Clinic Management System

Project Description:

The Clinic Management System is designed for a multi-specialty private clinic offering fields like internal medicine, ophthalmology, orthopedics, dentistry, dermatology, physical therapy, surgery, and gynecology. Each specialty has two doctors with different schedules.

Patients can register and book appointments based on available specialties and doctor schedules. They receive a queue number upon booking, which is automatically assigned. Doctors can view their appointments and patient details after logging in. Admins manage the system by adding/removing doctors, modifying schedules, and handling appointment cancellations.

Functional Requirements:

1. **User Registration**
    - Patients should be able to register by entering their personal information.
    - Patients should enter their name, unique email address, phone number, age, gender, and password.
2. **Login**
    - Patients, admins, and doctors can log in to the system with their email address and password.
3. **View Specialties**
    - Patients should be able to view a list of all available specialties.
4. **View Available Doctors**
    - Patients should be able to view a list of doctors in a selected specialty.
5. **View the Doctor's Available Days**
    - Patients should be able to view available days for a specific doctor.
6. **Make Appointment**
    - Patients should be able to book an appointment with a doctor for a specific day, receiving a queue number.
7. **View Appointment History**
    - Patients can view a list of their past appointments.

8. **Set Appointment Limit for Doctors Per Day**
    - Admins should be able to set a maximum number of appointments per day for each doctor. Once the limit is reached, the system should automatically close bookings for that day for the specified doctor.
9. **View Doctor Schedule**
    - Doctors should be able to view their available days based on their schedule.
10. **View Doctor Appointment**
    - Doctors can view all their appointments.
11. **Add Doctor**
    - Admins should be able to add a new doctor to the system, assigning the doctor's specialty.
12. **Edit Doctor**
    - Admins should be able to edit each doctor's personal information and specialty.
13. **Remove Doctor**
    - Admins should be able to remove a doctor from the system if necessary.
14. **Add Specialty**
    - Admins should be able to add new specialties to the system.
15. **Remove Specialty**
    - Admins should be able to remove a specialty from the system if necessary.
16. **Add Appointment**
    - Admins should be able to add new appointments.
17. **Delete Appointment**
    - Admins should be able to delete a patient's appointment.
18. **Edit Appointment**
    - Admins should be able to edit a patient's appointment status (Pending or Completed).
19. **Make Reports**
    - Doctors can submit reports for specified appointments and make them completed.

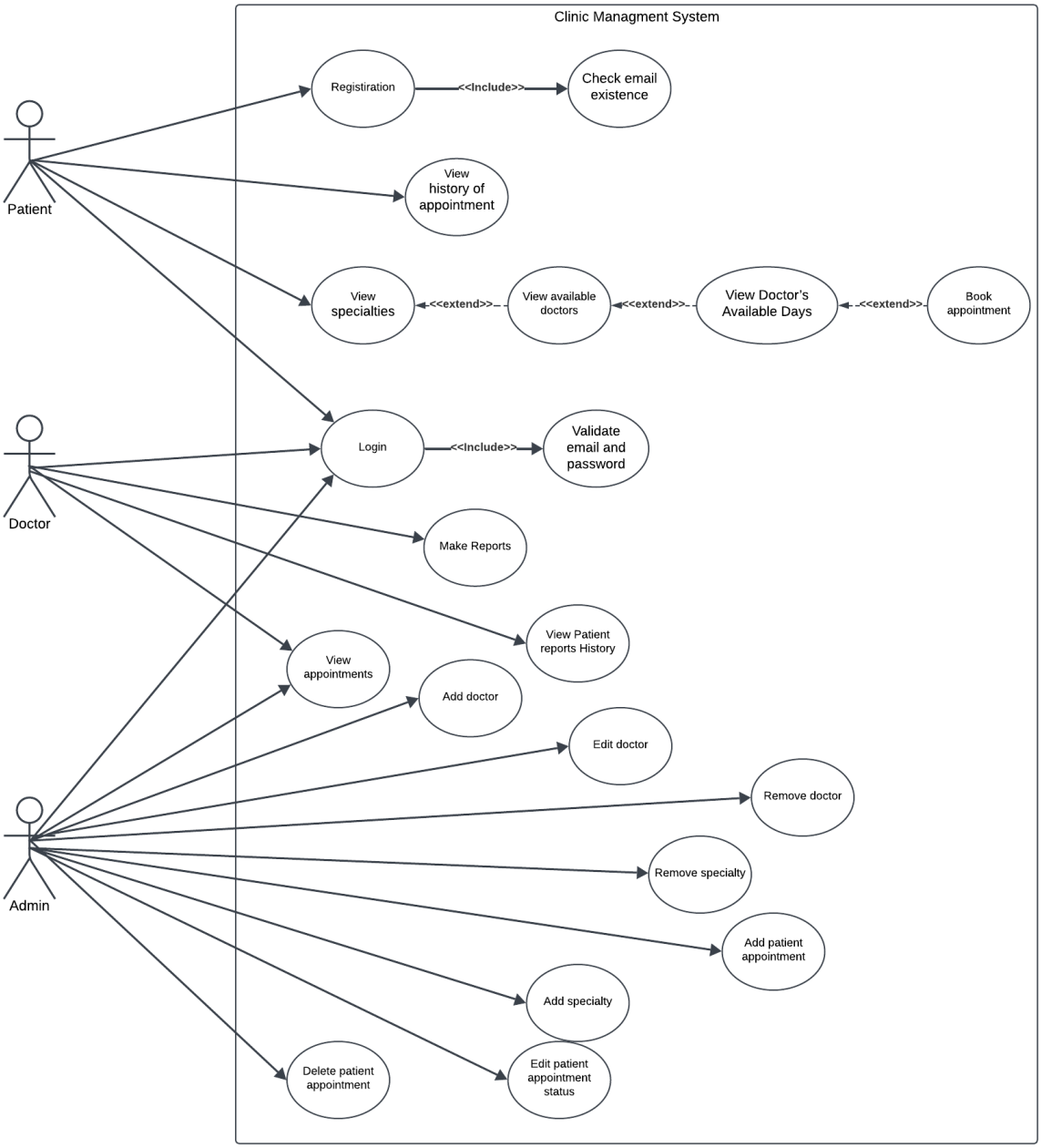
20. **View Patient Reports History**
    - Doctors can view previous appointment reports for each appointment.

Non-Functional Requirements:

1. **Reliability**
    - **Requirement**: The system must have an uptime of 99.9%, always ensuring availability for users.
    - **Implementation**:
        1. Implement error handling to manage exceptions and prevent application crashes.
        2. Ensure data integrity by implementing transaction management where necessary.
2. **Scalability**
    - **Requirement**: The system should handle increased numbers of patients, doctors, and appointments efficiently.
    - **Implementation**:
        1. Use efficient database queries and indexing to improve data retrieval times.
        2. Design the application to support modular updates and enhancements.
3. **Backup and Data Recovery**
    - **Requirement**: Ensure data is backed up regularly to prevent loss.
    - **Implementation**: Schedule daily backups to a secondary server or cloud storage (Google Drive).
4. **Performance**
    - **Requirement**: Pages should load within 0.01 seconds, and booking actions should be processed in under 0.01 seconds.
    - **Implementation**: Optimize database queries and implement indexing for frequently accessed data.

5. **Security**
    - **Requirement**: Ensure data privacy and security for patient and doctor information.
    - **Implementation**: Implement password hashing using a suitable encryption method.
6. **Usability**
    - **Requirement**: The system should have an intuitive interface, accessible to users with basic technical skills.
    - **Implementation**: Design a simple and responsive UI for desktop platforms.
7. **Maintainability**
    - **Requirement**: The system should be easy to maintain and update.
    - **Implementation**:
        1. Follow modular coding practices.
        2. Use version control systems to manage code changes and track history (Git).

Use Case Diagram:



Use Case Description:

Registration

- **Description: Allows patients to register by filling out a form and verifying their email.**
- **Actors: Patient**
- **Precondition: None**
- **Basic Steps:**
    1. **Patient fills out the registration form.**
    2. **Patient clicks the register button.**
    3. **System validates the fields.**
    4. **System checks for duplicate email.**
    5. **If email is unique, system hashes the password and inserts the user into the database.**
    6. **System shows a success message and redirects to the login page.**
- **Postcondition: Patient is registered and redirected to the login page.**
- **Exceptions:**
    - **If validation fails, the system shows an error message.**
    - **If email already exists, the system shows an error message.**

Login

- **Description: Allows users to log into the system by verifying their credentials.**
- **Actors: Patient, Doctor, Admin**
- **Precondition: User is already registered.**
- **Basic Steps:**
    1. **User enters email and password.**
    2. **User clicks login.**
    3. **System checks credentials.**
    4. **System grants access based on role.**

5. **System loads the appropriate dashboard.**

- **Postcondition: User is logged into the system with access to functionalities according to their role.**
- **Exceptions:**
    - **If fields are empty, the system shows an error message.**
    - **If email or password is incorrect, the system shows an error message.**

Make Appointment

- **Description: Allows patients to book an appointment with a doctor.**
- **Actors: Patient**
- **Precondition: Patient is logged in.**
- **Basic Steps:**
    1. **Patient selects a specialty.**
    2. **Patient selects a doctor.**
    3. **Patient selects a schedule.**
    4. **Patient submits the appointment.**
    5. **System inserts the appointment and updates the schedule limit.**
    6. **System shows a success message with a queue number.**
- **Postcondition: Appointment is booked and schedule is updated.**
- **Exceptions:**
    - **If fields are incomplete, the system shows an error message.**

Submit Report

- **Description: Allows doctors to submit reports for appointments.**
- **Actors: Doctor**
- **Precondition: Doctor is logged in and has selected an appointment.**
- **Basic Steps:**
    1. **Doctor enters report content.**
    2. **Doctor clicks submit.**
    3. **System checks if report content is provided.**
    4. **System inserts the report into the database.**

5. **System updates appointment status to 'Completed'.**
6. **System shows a success message.**

- **Postcondition: Report is submitted and appointment status is updated.**
- **Exceptions:**
  - **If report content is empty, the system shows an error message.**
  - **If insertion fails, the system shows an error message.**

Remove Specialty

- **Description: Allows admins to remove a specialty from the system.**
- **Actors: Admin**
- **Precondition: Admin is logged in.**
- **Basic Steps:**
  1. **Admin selects a specialty.**
  2. **Admin clicks remove specialty.**
  3. **System shows a confirmation dialog.**
  4. **Admin confirms removal.**
  5. **System deletes the specialty from the database.**
  6. **System shows a success message.**
- **Postcondition: Specialty is removed from the system.**
- **Exceptions:**
  - **If no specialty is selected, the system shows an error message.**
  - **If deletion fails, the system shows an error message.**

Add Specialty

- **Description: Allows admins to add a new specialty to the system.**
- **Actors: Admin**
- **Precondition: Admin is logged in.**
- **Basic Steps:**
  1. **Admin enters specialty name.**
  2. **Admin clicks add specialty.**
  3. **System validates input.**

4. **System inserts the specialty into the database.**
5. **System shows a success message.**

- **Postcondition: Specialty is added to the system.**
- **Exceptions:**
  - **If input is empty, the system shows an error message.**
  - **If insertion fails, the system shows an error message.**

Edit Doctor

- **Description: Allows admins to edit doctor details.**
- **Actors: Admin**
- **Precondition: Admin is logged in.**
- **Basic Steps:**
  1. **Admin selects a doctor and clicks edit.**
  2. **Admin modifies the form and submits.**
  3. **System validates input.**
  4. **System updates doctor details in the database.**
  5. **System shows a success message.**
- **Postcondition: Doctor details are updated.**
- **Exceptions:**
  - **If input is invalid, the system shows an error message.**

Add Doctor

- **Description: Allows admins to add a new doctor to the system.**
- **Actors: Admin**
- **Precondition: Admin is logged in.**
- **Basic Steps:**
  1. **Admin clicks add doctor.**
  2. **Admin fills out the form and submits.**
  3. **System validates input.**
  4. **System inserts doctor details into the database.**
  5. **System shows a success message.**

- **Postcondition: Doctor is added to the system.**
- **Exceptions:**
    - **If input is invalid, the system shows an error message.**

Delete Doctor

- **Description: Allows admins to delete a doctor from the system.**
- **Actors: Admin**
- **Precondition: Admin is logged in.**
- **Basic Steps:**
    1. **Admin selects a doctor and clicks delete.**
    2. **System shows a confirmation dialog.**
    3. **Admin confirms deletion.**
    4. **System deletes doctor records from the database.**
    5. **System shows a success message.**
- **Postcondition: Doctor is removed from the system.**
- **Exceptions:**
    - **If no doctor is selected, the system shows an error message.**

Set Appointment Limit

- **Description: Allows admins to set a daily appointment limit for doctors.**
- **Actors: Admin**
- **Precondition: Admin is logged in.**
- **Basic Steps:**
    1. **Admin enters appointment limit.**
    2. **Admin clicks set limit.**
    3. **System validates input.**
    4. **System updates the appointment limit in the database.**
    5. **System shows a success message.**
- **Postcondition: Appointment limit is set.**
- **Exceptions:**
    - **If input is invalid, the system shows an error message.**

View History of Appointment

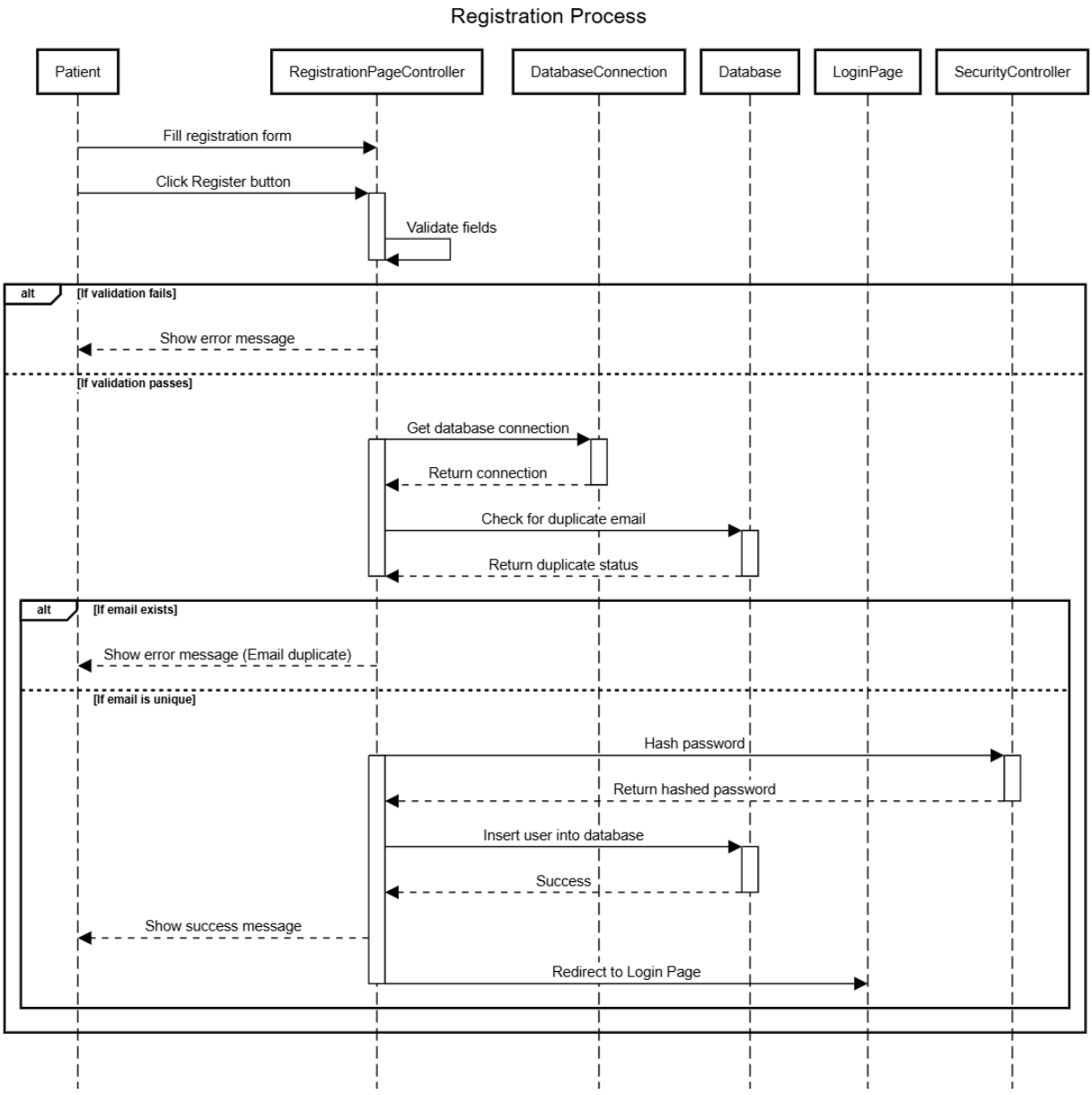
- **Description: Allows patients to view their appointment history.**
- **Actors: Patient**
- **Precondition: Patient is logged in.**
- **Basic Steps:**
    1. **Patient navigates to appointment history.**
    2. **System retrieves and displays appointment history.**
- **Postcondition: Appointment history is displayed.**
- **Exceptions:**
    - **If retrieval fails, the system shows an error message.**

View Patient Reports History

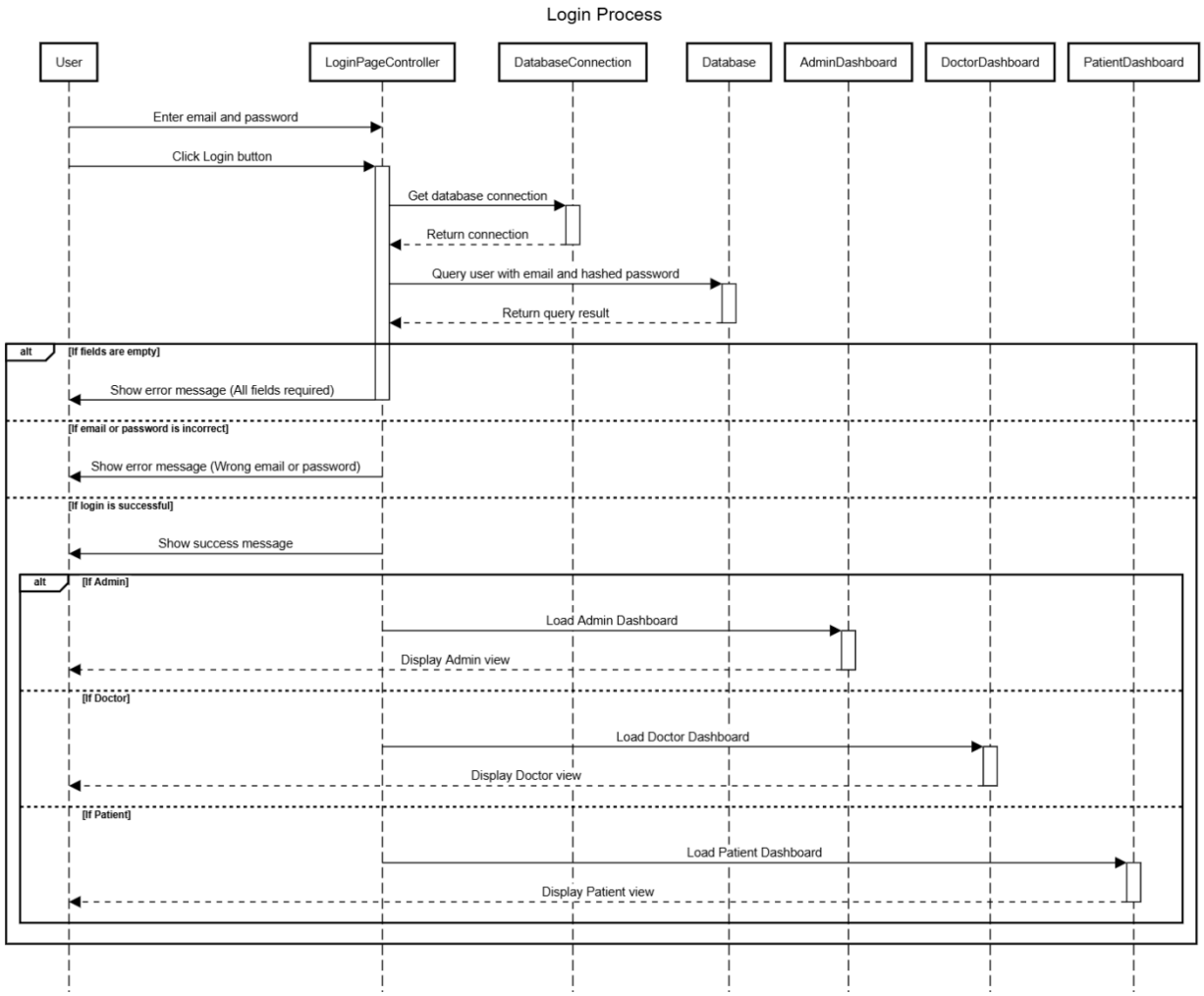
- **Description: Allows doctors to view the history of patient reports.**
- **Actors: Doctor**
- **Precondition: Doctor is logged in.**
- **Basic Steps:**
    1. **Doctor navigates to patient reports history.**
    2. **System retrieves and displays reports history.**
- **Postcondition: Reports history is displayed.**
- **Exceptions:**
    - **If retrieval fails, the system shows an error message.**

Sequence Diagrams:

1. Registration:

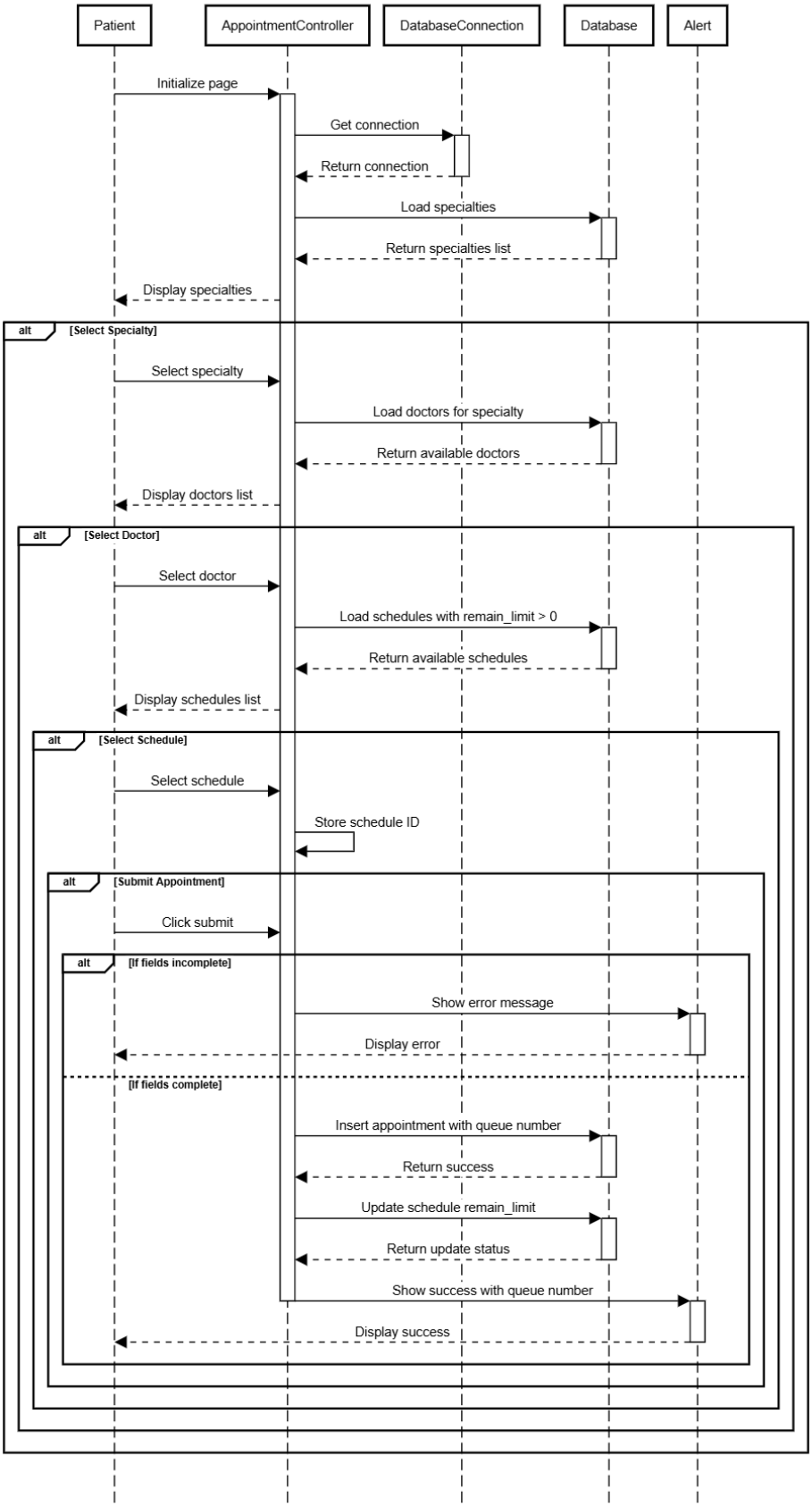


2. Login:

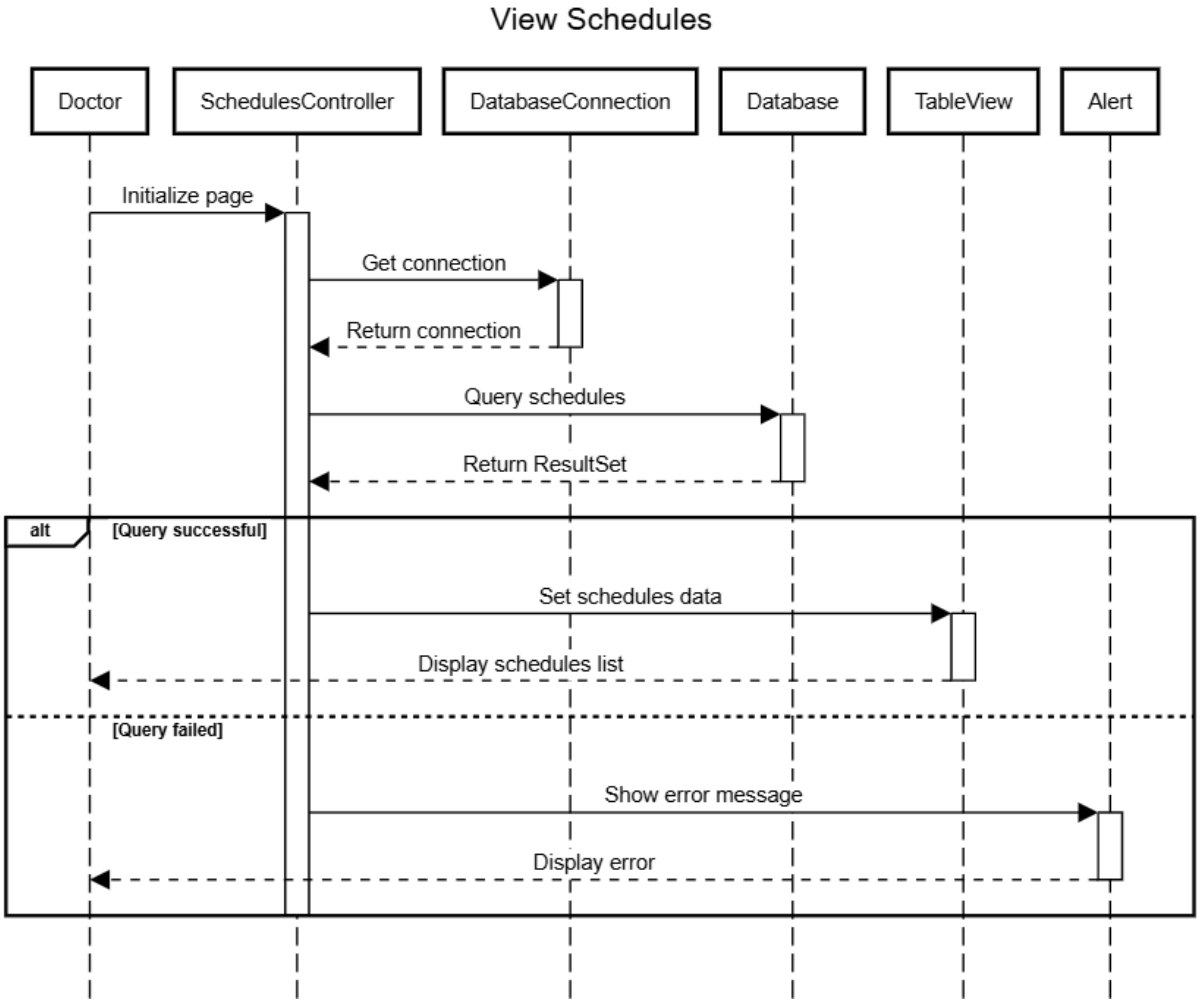

Login Process
User
LoginPageController
DatabaseConnection
Database
AdminDashboard
DoctorDashboard
PatientDashboard
Enter email and password
Click Login button
Get database connection
Return connection
Query user with email and hashed password
Return query result
alt
[If fields are empty]
Show error message (All fields required)
[If email or password is incorrect]
Show error message (Wrong email or password)
[If login is successful]
Show success message
alt
[If Admin]
Load Admin Dashboard
Display Admin view
[If Doctor]
Load Doctor Dashboard
Display Doctor view
[If Patient]
Load Patient Dashboard
Display Patient view

3. Make Appointment:

Make Appointment

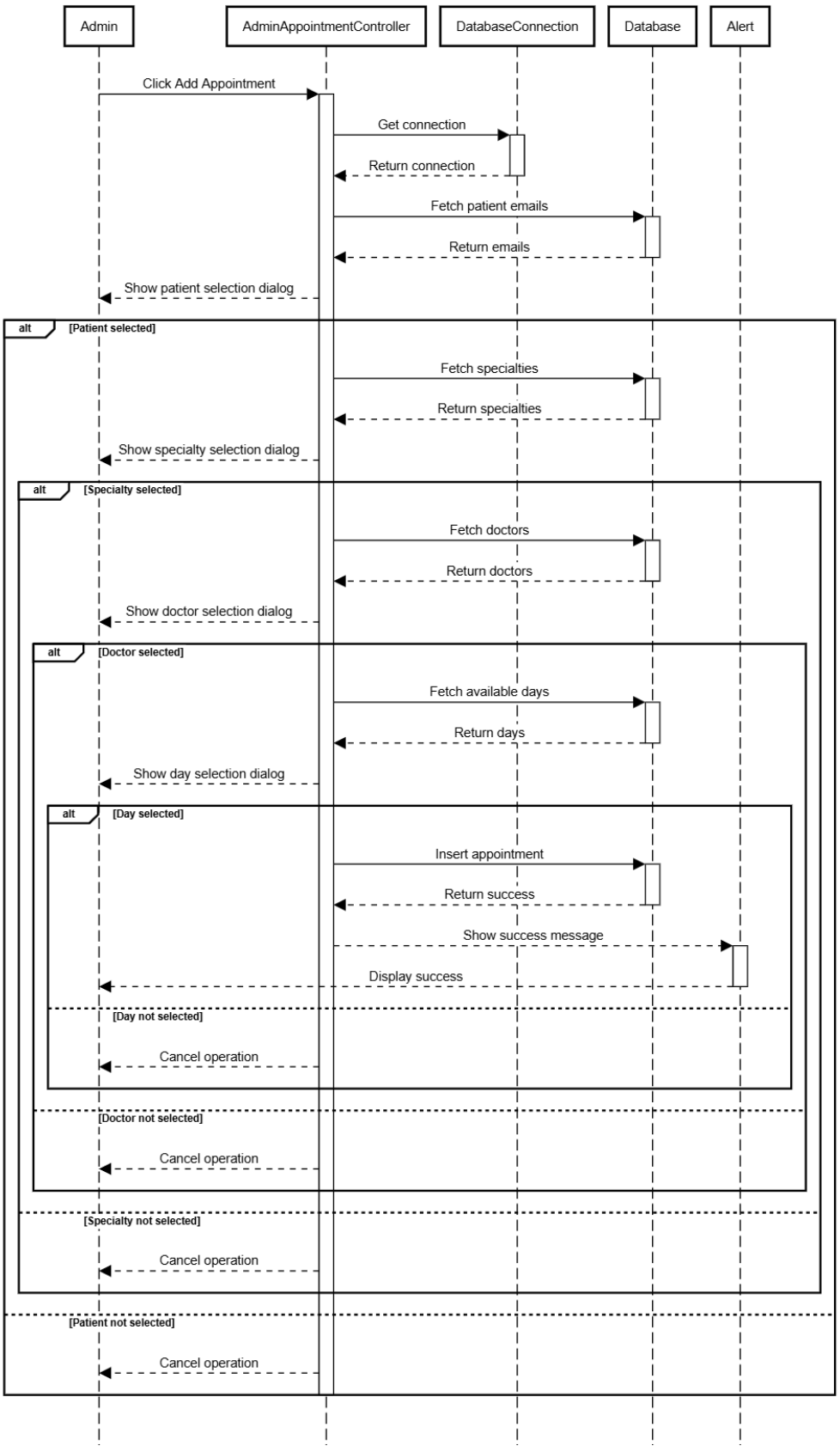


4. Doctor view Schedule:

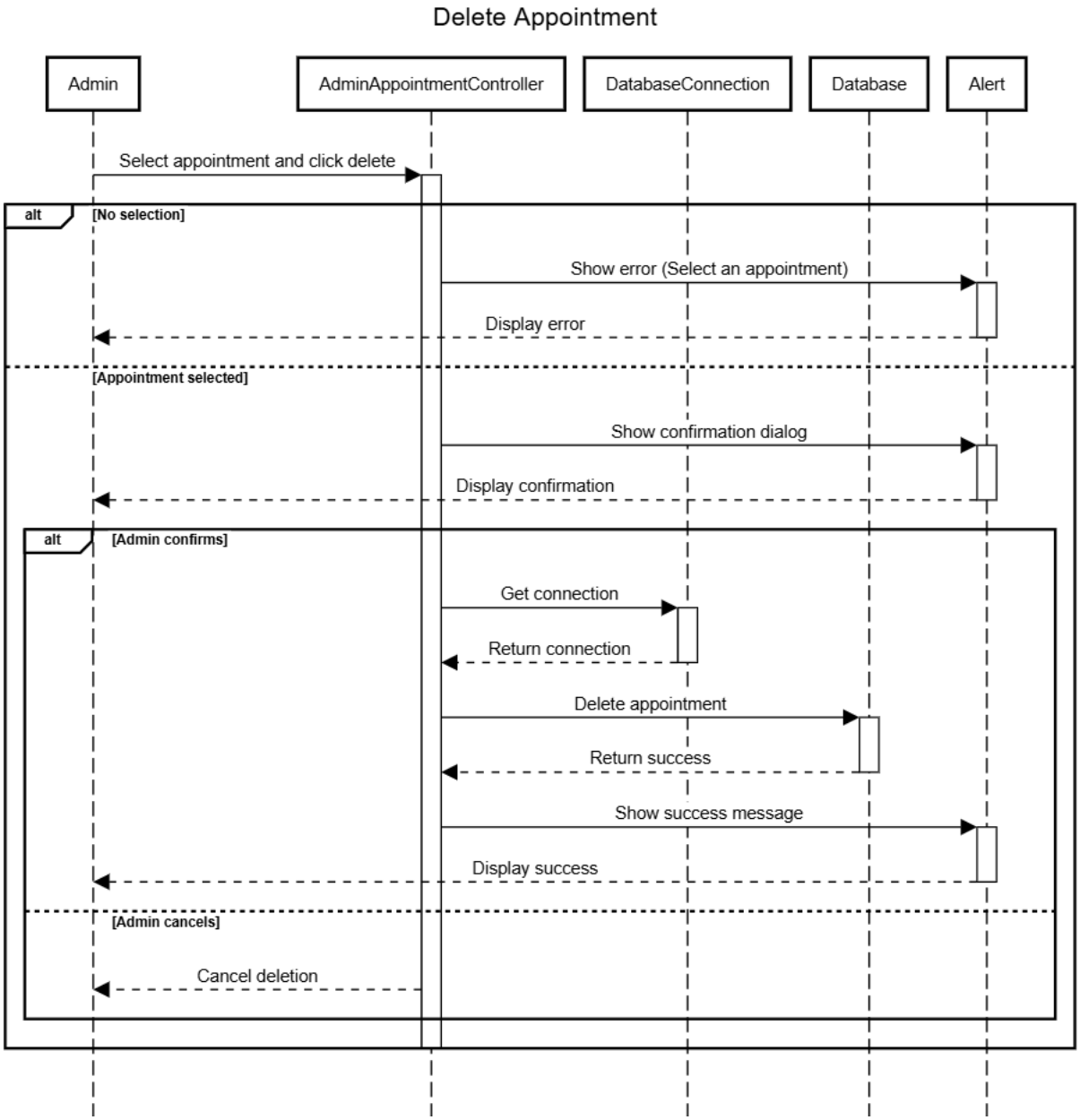


5. Add Patient Appointment:

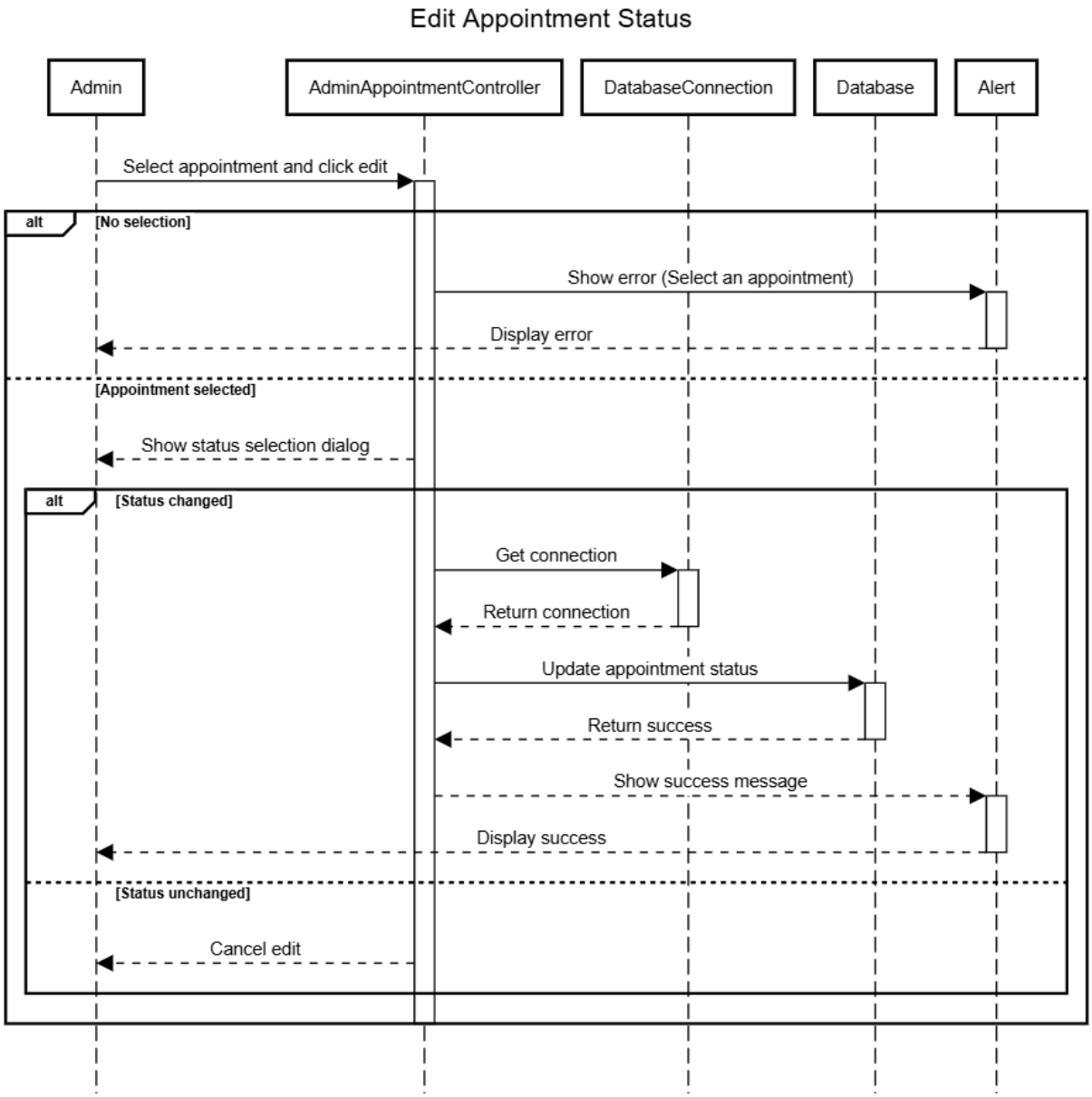
Add Appointment



6. Delete Patient Appointment:

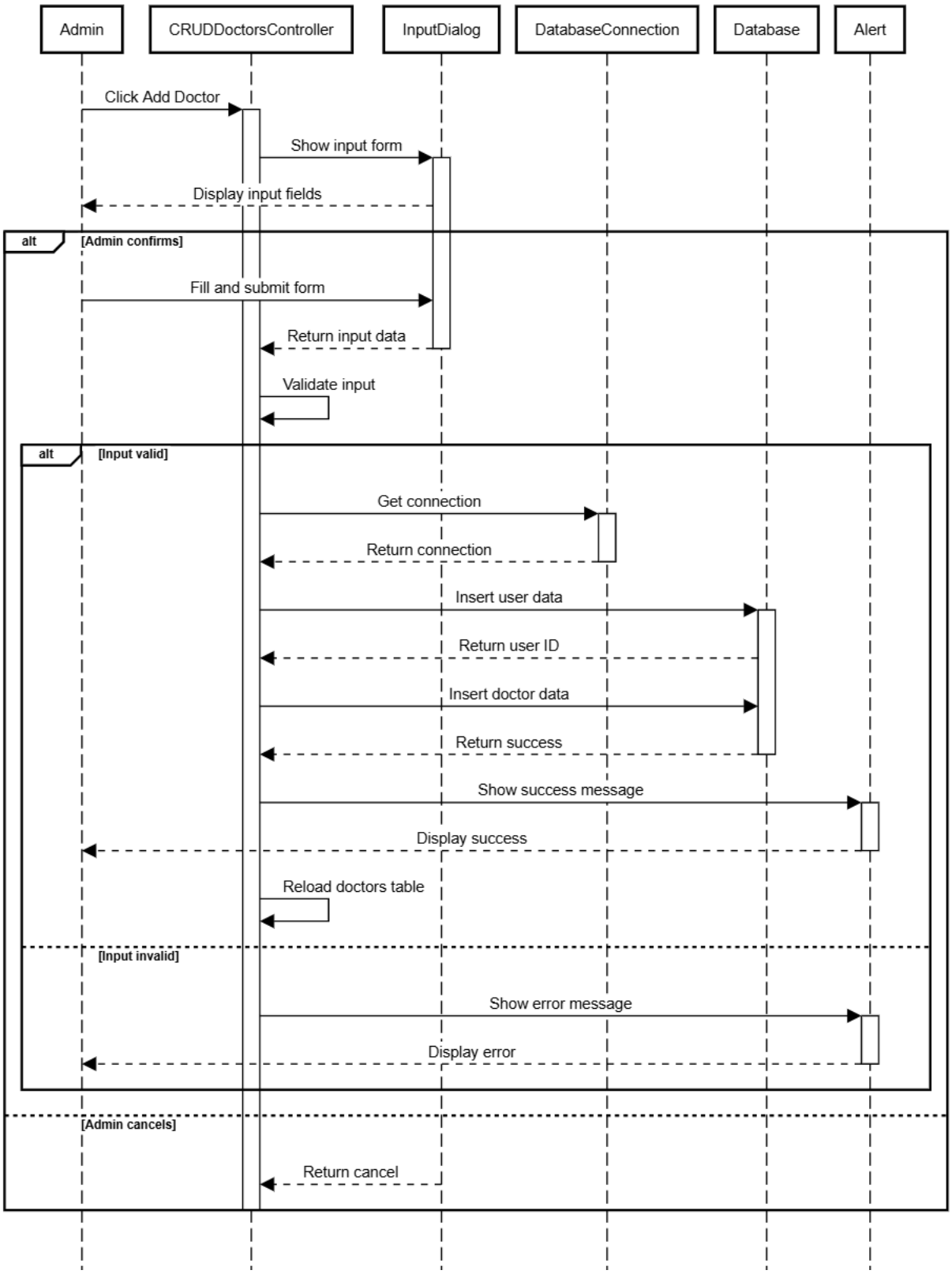


7. Edit Appointment Status:

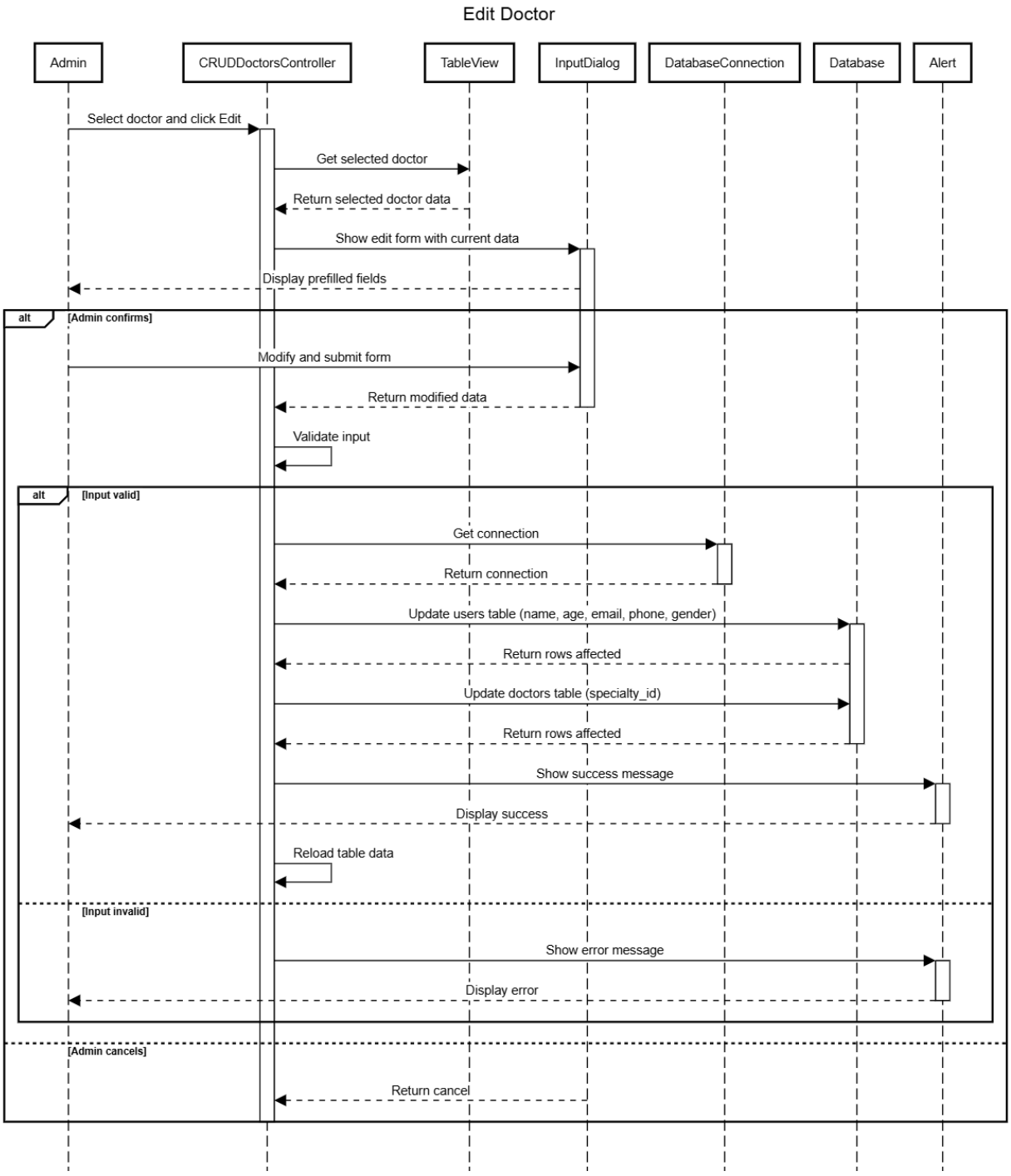


8. ADD Doctor:

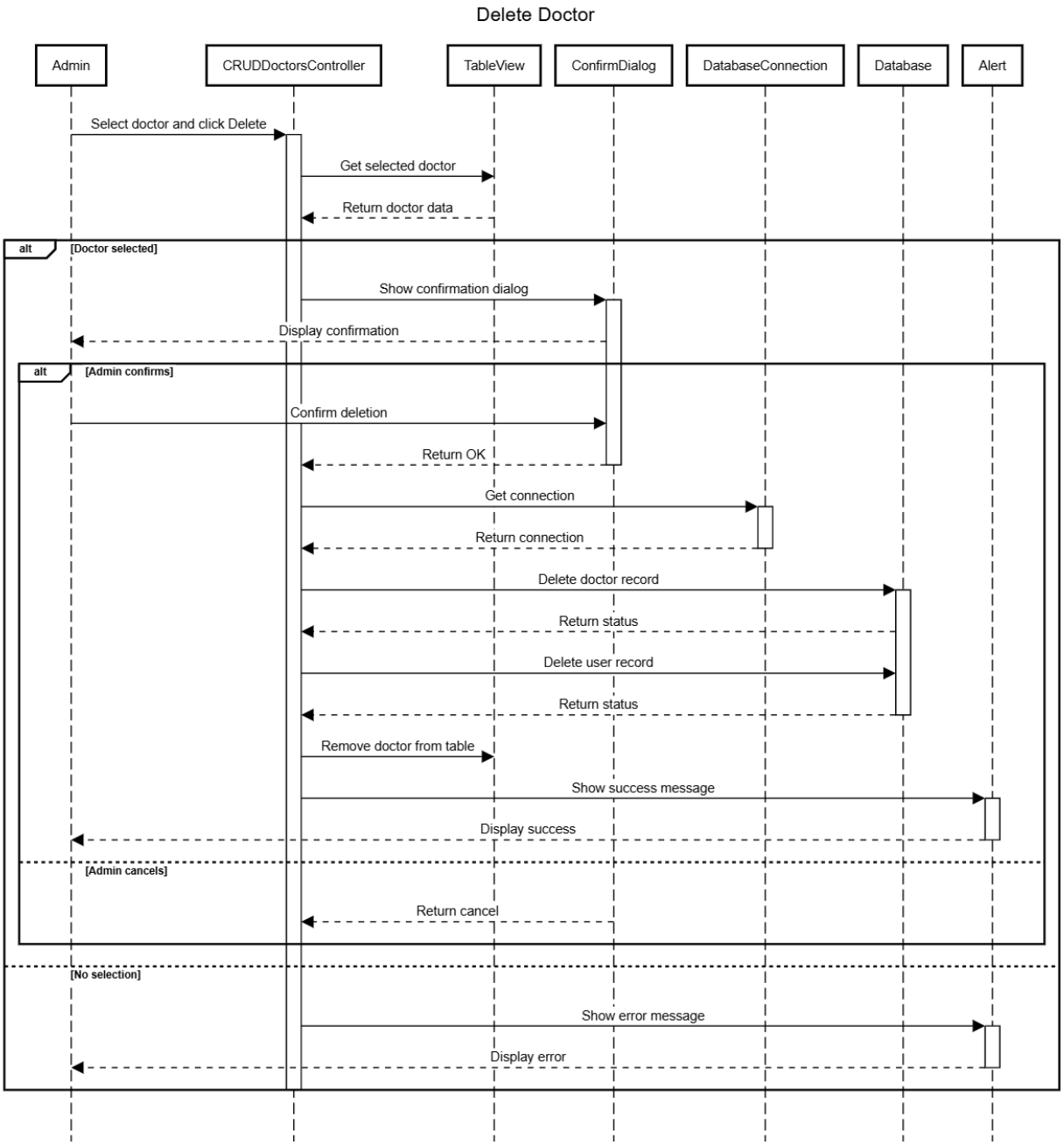
Add Doctor


Admin
CRUDDoctorsController
InputDialog
DatabaseConnection
Database
Alert
Click Add Doctor
Show input form
Display input fields
alt
[Admin confirms]
Fill and submit form
Return input data
Validate input
alt
[Input valid]
Get connection
Return connection
Insert user data
Return user ID
Insert doctor data
Return success
Show success message
Display success
Reload doctors table
[Input invalid]
Show error message
Display error
[Admin cancels]
Return cancel

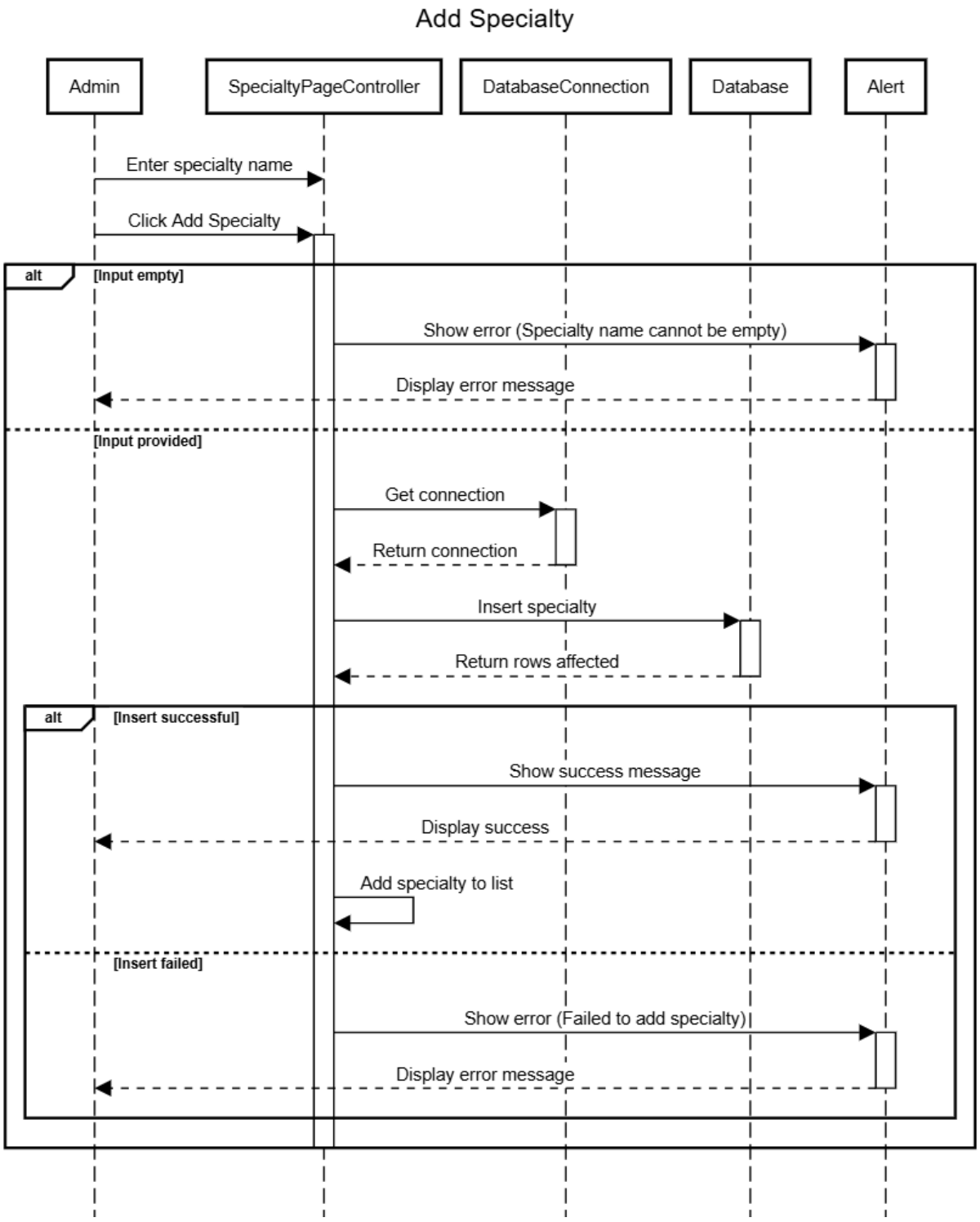
9. Edit Doctor:



10. Delete Doctor:

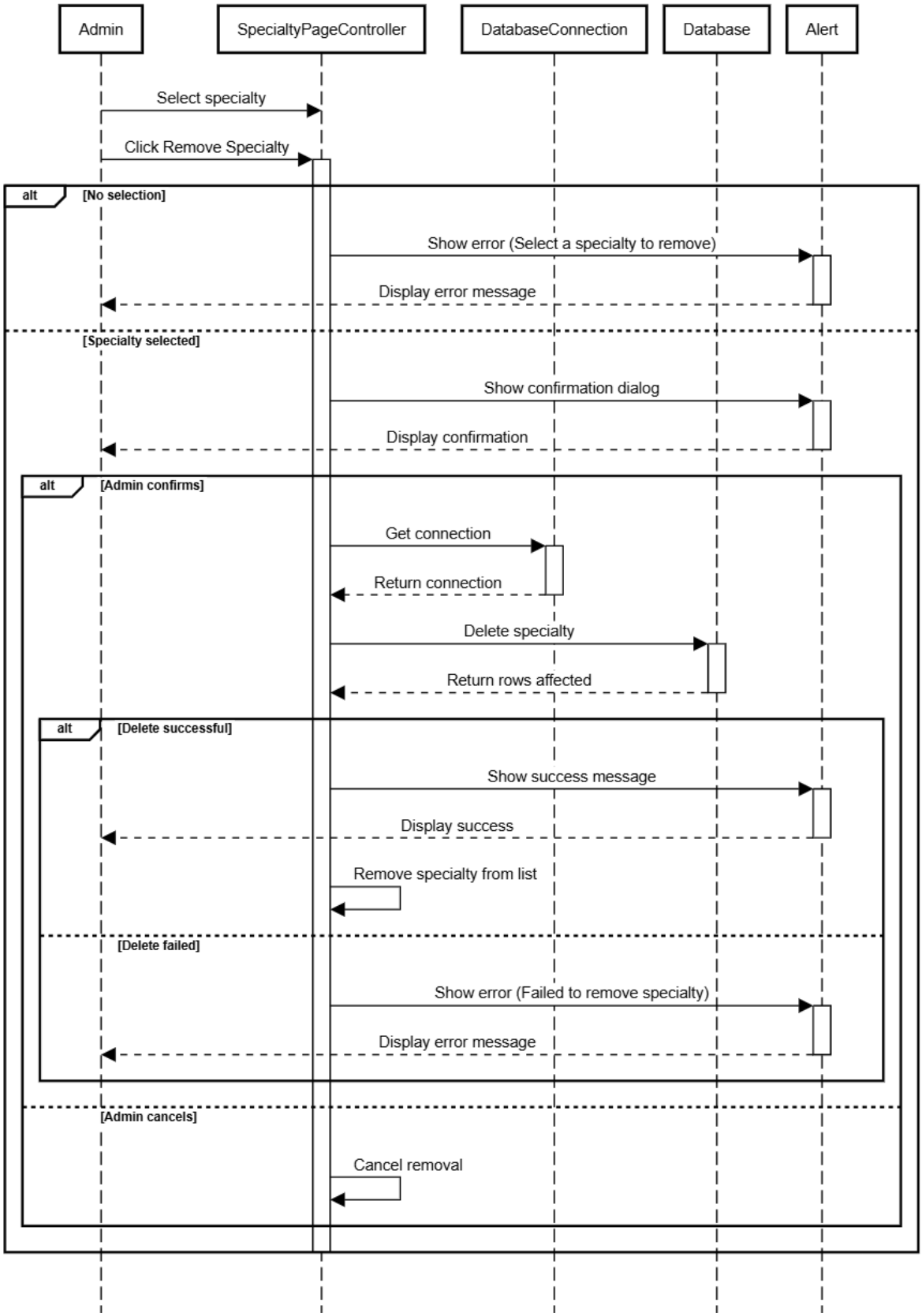


11. Add Specialty:

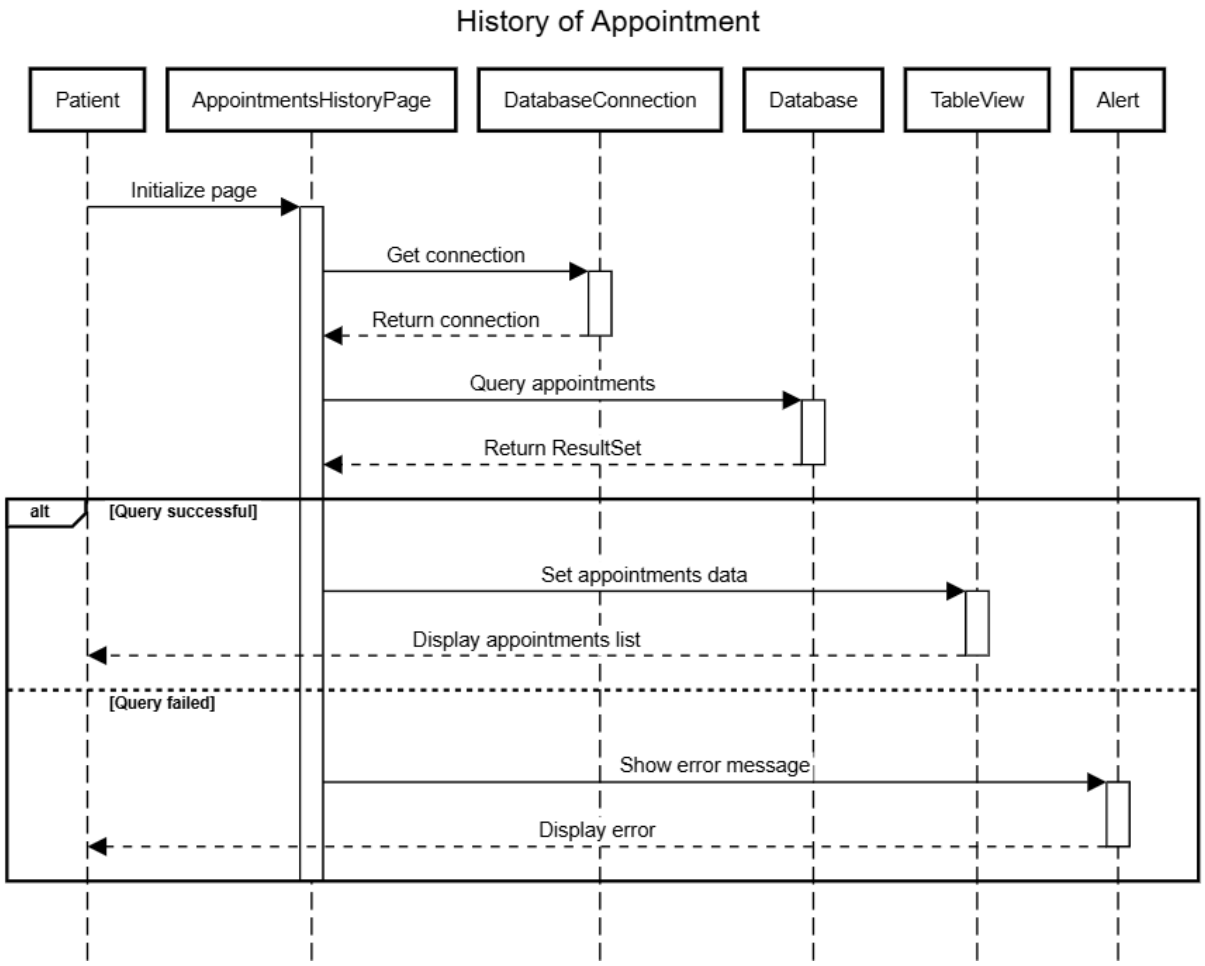


12. Remove Specialty:

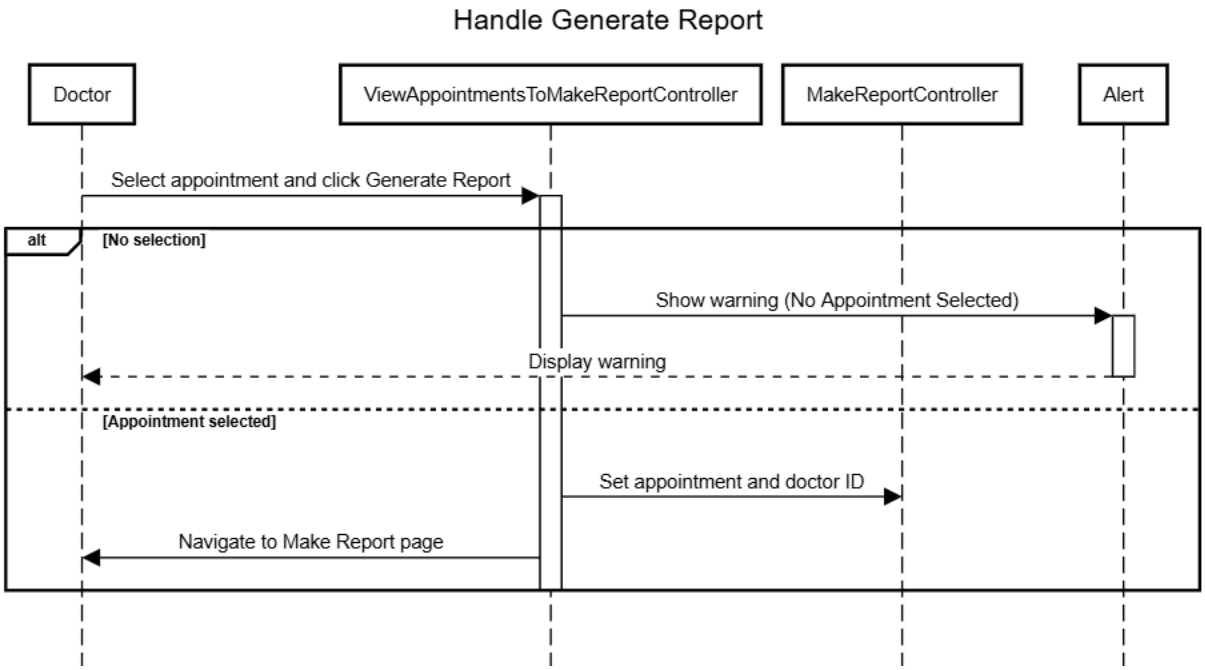
Remove Specialty



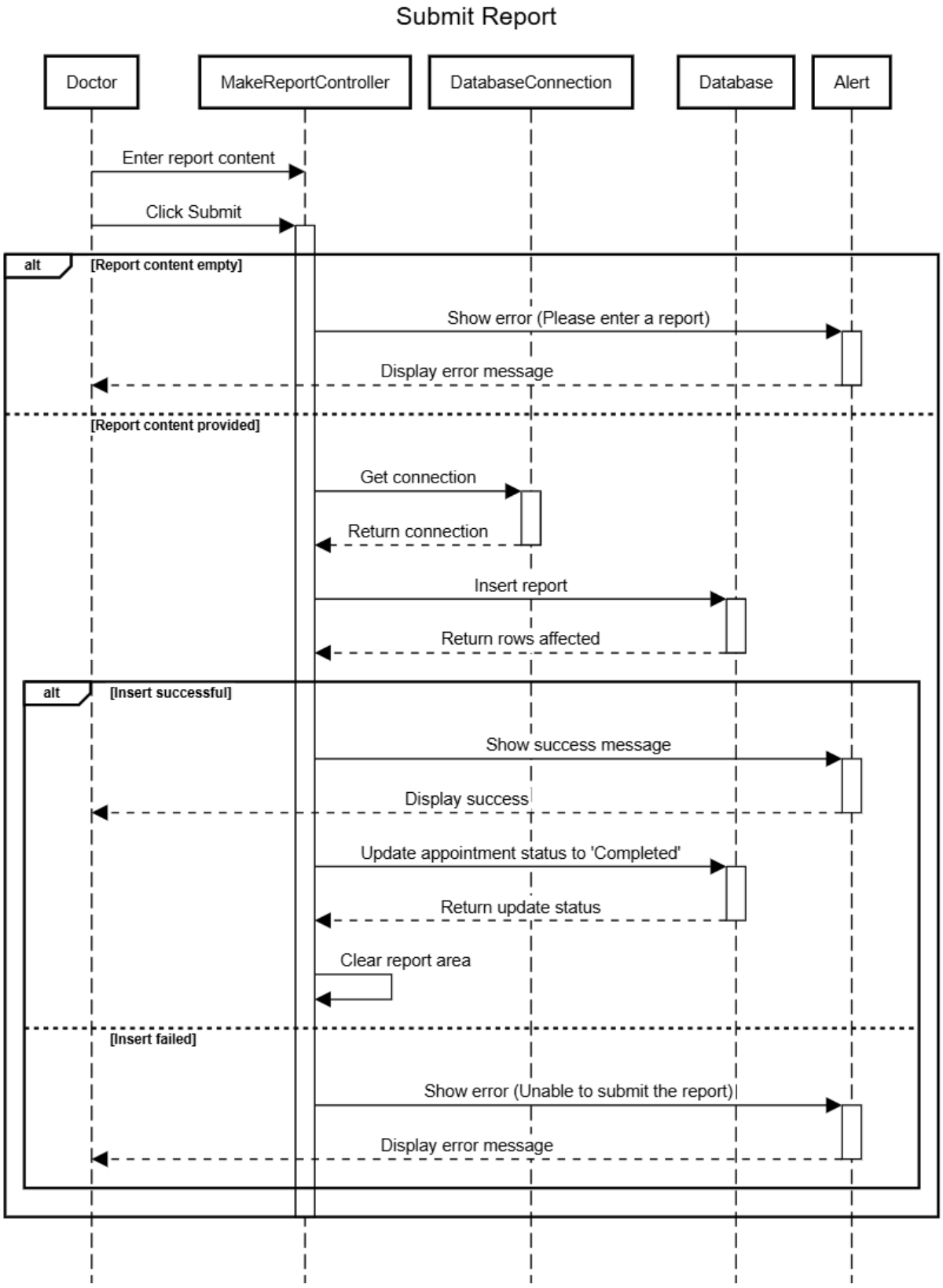
13. View Appointment History:



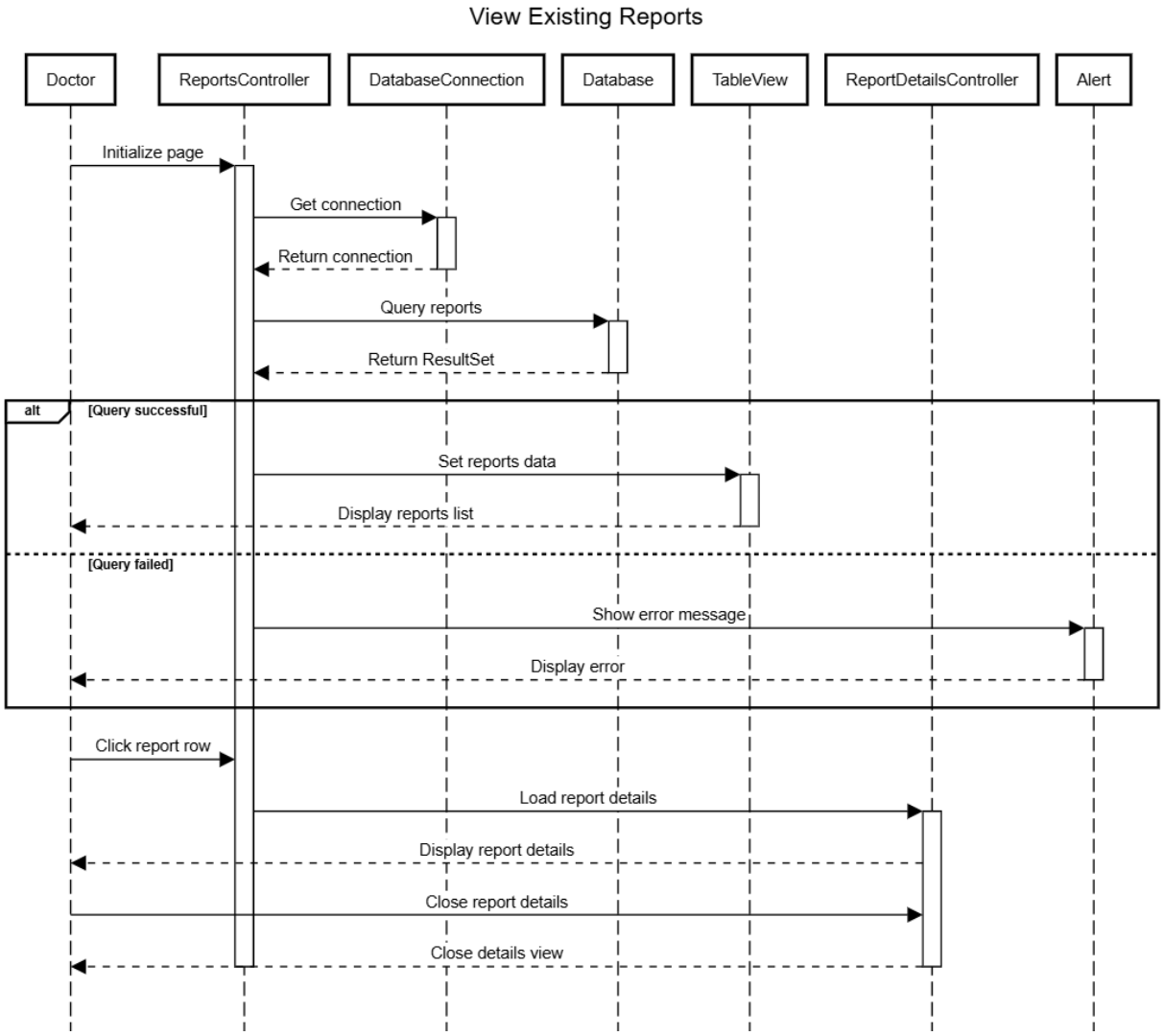
14. Generate Report:



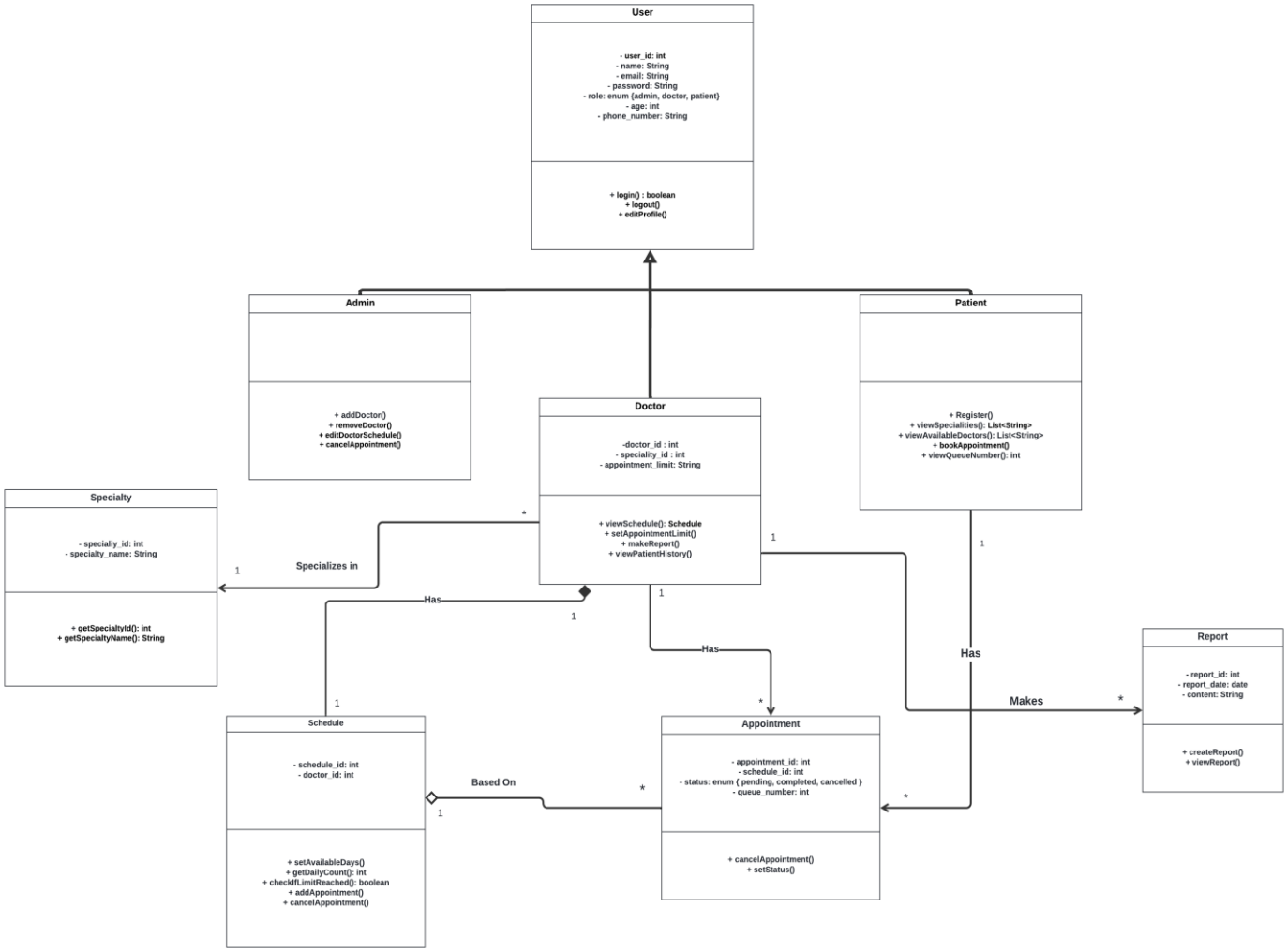
15. Make Report:



16. View History Report:



Class Diagram:



Object Diagram: