**Reviewed by:**

**Approved by:**

Revision / Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Ver.** | **Date** | **Changed by** | **Modifications** |
| 1.0 | 21/07/2025 | Roopesh | Initial version of document, Database design is included in this document, review changes implemented |

Note: Present content to be deleted and revision history to be updated with project's document history, when used in Projects

**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 [4](#__RefHeading___Toc371459675)

2. Design Scope [4](#__RefHeading___Toc371459676)

3. Design Methodology [4](#__RefHeading___Toc371459677)

4. Design Notations [4](#__RefHeading___Toc371459678)

5. Design Considerations [4](#__RefHeading___Toc371459679)

6. Design Overview [5](#__RefHeading___Toc371459680)

7. Decomposition [13](#__RefHeading___Toc371459681)

DH-1-1 ProviderEO [13](#__RefHeading___Toc371459682)

DH-1-2 DoctorEO [13](#__RefHeading___Toc371459683)

DH-1-3 CustomerEO [13](#__RefHeading___Toc371459684)

DH-1-4 AdminEO [13](#__RefHeading___Toc371459685)

DH-1-5 InsuranceTeamEO [13](#__RefHeading___Toc371459686)

DH-1-6 InsurancePlansEO [13](#__RefHeading___Toc371459687)

DH-1-7 Notification [14](#__RefHeading___Toc371459688)

DH-1-8 ProviderServiceImpl [14](#__RefHeading___Toc371459689)

DH-1-9 DoctorServiceImpl [14](#__RefHeading___Toc371459690)

DH-1-10 CustomerServiceImpl [14](#__RefHeading___Toc371459691)

DH-1-11 LoginServiceImpl [14](#__RefHeading___Toc371459692)

DH-1-12 OtpService [14](#__RefHeading___Toc371459693)

DH-1-13 AdminServiceImpl [15](#__RefHeading___Toc371459694)

DH-1-14 InsurancePlansServiceImpl [15](#__RefHeading___Toc371459695)

DH-1-15 HospitalNotificationListener [15](#__RefHeading___Toc371459696)

DH-1-16 EmailService [15](#__RefHeading___Toc371459697)

DH-1-17 ProviderController [17](#__RefHeading___Toc371459705)

DH-1-18 DoctorController [17](#__RefHeading___Toc371459706)

DH-1-19 CustomerController [17](#__RefHeading___Toc371459707)

DH-1-20 NotificationController [17](#__RefHeading___Toc371459708)

DH-1-21 InsurancePlansController [17](#__RefHeading___Toc371459709)

DH-1-22 LoginController [17](#__RefHeading___Toc371459709)

8. Interface Design [18](#__RefHeading___Toc371459710)

8.1 User Interface [18](#__RefHeading___Toc371459711)

9. Data design [19](#__RefHeading___Toc371459712)

9.2 Data structure (data types, arrays, and structures) [20](#__RefHeading___Toc371459713)

10. Reusability [20](#__RefHeading___Toc371459714)

11. Design Alternatives [20](#__RefHeading___Toc371459715)

13. Additional Hardware and Software required [20](#__RefHeading___Toc371459716)

14. Testing Strategy [20](#__RefHeading___Toc371459717)

15. Traceability Matrix [20](#__RefHeading___Toc371459718)

16. References [21](#__RefHeading___Toc371459719)

# Introduction

The ***Provider Finder & Network Lookup*** project is designed to enable insurance members and users to efficiently search for healthcare providers—such as doctors and hospitals—who are within their insurance networks. The goal is to minimize out-of-network visits that can result in unexpected high costs, and to provide clarity and transparency regarding insurance coverage for medical services. The system empowers users to locate in-network providers based on specialty, location, and insurance plan, ensuring informed decision-making for medical care

# Design Scope

* The scope of this project involves the development of a comprehensive system for the efficient search, verification, and management of healthcare providers within an insurance network, including the following functional areas:
* **Search Functionality**: Users can search and filter healthcare providers by specialty, geographic location, and accepted insurance plans. Advanced filters such as provider ratings, distance, and availability will be supported, with results displayed in both list and interactive map views.
* **Provider Profiles:** Structured and detailed profiles for doctors, hospitals, and clinics including specialties, qualifications, experience, ratings/reviews, offered services, and accepted insurance plans.
* **Map**  **Integration:** Integration with mapping APIs to display search results geographically, support location-based filtering, and provide directions to provider locations.
* **Admin and Provider Management:** Tools for administrators to add, update, validate, and remove provider details, manage provider affiliations, and maintain network accuracy.
* **Insurance Team :** Insurance Team Manages the Customer and Insurance Plans which involves to add,update,validate and remove the Customer and insurance Plans details.

# Design Methodology

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

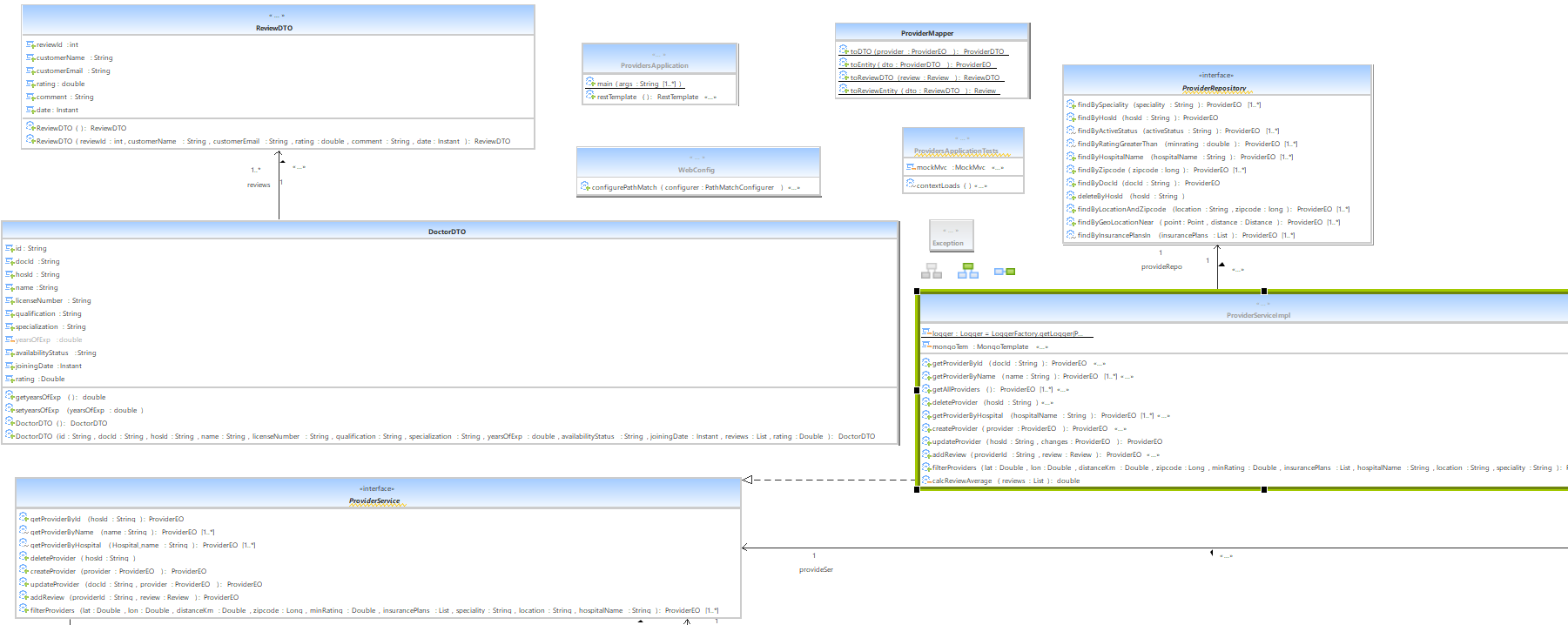
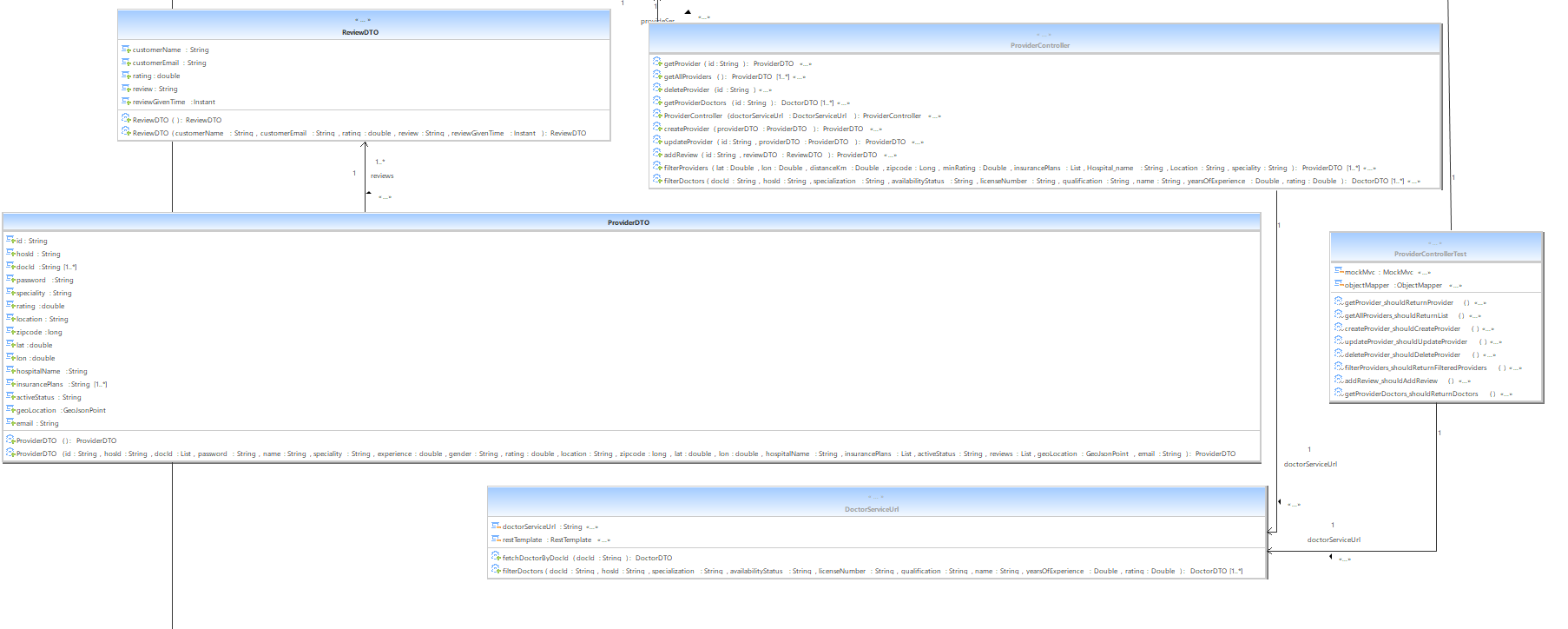
# Design Notations

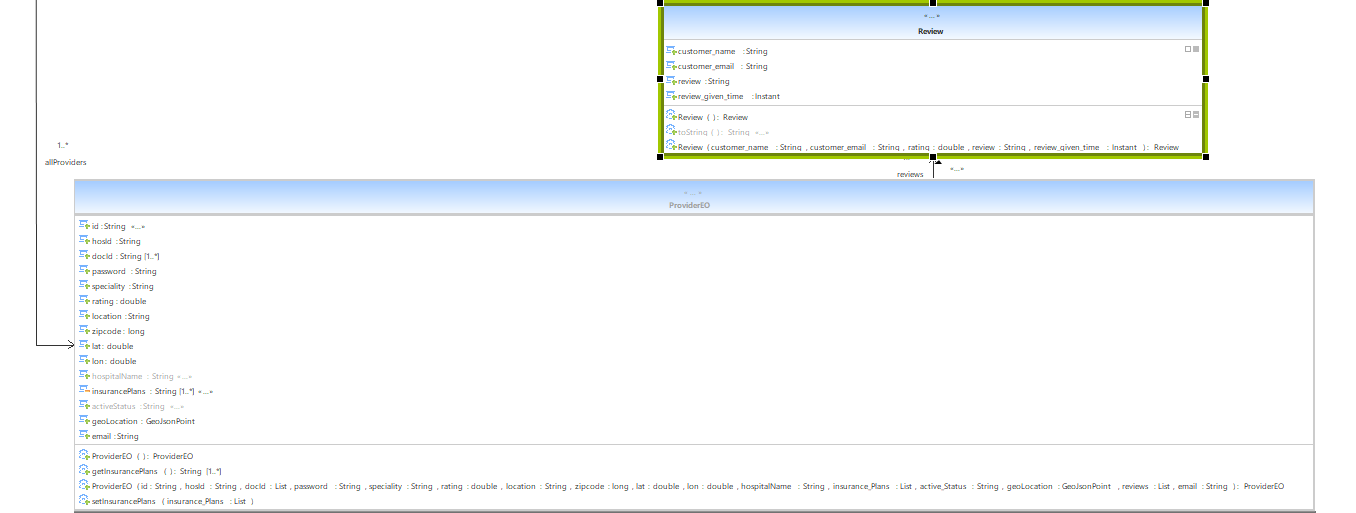
The naming conventions followed conform to Unified Modelling Language (UML) as Object Oriented Analysis and Design (OOAD) is followed.

# Design Considerations

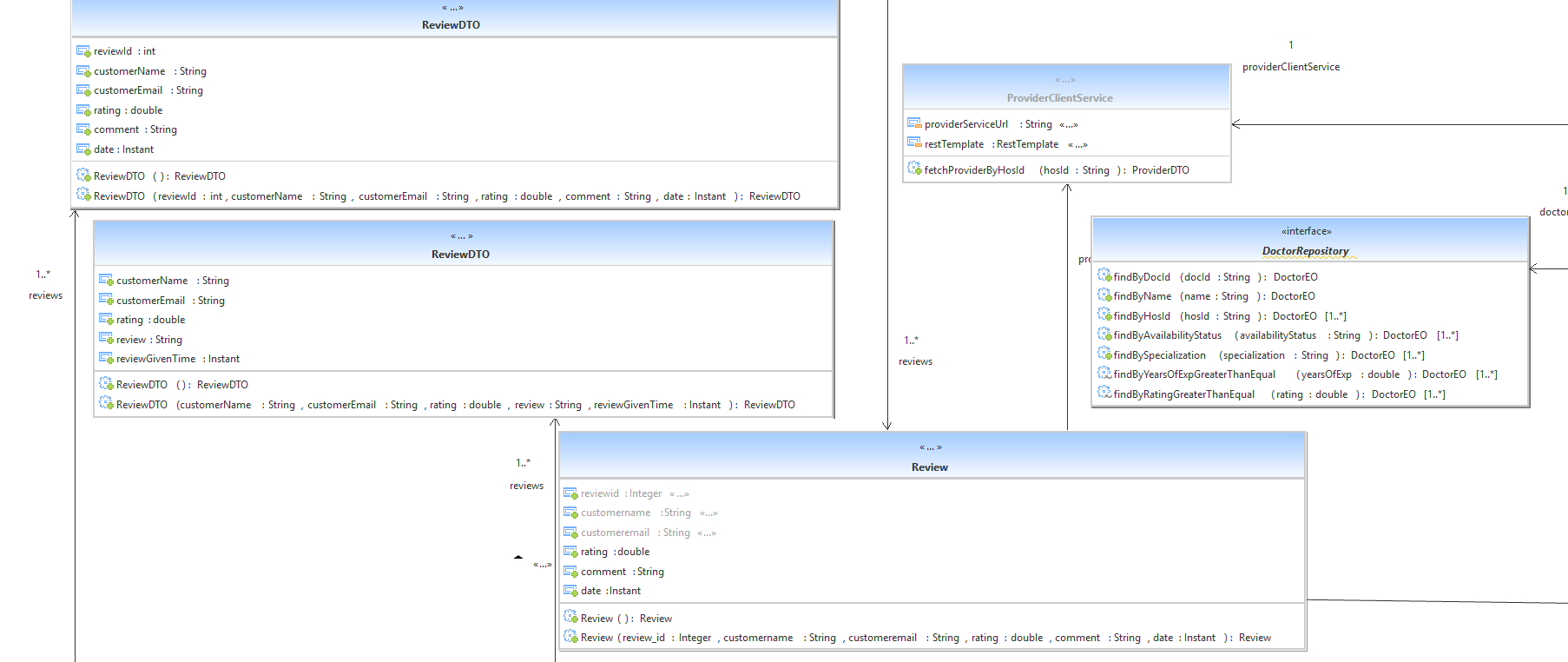
Not Applicable.

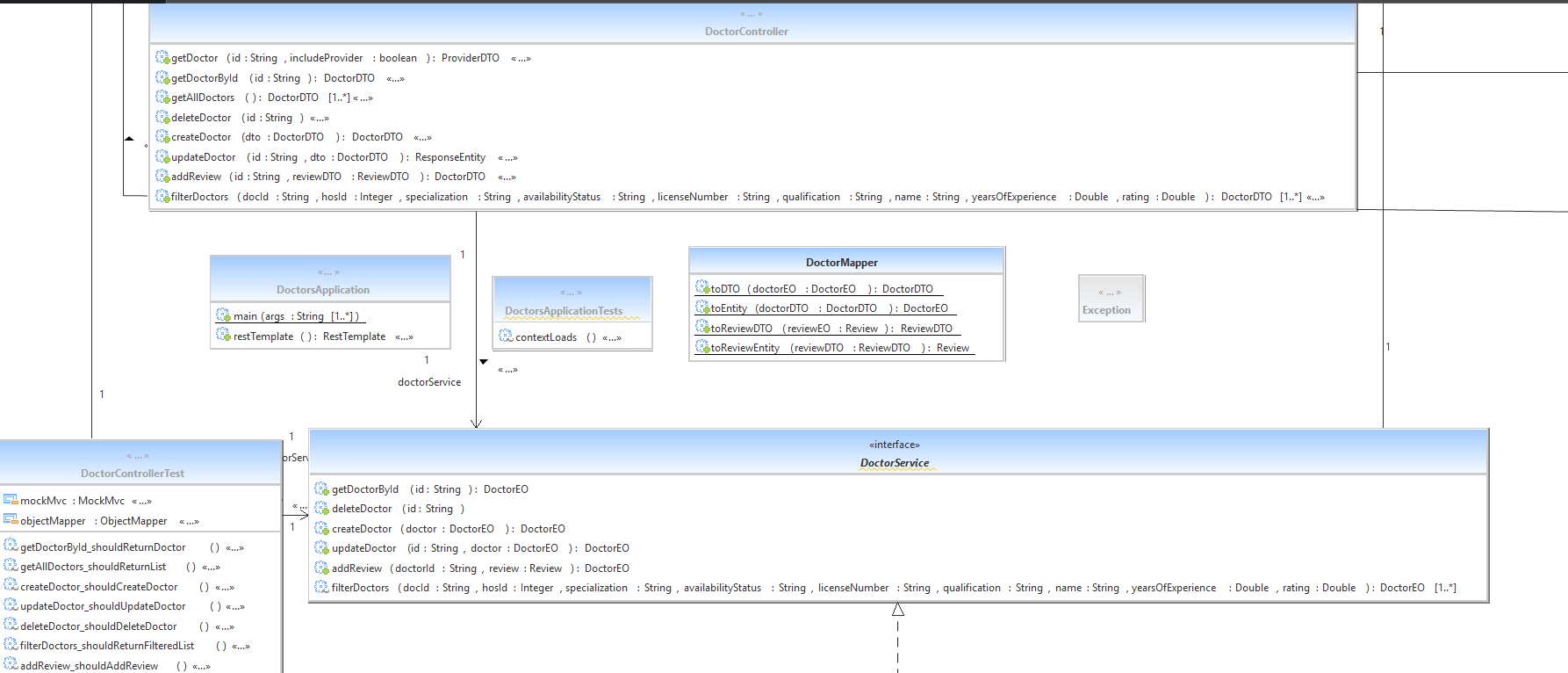
# 6. Design Overview

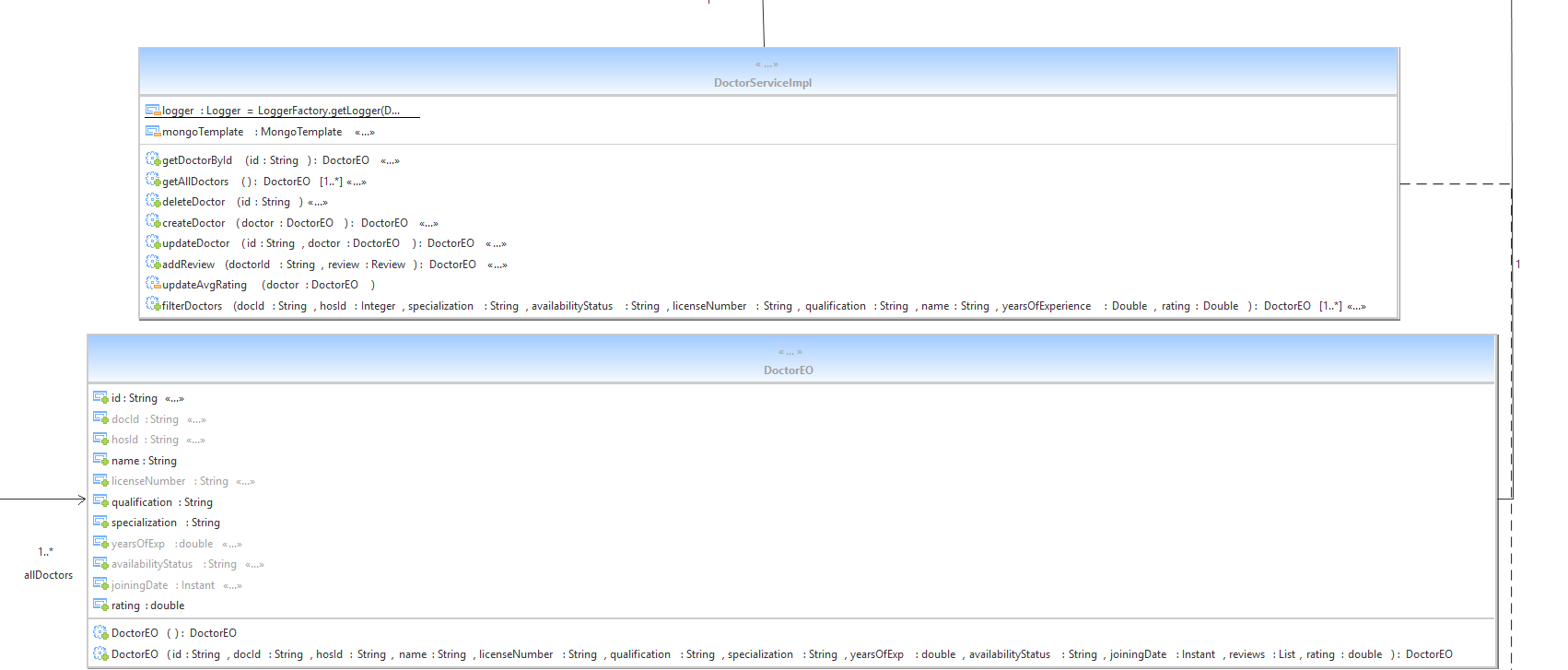
 provider Microservice :

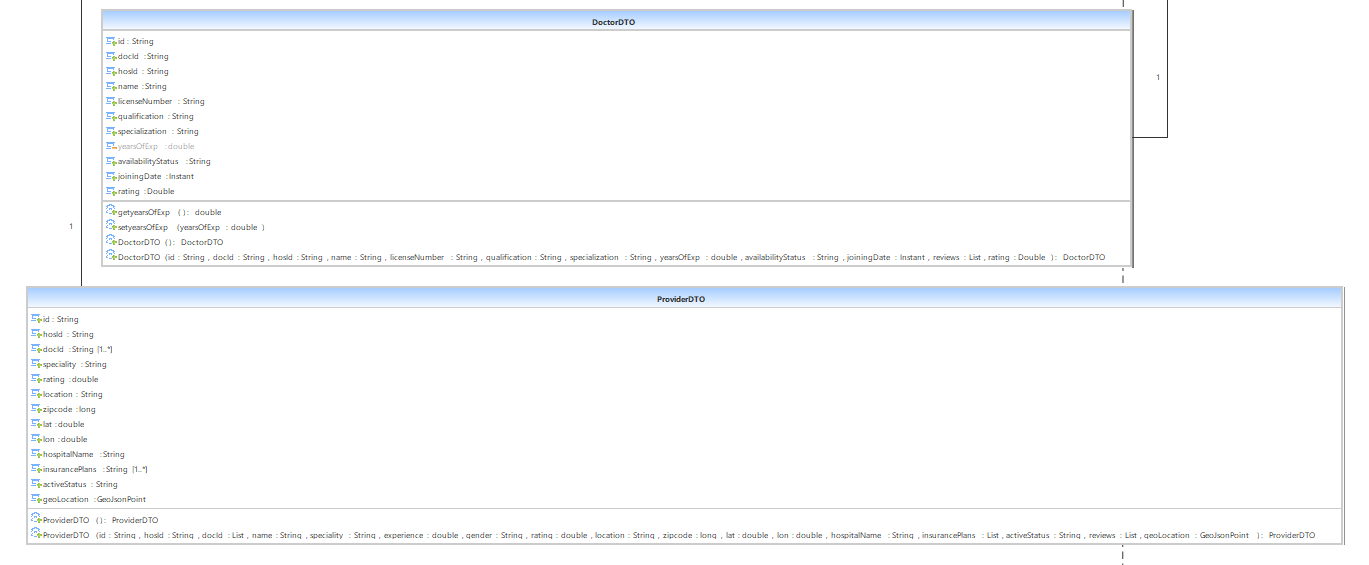


Doctor Microservice :

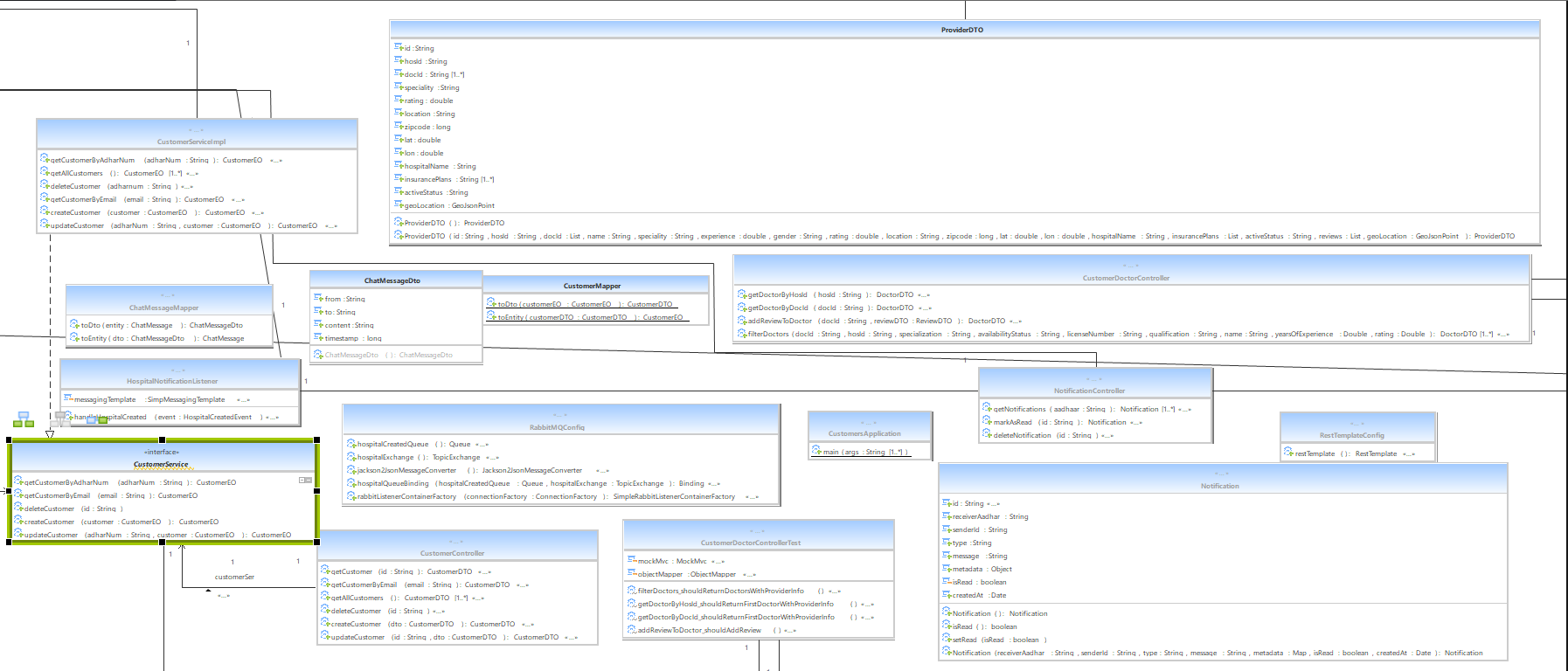
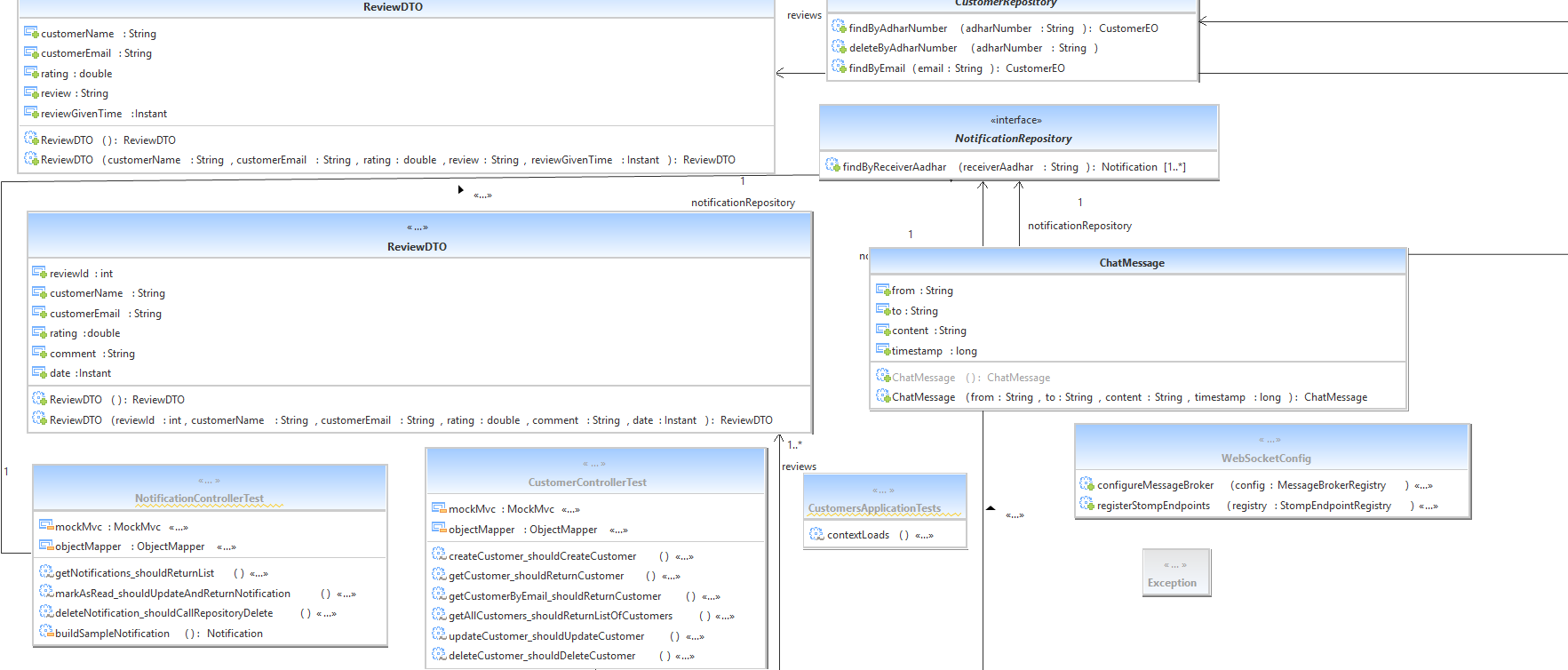








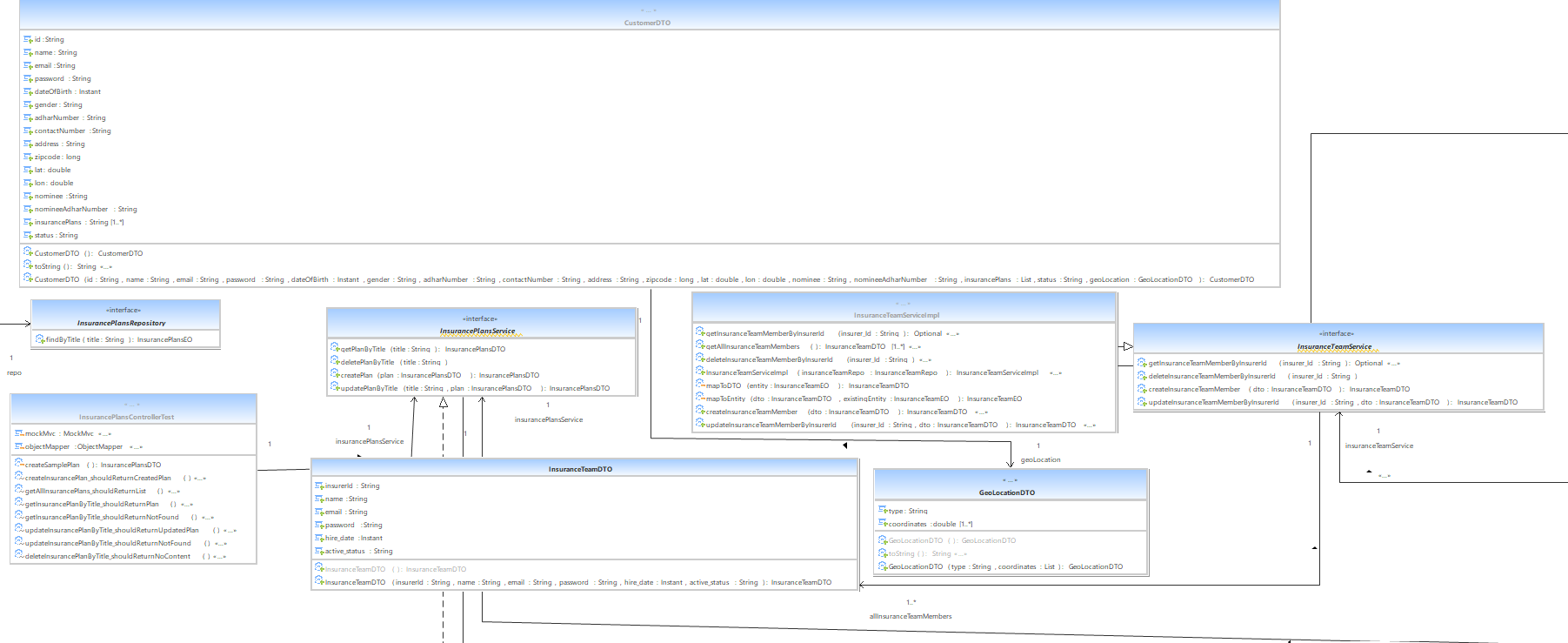
Customer Microservice :

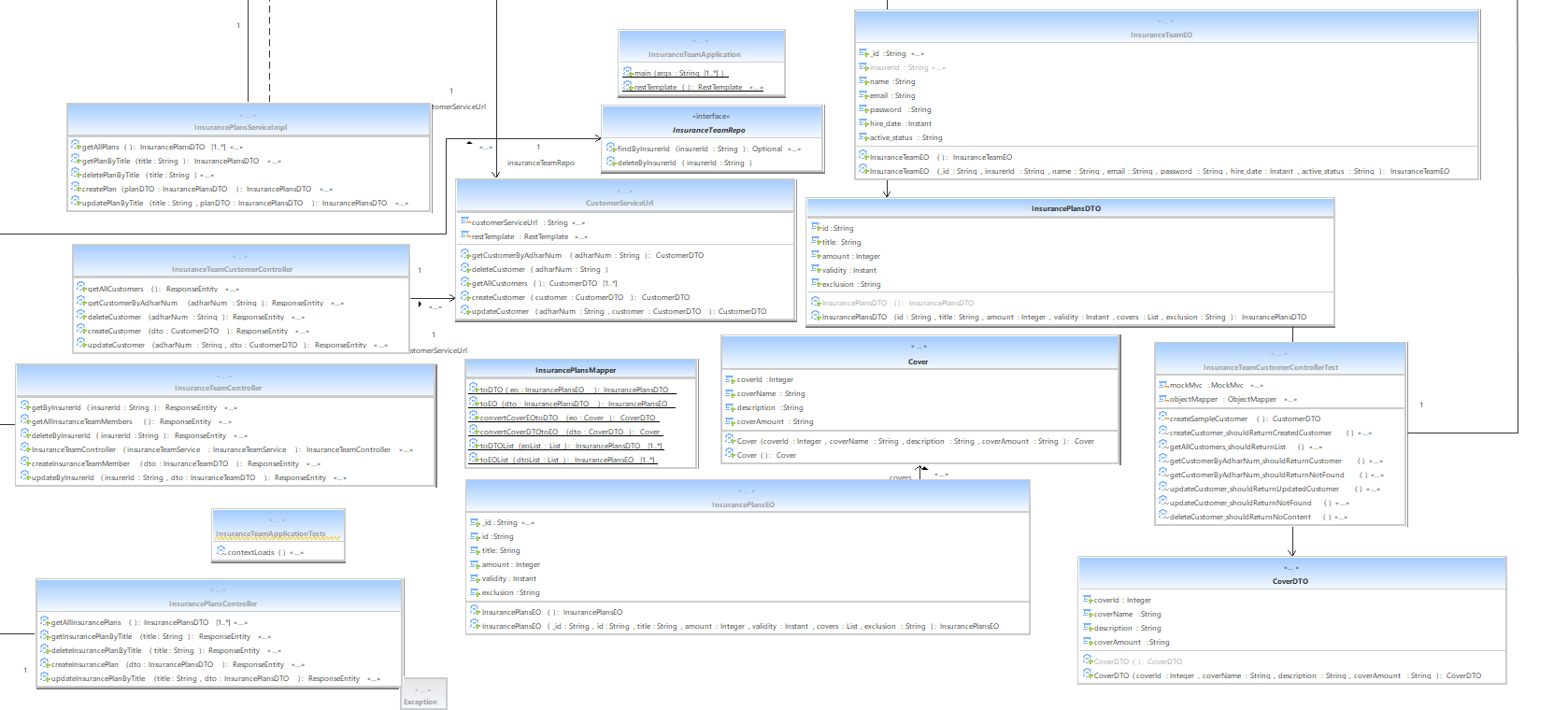


# 

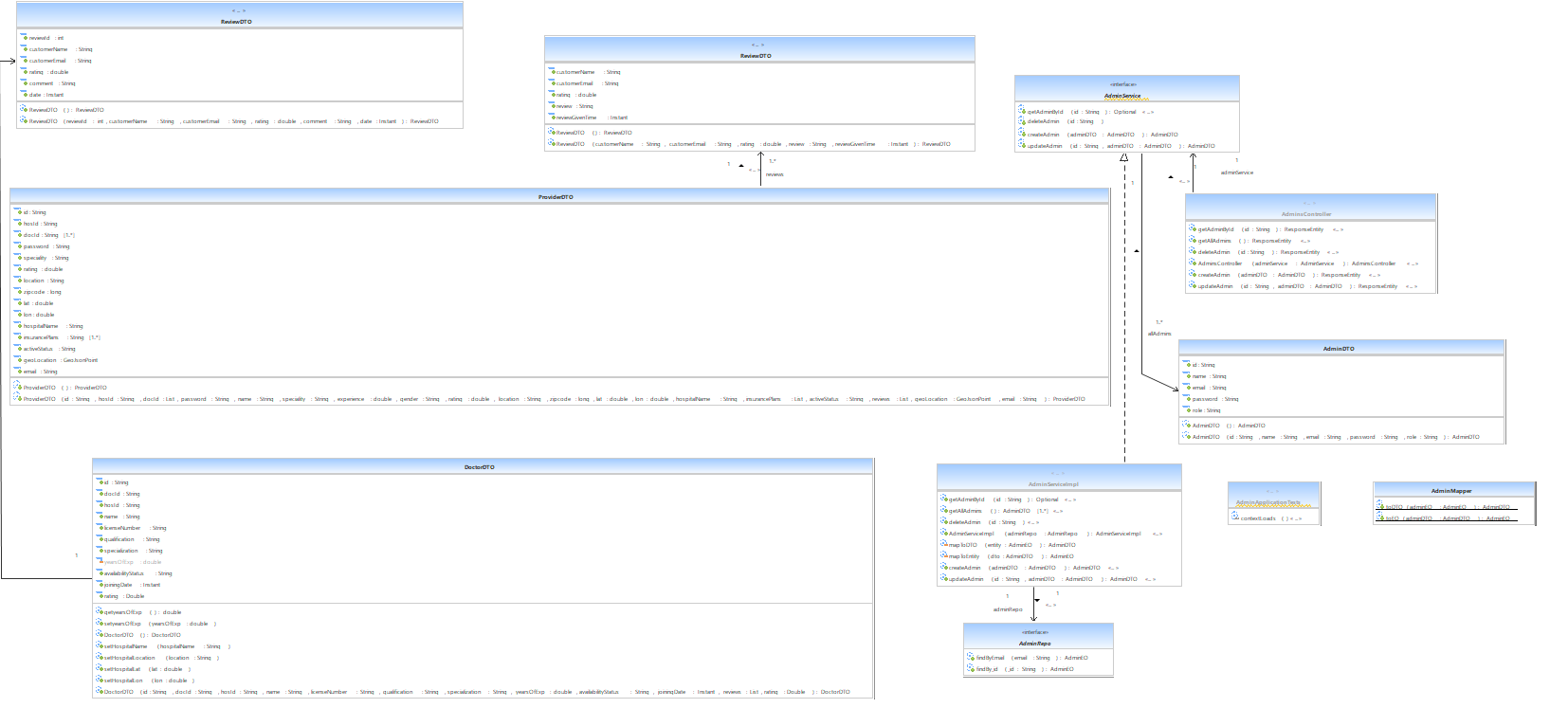
# 

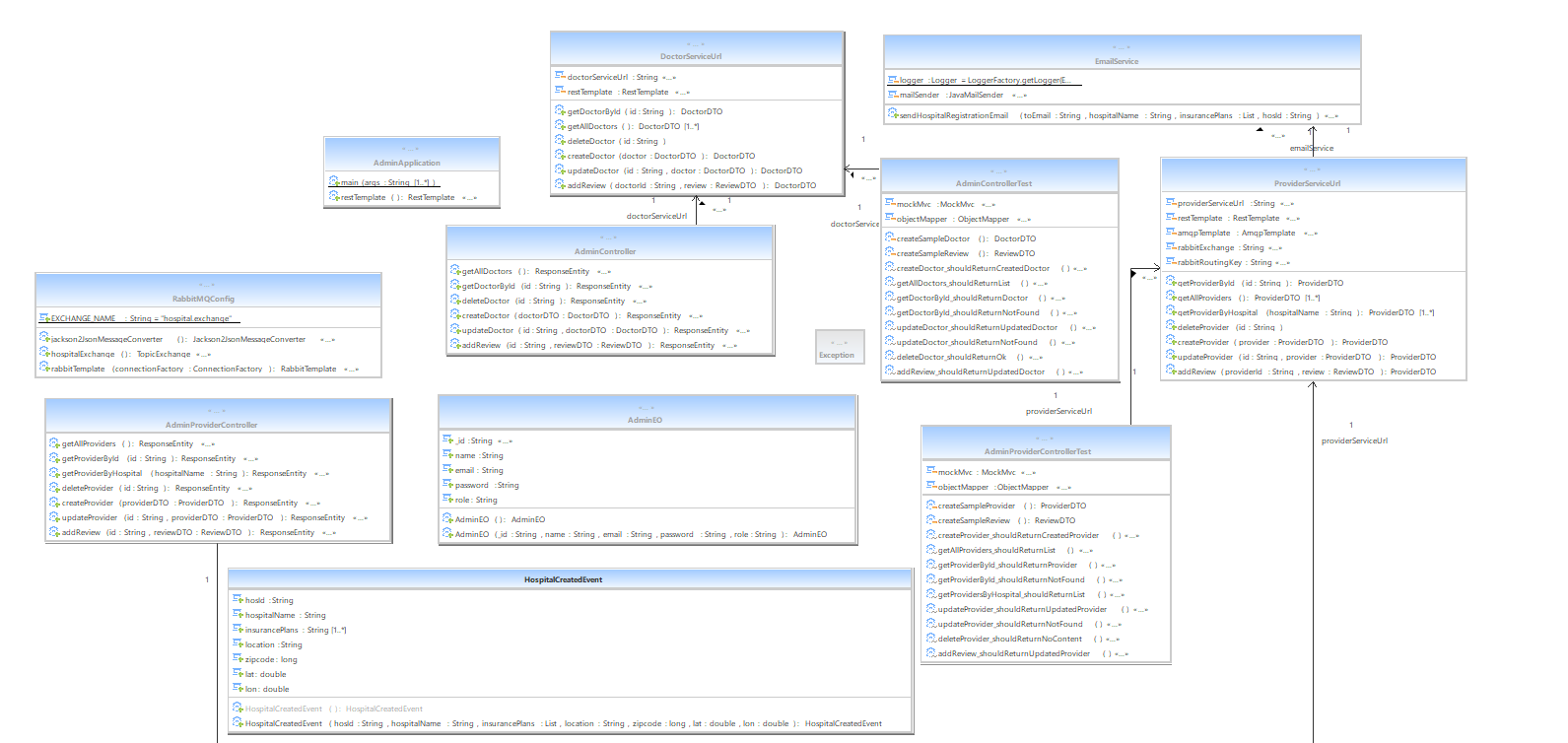
Insurance Team Microservice :





Admin MicroService :





Login Microservice :

# 

# 7. Decomposition

## DH-1-1 ProviderEO

* Inputs: Sets the Providers details .
* Outputs: Gets the Providers details.
* Scope: Specific
* Description :This class is a model class for the database table Providers.

## DH-1-2 DoctorEO

* Inputs: Sets the doctors details .
* Outputs: Gets the doctors details.
* Scope: Specific
* Description :This class is a model class for the database table Doctors.

## DH-1-3 CustomerEO

* Inputs: Sets the Customers details .
* Outputs: Gets the Customers details.
* Scope: Specific
* Description :This class is a model class for the database table Customer.

## DH-1-4 AdminEO

* Inputs: Sets the Admin details .
* Outputs: Gets the Admin details.
* Scope: Specific
* Description :This class is a model class for the database table Admin.

## DH-1-5 InsuranceTeamEO

* Inputs: Sets the InsuranceTeamEO details .
* Outputs: Gets the InsuranceTeamEO details.
* Scope: Specific
* Description :This class is a model class for the database table Insurance\_Team.

## DH-1-6 InsurancePlanEO

* Inputs: Sets the InsurancePlanEO details .
* Outputs: Gets the InsurancePlanEO details.
* Scope: Specific
* Description :This class is a model class for the database table Insurance\_Plans.

## DH-1-7 Notification

* Inputs: Sets the Notifications details .
* Outputs: Gets the Notification details.
* Scope: Specific
* Description :This class is a model class for the database table Notifications.

## DH-1-8 ProviderServiceImpl

* Inputs: Accepts provider details (name, hospital ID, location, GPS coordinates, speciality, insurance plans, ZIP code, reviews) and filter parameters.
* Outputs: Returns provider records, filtered search results, and updated ratings.
* Scope: Specific
* Description: Implements business logic for provider management using MongoDB, including CRUD, geolocation filtering, insurance filtering, and rating calculation.

This class implements the UserMasterDAO interface.

## DH-1-9 DoctorServiceImpl

* Inputs: Accepts doctor details (ID, hospital ID, license number, qualification, specialization, experience, status, reviews) and filter parameters.
* Outputs: Returns doctor records, filtered lists, and updated ratings.
* Scope: Specific
* Description: Handles CRUD and search operations for doctors with multi-criteria filtering, review aggregation, and MongoDB persistence.

## DH-1-10 CustomerServiceImpl

* Inputs: Accepts customer details (Aadhar number, name, email, personal info, nominee, insurance plans, location).
* Outputs: Returns customer records after creation, update, or retrieval.
* Scope: Specific
* Description: Provides CRUD operations for customer profiles with MongoDB persistence and geolocation data support.

## DH-1-11 LoginServiceImpl

* Inputs: Accepts user identifier and password for authentication.
* Outputs: Returns authenticated user object (Provider, InsuranceTeam, Admin, or Customer) or null if not found.
* Scope: Specific
* Description: Implements login and authentication by checking multiple user repositories for matching credentials.

## DH-1-12 OtpService

* Inputs: Accepts email for OTP generation and validation, along with the OTP code for verification.
* Outputs: Sends OTP email notifications; returns boolean success/failure for generate and validate operations.
* Scope: Specific
* Description: Handles OTP generation, email dispatch using JavaMailSender, OTP storage with expiry in multiple user entities, and validation for secure login flows.

## DH-1-13 AdminServieImpl

* Inputs: Accepts admin details (ID, name, email, password, role).
* Outputs: Returns admin details and lists of admins.
* Scope: Specific
* Description: Manages admin accounts with CRUD operations, mapping between DTOs and entities, and MongoDB persistence.

## DH-1-14 InsurancePlansServiceImpl

* Inputs: Accepts insurance plan DTOs and plan title for CRUD operations.
* Outputs: Returns created, updated, or fetched InsurancePlansDTO objects; deletes plans by title.
* Scope: Specific
* Description: Manages insurance plan data with DTO-to-entity mapping, supporting create, read, update, and delete operations via repository.

## DH-1-15 HospitalNotificationListener

* Inputs: Receives HospitalCreatedEvent messages from RabbitMQ queue.
* Outputs: Persists notifications to MongoDB and pushes real-time notifications via WebSocket to all customers.
* Scope: Specific
* Description: Listens to hospital creation events, constructs notification messages, stores notifications, and broadcasts them to subscribers

## DH-1-16 EmailService

* Inputs: Receives email address, hospital name, insurance plans, and hospital ID for sending registration confirmation emails.
* Outputs: Sends out email notifications asynchronously and logs email sending results.
* Scope: Specific
* Description: Provides asynchronous email notification service for hospital registration success using JavaMailSender with error handling and logging.

## DH-1-17 ProviderController

* Inputs: Accepts ProviderDTO bodies, path variables for provider IDs, filtering request parameters (with URL decoding), and provider reviews.
* Outputs: Returns ProviderDTO objects, lists, filtered sets, and related doctors; supports adding reviews.
* Scope: Specific
* Description: Provides REST API for provider entity management—CRUD operations, filtering with decoded parameters, related doctor fetching via service calls, and review additions.

## DH-1-18 DoctorController

* Inputs: Accepts DoctorDTO in request bodies, path variables for doctor IDs, and optional request parameters for filters and associations.
* Outputs: Returns DoctorDTO objects, lists of doctors, and provider info optionally; supports ResponseEntity for update status.
* Scope: Specific
* Description: Exposes doctor-related APIs for CRUD, filtering with URL-decoded parameters, adding reviews, and fetching linked providers, coordinating with service and client service.

## DH-1-19 CustomerController

* Inputs: Accepts CustomerDTO from request bodies and path/email parameters for retrieval, update, and deletion actions.
* Outputs: Returns CustomerDTO objects or void for deletions; lists of customers for bulk retrieval.
* Scope: Specific
* Description: REST controller managing customer API endpoints—handles create, read (by ID and email), update, delete, and list operations, using service layer and entity-DTO mapping.

## DH-1-20 NotificationController

* Inputs: Accepts customer Aadhaar number and notification ID path variables for retrieval, read marking, and deletion.
* Outputs: Returns lists of Notification objects, updated notification, or void upon deletion.
* Scope: Specific
* Description: REST controller for notification management—fetches customer notifications, marks as read, and deletes notifications via repository.

## DH-1-21 InsurancePlansController

* Inputs: Accepts InsurancePlansDTO in request bodies and plan title as path variables for specific operations.
* Outputs: Returns created, fetched, updated InsurancePlansDTO objects, lists of all plans, or HTTP status responses for delete and not found cases.
* Scope: Specific
* Description: REST controller that exposes CRUD endpoints for insurance plans, coordinating with the service layer for creation, retrieval by title or all, updates, and deletions.

## DH-1-22 LoginController

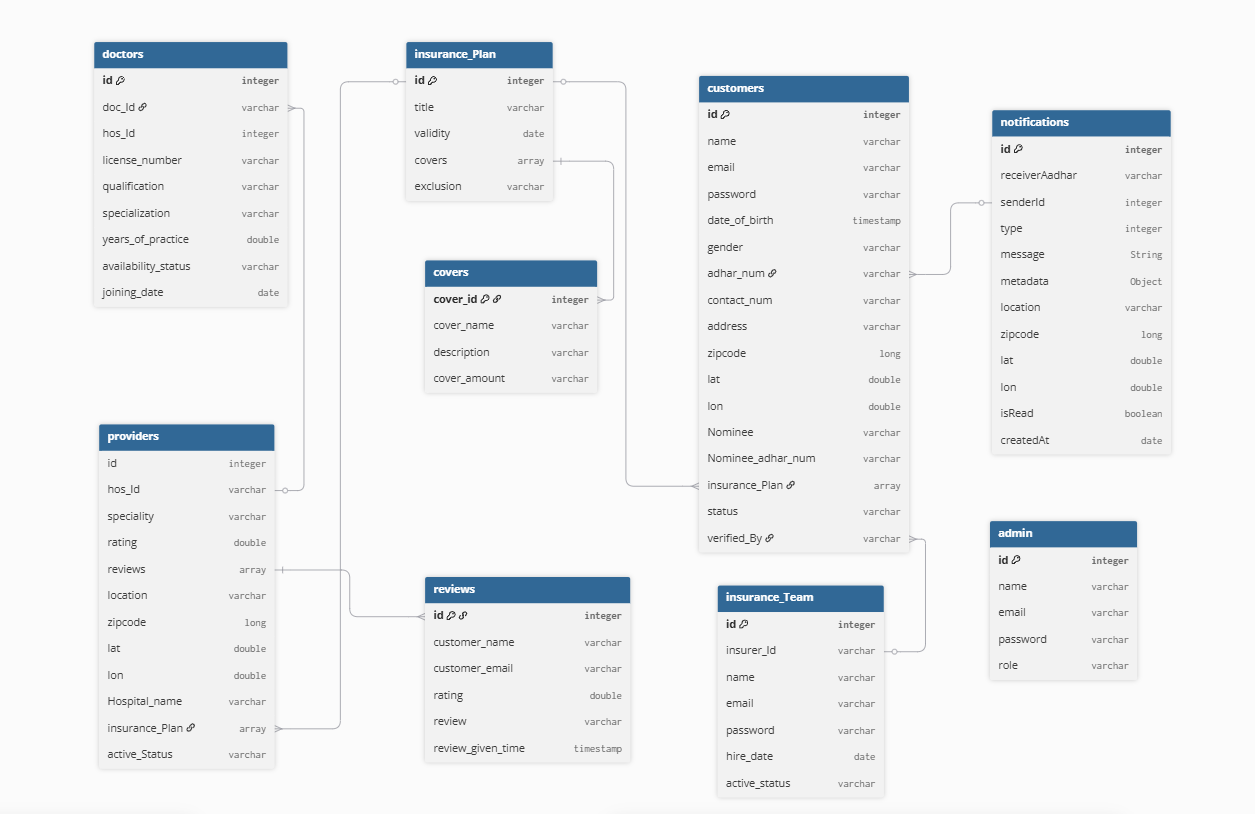
* Inputs: Accepts JSON bodies for login credentials, email for password reset, OTP validation data, and reset password details.
* Outputs: Returns authentication results, OTP generation status, validation results, password reset outcomes with HTTP status codes.
* Scope: Specific
* Description: Manages authentication endpoints including login, forgot password (OTP generation), OTP validation, and password reset for multiple user types, coordinating with services and repositories.

# 8. Interface Design

## 8.1 User Interface

NA

# 9. Data design



<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

# 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*](https://iqgateway-my.sharepoint.com/:w:/r/personal/roopesh_burra_iqgateway_com/_layouts/15/Doc.aspx?sourcedoc=%7B238D7732-48D4-442D-92B0-0FB43C05F303%7D&file=Provider_Finder%20-%20SRS.docx&action=default&mobileredirect=true&wdOrigin=APPHOME-WEB.DIRECT%2CAPPHOME-WEB.BANNER.UPLOAD&wdPreviousSession=7967203b-4ea7-42d0-8250-5d2006e4f004&wdPreviousSessionSrc=AppHomeWeb&ct=1755282420127) |
| 2. | Low Level Design – Provider Finder & Network Lookup | 10/08/2025 | 1.0 | [LLD](https://iqgateway-my.sharepoint.com/:w:/r/personal/roopesh_burra_iqgateway_com/_layouts/15/Doc.aspx?sourcedoc=%7B8971D731-8839-40E8-909F-2F60B836B8B5%7D&file=Provider_Finder%20-%20LLD.docx&action=default&mobileredirect=true&wdOrigin=APPHOME-WEB.DIRECT%2CAPPHOME-WEB.BANNER.UPLOAD&wdPreviousSession=7967203b-4ea7-42d0-8250-5d2006e4f004&wdPreviousSessionSrc=AppHomeWeb&ct=1755282757179) |
| 3. | Provider Finder & Network Lookup/Readme | 11/08/2025 | 1.0 | [ReadMe](https://iqgateway-my.sharepoint.com/:w:/r/personal/roopesh_burra_iqgateway_com/_layouts/15/Doc.aspx?sourcedoc=%7B48D45E4F-76BA-431A-8A0D-9EE2C8B4E97F%7D&file=Provider_Finder%20-%20Readme.docx&action=default&mobileredirect=true&wdOrigin=APPHOME-WEB.DIRECT%2CAPPHOME-WEB.BANNER.UPLOAD&wdPreviousSession=7967203b-4ea7-42d0-8250-5d2006e4f004&wdPreviousSessionSrc=AppHomeWeb&ct=1755282773025) |

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