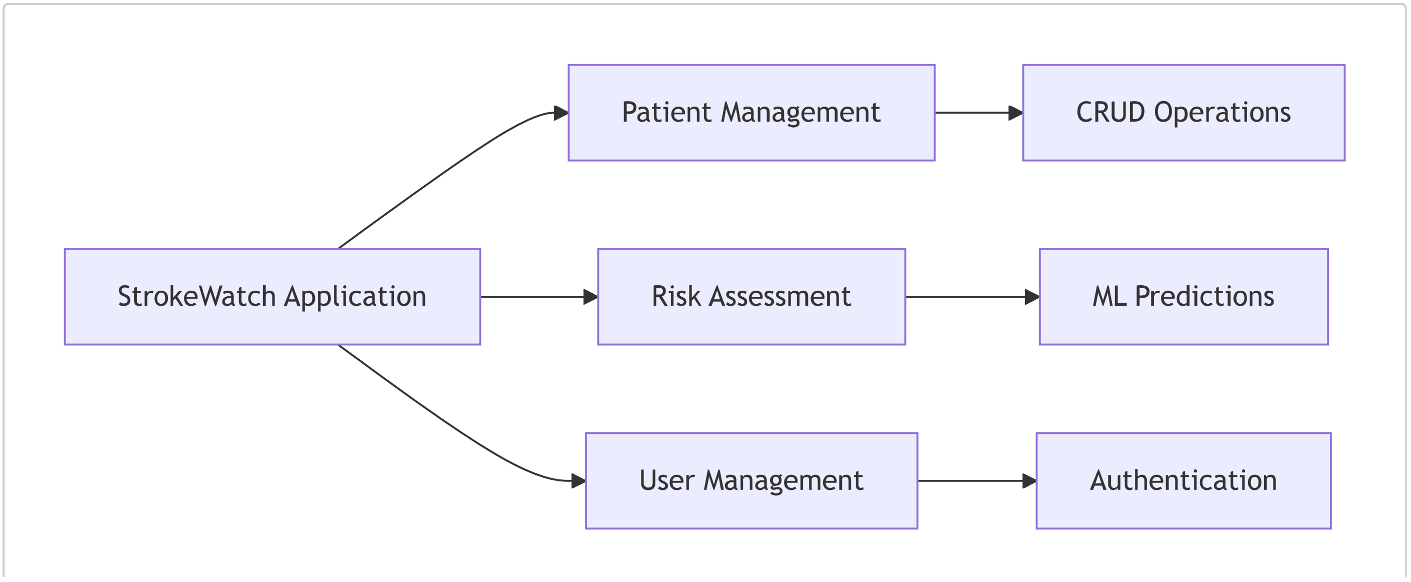
🏥 StrokeWatch Documentation

Comprehensive System Documentation & Testing Guide

📑 Table of Contents

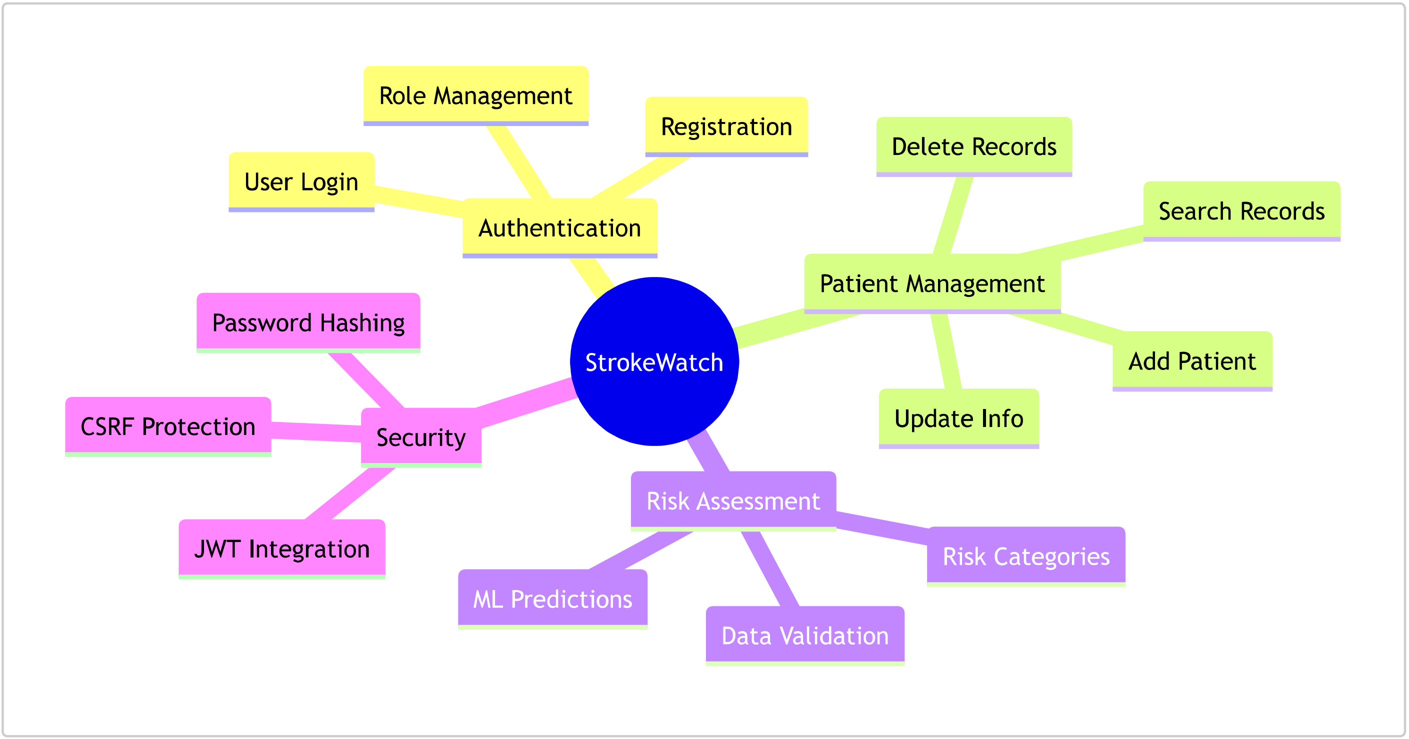
1. Project Overview

1.1 Introduction



StrokeWatch combines web technologies with machine learning for healthcare stroke risk assessment and patient management.

1.2 Core Features



1.3 Tech Stack Overview

Backend Components

- 🔷 Flask 3.1.0
- 🔶 SQLAlchemy 3.1.1
- 🔷 MongoEngine 0.29.1
- 🔶 TensorFlow 2.18.0
- 🔷 Keras 3.6.0

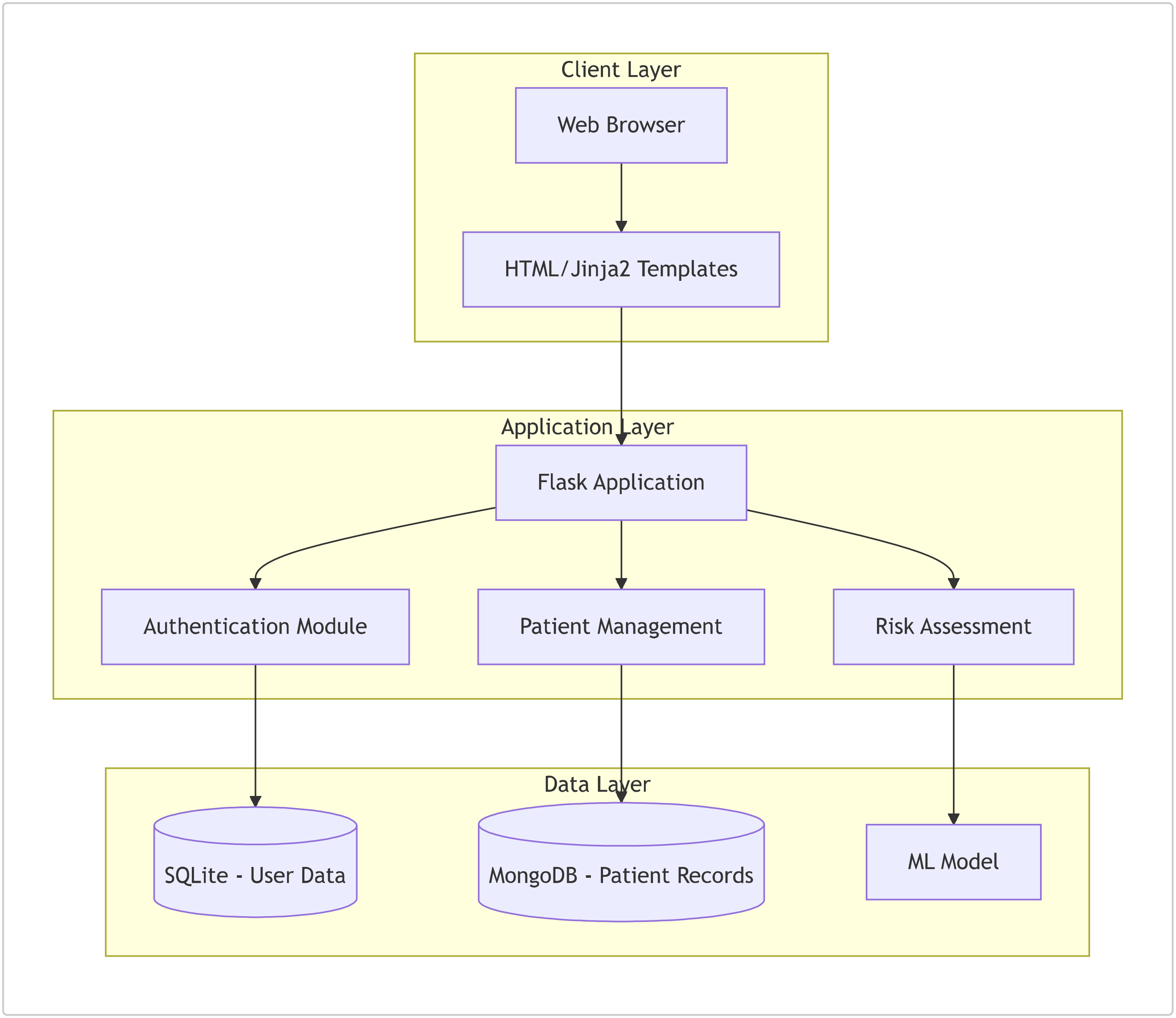
Security Components

- 🛡️ Flask-JWT-Extended 4.6.0
- 🔑 Flask-Login 0.6.3
- 🔐 Flask-Bcrypt 1.0.1
- 🛡️ Flask-WTF 1.2.2

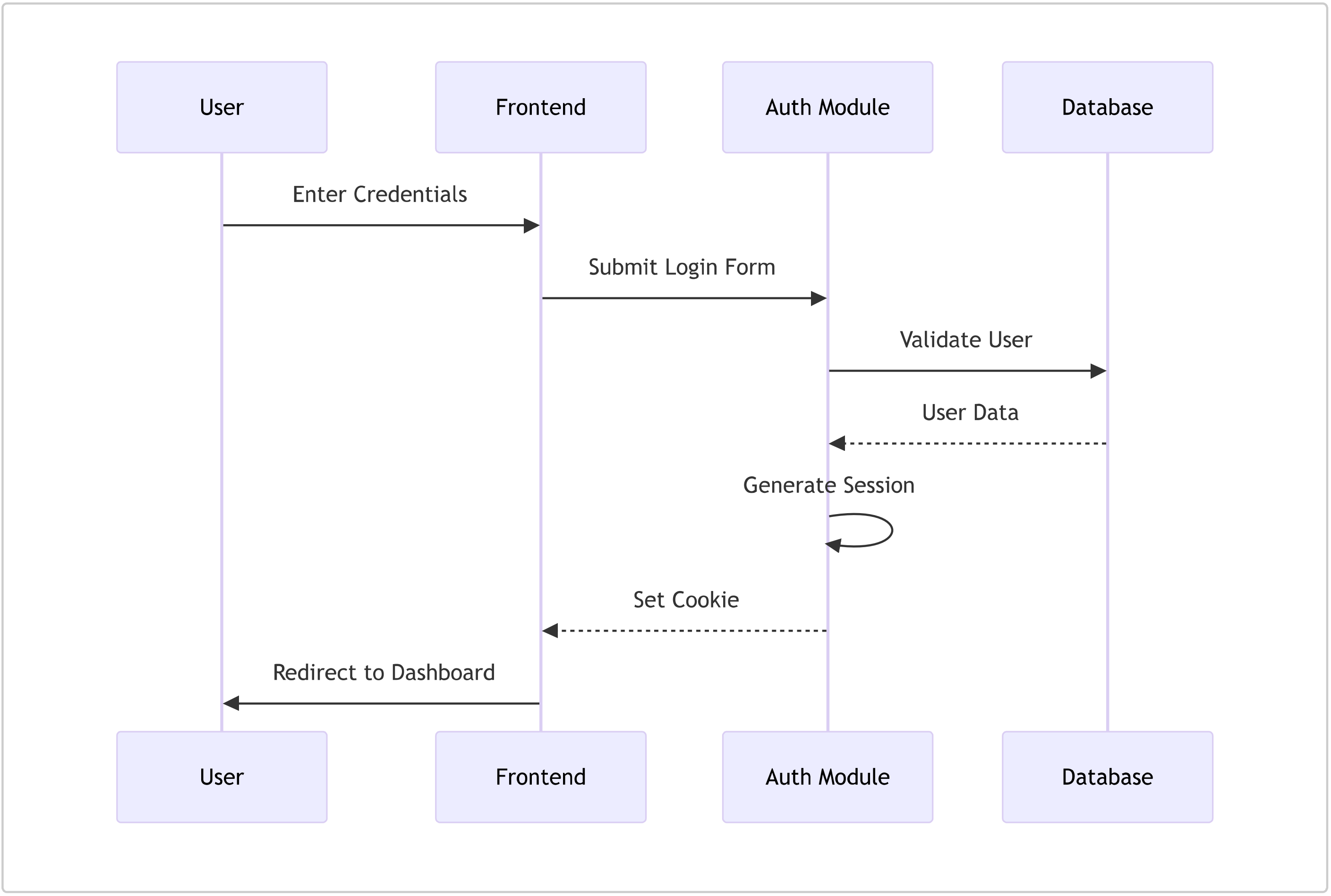
---

2. Technical Architecture

2.1 System Architecture



2.2 Authentication Flow



---

3. Setup and Installation

Environment Setup

```
# Clone repository
git clone https://github.com/CS-LTU/com7033-assignment-MRAWAISANWAR.git

# Create virtual environment
python -m venv venv
source venv/bin/activate  # Unix
venv\Scripts\activate     # Windows

# Install dependencies
pip install -r requirements.txt
```

Configuration

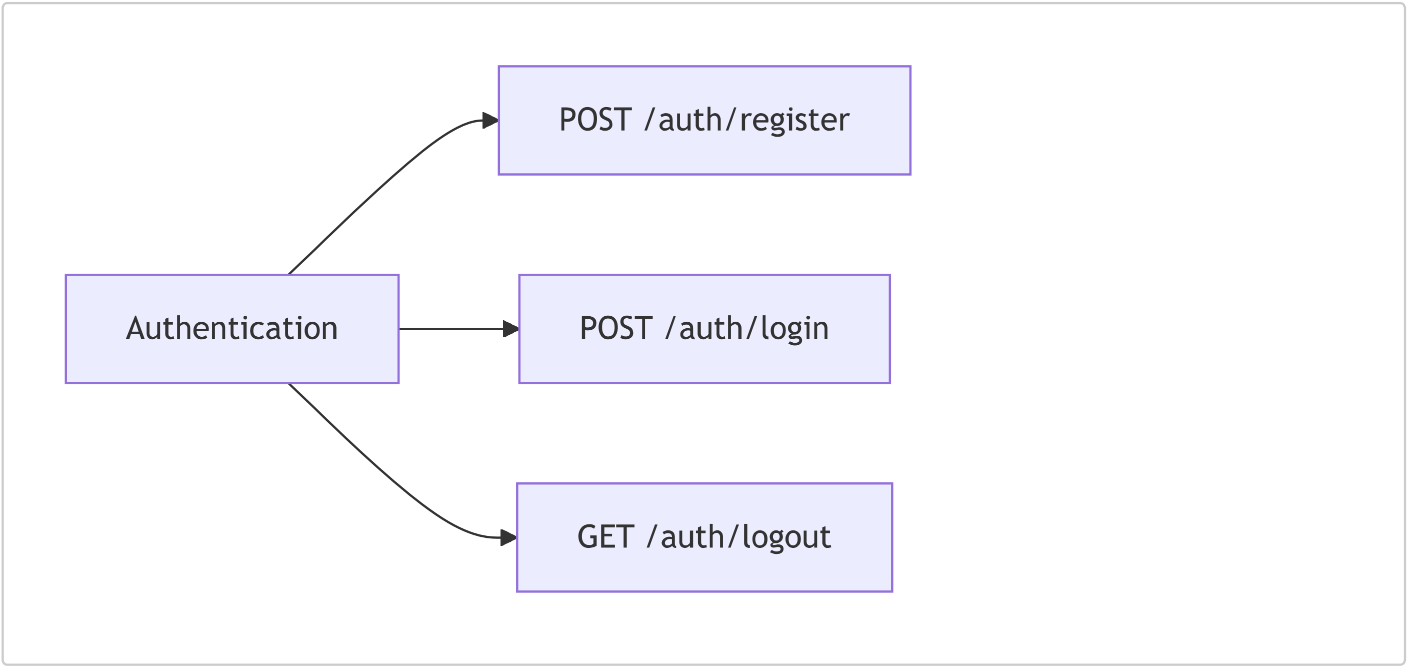
Add following to .env file:

```
FLASK_ENV=development
SECRET_KEY=your_secret_key
MONGO_URI=mongodb://localhost:27017/stroke_prediction
SQLITE_DATABASE_URI=sqlite:///stroke_prediction.db
```

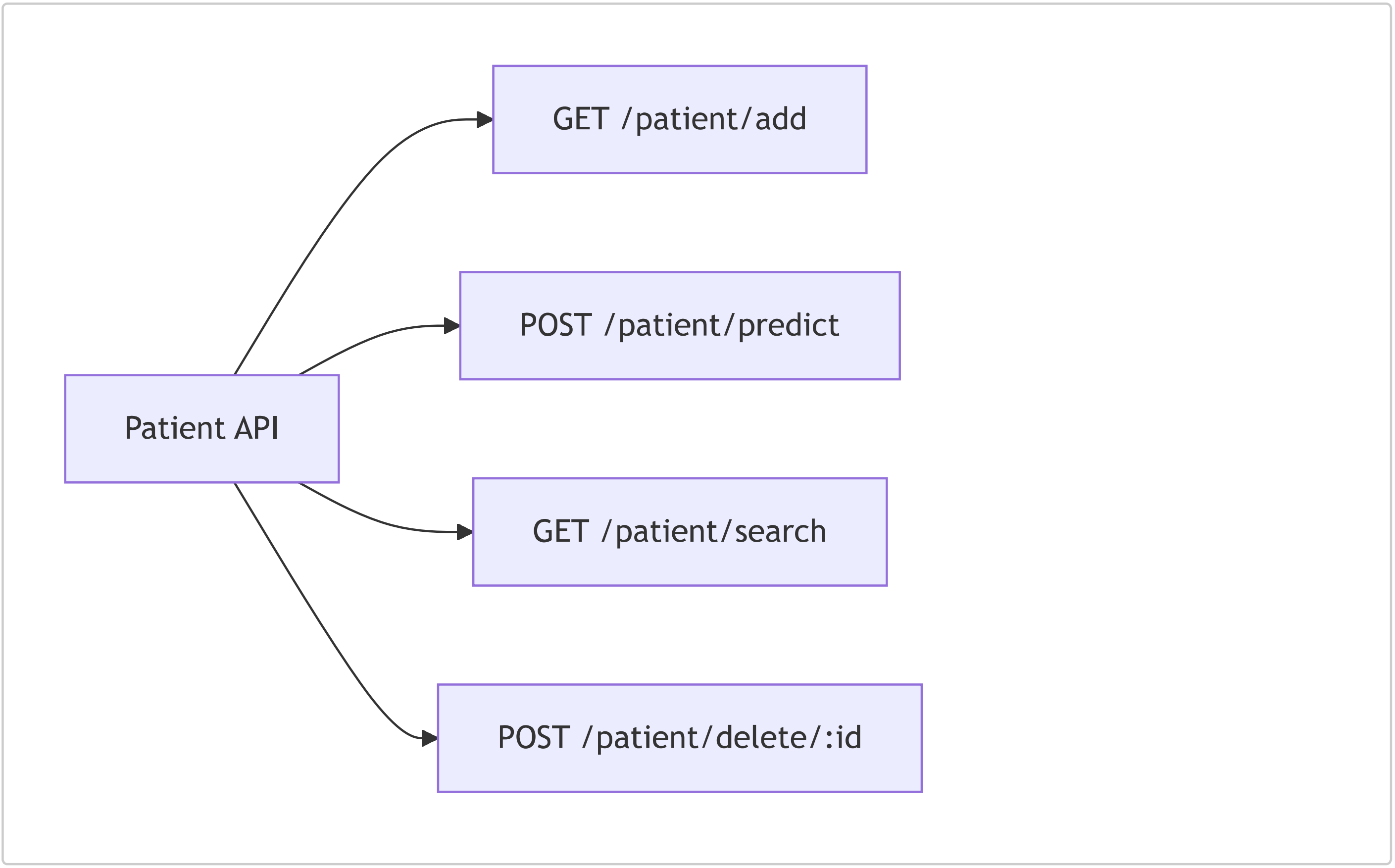
---

4. API Integration

Authentication Endpoints



Patient Management Endpoints



Example Requests

Register User

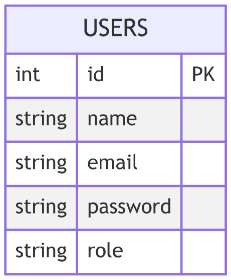
```
POST /auth/register
Content-Type: application/json

{
    "name": "John Doe",
    "email": "john@example.com",
    "password": "secure_password",
    "role": "doctor"
}
```

Add Patient

```
POST /patient/predict
Content-Type: application/json

{
    "name": "Patient Name",
    "age": 45,
    "gender": "Male",
        "hypertension": "1",
        ...
    }
```

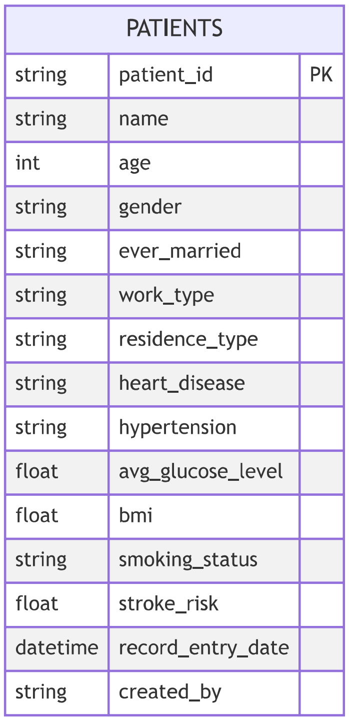
5. Database Design

User Schema (SQLite)



USERS		
int	id	PK
string	name	
string	email	
string	password	
string	role	

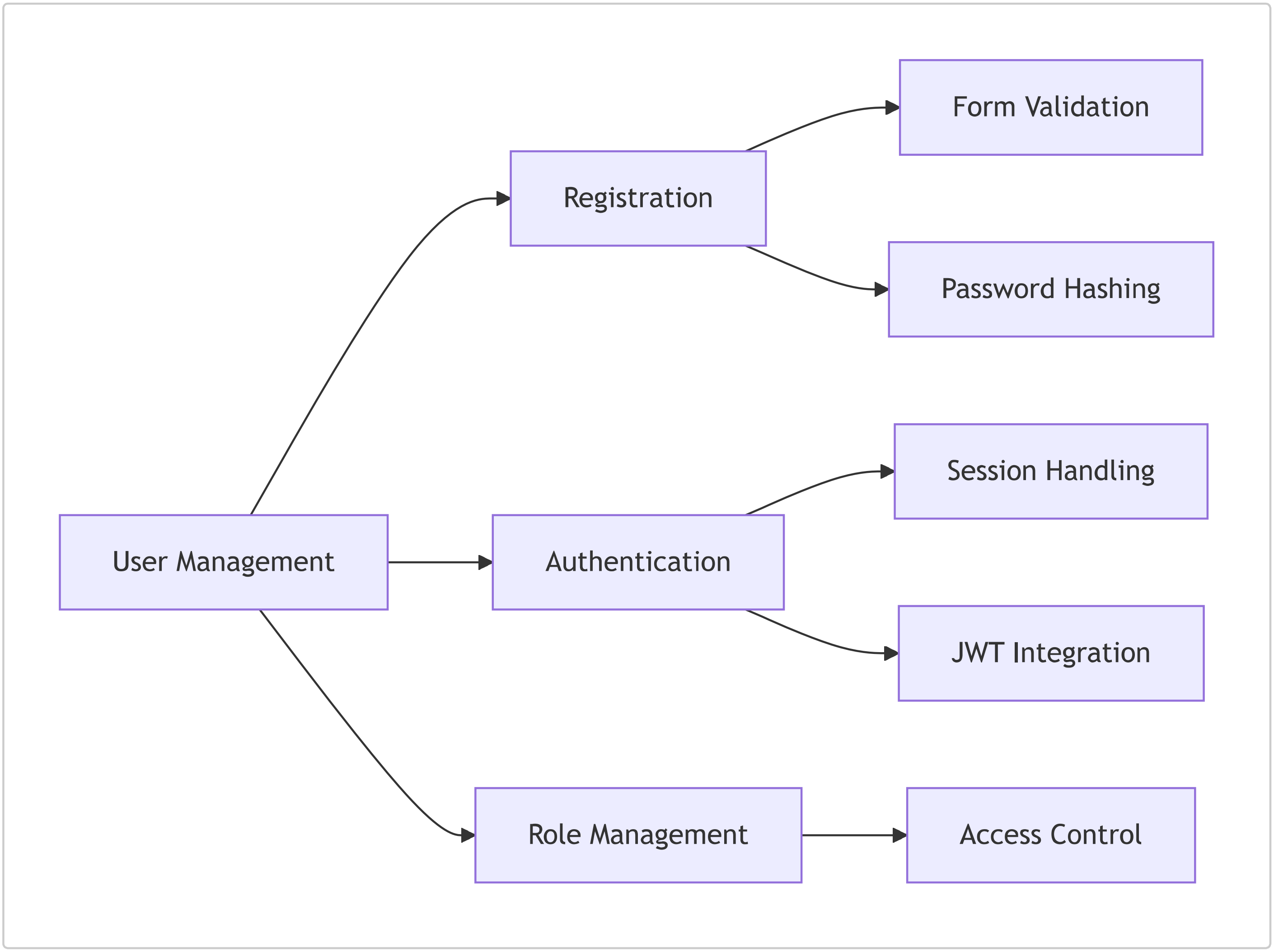
Patient Schema (MongoDB)



PATIENTS		
string	patient_id	PK
string	name	
int	age	
string	gender	
string	ever_married	
string	work_type	
string	residence_type	
string	heart_disease	
string	hypertension	
float	avg_glucose_level	
float	bmi	
string	smoking_status	
float	stroke_risk	
datetime	record_entry_date	
string	created_by	

6. Core Features

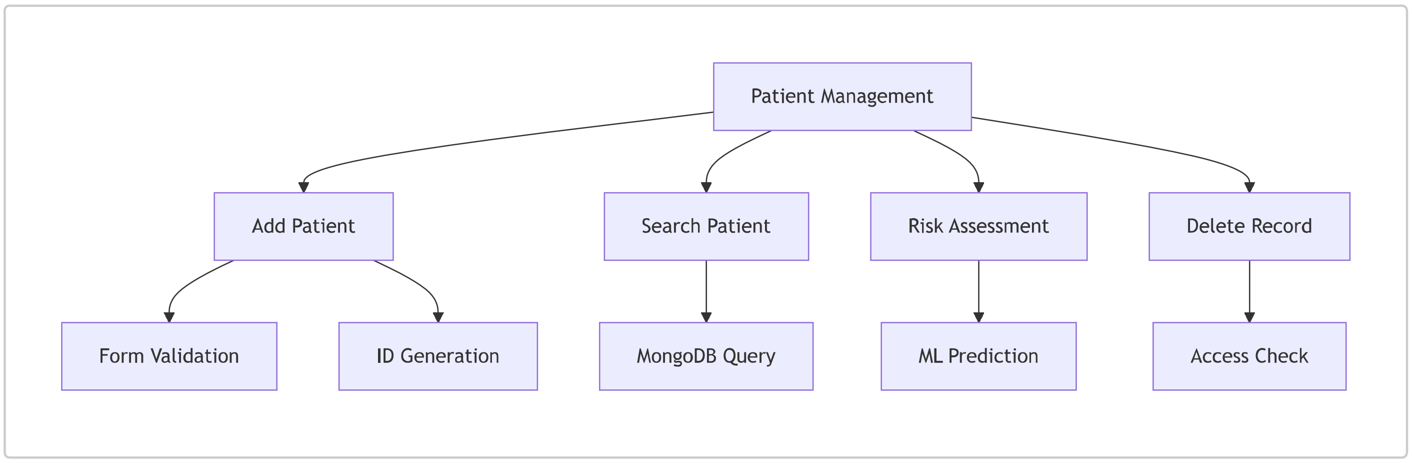
6.1 User Management



User Model Implementation

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(150), nullable=False)
    email = db.Column(db.String(150), unique=True)
    password = db.Column(db.String(150), nullable=False)
    role = db.Column(db.String(50), default="doctor")
```

6.2 Patient Management

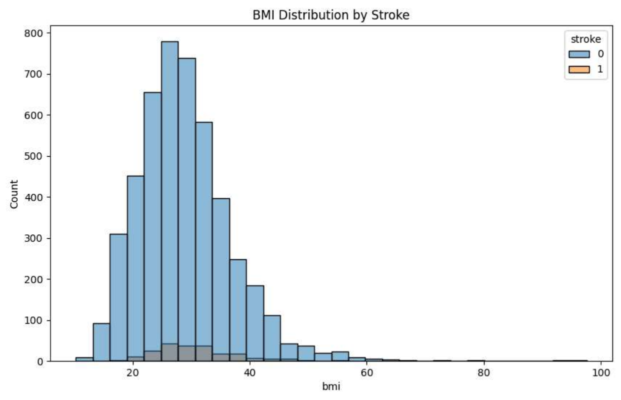


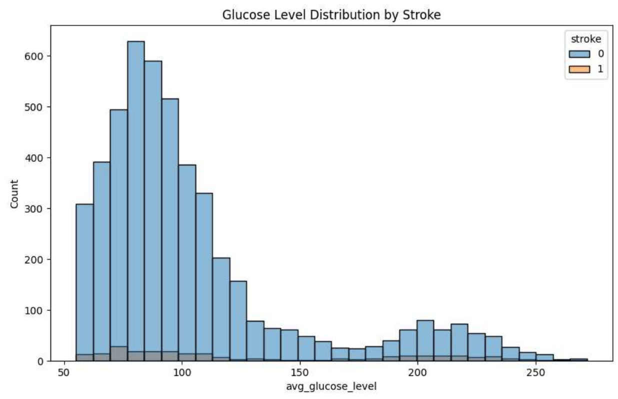
---

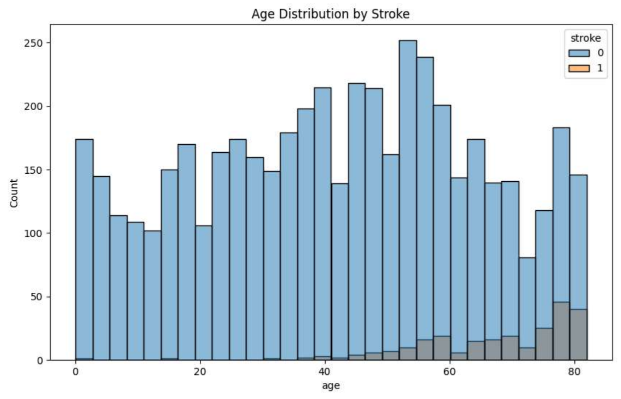
7. Dataset Analysis

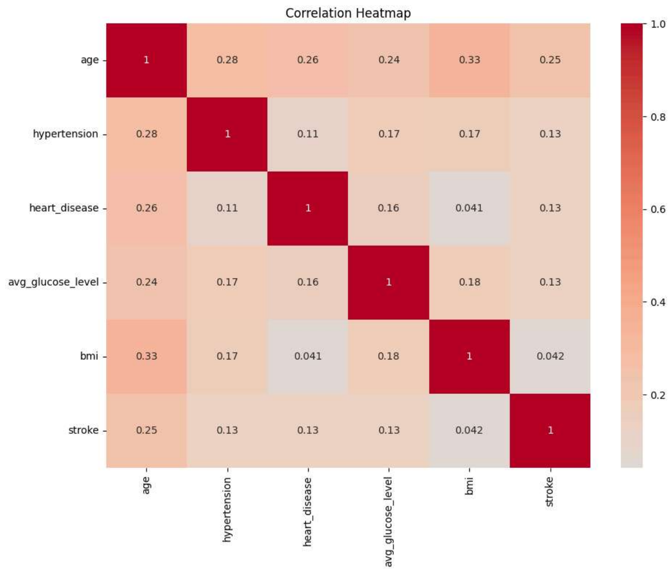
7.1 Dataset Analysis

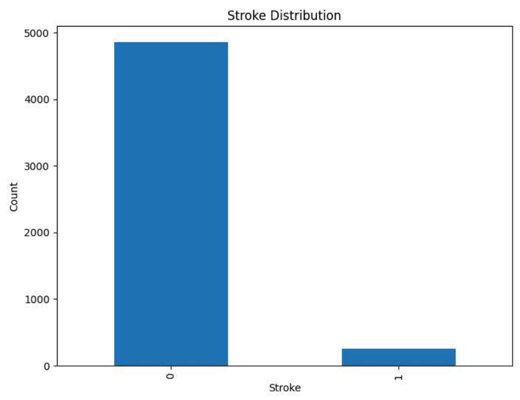
(Provided Dataset was highly imbalanced)

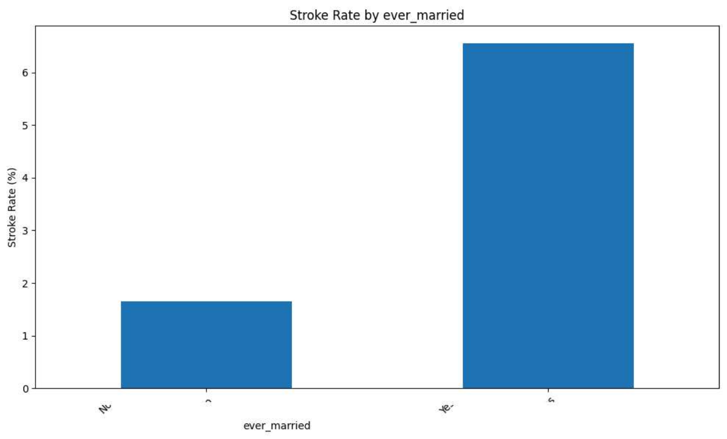

BMI Distribution by Stroke
stroke
0
1
Count
bmi
0
100
200
300
400
500
600
700
800
20
40
60
80
100

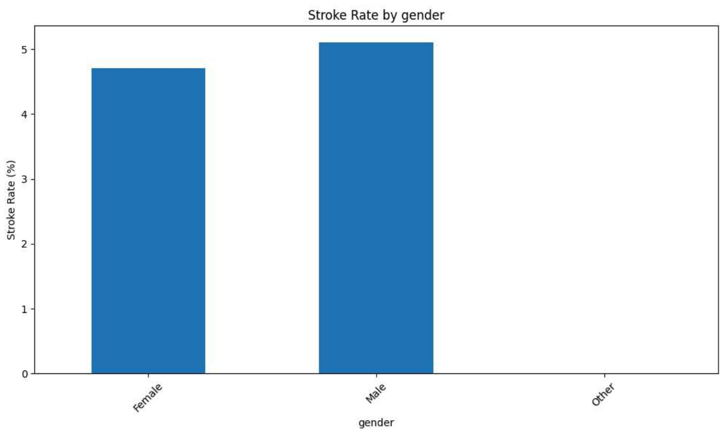

Glucose Level Distribution by Stroke
stroke
0
1
Count
avg_glucose_level
0
100
200
300
400
500
600
50
100
150
200
250

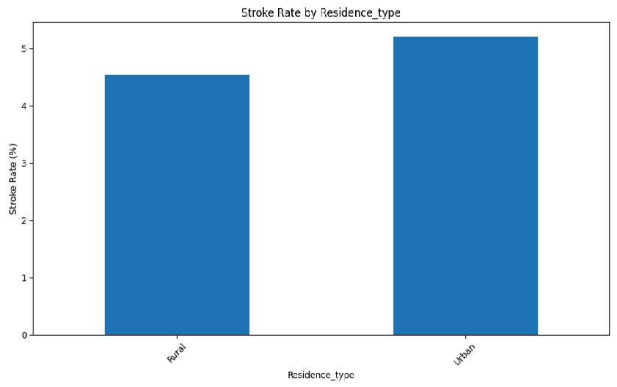

Age Distribution by Stroke
stroke
0
1
Count
age
0
50
100
150
200
250
0
20
40
60
80

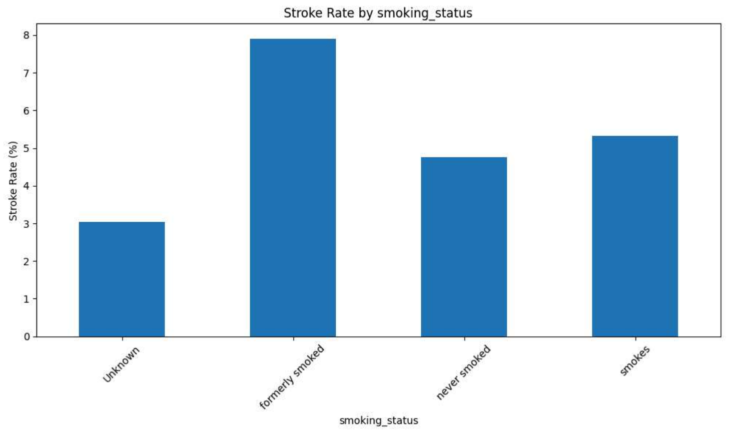

Correlation Heatmap
age
hypertension
heart_disease
avg_glucose_level
bmi
stroke
age: 1, 0.28, 0.26, 0.24, 0.33, 0.25
hypertension: 0.28, 1, 0.11, 0.17, 0.17, 0.13
heart_disease: 0.26, 0.11, 1, 0.16, 0.041, 0.13
avg_glucose_level: 0.24, 0.17, 0.16, 1, 0.18, 0.13
bmi: 0.33, 0.17, 0.041, 0.18, 1, 0.042
stroke: 0.25, 0.13, 0.13, 0.13, 0.042, 1
1.0
0.8
0.6
0.4
0.2





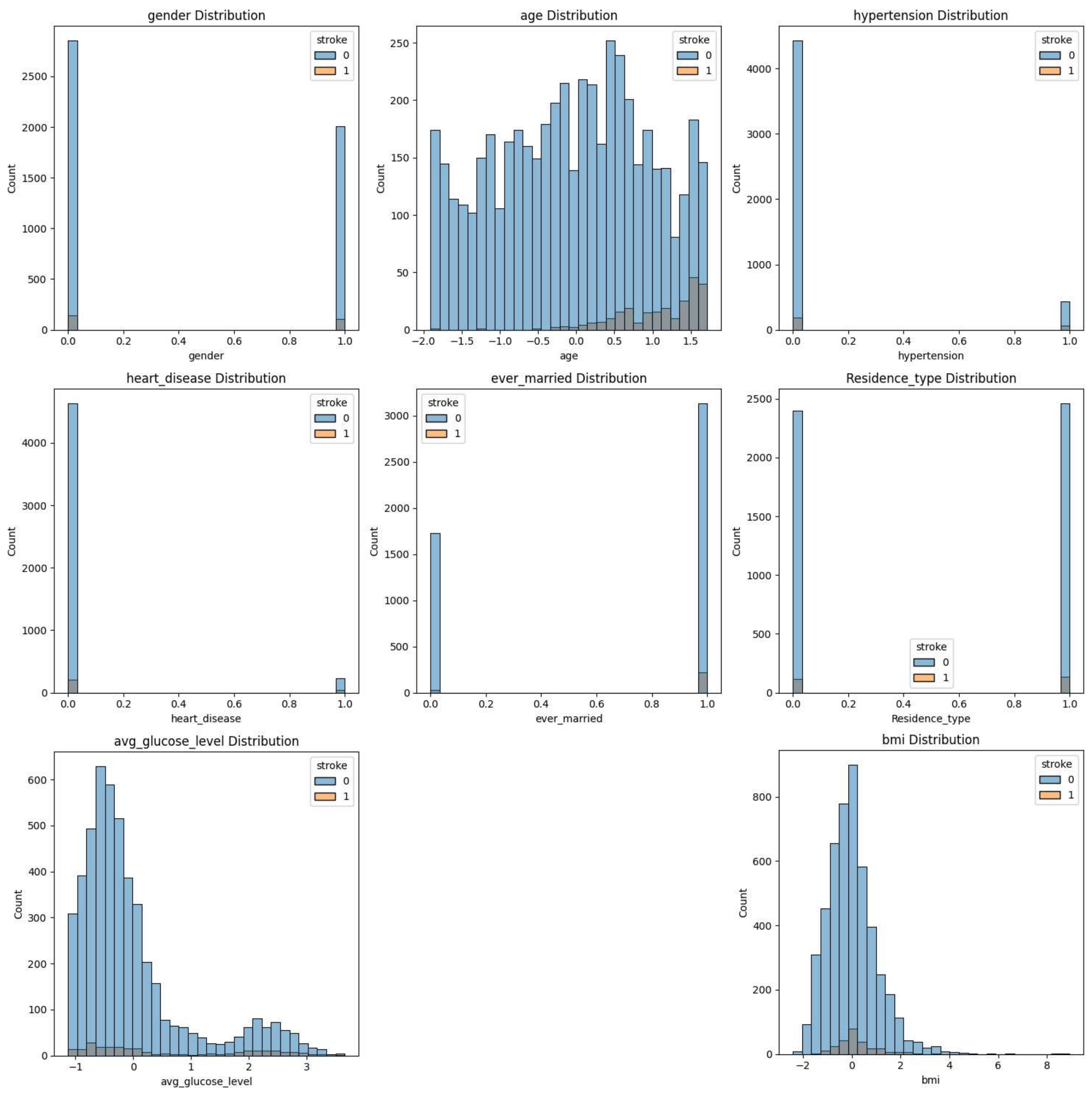


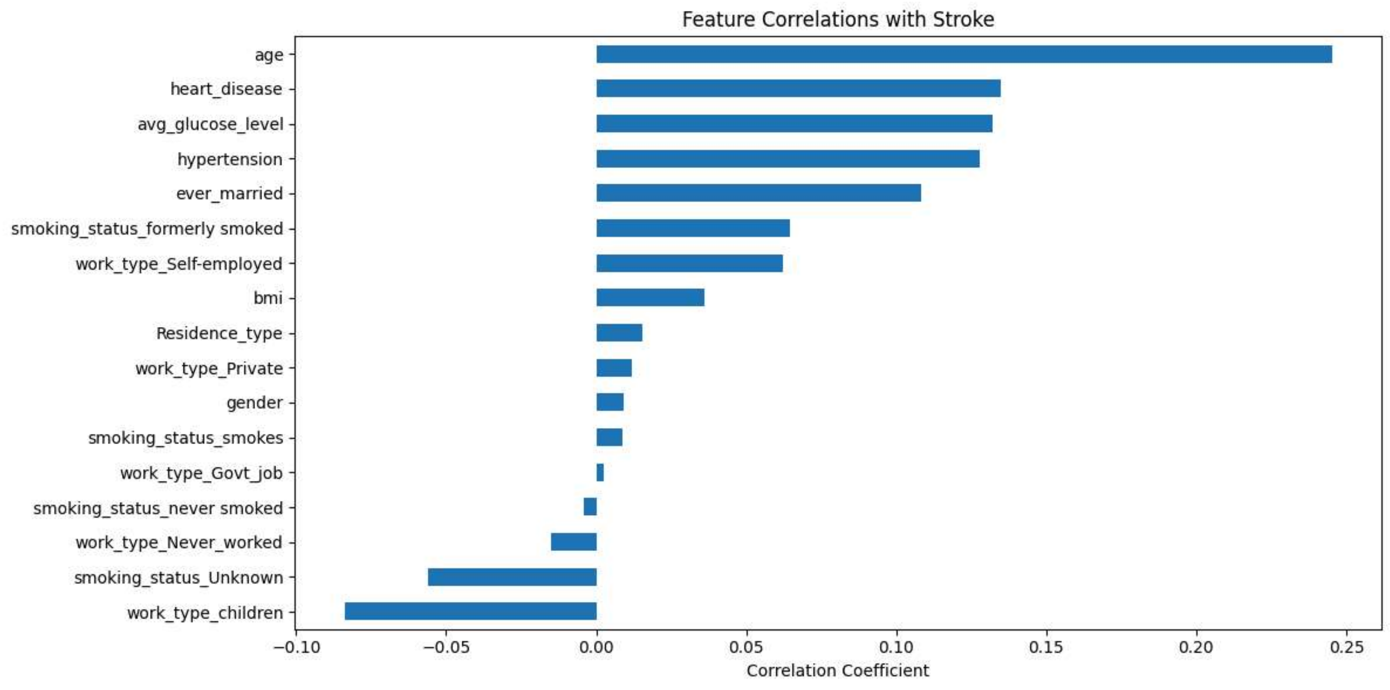




7.2 Processed Dataset Analysis

Feature Destributaions

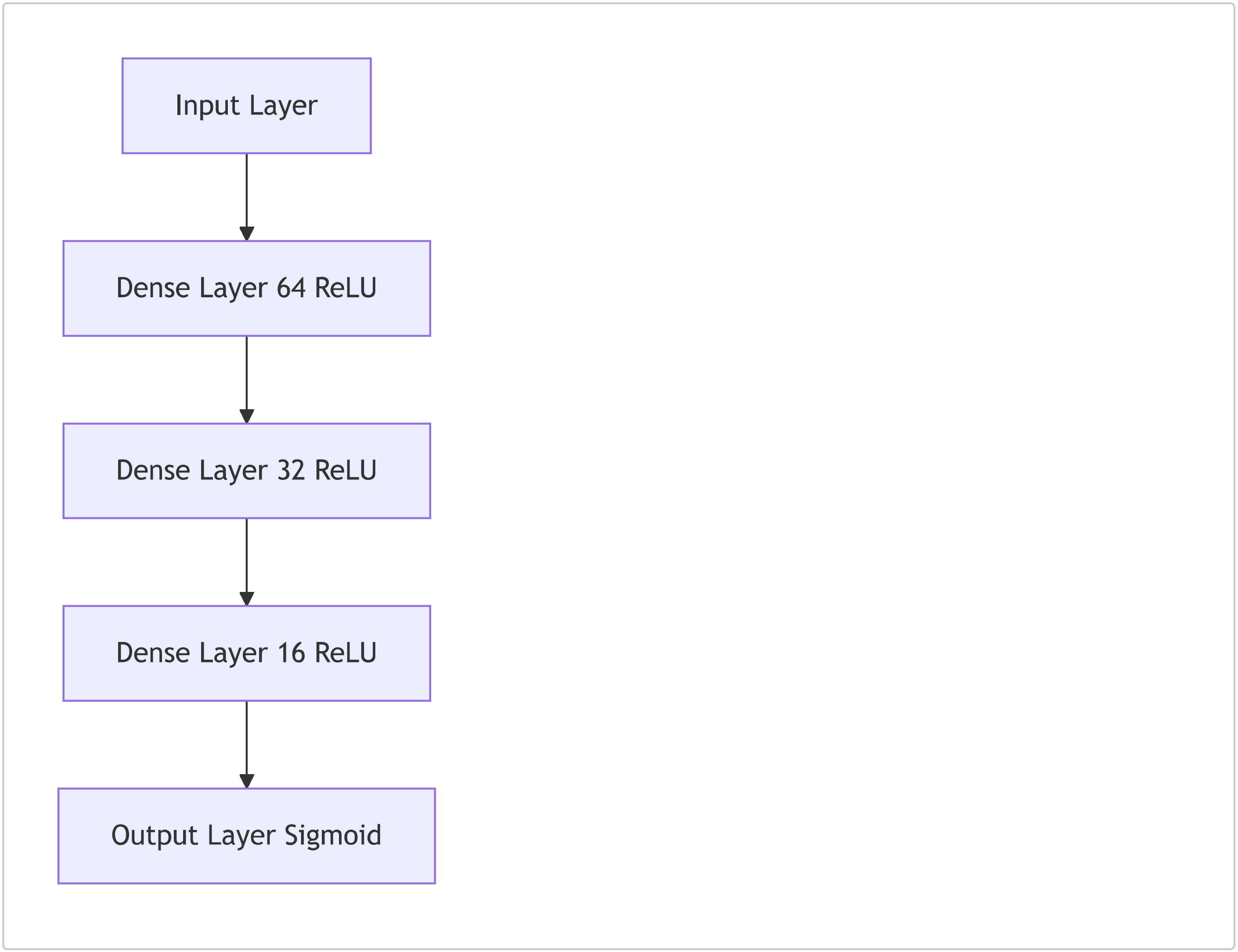




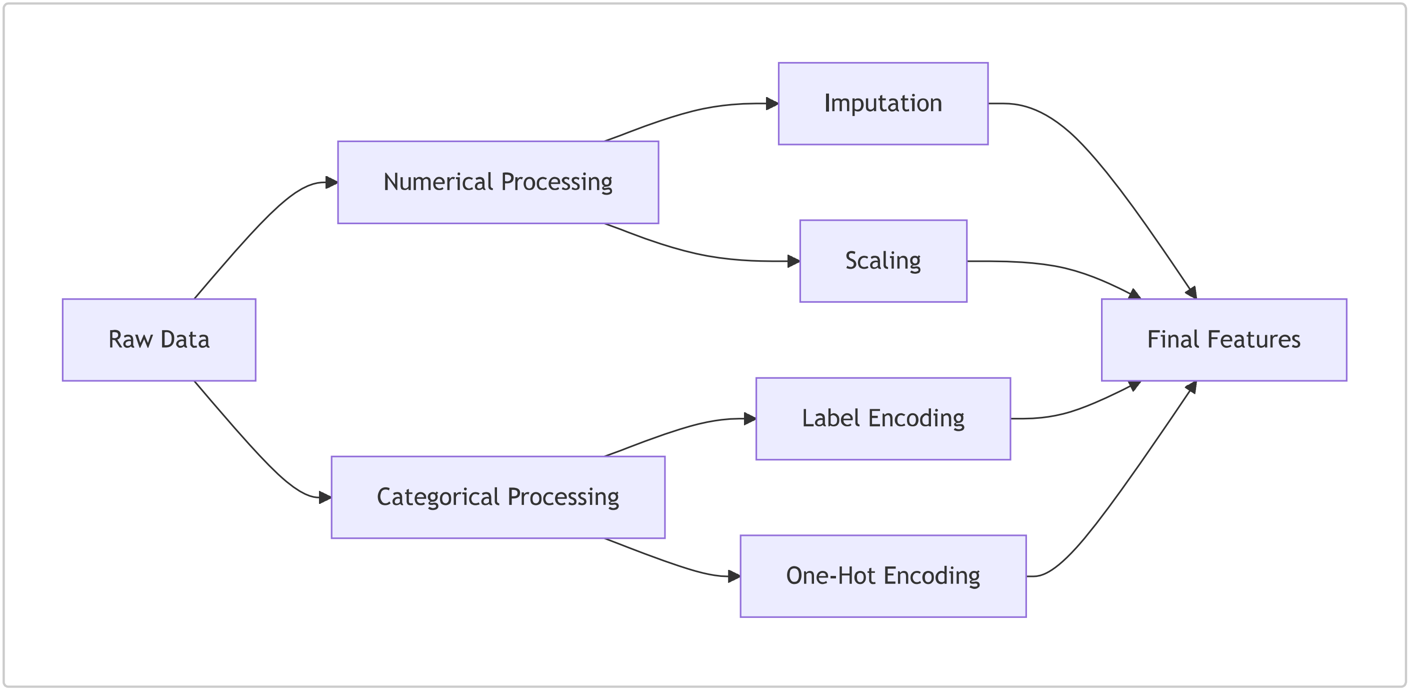
---

8. Machine Learning

8.1 Model Architecture



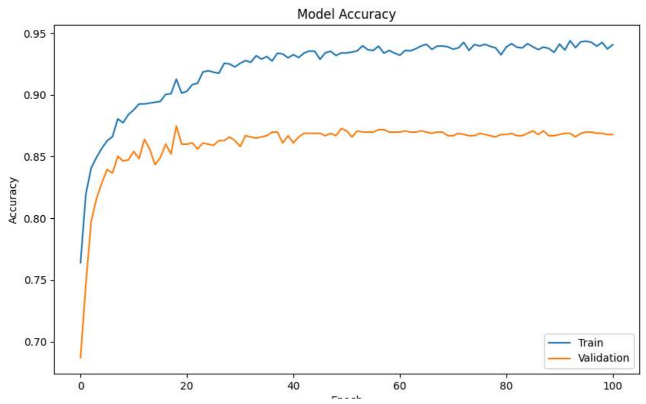
8.2 Feature Processing Pipeline

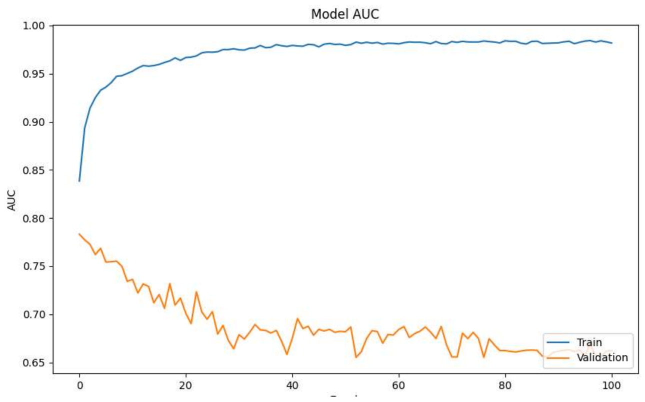


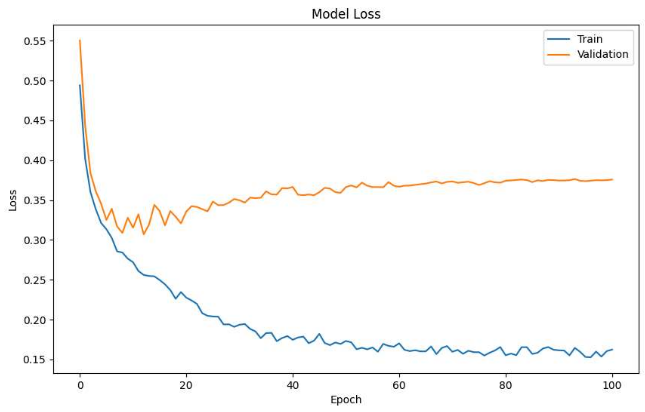
8.3 Model Performance

Model Evaluation

```
Accuracy:   69%
AUC-ROC:    80%
Precision:  98%
Recall:     69%
F1 Score:   81%
```



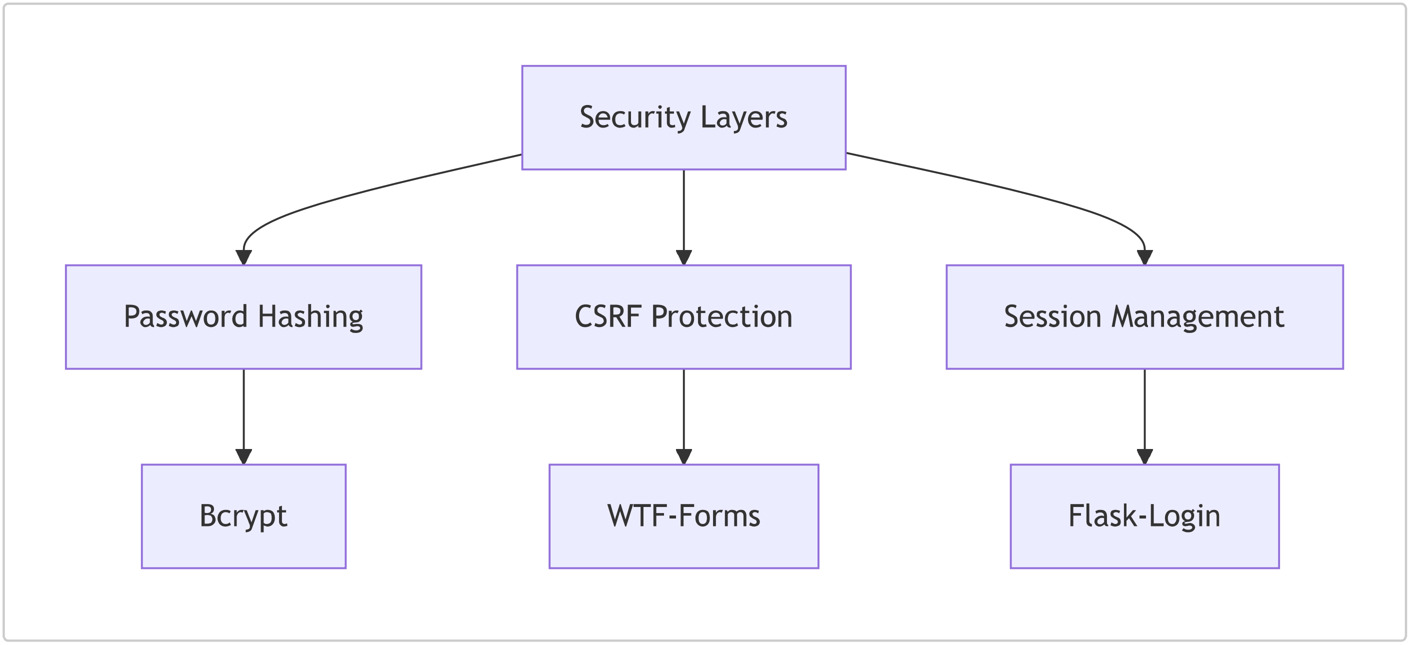




---

9. Security Implementation

9.1 Authentication Security



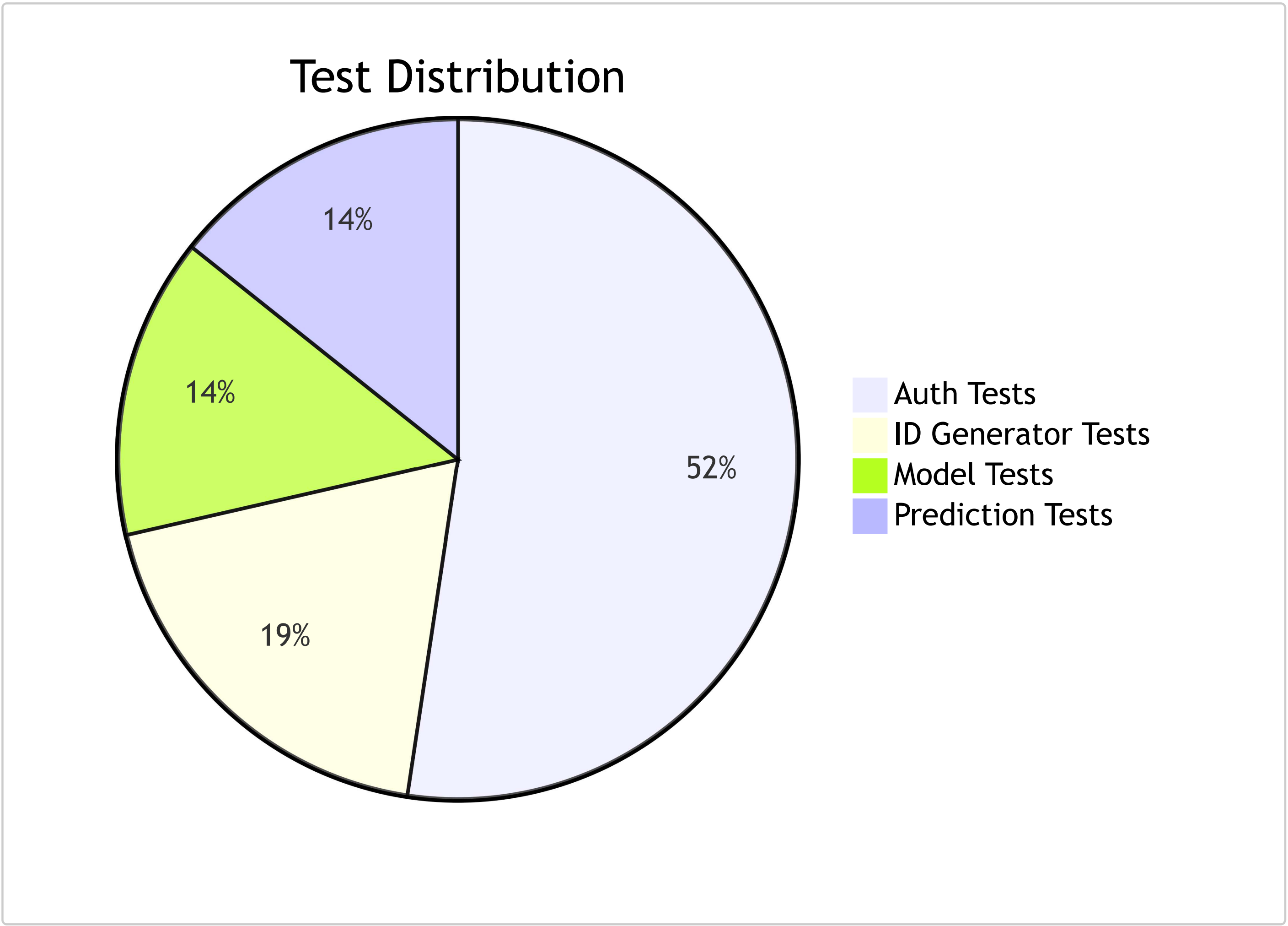
9.2 Data Protection

```python
# CSRF Protection Setup
app.config['WTF_CSRF_ENABLED'] = True
app.config['WTF_CSRF_TIME_LIMIT'] = 3600
app.config['WTF_CSRF_SSL_STRICT'] = True

# Password Hashing
def set_password(self, password):
    self.password = bcrypt.generate_password_hash(password).decode('utf-8')
```

---

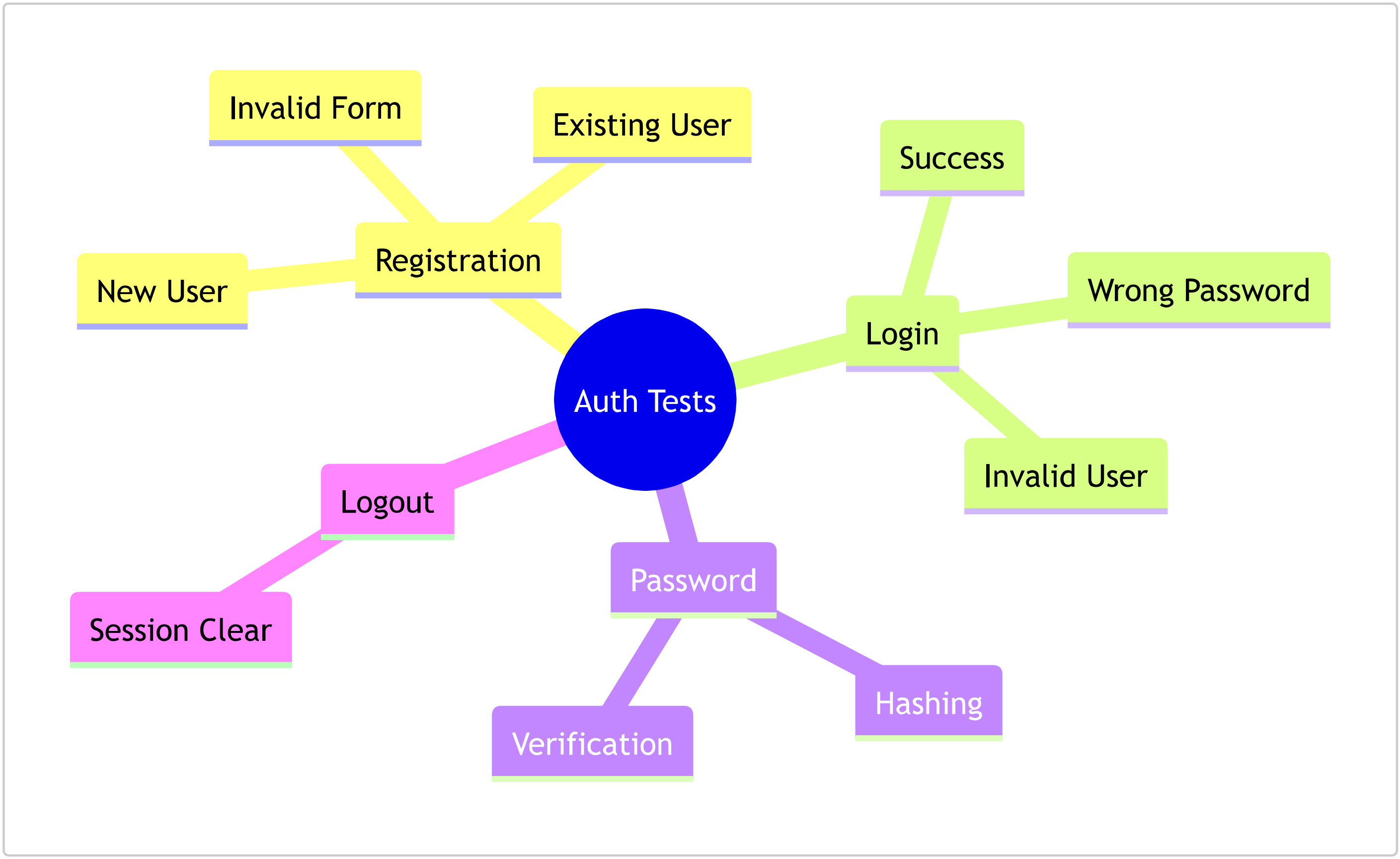
10. Testing & Quality Assurance

10.1 Test Coverage Overview

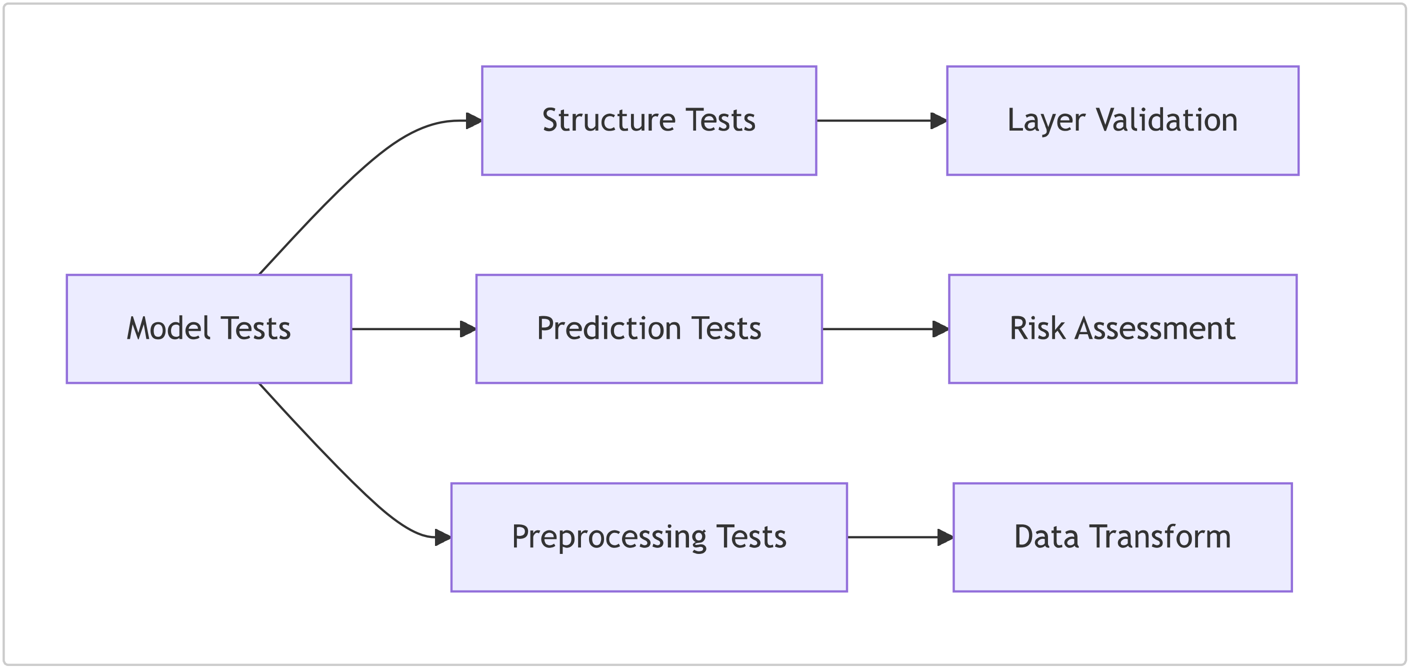


10.2 Test Categories

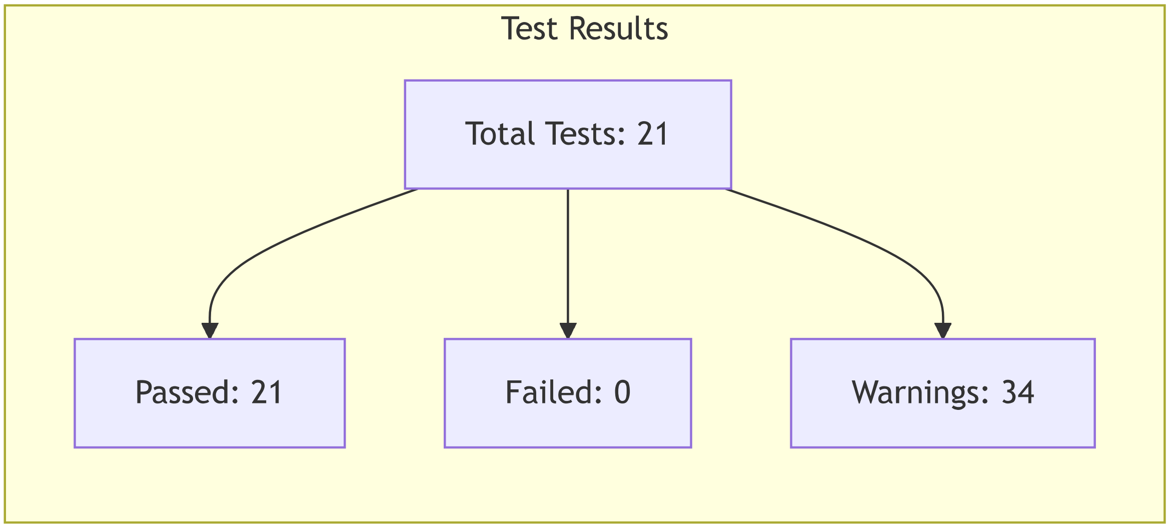
Authentication Testing



Model Evaluation Testing

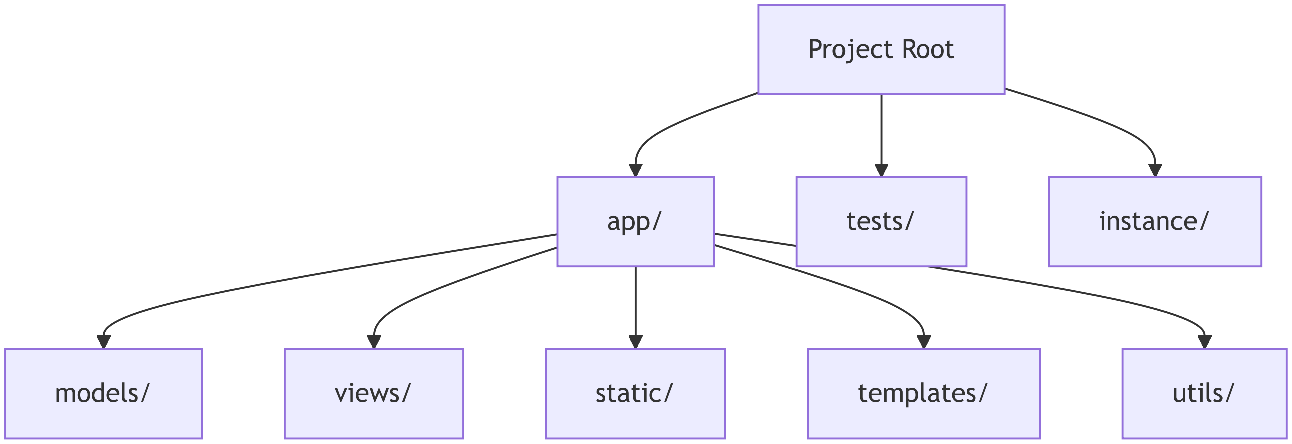


10.3 Test Results

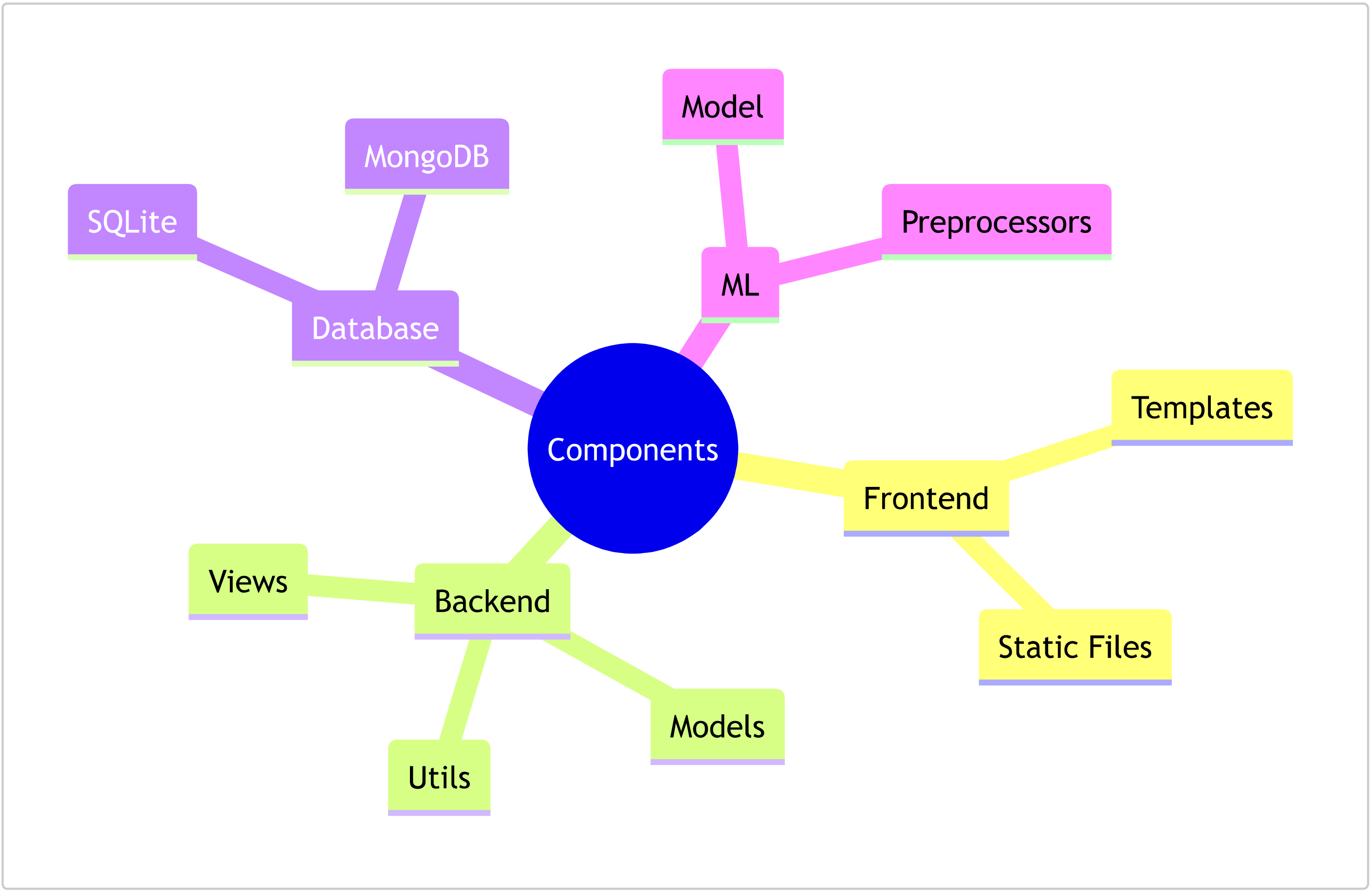


11. Code Structure

11.1 Project Layout



11.2 Key Components



11.3 Detailed Structure

```
├── app/
│   ├── forms/
│   │   └── patient_form.py
│   ├── models/
│   │   ├── patient.py
│   │   └── user.py
│   ├── static/
│   │   ├── css/
│   │   │   ├── home.css
│   │   │   └── styles.css
│   │   ├── js/
│   │   │   ├── home.js
│   │   │   ├── main.js
│   │   │   ├── mainC..js
│   │   │   ├── mainO.js
│   │   │   └── patient.js
│   │   └── models/
│   │       ├── model_metrics.json
│   │       ├── preprocessors.pkl
│   │       ├── stroke_prediction_model_Best.keras
│   │       └── stroke_prediction_model_Final.keras
│   ├── templates/
│   │   ├── auth/
│   │   │   ├── login.html
│   │   │   └── register.html
│   │   ├── partials/
│   │   │   └── navbar.html
│   │   ├── patient/
│   │   │   └── add_patient.html
│   │   ├── profile/
│   │   │   └── settings.html
│   │   ├── base.html
│   │   ├── home.html
│   │   └── patient_details.html
│   ├── utils/
│   │   ├── decorators.py
│   │   ├── id_generator.py
│   │   └── prediction.py
│   ├── views/
│   │   ├── auth.py
│   │   ├── process_patient.py
│   │   └── profile.py
│   └── __init__.py
├── instance/
│   └── stroke_prediction.db
├── .env
├── InitializeSQLlite.py
├── MongoDB_Schema.py
└── run.py
```

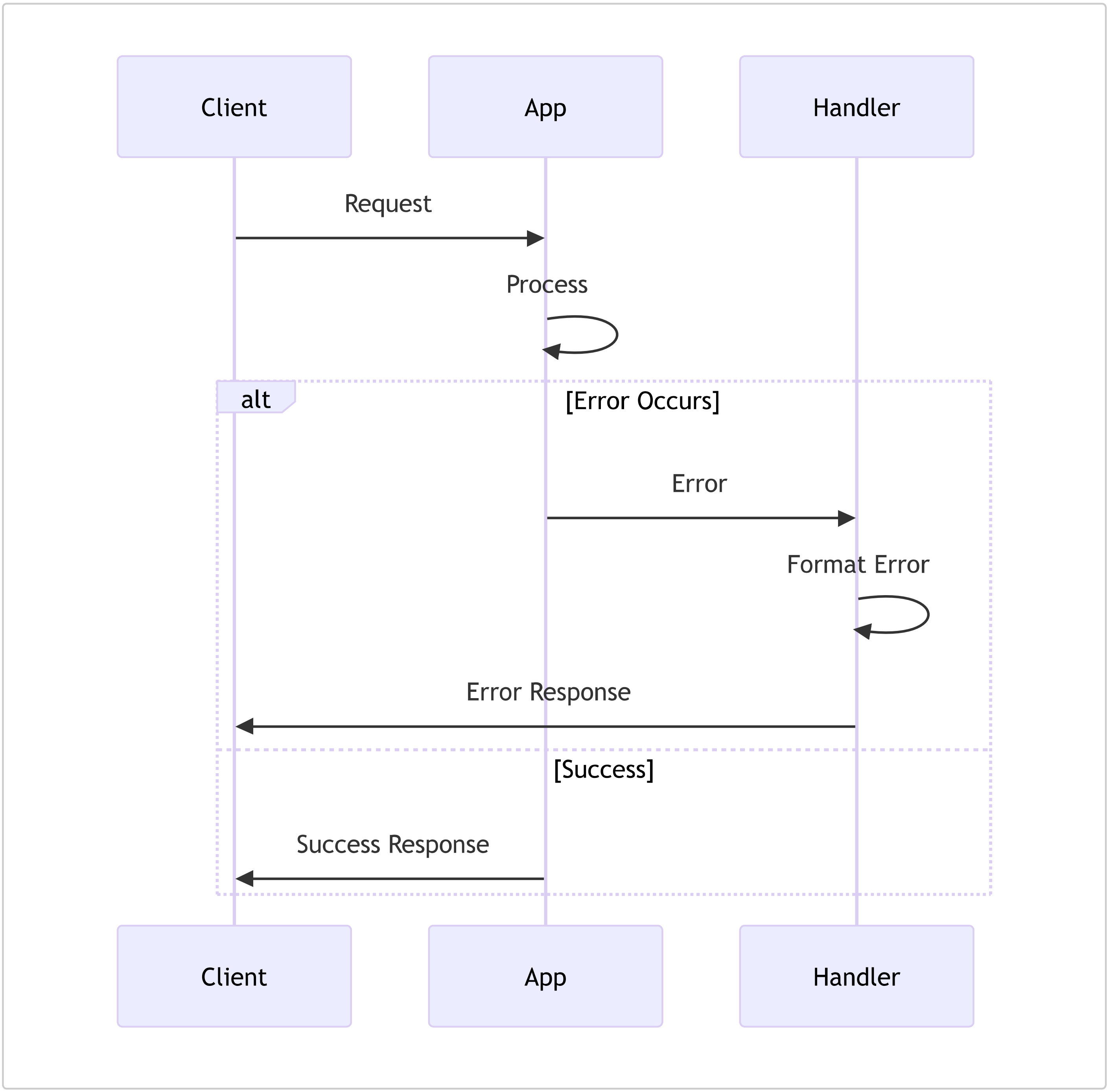
---

12. Error Handling

12.1 Global Error Handlers

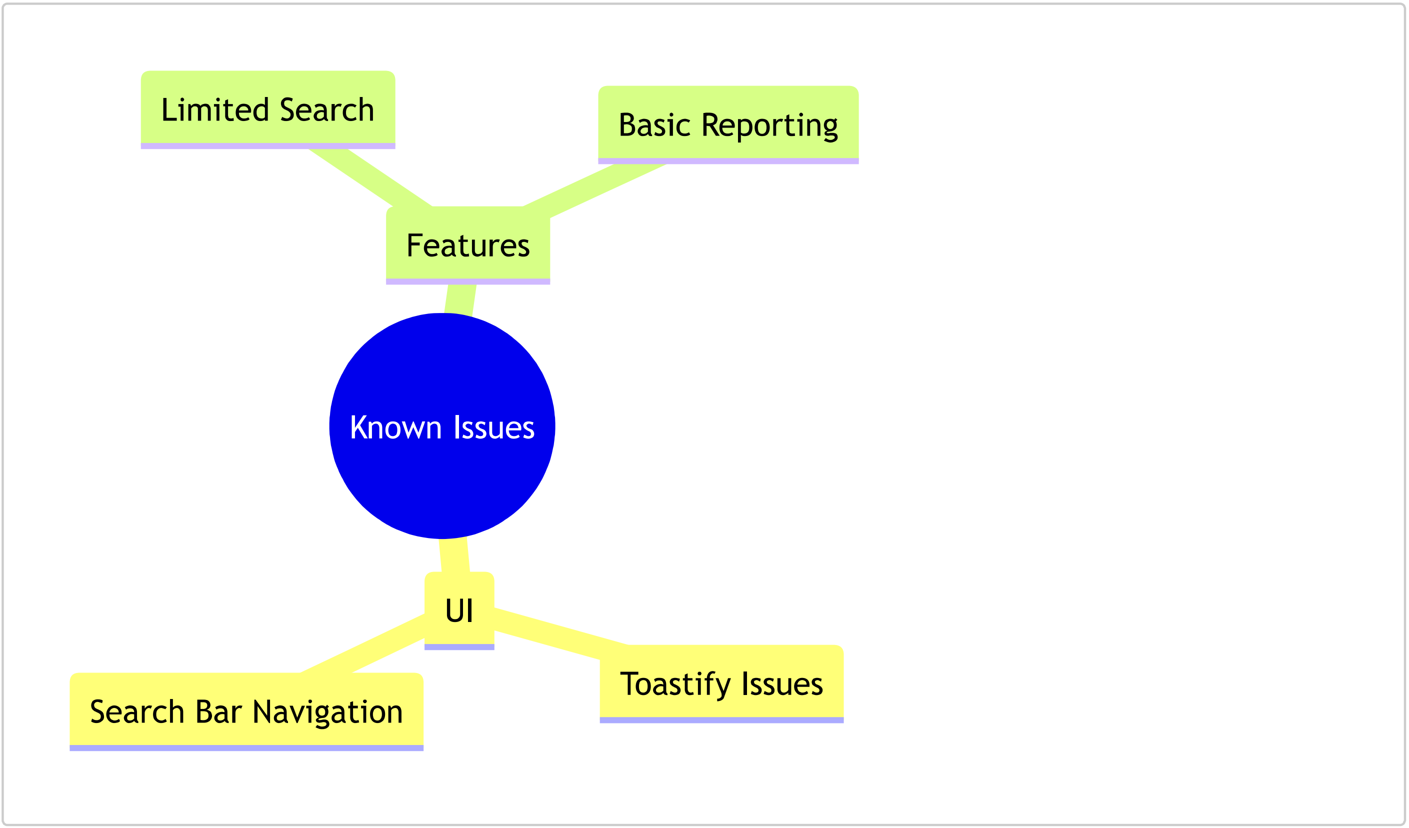
```
@app.errorhandler(CSRFError)
def handle_csrf_error(e):
    return jsonify({
        "error": "CSRF token missing or invalid",
        "message": str(e)
    }), 400

@app.errorhandler(500)
def handle_server_error(e):
    return jsonify({
        "error": "Internal Server Error",
        "message": "An unexpected error occurred"
    }), 500
```

12.2 Error Flow



13. Known Issues

13.1 Current Limitations



Additional Resources

Quick Reference

- 📘 API Documentation
- 🔧 Setup Guide
- 🧪 Testing Guide
- ⚙️ Configuration

Contact

- 📧 Email Support: 2410816@leedstrinity.ac.uk