Technical Documentation – STUDI-web-app

Table of Contents

1. Initial Technological Considerations

The primary objective for developing the application for SoigneMoi hospital was to enhance efficiency and streamline operations. The key considerations included:

a. **Scalability**: The application needed to handle an increasing number of users and data without performance degradation. We chose PHP with PDO for the backend and MySQL for the database due to their scalability features.

b. **Security**: Protecting patient data is paramount. We implemented secure authentication mechanisms and data encryption.

c. **User Experience**: The application had to be intuitive and user-friendly. We used HTML5, CSS, and JavaScript to ensure a responsive design.

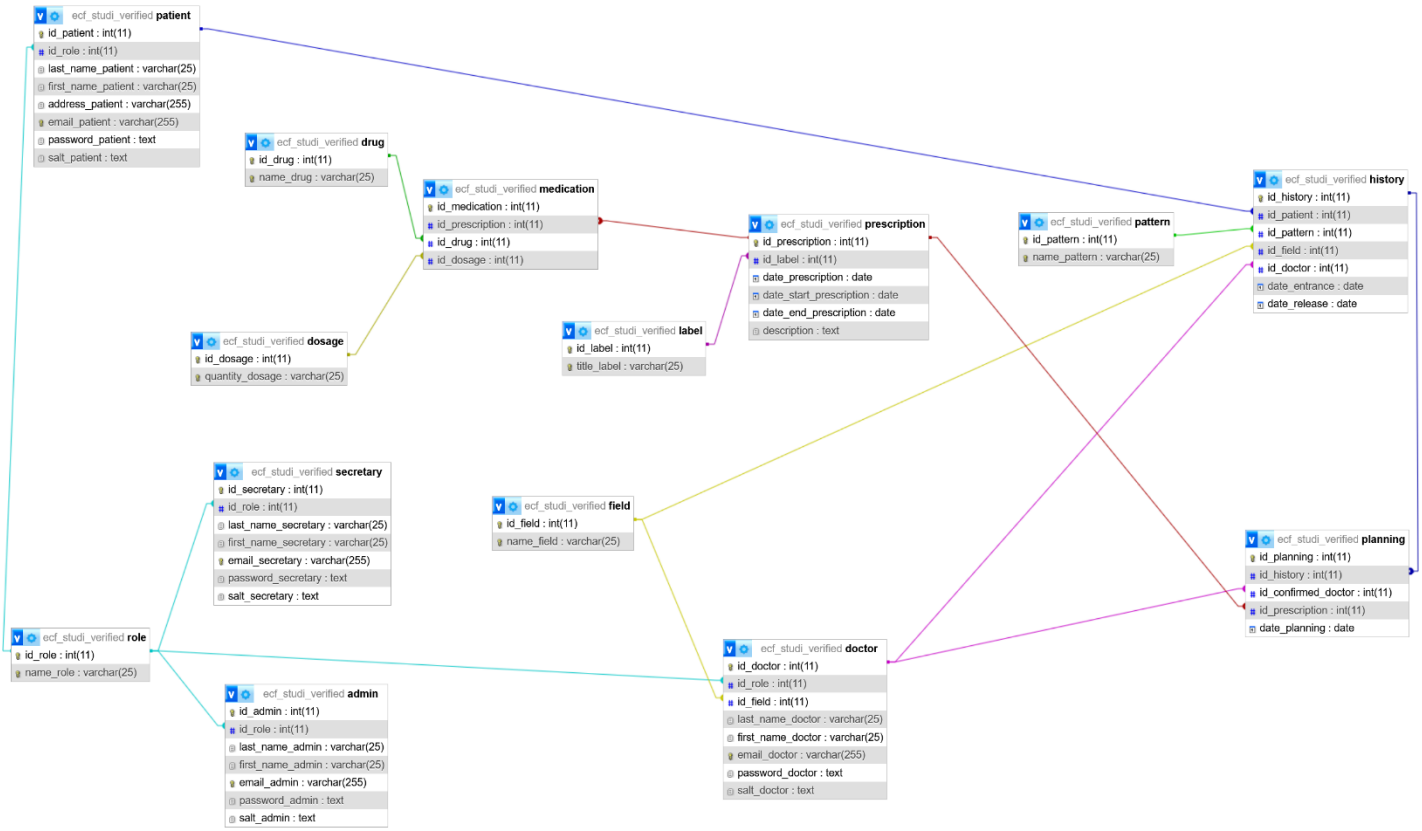
2. Work Environment Configuration

a. **Development Environment**:
   - **Operating System**: MacOS
   - **Web Server**: Apache 2.4.58
   - **Programming Language**: PHP 8.2.4
   - **Database**: MySQL 15.1
   - **Version Control**: Git
   - **Development Tools**: Visual Studio Code, phpMyAdmin

b. **Deployment Environment**:
   - **Platform**: Fly.io
   - **CI/CD**: GitHub Actions

3. Conceptual Data Model (MCD)

The conceptual data model defines the main entities and their relationships:



4. Use Case Diagram

The use case diagram represents the interactions between different user types and the system.

Key use cases:

a. **For visitor:**

- Create an account
- Log in

a. **For patient:**

- View its history
- Book a stay

c. **For admin:**

- View all doctors
- View all plannings from today
- Add a new doctor
- Manage doctor schedules

5. Sequence Diagram

The sequence diagram illustrates the flow of operations for:

a. **Creation of account:**
   1. **Visitor** enters last name, first name, postal address, email and password then validates the reCAPTCHA.
   2. **System** checks entries and reCAPTCHA keys.
   3. **System** validates the entries or not.
   4. **System** saves admission information or not.
   5. **System** confirms the admission creation to the visitor or not.

b. **Log in:**
   1. **Visitor** enters email and password then validates the reCAPTCHA.
   2. **System** checks entries and reCAPTCHA keys.
   3. **System** validates the entries or not.
   4. **System** confirms the log in to the visitor or not.

c. **View patient history:**
   1. **Patient** clicks on the specific button from home.
   2. **System** displays the list.

d. **Book a stay:**
   1. **Patient** enters a start and end dates, the reason, the required specialty and the desired doctor.
   2. **System** checks entries.
   3. **System** validates the entries or not.
   4. **System** saves stay information or not.
   5. **System** confirms the new stay to the patient or not.

e. **View all doctors:**
   1. **Admin** clicks on the specific button from home.
   2. **System** displays the list.

f. **View all plannings from today:**
   1. **Admin** clicks on the specific button from home.
   2. **System** displays the list.

g. **Add a new doctor:**
   1. **Admin** enters the last and first names with specialty of the new doctor.
   2. **System** checks entries.
   3. **System** validates the entries or not.
   4. **System** saves doctor information or not.

5. **System** confirms the new doctor to the admin or not.

h. **Manage doctor schedules:**

1. **Admin** chooses the ID and the doctor who will replace.
2. **System** checks entries.
3. **System** validates the entries or not.
4. **System** saves the change or not.
5. **System** confirms the change to the admin or not.

6. Plan Explanation

The test plan was designed to ensure comprehensive testing of the application, covering unit tests, integration tests, and user acceptance tests (UAT).

1. **Unit Tests**:
    - **Objective**: Verify individual components and functions. The main objective is to test all functions in 4. Use Case Diagram.
    - **Tools**: PHP and local database.
2. **Integration Tests**:
    - **Objective**: Ensure modules and components work together.
    - **Approach**: Manual tests from simulated website.
3. **User Acceptance Tests (UAT)**:
    - **Objective**: Validate the application against user requirements. The main objective is to test all functions in 4. Use Case Diagram.
    - **Approach**: Conducted with actual users to simulate real-world scenarios.
4. **Security Tests**:
    - **Objective**: Identify vulnerabilities and ensure data protection.
    - **Approach**: Manual tests from simulated website with SQL injection and Cross-Site Scripting.
5. **Performance Tests**:
    - **Objective**: Ensure the application performs well under expected load.
    - **Approach**: Manual tests from simulated website.