing cash payments (single or multiple), viewing payment and invoice histories, data lookup for customers, invoices, policies, and searching customers by name.

**DH‑1‑16 InsurerService**

**Inputs**:  
Customer ID, invoice ID(s), insurer ID, list of policy IDs, due date, months, Razorpay payment/order/subscription IDs, search strings, payment initiation/verification data, webhook payload details, and cash payment requests.

**Outputs**:  
Created invoice objects, Razorpay order IDs, payment records, updated customer payment histories, lists of customers/policies/invoices/payments, acknowledgements for successful payment processing, and notification triggers saved in the database.

**Scope**:  
Specific

**Description**:  
A Spring @Service component containing insurer‑side business logic: creating invoices with dynamic tax calculation, integrating with Razorpay for order/payment link generation, initiating and verifying payments, fetching unpaid customers/policies, retrieving invoice/payment histories, searching customers, and handling cash payments (single and multiple). It also updates MongoDB collections (customers, invoices, payments, notifications) and sends payment‑related notifications.

**DH‑1‑17 LoginController**

**Inputs**:  
Login credentials (email, password), forgot password request (email), reset password request (email, otp, newPassword).

**Outputs**:  
LoginResponse DTO for authenticated users, status messages for OTP sent or invalid OTP, confirmation for password reset, or HTTP error codes for failed operations.

**Scope**:  
Specific

**Description**:  
A Spring REST controller handling login, forgot password, and password reset flows for both insurer and customer accounts. It delegates authentication to LoginService, OTP generation/validation to OtpService, and interacts with the CustomerRepository for updating passwords.

**DH‑1‑18 LoginService / LoginServiceImpl**

**Inputs**:  
Email and password credentials passed from the controller.

**Outputs**:  
Authenticated Insurer or Customer object on success, or null if authentication fails.

**Scope**:  
Specific

**Description**:  
Business service for verifying login credentials. Determines user type based on email pattern (insurer vs gmail), retrieves records from InsurerRepository or CustomerRepository, and validates the password.

**DH‑1‑19 OtpService**

**Inputs**:  
Email for OTP generation, and OTP verification request (email, otp).

**Outputs**:  
Boolean flag indicating OTP sent successfully, or OTP validation result.

**Scope**:  
Specific

**DH‑1‑20 CustomerRepository (Login Module)**

**Inputs**:  
Email or ID values to query customers collection.

**Outputs**:  
Matching Customer entity or null if not found.

**Scope**:  
Specific

**Description**:  
Spring Data MongoDB repository providing CRUD and query methods for the Customer collection, with a custom method findByEmail.

**DH‑1‑21 InsurerRepository (Login Module)**

**Inputs**:  
Email or ID values to query insurers collection.

**Outputs**:  
Matching Insurer entity or null if not found.

**Scope**:  
Specific

**Description**:  
Spring Data MongoDB repository for accessing Insurer entity data, including custom finder findByEmail.

**DH‑1‑22 NotificationController**

**Inputs**:  
HTTP POST requests containing recipient email, subject/content; autopay status details; payment status details; or policy expiry details.

**Outputs**:  
HTTP responses confirming successful email sending or error messages for missing fields.

**Scope**:  
Specific

**Description**:  
Spring REST controller that exposes endpoints for sending different types of notification emails — generic emails, auto‑pay status changes, payment status alerts, and policy expiry reminders. Delegates actual sending to NotificationService.

**DH‑1‑23 NotificationService**

**Inputs**:  
Target email address, email subject/content, and contextual data like policy ID, policy name, invoice ID, status, or expiry date.

**Outputs**:  
Outbound email messages sent via JavaMailSender to customers.

**Scope**:  
Specific

**Description**:  
Service component responsible for formatting and sending notification emails. Supports generic mails, auto‑pay status messages, payment success/failure alerts, and policy expiry reminders. Uses JavaMailSender and from\_email configuration from application properties.

# 8. Interface Design

NA

## 8.1 User Interface

NA

# 9. Data design

A screenshot of a computer

AI-generated content may be incorrect.

<The entity relationship diagram prepared for the database has to be defined here.>

## 9.2 Data structure (data types, arrays, and structures)

Not applicable.

# 10. Reusability

* LOGIN
* Email Service

# 11. Design Alternatives

NA

**12. Design Feasibility**

We have used the OOAD approach in this project. This methodology has been chosen based on our analogy of the user requirements, feasibility study and based on the experience of the co-ordinators. It has been seen that several other project groups developing similar projects have chosen the same methodology.

The OOAD assures properties such as reusability, modularity, efficiency.

# Additional Hardware and Software required

This requirement is based on the future stages of development. Therefore as of now this is not applicable

# 14. Testing Strategy

1. Testing Levels & Types

a. Unit Testing (White Box): Performed using JUnit for each backend microservice (Login Authentication, Hospital Staff, Procurement Officer, System Admin). Unit tests validated the internal logic, method outputs, and database interactions.

b. Integration Testing: Verified interfaces and data flow between microservices and the frontend.

c. Regression Testing: Conducted after new features or fixes to ensure existing functionality still behaved correctly.

d. Manual Functional Testing (Black Box): Conducted using Postman to send and validate REST API requests/responses, and Swagger UI for interactive API validation and documentation verification.

**2. Tools & Frameworks**

a. JUnit – automated unit tests in Java for Spring Boot services.

b. Postman – manual and automated API request/response validation.

c. Swagger – API documentation testing and quick endpoint verification

# 15. Traceability Matrix

As per the requirements-HLD tagging shown in the document “Requirement\_Traceability.xls” each of the requirements has been mapped to the appropriate classes. Both the requirements and classes have been tagged according to the tag standards of RBIN.

# 16. References

List of all external sources of information referenced in this document.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No.** | **Description** | **Date** | **Vers.** | **Location** |
| 1. | Software Requirements Specification Document | 11/08/2025 | 1.0 | *SRS.doc* |
| 2. | High Level Design –Billing And Premium Payment Portal | 10/10/2023 | 1.0 |  |
| 3. | Low Level Design – Billing And Premium Payment Portal | 25/07/2025 | 1.0 |  |
| 4 | Billing And Payment Portal / Readme | 11/08/2025 | 1.0 |  |

Description, date, and version shall uniquely identify the information source, and the location shall specify where it is to be found.