Software Design Specifications

for

Machine Learning Based Heart Stroke Risk Prediction System

Prepared by:

Team 7
Members:

Challa Shreyas Reddy (SE22UCSE068) – Frontend

Challa Shishir Reddy (SE22UCSE067) – Full Stack

B.N. Varun(SE22UCSE042) – Frontend

Dakoji Praneeth (SE22UCSE074) – Backend

B. Nihaal Patnaik(SE22UCSE050) – Backend

B.V.V. Nitesh Reddy(SE22UCSE046) – Project Manager

Document Information

Title:	
Project Manager:Nitesh	**Document Version No: 2**
	Document Version Date: 07/05/2025(DD/MM/YYYY)
Prepared By:Team 7	**Preparation Date:**

Version History

Ver. No.	**Ver. Date**	**Revised By**	**Description**	**Filename**
1	08/04/2025	Team 07	First version of Software Design Specification	
2	07/05/2025	Team 07	Second version of Software Design Specification	

Table of Contents

1 Introduction

1.1 Purpose

This document describes the detailed design of the Heart Stroke Risk Assessment System. It is intended for developers, testers, system architects, and project stakeholders. It elaborates on system components, design models, external interfaces, and non-functional considerations, and ensures traceability to the SRS.

1.2 Scope

This document covers the full design blueprint of the web-based heart stroke risk assessment platform, including backend logic, data storage, security protocols, and the integration of ML models for risk prediction.

1.3 Definitions, Acronyms, and Abbreviations

- SRS - *Software Requirements Specification*, a document that outlines the functional and non-functional requirements of the system.
- ML - *Machine Learning*, a subset of artificial intelligence that uses data-driven models to make predictions or decisions.
- DPDP Act - *Digital Personal Data Protection Act*, a legal framework in India governing personal data collection, processing, and storage.
- HDM Policy - *Health Data Management Policy*, a guideline under the Ayushman Bharat Digital Mission for the handling of health-related data.
- ABDM - *Ayushman Bharat Digital Mission*, a government initiative aimed at digitizing healthcare infrastructure and systems.
- OTP - *One-Time Password*, a temporary password used for user authentication.
- API - *Application Programming Interface*, a set of functions that allows different software components to communicate.
- AWS - *Amazon Web Services*, a cloud computing platform used for scalable data storage and processing.
- GCP - *Google Cloud Platform*, another widely used cloud service provider.
- S3 - *Simple Storage Service*, a cloud storage solution provided by AWS.

- MFA - *Multi-Factor Authentication*, a security process that requires two or more methods of identity verification.
- REST APIs - *Representational State Transfer Application Programming Interfaces*, which allow stateless communication between systems over HTTP.
- Django ORM - *Django ORM is a powerful tool that allows you to interact with your database using Python code instead of writing raw SQL queries. It maps Python classes (models) to database tables, and Python objects to rows in those tables.*
- AES-256 - *Advanced Encryption Standard with a 256-bit key*, widely used for encrypting sensitive data.
- SSL/TLS - *Secure Sockets Layer* and *Transport Layer Security*, protocols used to secure data transmitted over the internet.

1.4 References

Digital Personal Data Protection Act, 2023 (India)
https://www.meity.gov.in/data-protection-framework

OWASP Security Guidelines
https://owasp.org/www-project-top-ten/

Health Data Management Policy – Ayushman Bharat Digital Mission

Django Documentation
https://docs.djangoproject.com

scikit-learn Documentation
https://scikit-learn.org/stable/documentation.html

joblib Documentation
https://joblib.readthedocs.io/en/latest/

AWS Documentation (S3)
https://docs.aws.amazon.com/s3/

PostgreSQL Documentation
https://www.postgresql.org/docs/

2 Use Case View

2.1 Use Case

- User Registration and Login
- Input Health and Lifestyle Data
- Stroke Risk Prediction
- Generate and Review Recommendations
- Send Emergency Alert
- View and Download Health Reports

Refer to SRS Section 3.3 (page 6) for more information.

3 Design Overview

3.1 Design Goals and Constraints

- The system should conform to The Digital Personal Data Protection Act, 2023 (DPDP Act) and Health Data Management (HDM) Policy under the Ayushman Bharat Digital Mission (ABDM) to provide secure processing and storage of personal health data.
- Machine Learning Model Choice: Demands highly accurate and low-bias trained AI models.
- Real-time prediction capability
- Multi-device accessibility

3.2 Design Assumptions

- User Accessibility: The user will be provided with a computer or smartphone with an internet connection to use for entering health information and obtaining risk assessments.
- Data Availability: The system takes it for granted that current and accurate patient health information is available to input, either by user or through linkage from healthcare databases.
- Periodic model updates: The models will require periodic retraining to stay accurate as new medical research and patient information become available.
- User Compliance: Users follow system recommendations

3.3 Significant Design Packages

- auth – Handles user registration, login, OTP verification
- Risk Engine – Interfaces with ML model to assess stroke risk
- data_manager – Manages user input and report storage

3.4 Dependent External Interfaces

The table below lists the public interfaces this design requires from other modules or applications.

External Application and Interface Name	**Module Using the Interface**	**Functionality/ Description**
Authentication Service	OTP Verification Gmail SMTP	Used during login/registration/forgot password for authentication using email-based OTP.
Django Admin Console	Admin User Interface	Allows admin to manage users, view prediction history, and monitor data through Django admin.
Django ORM	Prediction History and Users	Stores structured stroke prediction data and user details.

3.5 Implemented Application External Interfaces (and SOA web services)

The table below lists the implementation of public interfaces this design makes available for other applications.

Interface Name	**Module Implementing the Interface**	**Functionality/ Description**
predict_risk	stroke_risk.views.assessment_step3	Accepts health data via POST request, invokes the ML model (stroke_model.pkl), and returns stroke risk predictions.
register	stroke_risk.views.register_view	Handles new user registration using RegisterForm, password hashing, and sends OTP via Gmail SMTP.
login	stroke_risk.views.longin_view + otp_login_view	Authenticates users with credentials and OTP. Session-based login flow is followed after OTP verification.
input_health	stroke_risk.views.assessment_step1/2/3 + form.StrokeRiskForm	Collects user health and lifestyle data in a multi-step process and preprocesses it for ML model input.
get_report	stroke_risk.views.history_view	Returns a downloadable PDF report of stroke risk predictions.

4 Logical View

4.1 Design Model

This section defines the high-level module structure. The backend will be structured as a Django application with modular apps such as:

This section defines the high-level module structure. The backend is structured as a Django application with the following integrated functional modules:

- Authentication
  Handles user registration, login, OTP verification, password reset, and session-based authentication using Django's built-in user model and Gmail SMTP.
- Multi-Step Stroke Prediction
  Implements a 3-step health assessment workflow. Collected data is preprocessed and passed to a trained machine learning model (stroke_model.pkl) to predict stroke risk.

- User Data Management
  Stores user profiles and prediction history securely using Django ORM and the PredictionHistory model. Enables tracking and retrieval of past prediction results.

- Health Recommendation Interface
  Displays dynamic recommendations on the result page based on user BMI, glucose levels, and blood pressure status. No dedicated engine or ML model is used.

- Notification Service
  Sends email-based OTPs via Gmail SMTP during login and password reset flows. Also supports OTP expiry and resend features.

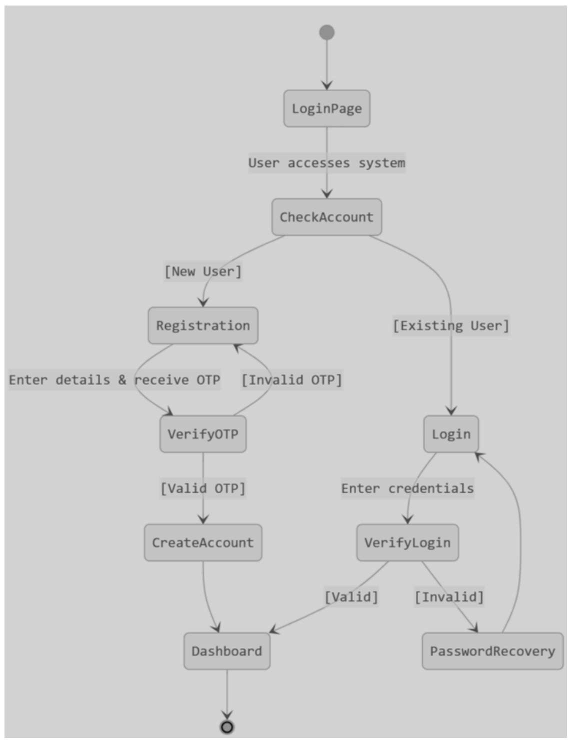
The frontend is implemented using Django templates (HTML/CSS) with minimal JavaScript for dark mode toggling, OTP handling, and risk chart visualization. It communicates with the backend using secure form submissions and Django session management.

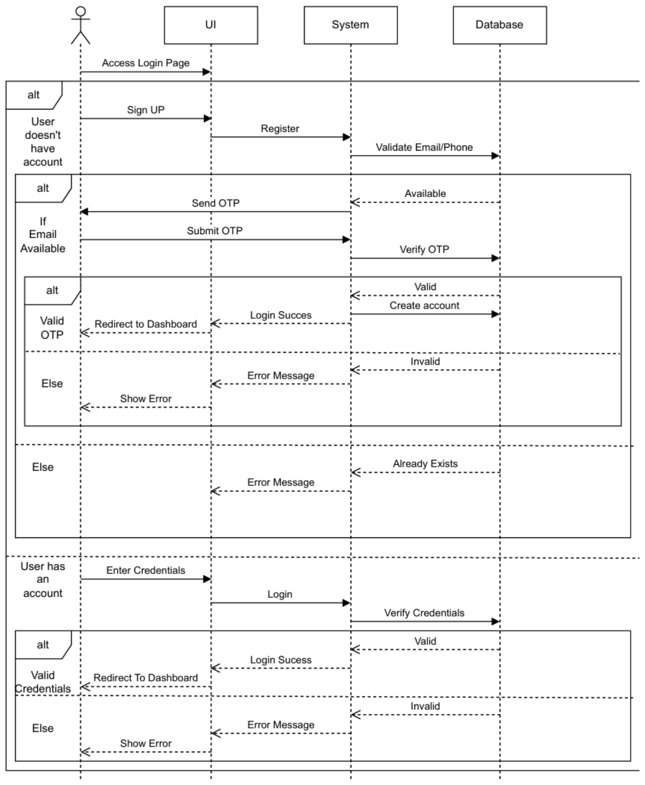
4.2 Use Case Realization

Refer to SRS Section 3.3 (page 6) for the complete step-by-step flows.

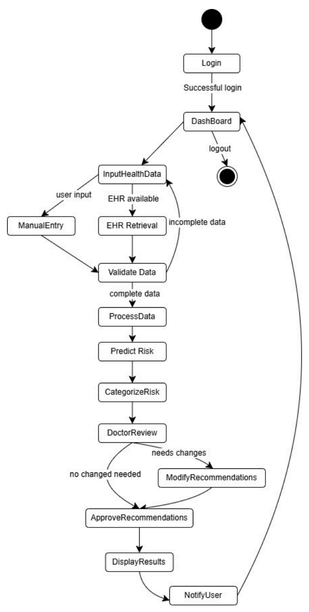
Below are the diagrams for the Use Cases:

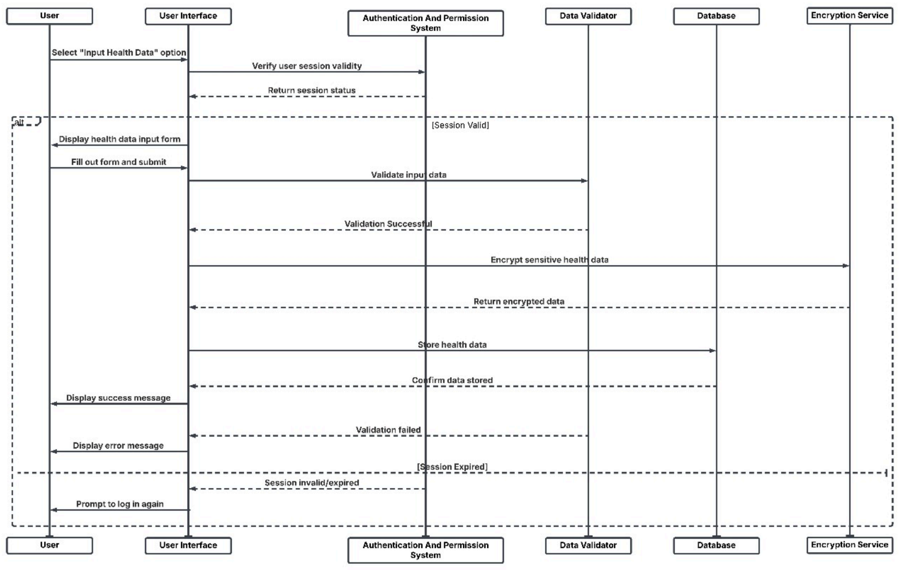
Use Case: User Registration and Login



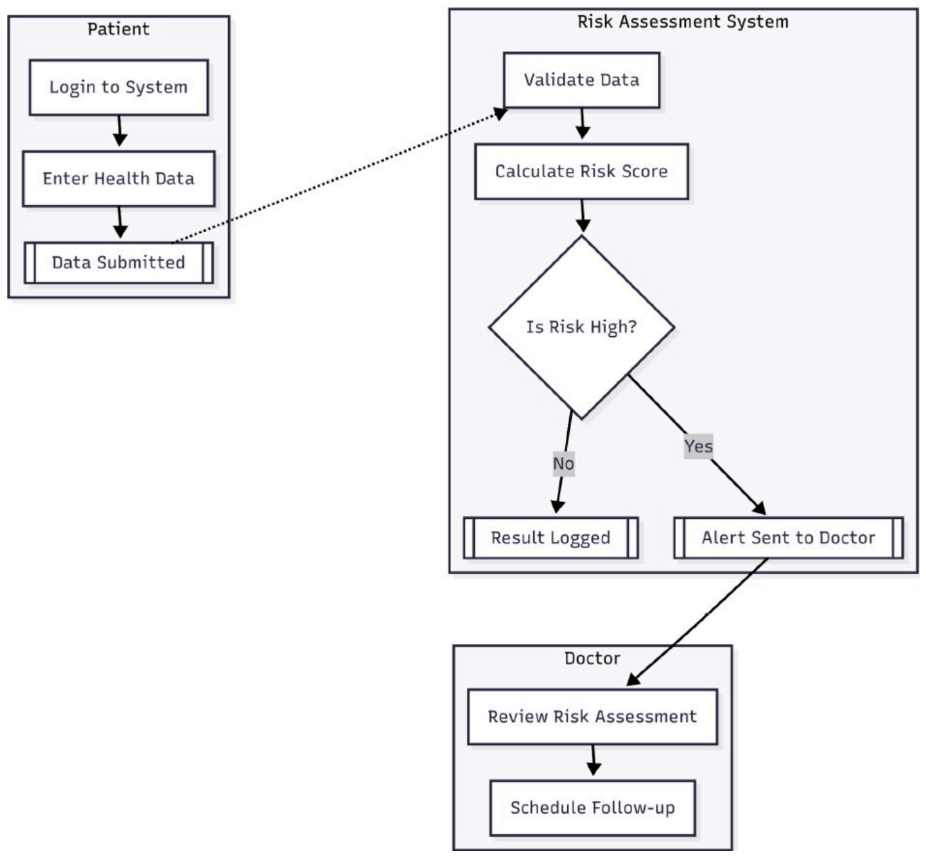


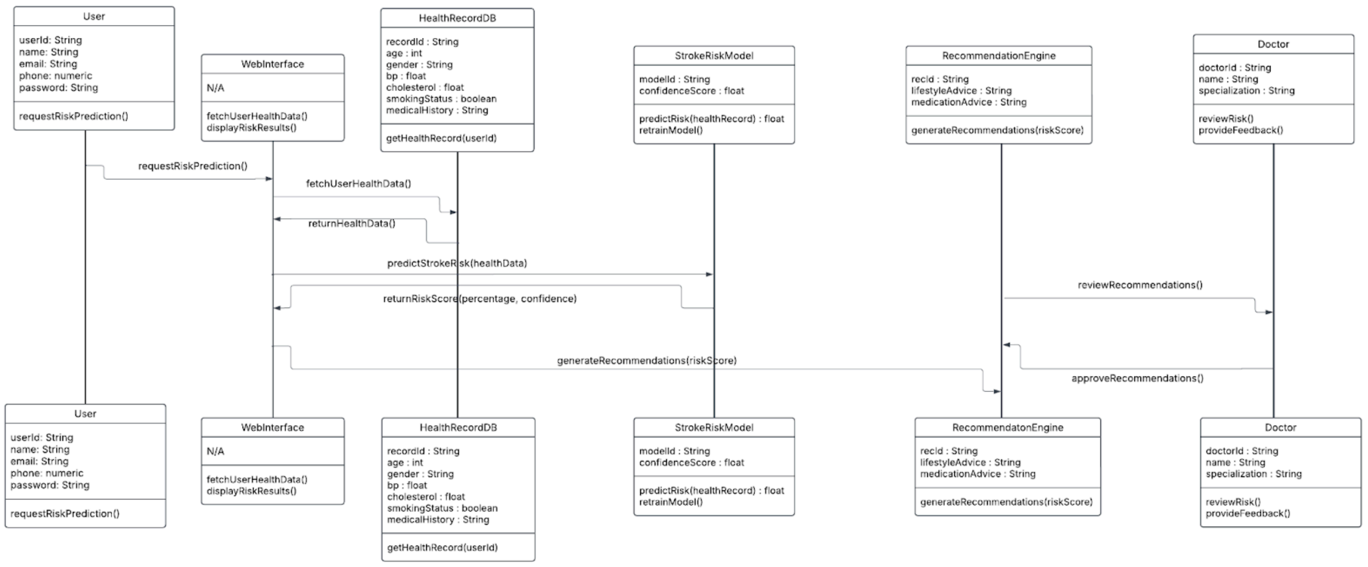
Use Case: Input Health and Lifestyle Data



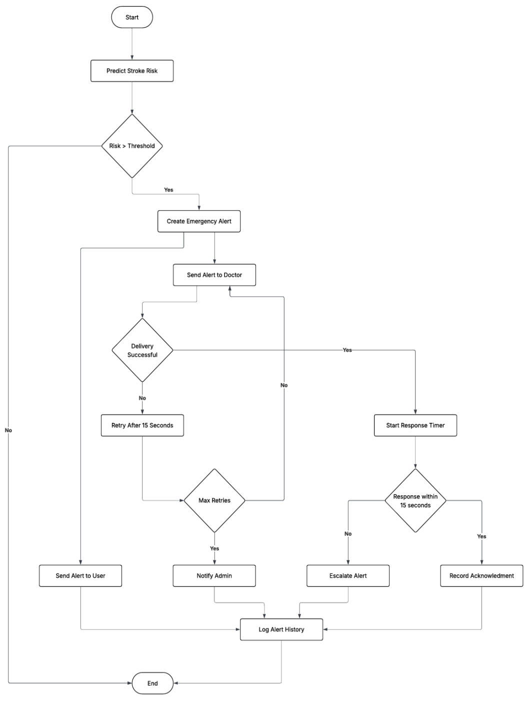


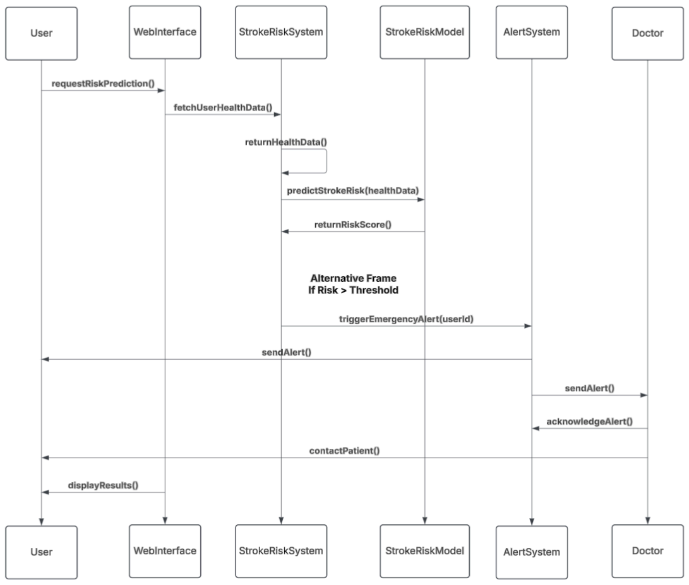
Use Case: Stroke Risk Prediction



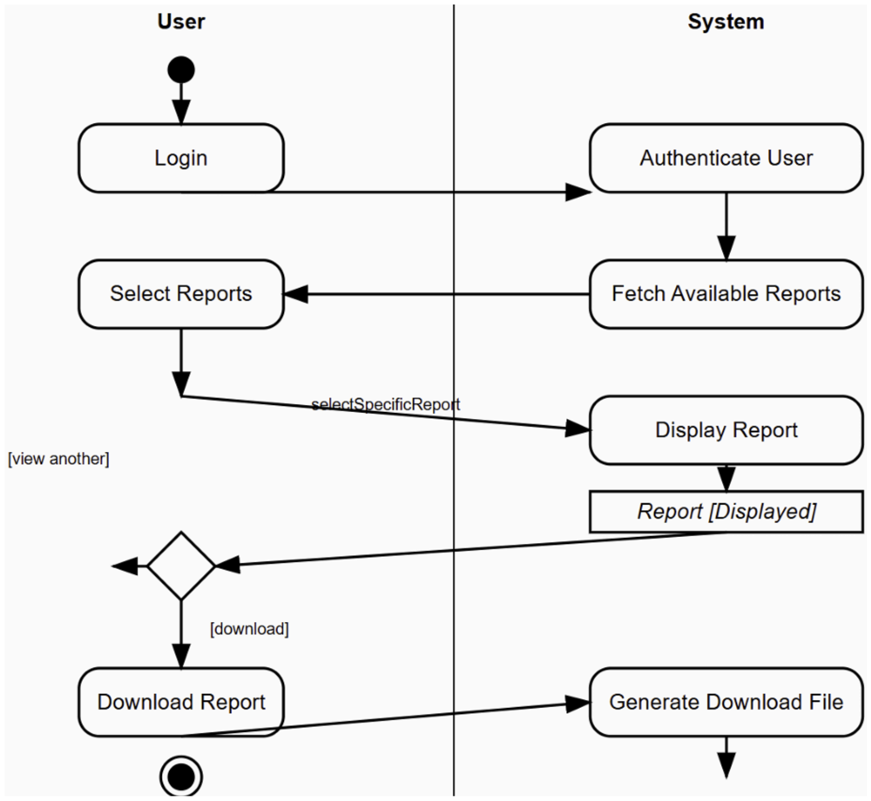


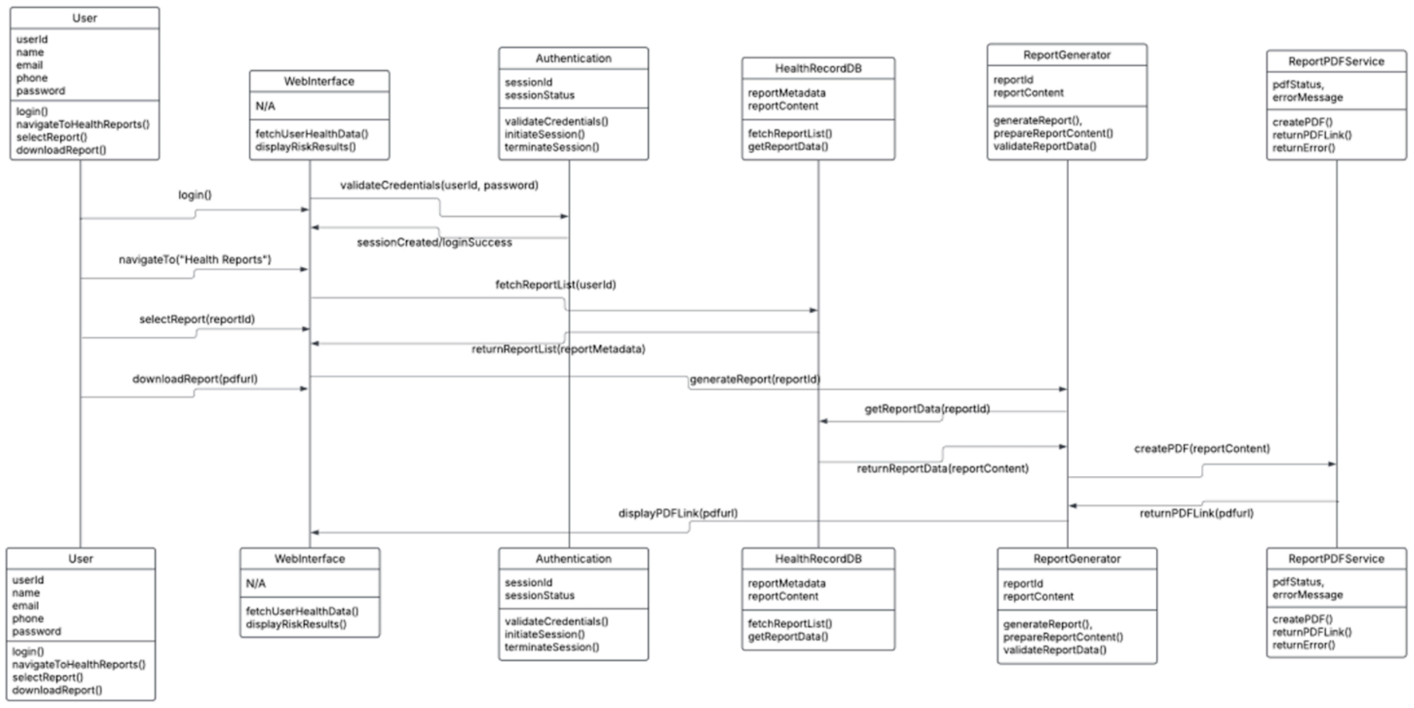
Use Case: Send Emergency Alert





Use Case: View and Download Health Reports





5 Data View

5.1 Domain Model

The core entities of the system and their relationships are outlined below. These entities represent persistent data in the system:

Primary Entities:

1. **User**
   - Attributes: user_id, name, email, password_hash
   - Relationships: One-to-Many with HealthRecord, PredictionHistory
2. **PredictionHistory**
   - Attributes: prediction_id, user_id, risk_score, risk_category, timestamp
   - Relationships: Belongs to User

5.2 Data Model (persistent data view)

Data is managed using Django's ORM, which handles all database interactions through defined models. The current implementation stores user information and prediction history using Django's default development database. Sensitive fields like passwords are securely handled through Django's built-in hashing and session mechanisms. No custom encryption is applied.

5.2.1 Data Dictionary

Name	Type	Category	States / Values	Description	Related Operations / Requirements
`user_id`	Integer	State Variable	Auto-generated	Unique identifier for each user	Used for linking predictions to user
`email`	String	Input	Valid email format	User's email address	Used for registration, login, and OTP delivery

`password_hash`	String	Constant	Encrypted (Django-managed)	Secure password storage	Used for login; never stored in plaintext
`age`	Integer	Input	≥ 0	User’s age	Input for ML model
`gender`	Enum	Input	Male, Female, Other	User’s gender	Input for ML model
`hypertension`	Boolean	Input	0 (No), 1 (Yes)	Whether user has hypertension	Input for prediction
`heart_disease`	Boolean	Input	0 (No), 1 (Yes)	Heart disease status	Input for prediction
`ever_married`	Boolean	Input	0 (No), 1 (Yes)	Marital status	Input for socio-demographic correlation
`work_type`	Enum	Input	Private, Self-employed, Govt_job, etc.	Type of employment	Used in ML model
`residence_type`	Enum	Input	Urban, Rural	Type of living environment	Used in ML model
`avg_glucose_level`	Float	Input	Positive float	Average glucose level	Critical ML input
`bmi`	Float	Input	Calculated or entered	Body Mass Index	Used in ML and recommendation
`smoking_status`	Enum	Input	never smoked, formerly smoked, smokes, Unknown	Smoking behavior	Input feature for ML

`result_percent`	Float	Output	0.00 to 100.00	Stroke risk prediction	Displayed to user
`result_label`	Enum	Output	Low Risk, Moderate Risk, High Risk	Risk category	Based on `result_percent` threshold
`created_at`	DateTime	State Variable	Auto-generated	Timestamp of prediction	Used in history view
`recommendations`	List[String]	Output	Free-text list	Static lifestyle advice (rendered, not stored)	Shown post-prediction
`otp` (in memory)	Integer	State Variable	6-digit integer	OTP for authentication	Expires in 2 minutes; not stored in DB

6 Exception Handling

- **Login failures** → Display appropriate error messages (e.g., "Invalid username or password").
- **OTP verification fails** → User is shown "Invalid or expired OTP"; retry possible; supports resend functionality via AJAX.
- **ML model failure** → Displays error message on result page (handled in try/ecpect block); form is re-rendered with explanation.
- **Unexpected server errors** → Session is cleared; user is redirected to login.

7 Configurable Parameters

This table describes the simple configurable parameters (name/value pairs).

Configuration Parameter Name	Definition and Usage	Dynamic?
SESSION_TIMEOUT	Logout after 2 mins of inactivity	Yes
MLERT_THRESHOLD	Stroke risk percentage triggering alerts (e.g., 80%)	Yes
RETRAIN_INTERVAL	Time interval for model retraining (e.g., 30 days)	No

8 Quality of Service

8.1 Availability

- Application runs locally for demo and development purposes.

8.2 Security and Authorization

- **Authentication:** Email + OTP-based login and password reset provide basic verification.
- **Session Management:** Secure sessions are used to maintain login state.
- **Data Protection:** Passwords are hashed via Django's default user model, but no custom encryption (e.g., AES) is applied.
- **Compliance:** Not aligned with DPDP Act or ABDM regulations; current version is academic and not ready for real-world deployment.

8.3 Load and Performance Implications

- System is designed for **single-user interaction** at a time; concurrency is not implemented or tested.
- ML model prediction completes in under **3 seconds** in local execution.
- No use of caching layers or asynchronous queues; all requests are synchronous.

8.4 Monitoring and Control

- **Prediction logs** are stored per user via the PredictionHistory model.
- No monitoring for API response times, system uptime, or error tracking is implemented.
- The system uses the built-in Django admin dashboard for managing users and viewing prediction records.