TRƯỜNG ĐẠI HỌC FPT MINISTRY OF EDUCATION AND TRAINING

Software Requirements Specification for Vaccination System

Release 1.0

Version 1.0 approved

Prepared by: Doan Phuong Anh

Process Impact

April 18, 2025

TABLE OF CONTENTS

Definition and Acronyms

Acronym	Definition
PWM	Psychology website
BA	Business Analysis
BR	Business Rule
ERD	Entity Relationship Diagram
GUI	Graphical User Interface
PM	Project Manager
SDD	Software Design Description
SPMP	Software Project Management Plan
SRS	Software Requirement Specification
UAT	User Acceptance Test
UC	Use Case
API	Application Program Interface

I. Project Introduction

1. Overview

1.1 Project Information

- Project Name: Vaccination System
- Project Type: Web Application
- Development Approach: ASP.NET Core with C#

1.2 Project Team

- Team Member:	Trần Duy Khải:	SE183260
	Nguyễn Song Kha:	SE171700
	Đoàn Phương Anh:	SE183812
	Ngô Lê Thảo Nguyên:	SE183870
	Hà Đức Bảo:	SE172197
	Nguyễn Minh Thư:	SE183335
- Supervisor:	Dr. Phạm Thanh Trí	

2. Product Background

Currently, managing vaccination information poses significant challenges due to the limitations of traditional systems, including data loss, difficulties in retrieving information, and complex registration procedures. This issue becomes particularly critical in the context of large-scale vaccination campaigns, which require an efficient system for data tracking and management.

The SWD Vaccination System was initiated to address these challenges by developing a modern vaccination management platform. This platform aims to optimize workflows for healthcare personnel while providing a more convenient experience for the public.

3. Existing Systems

Currently, many vaccination management systems rely on manual processes or fragmented software solutions with limited integration.

Reference systems: VNVC, CDC, and other electronic vaccination applications.

4. Business Opportunity

Target Market: Hospitals, clinics, healthcare centers, and regulatory health agencies.

Current Issues:

- Manual processes are time-consuming and prone to errors.
- Existing systems fail to adequately support vaccination scheduling and patient tracking needs.
- Lack of data integration between different healthcare entities.

Our Solution:

- Develop a digital vaccination management system that ensures secure data tracking and storage.

- Integrate appointment scheduling and automated reminders.
- Provide a user-friendly interface to enhance the overall user experience.

Market Trends: The digitalization of healthcare has become an inevitable trend, with many countries adopting technology to enhance the efficiency of healthcare management.

5. Software Product Vision

The SWD Vaccination System aims to develop a comprehensive vaccination management system that enhances healthcare facilities' efficiency while providing users a seamless experience.

The product will help:

- Healthcare professionals efficiently manage vaccination schedules and records.
- Individuals conveniently book appointments and receive timely reminders.
- Health authorities gain a robust tool for tracking and reporting vaccination data.

This project aligns with the digital transformation trends in healthcare and has the potential for expansion into other countries.

6. Project Scope & Limitations

Project Scope

- **Key Features:**
  - Registration and management of vaccination schedules.
  - Management of patient records and vaccination history.
  - Automated reminders via SMS or email.
  - Reporting and analytics for vaccination data.
  - Integration with existing healthcare management systems.
- **Platform:** Web-based.
- **Target Users:** Healthcare professionals, the public, and health regulatory authorities.

Project Limitations

- Does not include medical diagnosis or prescription functionalities.
- Limited by budget and implementation timeframe, estimated at [number of months].
- Must comply with healthcare data security regulations such as [HIPAA/GDPR or local regulations].

II. Project Management Plan

1. Overview

1.1 Scope & Estimation

Risk	Complexity	Estimated Effort (man-days)
Account Management & Authentication	Simple	5

Child Management	Simple	4
Vaccination Schedule Management	Medium	8
Vaccination Service Package Management	Simple	5
Payment	Complex	4
Holiday Management	Simple	3
Notification Management	Simple	4
Feedback & Review Management	Simple	3
WareHouse Management	Complex	10

1.2 Project Objectives

The primary objective of the SWD Vaccination System is to develop an efficient and reliable vaccination management system that optimizes healthcare workflows. The project aims to:

- **Quality:** Ensure a secure and user-friendly system that meets healthcare industry standards.
- **Time:** Deliver the complete system within 3 months.
- **Cost:** Allocate effort distribution for project activities as follows:
  - Requirement Analysis: Identify system requirements and functionalities.
  - Design: Design the user interface, database, and system architecture.
  - Coding: Develop the code and implement system features.
  - Testing: Verify and fix errors before deployment.
  - Project Management: Coordinate, monitor progress, and ensure timely project completion.

1.3 Project Risks

Risk	**Description**	**Mitigation Plan**
Data Security	Risk of unauthorized access to sensitive patient data	Implement encryption and secure authentication mechanisms
Schedule Risks	The technology is more complex than expected, requiring more development time.	Develop a detailed plan with specific timelines. Have contingency plans in place to mitigate schedule risks.

Scope Creep	Additional requirements leading to project delays	Define clear scope and apply change management processes

2. Management Approach

2.1 Project Process

The project will follow the Agile Scrum methodology over a 10-week period, divided into the following phases:

Weeks 1 - 3: System Design & Requirement Analysis

During this phase, the team will focus on analyzing and designing the system to ensure a clear development roadmap. Key activities include:

- Requirement Analysis: Identify system functionalities and define user needs.
- System Design: Create essential diagrams such as:
  - Entity-Relationship Diagram (ERD) for database structure.
  - Database Schema Design to define data storage.
  - Use Case Diagrams to visualize system interactions.
  - High-level System Architecture to outline system components and communication flow.
- Documentation: Prepare technical specifications and requirement documents.

Weeks 4 - 8: Development & Implementation

The system will be developed in iterations using Agile Sprints to ensure incremental progress. Key activities include:

- Sprint Planning: Break down features into manageable tasks for each sprint.
- Feature Development: Implement functionalities according to the predefined requirements.
- Code Review & Continuous Integration: Ensure code quality and smooth integration of components.
- Midway Testing: Conduct unit testing for individual modules to identify early-stage issues.

Week 9: System Testing & Bug Fixing

Before deployment, the system will undergo comprehensive testing to ensure functionality, security, and performance. Key activities included.

- Functional Testing: Verify that each feature works as intended.
- Integration Testing: Ensure smooth interaction between different modules.
- Performance Testing: Assess system response times and scalability.
- Bug Fixing & Optimization: Resolve any identified issues before final deployment.

Week 10: Final Review & Presentation

The final phase will focus on:

- User Acceptance Testing (UAT): Gather stakeholder feedback.

- Final Documentation & Report: Summarize system design, implementation, and testing results.
- Project Presentation & Demonstration: Showcase the completed system with a live demo.

This structured 10-week approach ensures a well-planned, efficient, and high-quality project development lifecycle.

2.2 Quality Management

To maintain high-quality standards, the project will:

- Follow coding best practices.
- Conduct regular code reviews.
- Implement automated testing.
- Perform user acceptance testing (UAT) before deployment.

2.3 Training Plan

The team will organize training sessions for members who require additional knowledge in:

- Security compliance and data privacy.
- Cloud-based deployment strategies.
- Performance optimization techniques.

3. Project Deliverables

- **Internal Deliverables:** Requirement documents, system architecture, test cases, progress reports.
- **External Deliverables:** Web application, user manuals, API documentation.

4. Responsibility Assignments

5. Project Communications

Role	Responsibilities
Backend Developer	Develop APIs and manage database interactions
Frontend Developer	Implement UI/UX and client-side logic

The project team will use:

- **Zalo** for daily communications.
- **Github** for task tracking and project management.
- **Google Docs** for document sharing.
- **Google Sheet** for testing documentation

6. Configuration Management

6.1 Document Management

- Use **Google Docs** to store and track document versions.

6.2 Source Code Management

- Use **GitHub** for version control and code collaboration.

6.3 Tools & Infrastructure

Tool	**Purpose**
GitHub	Source code management
Azure	Containerization
SQL Server	Database management

III. Software Requirement Specification

1. Product Overview

The Vaccine Management System is a comprehensive digital solution designed to streamline the appointment scheduling, vaccine administration, and invoicing process for healthcare providers. The system aims to enhance operational efficiency, improve patient experience, and ensure accurate record-keeping for vaccine doses, health records, and payments.

Anticipated Users

- Customers: Individuals booking vaccination appointments for themselves or their children.
- Staff (Receptionist, Data Entry, Nurse): Support personnel who manage appointment scheduling, check-ins, and be responsible for administering vaccines and updating health records.
- Administrators: Users with full access to manage system configurations, reports, and user permissions.

System Environment

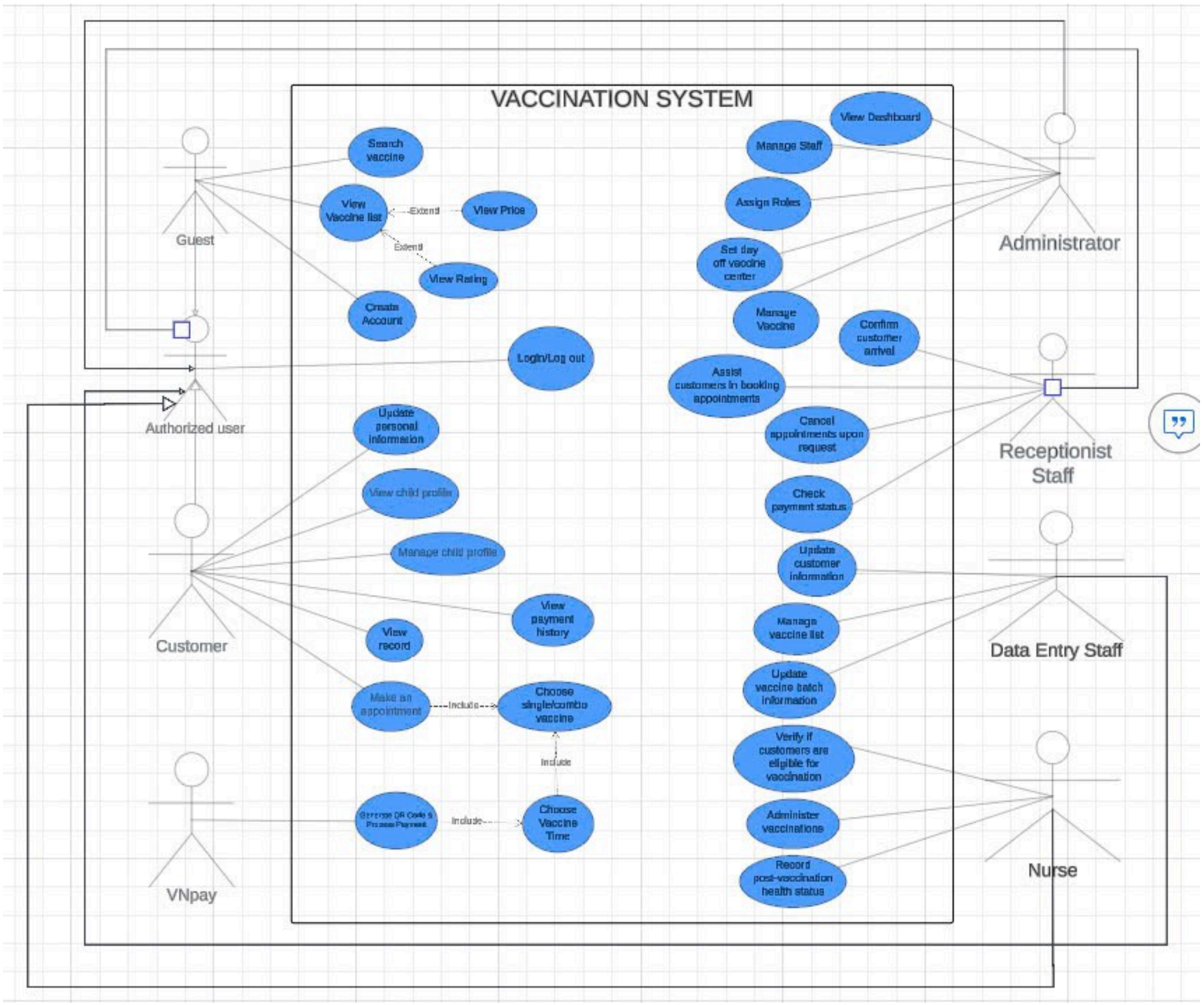
The system is designed as a web-based application, ensuring accessibility via modern browsers across different devices. It integrates with external systems for secure payment processing, electronic medical records, and inventory tracking of vaccines.

Known Constraints, Assumptions, and Dependencies

- Constraints:
    - Requires stable internet connectivity for real-time updates and transactions.
    - Must comply with healthcare regulations and data privacy laws (e.g., HIPAA, GDPR).
- Assumptions:
    - Users have basic digital literacy to navigate the system.

    - All vaccine-related data, including batch information, is accurately recorded and maintained.
- Dependencies:
    - Integration with external payment gateways for invoice processing.
    - Compatibility with healthcare databases for retrieving doctor and patient records.
    - Connection to inventory management systems for tracking vaccine stock levels.

2. User Requirements



2.1 Guest Use Cases

- Search for Vaccines: Unregistered users can look up information about available vaccines.
- View Vaccination Centers: Browse and review healthcare facilities.
- Register an Account: Create an account to become a customer.

2.2 Customer Use Cases

- Login/Logout: Customers can log in and log out of the system.
- Update Personal Profile: Modify personal details.
- Book Vaccination Appointments: Select a vaccine type, healthcare facility, and appointment time.

- Make Payments: Complete payments for vaccination services.
- View Vaccination History: Review past vaccination records.

2.3 Administrator Use Cases

- Manage Staff: Add, edit, or remove staff members.
- Approve Appointments: Confirm or reject customer appointment requests.
- Verify Payments: Check and approve payment transactions.
- View Reports: Access reports on appointments, staff, and customers.

2.4 Receptionist Staff Use Cases

- Check Appointment Status: Confirm or update the status of customer appointments.
- Verify Customer Arrival: Validate customer identity and confirm arrival for vaccination.
- Cancel Appointments: Cancel appointments upon customer request or valid reasons.

2.5 Data Entry Staff Use Cases

- Input Medical Information: Update customers' health information before and after vaccination.
- Input relating vaccine information: Vaccine, Vaccine Batch, Vaccine Combo

2.6 Nurse Use Cases

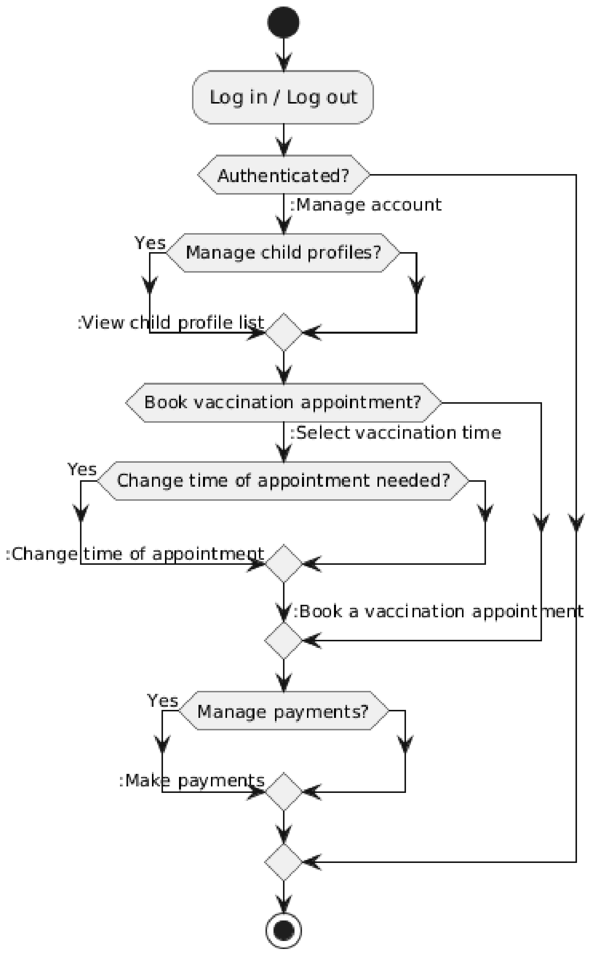
- Verify Patient Information: Confirm customer details before administering the vaccine.
- Assess Health Condition: Evaluate the customer’s health before vaccination.
- Administer Vaccination: Perform the vaccination and update the system accordingly.

3. Functional Requirements

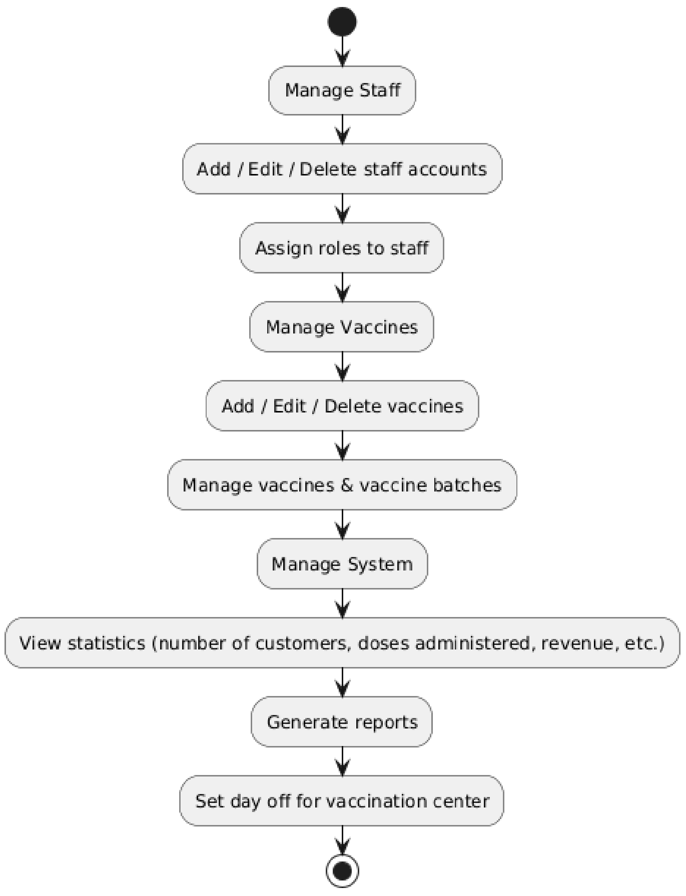
3.1 System Functional Overview

3.1.1 Screen flow

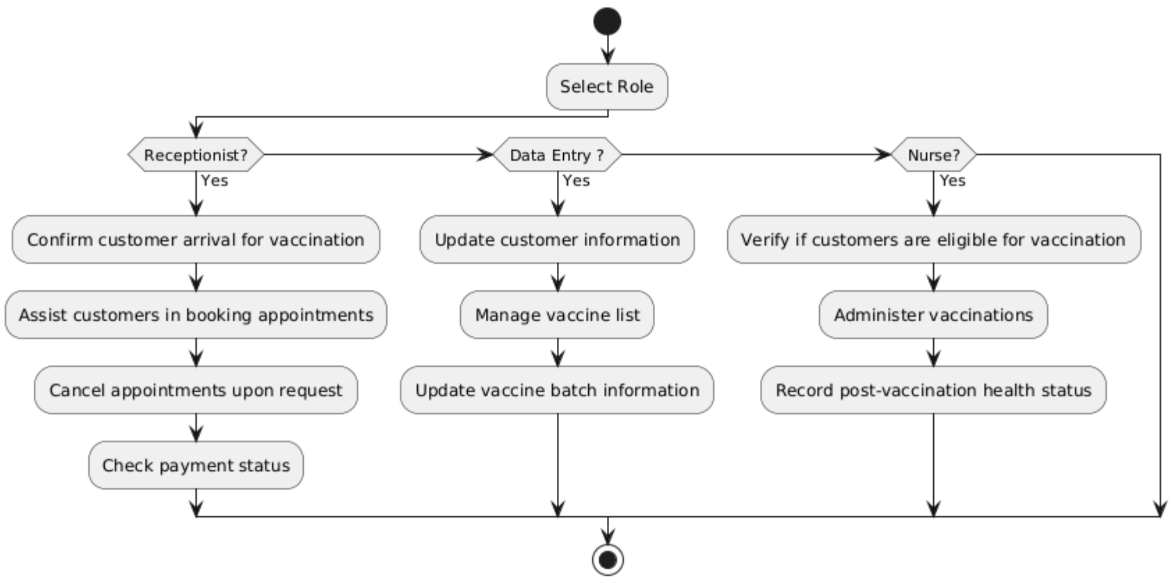
3.1.1.1 Screen flow for customer



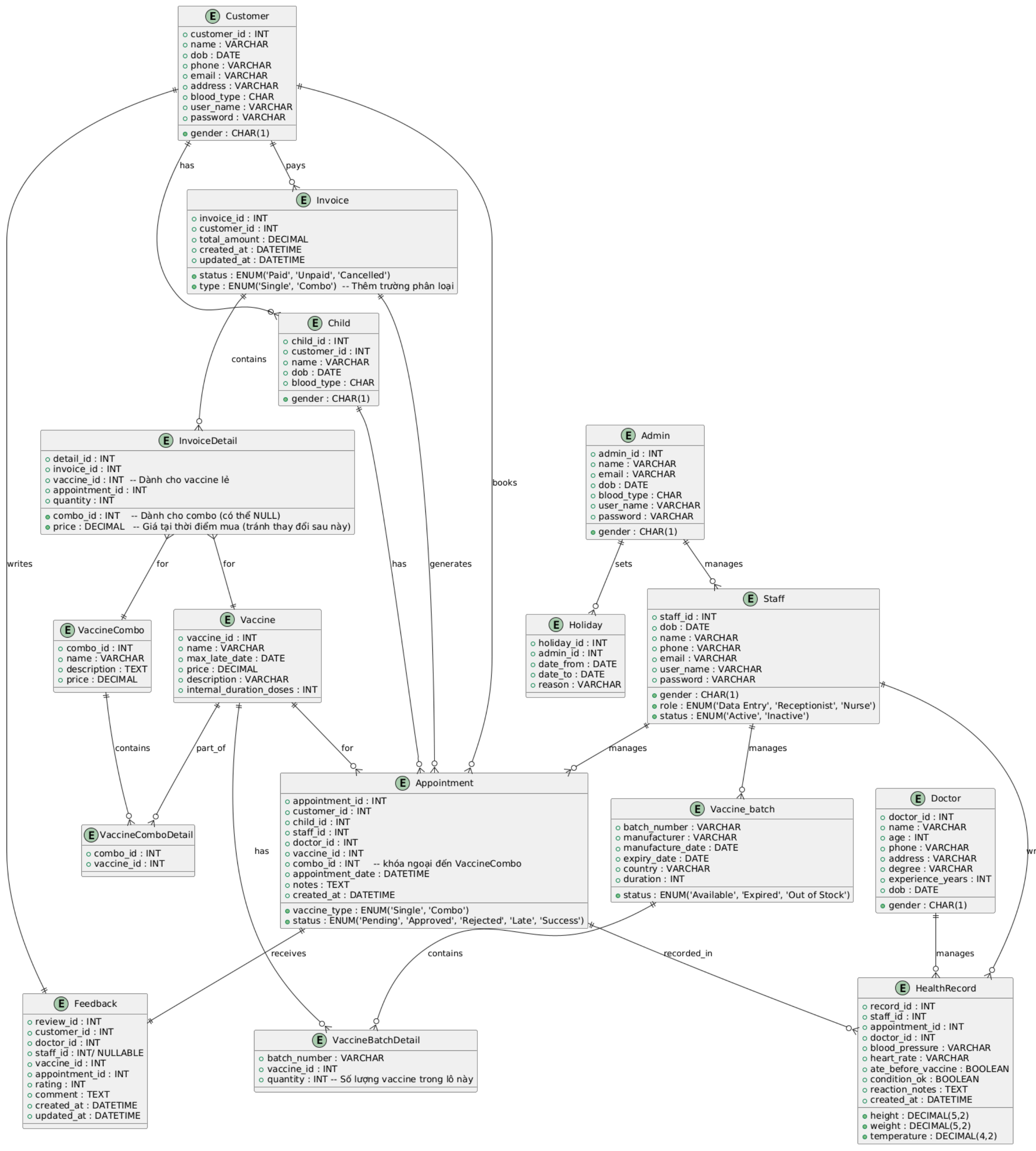
3.1.1.2 Screen flow for admin


Manage Staff
Add / Edit / Delete staff accounts
Assign roles to staff
Manage Vaccines
Add / Edit / Delete vaccines
Manage vaccines & vaccine batches
Manage System
View statistics (number of customers, doses administered, revenue, etc.)
Generate reports
Set day off for vaccination center

3.1.1.3 Screen flow for staff


Select Role
Receptionist?
Yes
Data Entry ?
Yes
Nurse?
Yes
Confirm customer arrival for vaccination
Assist customers in booking appointments
Cancel appointments upon request
Check payment status
Update customer information
Manage vaccine list
Update vaccine batch information
Verify if customers are eligible for vaccination
Administer vaccinations
Record post-vaccination health status

3.1.2 ERD



3.2 User Management

This feature allows administrators to manage user accounts, including staff and customers.

3.2.1 Create User Account

- Allows admins to create new user accounts.
- Inputs: Full Name, Email, Phone Number, Role (Customer, Staff, Admin).
- Validation: Email format, unique phone number.
- Outputs: Success or failure message.

3.2.2 Edit User Account

- Allows modification of existing user details.

- Editable fields: Name, Email, Role.
- Validation: Ensure role-based restrictions.
- Outputs: Success or failure message.

3.2.3 Delete User Account

- Allows admins to remove users from the system.
- Validation: Admin-only access.
- Outputs: Confirmation prompt, success/failure message.

3.2.4 Assign Roles to Staff

- Assign different roles to staff (Receptionist, Nurse, Data Entry).
- Inputs: Staff ID, Role Selection.
- Outputs: Role successfully updated.

3.3 Appointment Management

This feature supports scheduling, rescheduling, and managing vaccination appointments.

3.3.1 Book Appointment

- Customers select a vaccine and schedule an appointment.
- Inputs: Vaccine selection, preferred date/time.
- Validation: Check vaccine availability.
- Outputs: Appointment confirmation.

3.3.2 Change Appointment Time

- Allows customers to reschedule appointments.
- Inputs: Appointment ID, new date/time.
- Validation: Time slot availability.
- Outputs: Updated appointment details.

3.3.3 Cancel Appointment

- Customers or staff can cancel an appointment.
- Inputs: Appointment ID.
- Validation: Ensure valid cancellation window.
- Outputs: Cancellation confirmation.

3.4 Payment Processing

Handles payment methods, invoices, and transaction history.

3.4.1 View Invoice

- Customers can view generated invoices.
- Inputs: Customer ID.
- Outputs: List of unpaid invoices.

3.4.2 Make Payment

- Customers pay via online (credit/debit) or offline (cash at clinic).

- Inputs: Payment method selection, invoice ID.
- Validation: Check payment details.
- Outputs: Payment confirmation.

3.4.3 View Payment History

- Customers and staff can check past payments.
- Inputs: User ID.
- Outputs: List of previous transactions.

3.5 Vaccine Management

Enables staff to manage vaccine inventory and batch details.

3.5.1 Add/Edit/Delete Vaccine

- Data Entry Staff can update vaccine details.
- Inputs: Vaccine Name, Type, Dosage, Expiry Date.
- Outputs: Updated vaccine list.

3.5.2 Manage Vaccine Batches

- Track vaccine batch numbers and stock levels.
- Inputs: Batch ID, Expiry Date, Quantity.
- Outputs: Updated batch inventory.

3.6 Reporting & Statistics

Allows admins to generate reports on system performance.

3.6.1 View System Statistics

- Admins can view customer count, revenue, doses administered.
- Outputs: Statistical dashboard.

3.6.2 Generate Reports

- Admins can generate downloadable reports.
- Inputs: Date range, report type.
- Outputs: PDF/Excel reports.

3.6.3 Set Day Off for Vaccination Center

- Admins can define non-working days.
- Inputs: Date selection.
- Outputs: Updated schedule.

4. Non-Functional Requirements

4.1 External Interfaces

This section ensures that the system will communicate properly with users and external hardware or software components. The system will provide:

- User Interface: A responsive web and mobile application interface that supports accessibility standards.

- External System Integration: APIs for third-party systems such as electronic health records (EHR) and payment gateways.
- Hardware Compatibility: Compatibility with barcode scanners for patient verification and thermal printers for vaccination records.
- Communication Interfaces: Email and SMS notifications for appointment reminders and confirmations.

4.2 Quality Attributes

The system must meet the following quality attributes:

- Performance: Must handle at least 1,000 concurrent users with a response time under 2 seconds.
- Scalability: Designed for future expansion to support more users and additional healthcare facilities.
- Security: Must comply with GDPR and HIPAA standards to protect user data.
- Reliability: Must have a 99.9% uptime guarantee with automated failover mechanisms.
- Maintainability: Modular code structure to facilitate easy updates and enhancements.
- Usability: Intuitive and user-friendly interface with multilingual support (English, Vietnamese).

5. Requirement Appendix

5.1 Business Rules

ID	**Rule Description**
BR1	Customers must have an account before booking an appointment
BR2	A customer cannot book two appointments at the same time
BR3	Payment must be completed before the appointment is confirmed
BR4	Medical staff must verify the health condition before administering the vaccine

5.2 Common Requirements

- The system must support a website platform.
- User data must be secured according to GDPR/HIPAA standards.
- System response time should be under 2 seconds to ensure a good user experience.
- The system must support at least 1,000 concurrent users.

5.3 Application Messages List

Message Code	Message Description

MSG001	Login successful.
MSG002	Signup successful.
MSG003	Incorrect username or password.
MSG004	Appointment confirmed.
MSG005	Payment successful.
MSG006	An error occurred, please try again later.

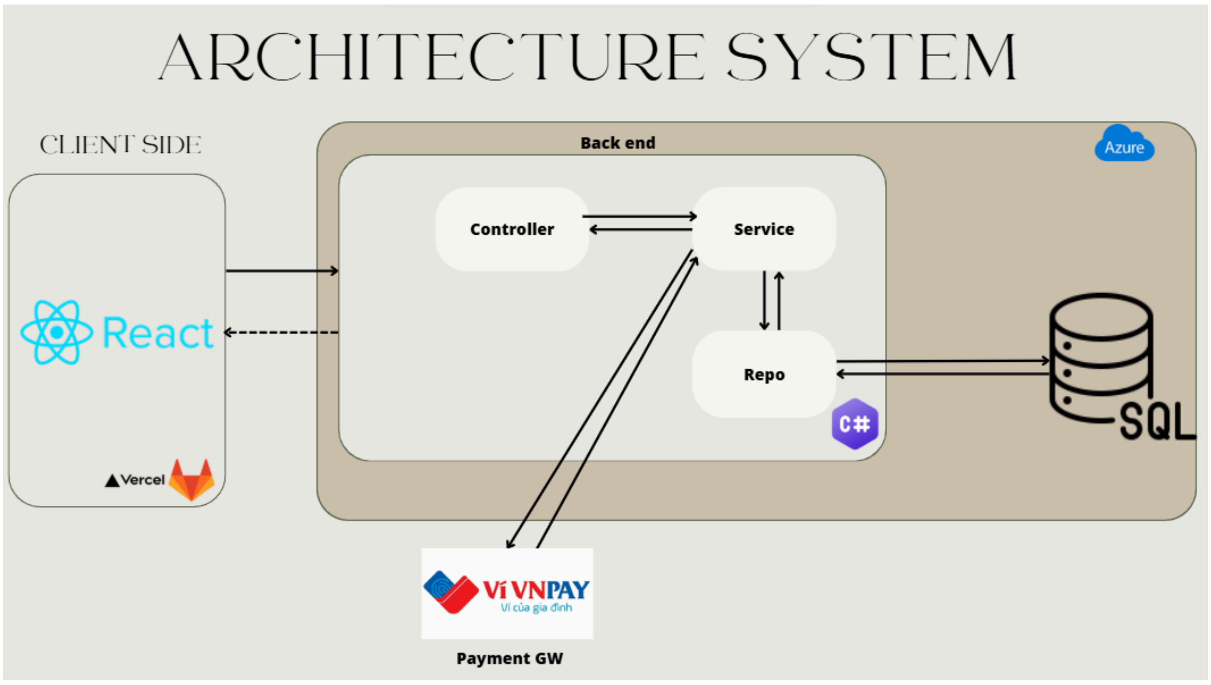
5.4 Other Requirements

- The system must have an automatic daily data backup feature.
- Support multiple languages (Vietnamese, English).
- Integrate online payment API with VNPay.
- Provide APIs for third-party system integration.

IV. Software Design Description

1. System Design

1.1 System Architecture



1.1.1 Client Side (Front-end)

- React

React is a JavaScript library used for building user interfaces (UI) in web applications. It enables the development of dynamic, interactive, and high-performance UI components. Within this architecture, React represents the user-facing layer through which end-users interact with the system.

- Vercel

Vercel is a cloud computing platform designed for deploying and hosting both static and dynamic web applications. It offers rapid deployment, automatic scaling, and performance optimization for React applications. In this architecture, Vercel serves as the hosting environment where the React application is deployed and made accessible to users.

1.1.2. Back-end (Server Side)

- Controller

The controller component is responsible for handling incoming requests from the client side (React). It processes the received data, coordinates the appropriate business logic, and returns responses accordingly. The controller acts as the interface between the front-end and the back-end services.

- Service Layer

The service layer contains the core business logic of the application. It is responsible for processing data, executing business rules, and interacting with the database layer. The service component acts as an intermediary between the controller and the repository.

- Repository (Repo)

The repository layer manages interactions with the SQL database. It performs operations such as querying, updating, and deleting records. The repository abstracts database access, allowing the service layer to remain independent of database-specific implementation details.

- C#

C# is the programming language utilized to develop the back-end components, including controllers, services, and repositories. It is a powerful, object-oriented language that is well-suited for building web applications and web services.

- SQL (Structured Query Language)

SQL is the database query language used to manage and manipulate data within the relational database. It facilitates data retrieval, updates, and deletions within the SQL Server database.

1.1.3. Azure (Cloud Infrastructure)

Azure is Microsoft’s cloud computing platform. In this architecture, Azure provides SQL Server as a cloud-based database service. It offers scalability, reliability, and security for data storage and management.

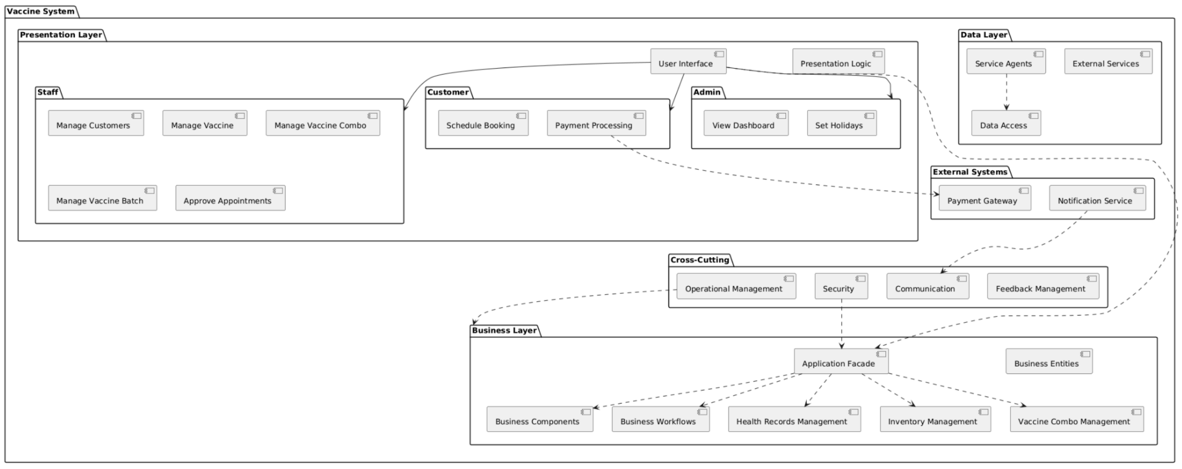
1.1.4. Payment Gateway

VNPAY Wallet: VNPAY is an online payment gateway that facilitates secure and seamless digital transactions. Within this architecture, the application integrates with VNPAY to process online payments.

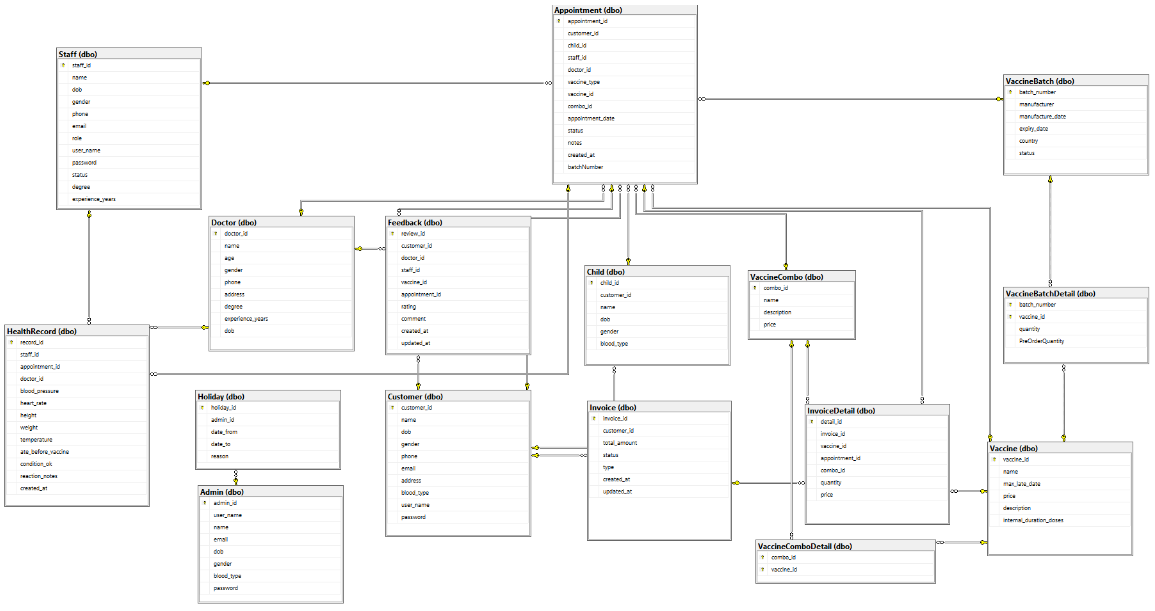
Data Flow

1. Users interact with the React-based front-end via a web browser.
2. React sends requests to the back-end controller.
3. The controller forwards the request to the service layer for business logic processing.
4. The service layer interacts with the repository to retrieve or update data from the SQL database hosted on Azure.
5. The service layer returns the processed data to the controller.
6. The controller sends the response back to React, enabling UI updates for the user.
7. The application may interact with VNPAY to facilitate online transactions.

1.2 Package Diagram



2. Database Design



Tables and Relationships:

Customer

- Description: Stores customer information, including personal details and login credentials
- Relationships:

    - One-to-Many with Child (one customer can have multiple children).
    - One-to-Many with Appointment (one customer can have multiple appointments).
    - One-to-Many with Invoice (one customer can have multiple invoices).
    - One-to-Many with Feedback (one customer can provide multiple reviews).

Child

- Description: Stores information about customers' children.
- Relationships:
    - Many-to-One with Customer (a child belongs to one customer).
    - One-to-Many with Appointment (a child can have multiple appointments).

Staff

- Description: Stores employee information, including personal details and login credentials.
- Relationships:
    - One-to-Many with Appointment (one staff member can create multiple appointments).
    - One-to-Many with HealthRecord (one staff member can create multiple health records).

Doctor

- Description: Stores information about doctors.
- Relationships:
    - One-to-Many with Appointment (one doctor can have multiple appointments).
    - One-to-Many with HealthRecord (one doctor can write multiple health records).
    - One-to-Many with Feedback (one doctor can receive multiple reviews).

Admin

- Description: Stores administrator information.
- Relationships:
    - One-to-Many with Holiday (one admin can create multiple holiday records).

Vaccine

- Description: Stores information about vaccines.
- Relationships:
    - One-to-Many with VaccineBatchDetail (one vaccine can belong to multiple batches).
    - One-to-Many with VaccineComboDetail (one vaccine can be included in multiple combos).
    - One-to-Many with Appointment (one vaccine can be used in multiple appointments).
    - One-to-Many with InvoiceDetail (one vaccine can appear in multiple invoice details).
    - One-to-Many with Feedback (one vaccine can receive multiple reviews).

VaccineBatch

- Description: Stores information about vaccine batches.

- Relationships:
    - One-to-Many with VaccineBatchDetail (one batch can contain multiple vaccines).

VaccineBatchDetail

- Description: Stores detailed relationships between vaccines and vaccine batches.
- Relationships:
    - Many-to-One with Vaccine (one detail belongs to one vaccine).
    - Many-to-One with VaccineBatch (one detail belongs to one batch).

VaccineCombo

- Description: Stores information about vaccine combos.
- Relationships:
    - One-to-Many with VaccineComboDetail (one combo can contain multiple vaccines).
    - One-to-Many with Appointment (one combo can be used in multiple appointments).
    - One-to-Many with InvoiceDetail (one combo can appear in multiple invoice details).

VaccineComboDetail

- Description: Stores detailed relationships between vaccines and vaccine combos.
- Relationships:
    - Many-to-One with Vaccine (one detail belongs to one vaccine).
    - Many-to-One with VaccineCombo (one detail belongs to one combo).

Appointment

- Description: Stores vaccination appointment information.
- Relationships:
    - Many-to-One with Customer (an appointment belongs to one customer).
    - Many-to-One with Child (an appointment is for one child).
    - Many-to-One with Staff (an appointment is created by one staff member).
    - Many-to-One with Doctor (an appointment is handled by one doctor).
    - Many-to-One with Vaccine (an appointment uses one vaccine).
    - Many-to-One with VaccineCombo (an appointment uses one combo).
    - One-to-Many with InvoiceDetail (an appointment can be included in invoice details).
    - One-to-Many with HealthRecord (an appointment can have health records).
    - One-to-Many with Feedback (an appointment can receive reviews).

Invoice

- Description: Stores payment invoice information.
- Relationships:
    - Many-to-One with Customer (an invoice belongs to one customer).
    - One-to-Many with InvoiceDetail (an invoice contains multiple details).

InvoiceDetail

- Description: Stores details of items in invoices.
- Relationships:
  - Many-to-One with Invoice (a detail belongs to one invoice).
  - Many-to-One with Vaccine (a detail may relate to one vaccine).
  - Many-to-One with Appointment (a detail may relate to one appointment).
  - Many-to-One with VaccineCombo (a detail may relate to one combo).

HealthRecord

- Description: Stores patient health records after vaccination.
- Relationships:
  - Many-to-One with Staff (a record is created by one staff member).
  - Many-to-One with Appointment (a record is related to one appointment).
  - Many-to-One with Doctor (a record is written by one doctor).

Feedback

- Description: Stores customer feedback about the service.
- Relationships:
  - Many-to-One with Customer (a review is submitted by one customer).
  - Many-to-One with Doctor (a review may be about one doctor).
  - Many-to-One with Staff (a review may be about one staff member).
  - Many-to-One with Vaccine (a review may be about one vaccine).
  - Many-to-One with Appointment (a review is related to one appointment).

Holiday

- Description: Stores holiday information.
- Relationships:
  - Many-to-One with Admin (a holiday is created by one admin).

Table Details

Appointment

- appointment_id (Primary Key): Unique identifier for the appointment.
- customer_id (Foreign Key): ID of the customer who booked the appointment.
- child_id (Foreign Key): ID of the child (if applicable).
- staff_id (Foreign Key): ID of the staff member who created the appointment.
- doctor_id (Foreign Key): ID of the doctor handling the appointment.
- vaccine_type: Type of vaccine (Single/Combo).
- vaccine_id (Foreign Key): ID of the single vaccine (if applicable).
- combo_id (Foreign Key): ID of the vaccine combo (if applicable).
- appointment_date: Date and time of the appointment.
- status: Appointment status (Pending, Approved, Rejected, Late, Success).

- notes: Notes about the appointment.
- created_at: Timestamp when the appointment was created.

Invoice

- invoice_id (Primary Key): Unique identifier for the invoice.
- customer_id (Foreign Key): ID of the customer.
- total_amount: Total payment amount.
- status: Invoice status (Paid, Unpaid, Cancelled).
- type: Invoice type (Single/Combo).
- created_at: Timestamp when the invoice was created.
- updated_at: Timestamp when the invoice was last updated.

InvoiceDetail

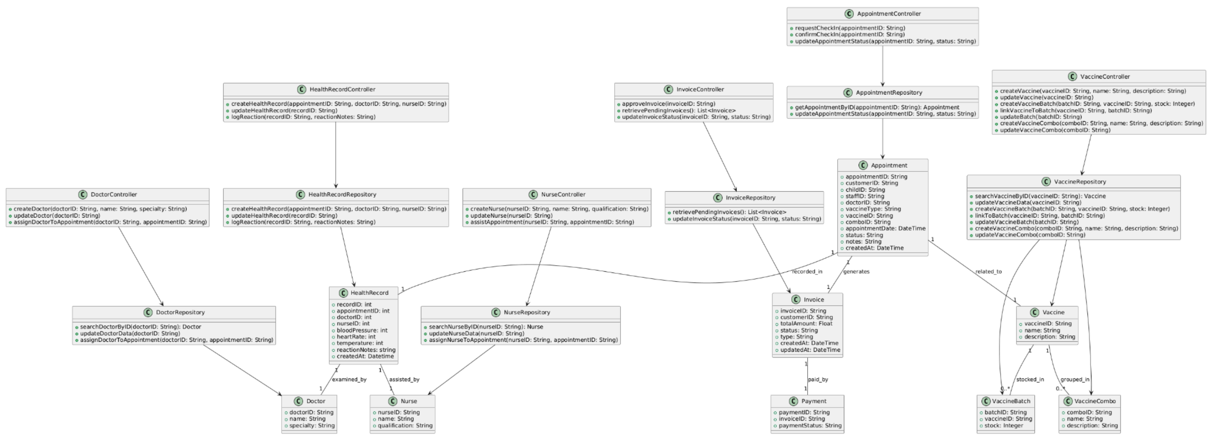
- detail_id (Primary Key): Unique identifier for the invoice detail.
- invoice_id (Foreign Key): ID of the invoice.
- vaccine_id (Foreign Key): ID of the vaccine (if applicable).
- appointment_id (Foreign Key): ID of the related appointment.
- combo_id (Foreign Key): ID of the vaccine combo (if applicable).
- quantity: Quantity of vaccines/combo.
- price: Price amount.

HealthRecord

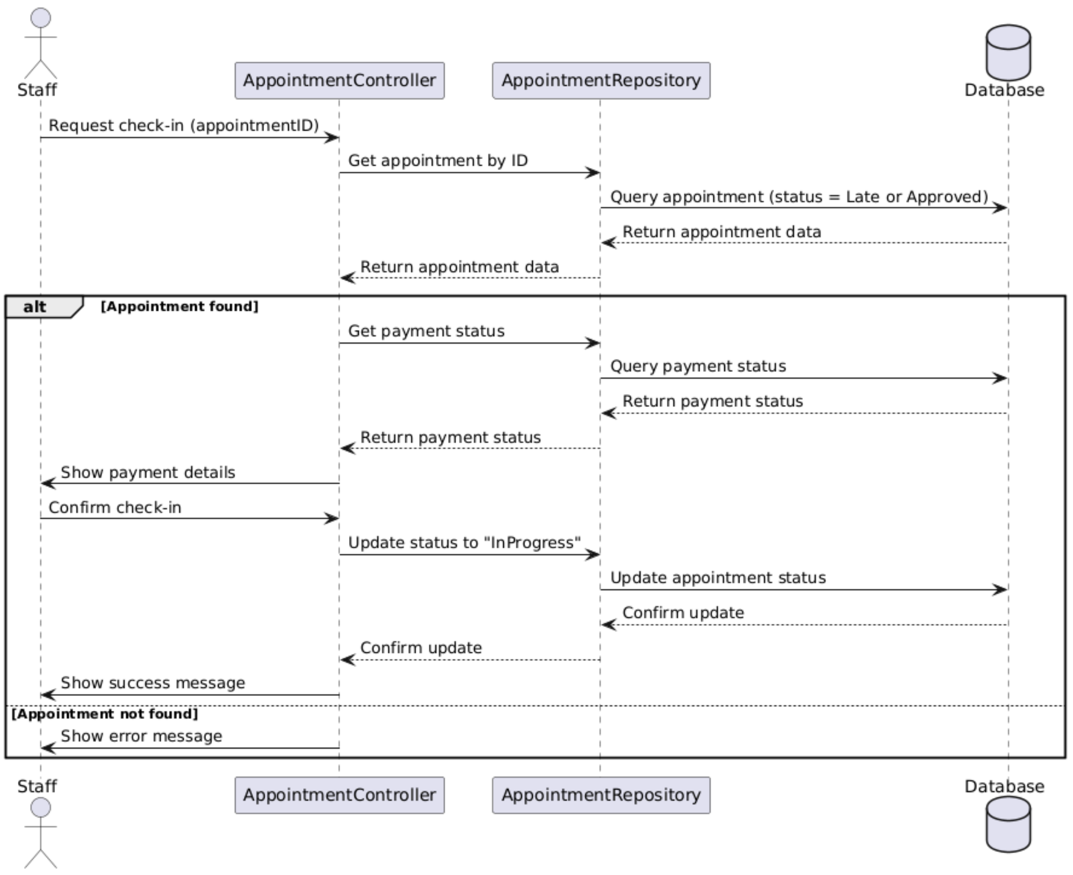
- record_id (Primary Key): Unique identifier for the health record.
- staff_id (Foreign Key): ID of the staff member who created the record.
- appointment_id (Foreign Key): ID of the related appointment.
- doctor_id (Foreign Key): ID of the doctor who wrote the record.
- blood_pressure: Blood pressure.
- heart_rate: Heart rate.
- height: Height.
- weight: Weight.
- temperature: Temperature.
- ate_before_vaccine: Ate before vaccination (True/False).
- condition_ok: Health condition status (True/False).
- reaction_notes: Notes on post-vaccine reactions.
- created_at: Timestamp when the record was created.

3. Detailed Design

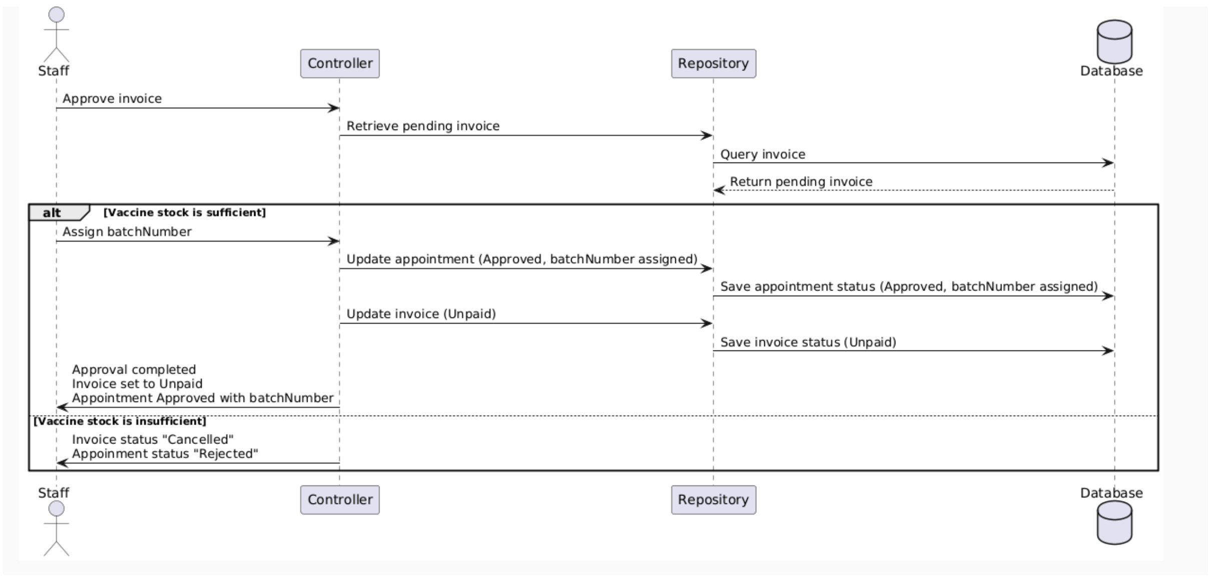
3.1 Staff handling process



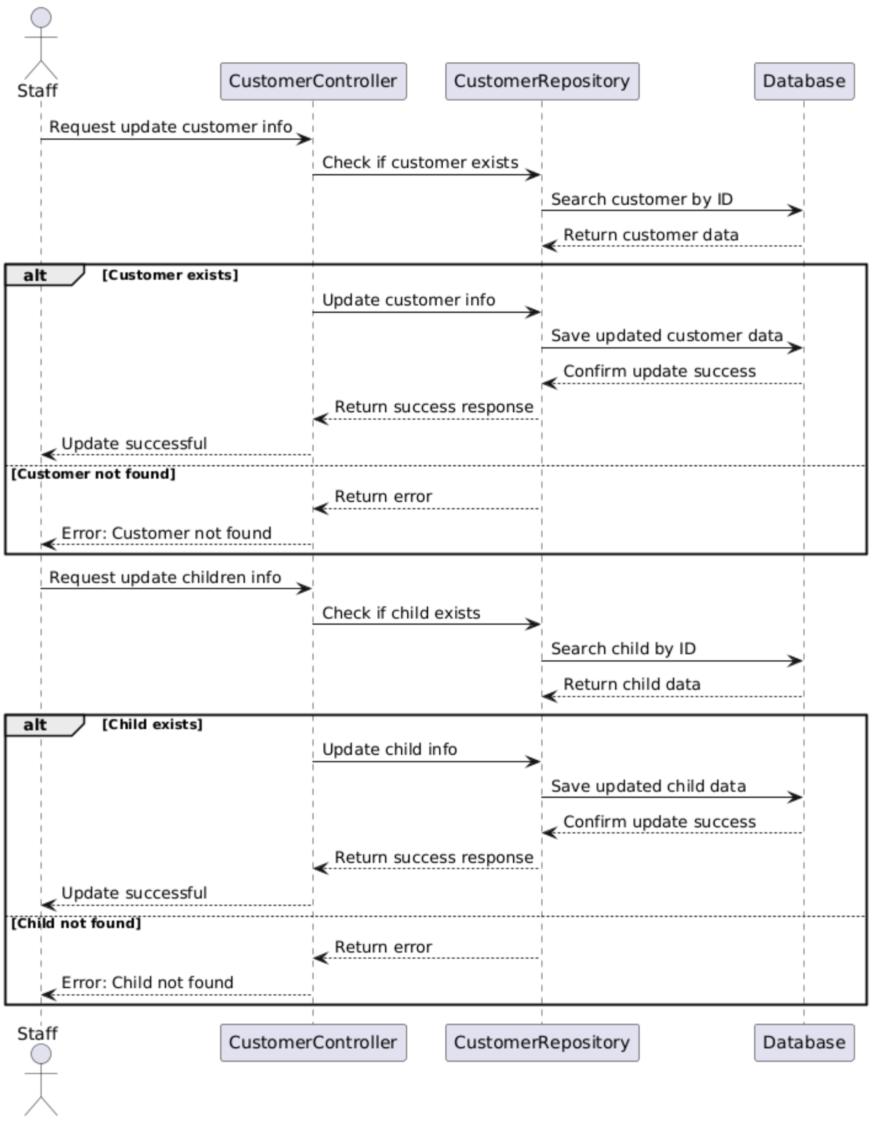
3.1.1 Sequence diagram for "Receptionist" check in for customer.



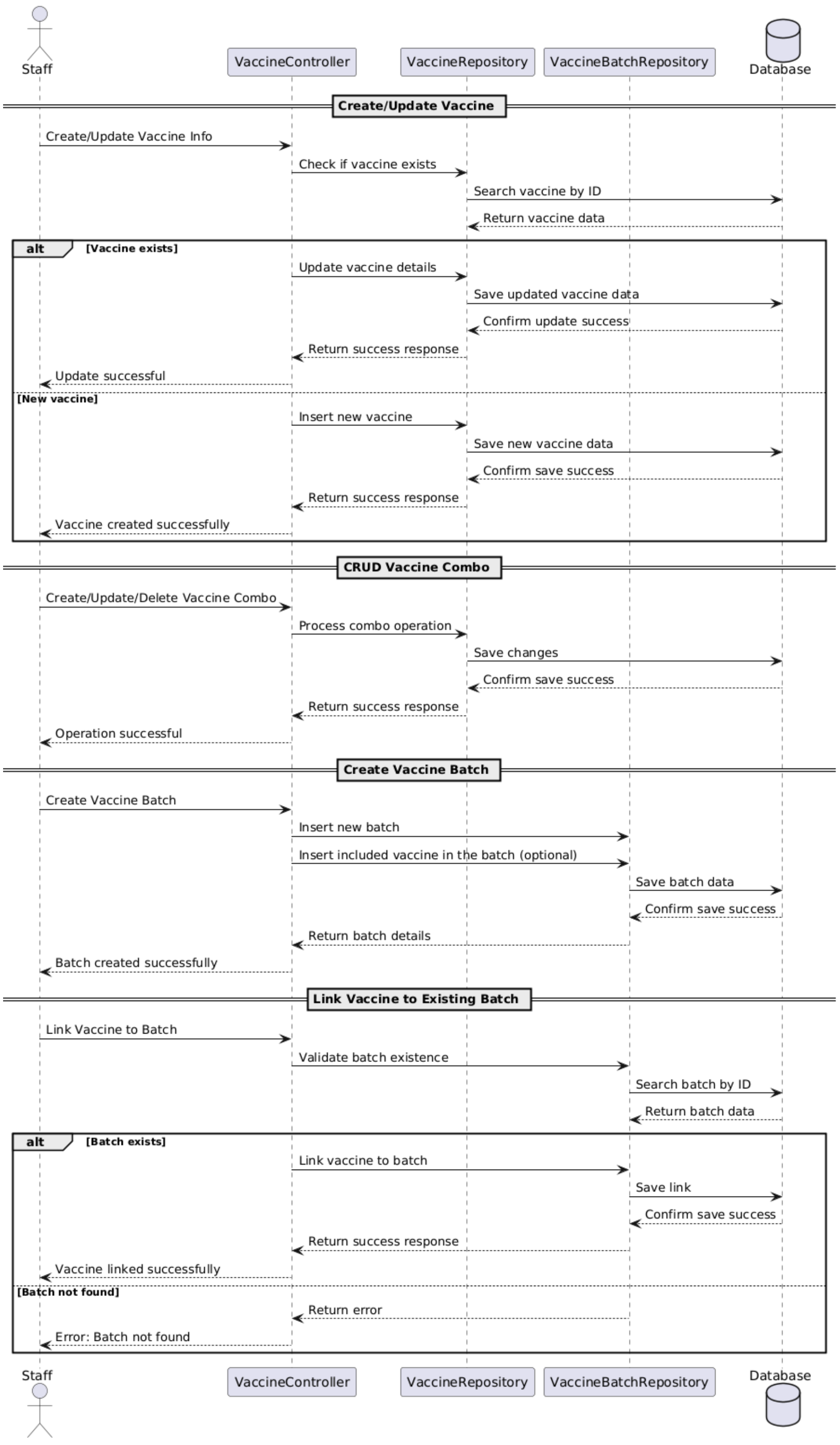
3.1.2 Sequence diagram for "Receptionist" approving invoice.


Staff
Controller
Repository
Database
Approve invoice
Retrieve pending invoice
Query invoice
Return pending invoice
alt
[Vaccine stock is sufficient]
Assign batchNumber
Update appointment (Approved, batchNumber assigned)
Save appointment status (Approved, batchNumber assigned)
Update invoice (Unpaid)
Save invoice status (Unpaid)
Approval completed
Invoice set to Unpaid
Appointment Approved with batchNumber
[Vaccine stock is insufficient]
Invoice status "Cancelled"
Appoinment status "Rejected"
Staff
Controller
Repository
Database

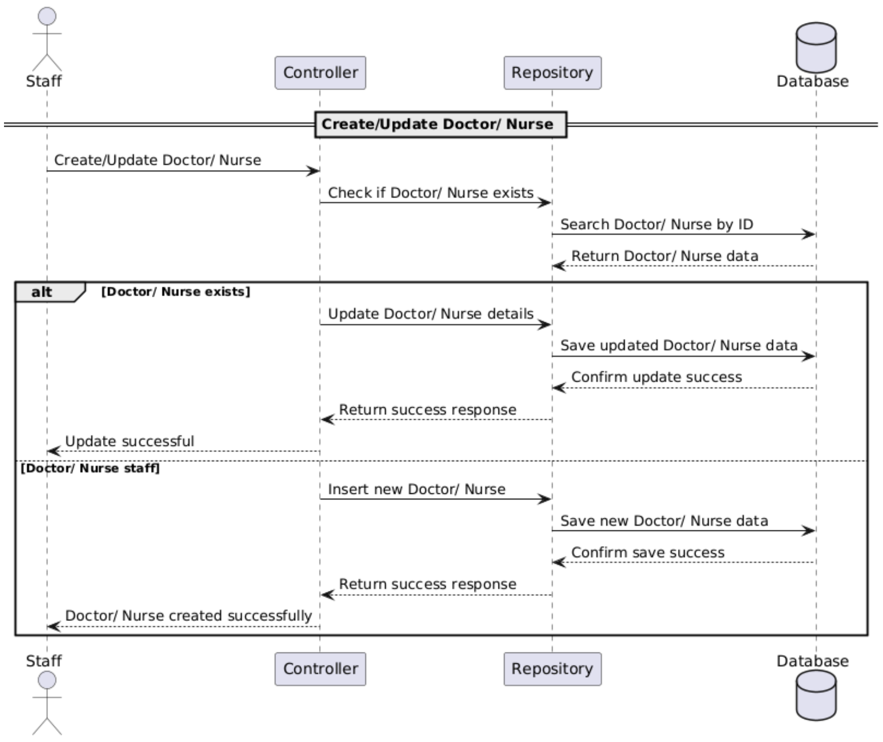
3.1.3 Sequence diagram for "Data Entry" update Customer and their own children information



3.1.4 Sequence diagram for "Data Entry" manage Vaccine- Vaccine Combo- Vaccine Batch



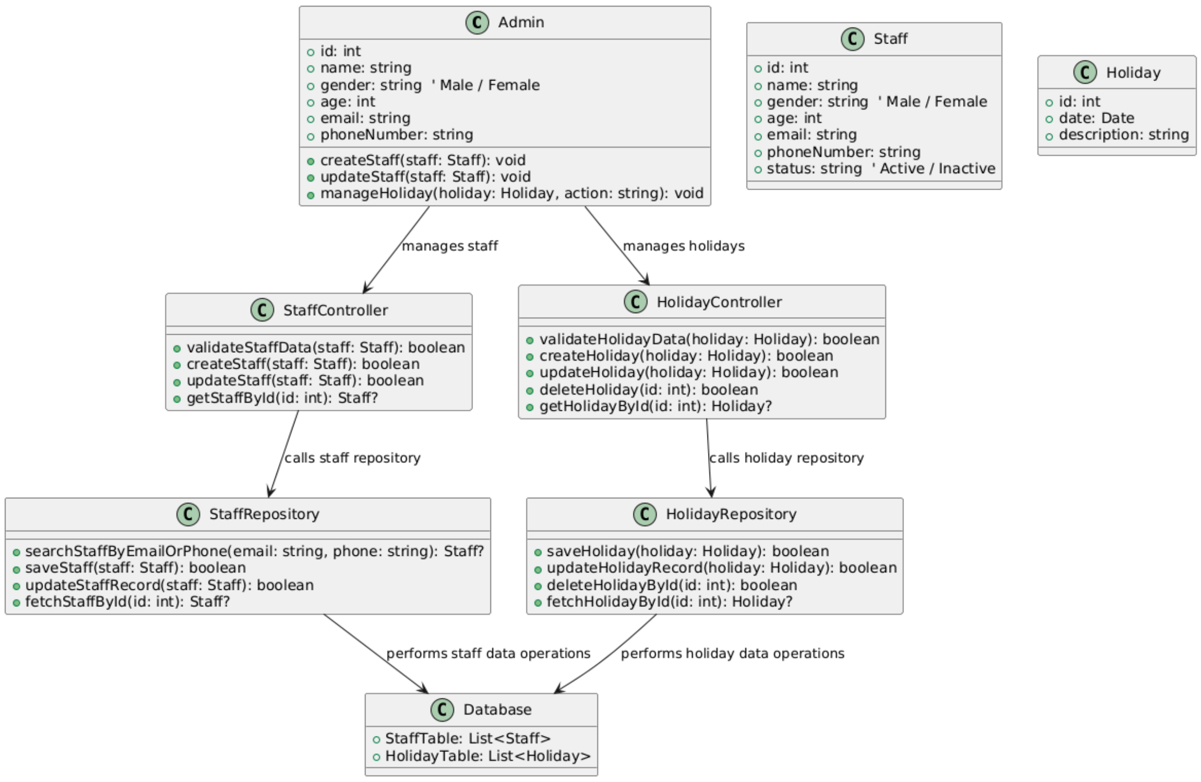
3.1.5 Sequence diagram for "Data Entry" manage Doctor and Nurse


Staff
Controller
Repository
Database
Create/Update Doctor/ Nurse
Create/Update Doctor/ Nurse
Check if Doctor/ Nurse exists
Search Doctor/ Nurse by ID
Return Doctor/ Nurse data
alt [Doctor/ Nurse exists]
Update Doctor/ Nurse details
Save updated Doctor/ Nurse data
Confirm update success
Return success response
Update successful
[Doctor/ Nurse staff]
Insert new Doctor/ Nurse
Save new Doctor/ Nurse data
Confirm save success
Return success response
Doctor/ Nurse created successfully

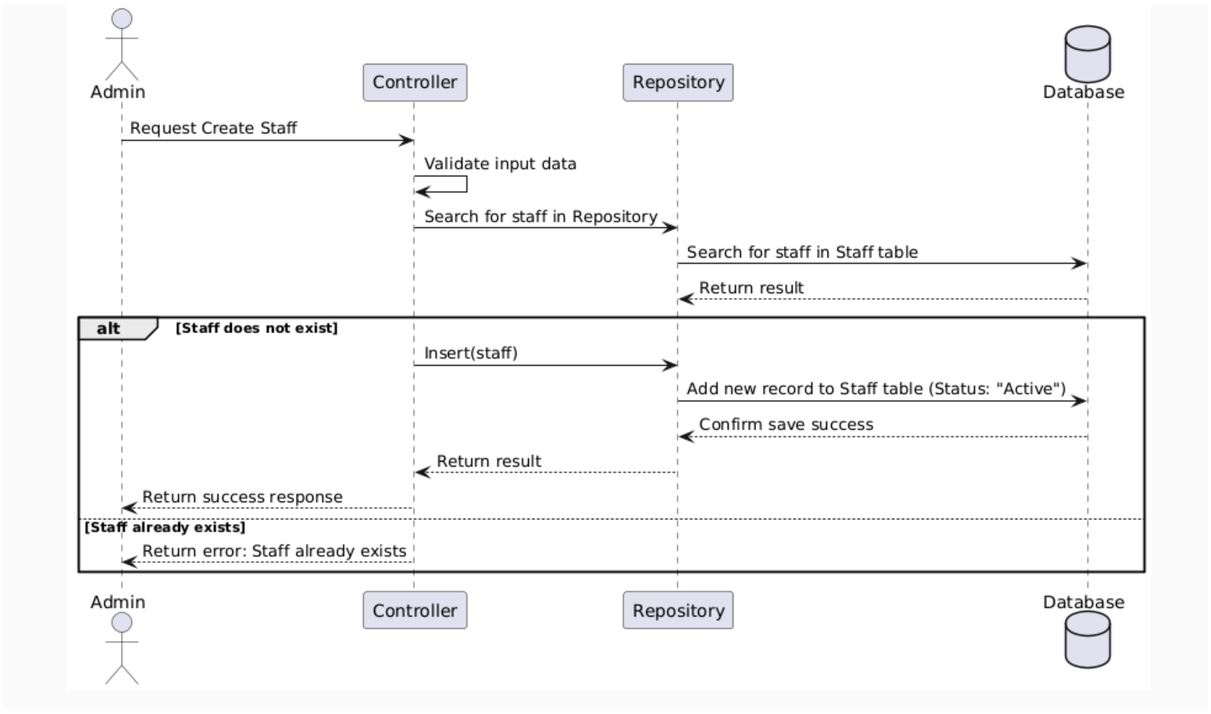
3.1.6 Sequence diagram for "Nurse" update the Health Record for customer and update the status of the appointment

0

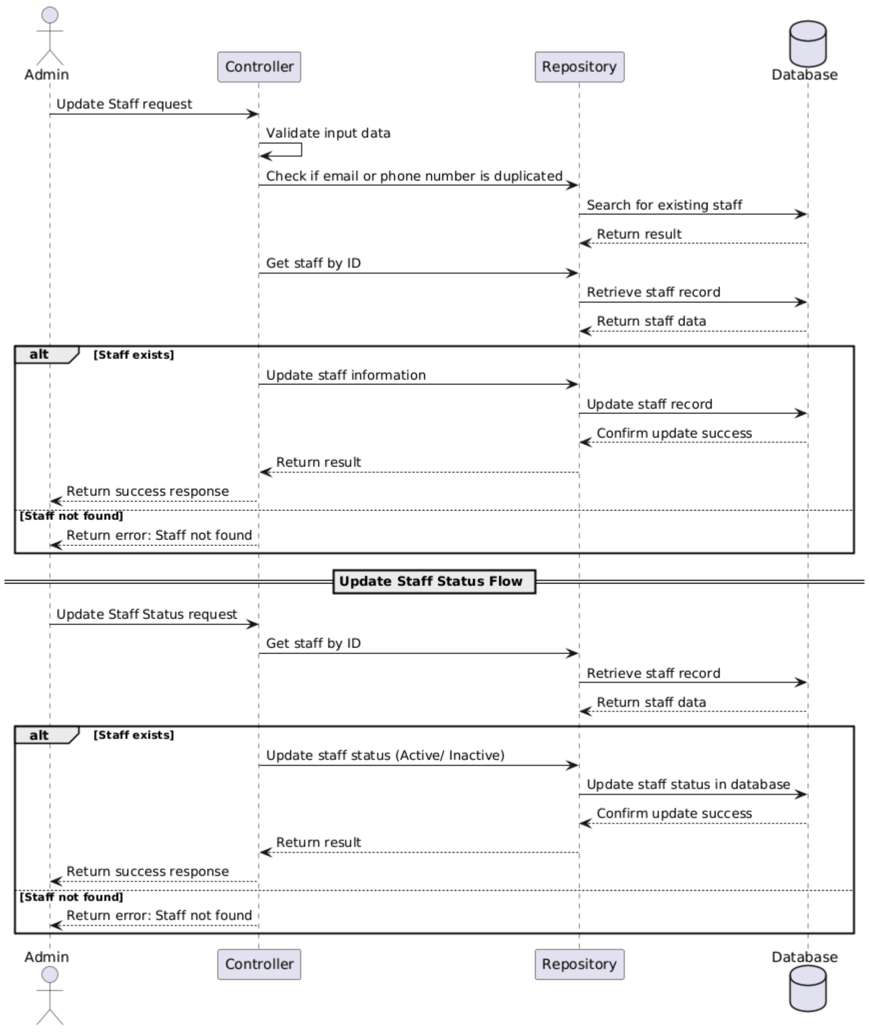
3.2 Admin handling process


Admin
id: int
name: string
gender: string ' Male / Female
age: int
email: string
phoneNumber: string
createStaff(staff: Staff): void
updateStaff(staff: Staff): void
manageHoliday(holiday: Holiday, action: string): void
Staff
id: int
name: string
gender: string ' Male / Female
age: int
email: string
phoneNumber: string
status: string ' Active / Inactive
Holiday
id: int
date: Date
description: string
manages staff
manages holidays
StaffController
validateStaffData(staff: Staff): boolean
createStaff(staff: Staff): boolean
updateStaff(staff: Staff): boolean
getStaffById(id: int): Staff?
HolidayController
validateHolidayData(holiday: Holiday): boolean
createHoliday(holiday: Holiday): boolean
updateHoliday(holiday: Holiday): boolean
deleteHoliday(id: int): boolean
getHolidayById(id: int): Holiday?
calls staff repository
calls holiday repository
StaffRepository
searchStaffByEmailOrPhone(email: string, phone: string): Staff?
saveStaff(staff: Staff): boolean
updateStaffRecord(staff: Staff): boolean
fetchStaffById(id: int): Staff?
HolidayRepository
saveHoliday(holiday: Holiday): boolean
updateHolidayRecord(holiday: Holiday): boolean
deleteHolidayById(id: int): boolean
fetchHolidayById(id: int): Holiday?
performs staff data operations
performs holiday data operations
Database
StaffTable: List<Staff>
HolidayTable: List<Holiday>

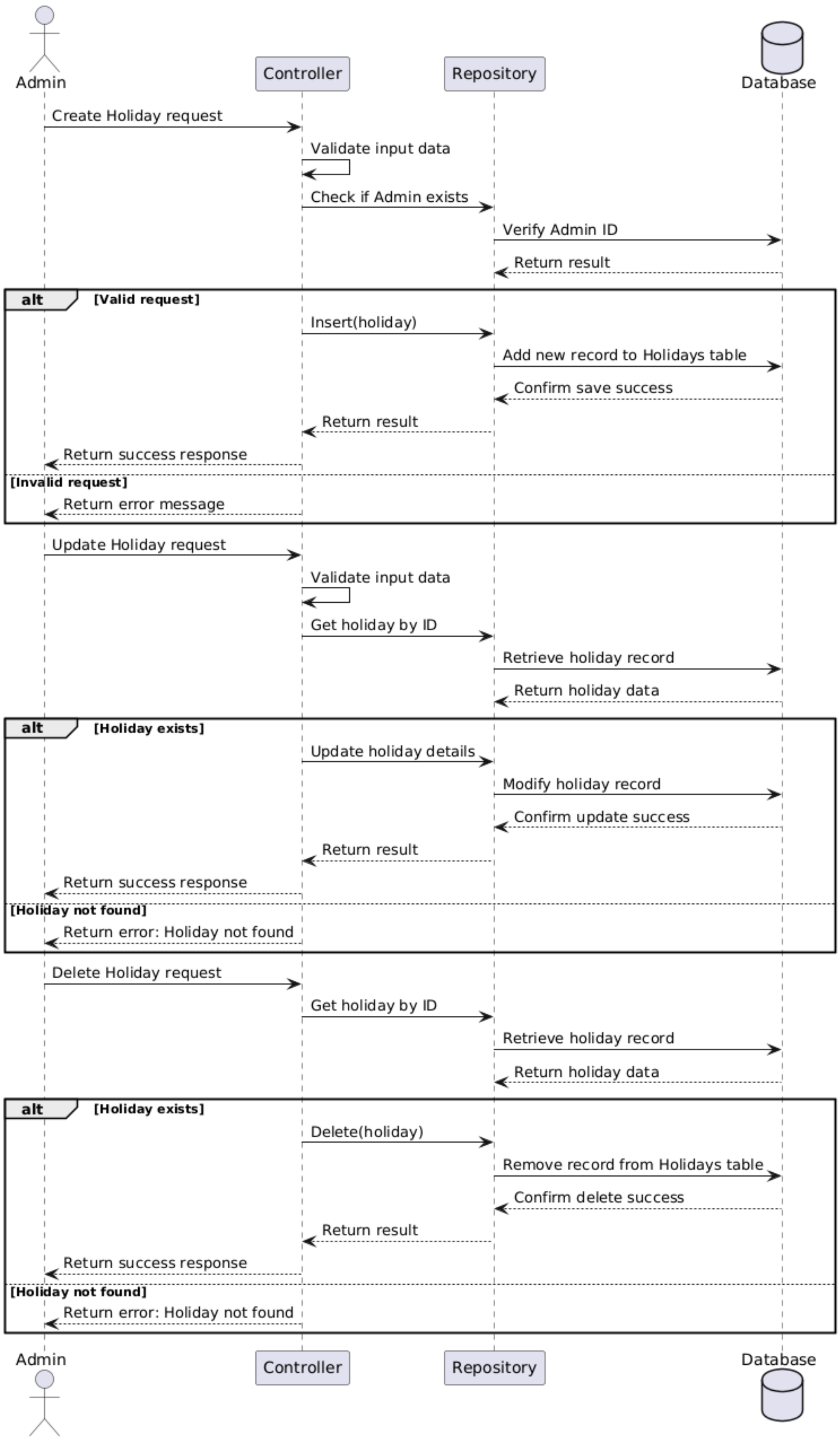
3.2.1 Sequence for admin creating staff account



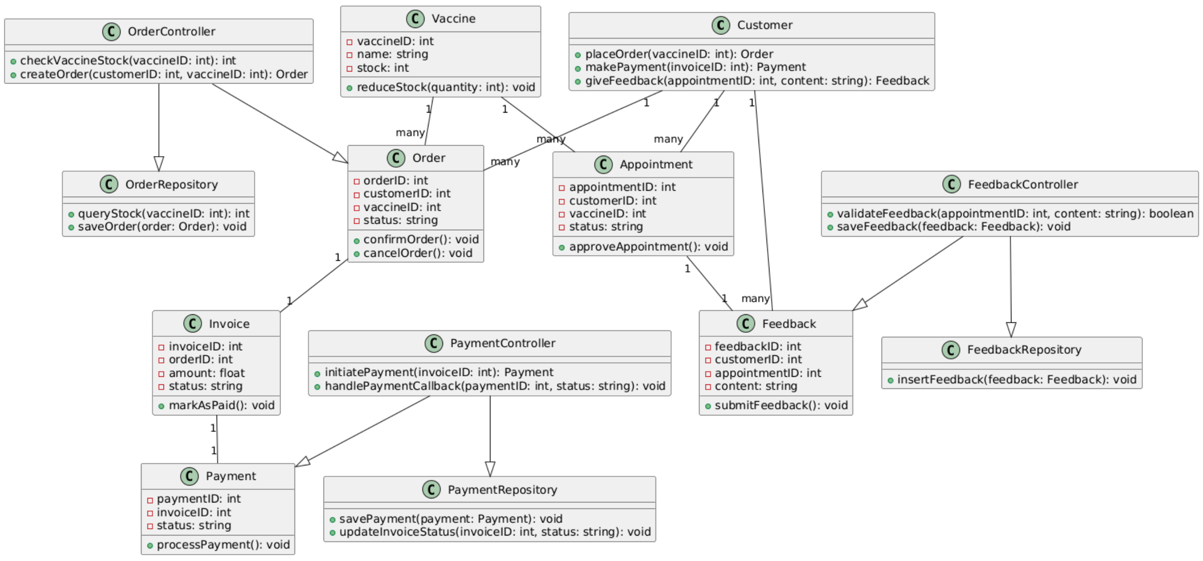
3.2.2 Sequence for admin updating staff account



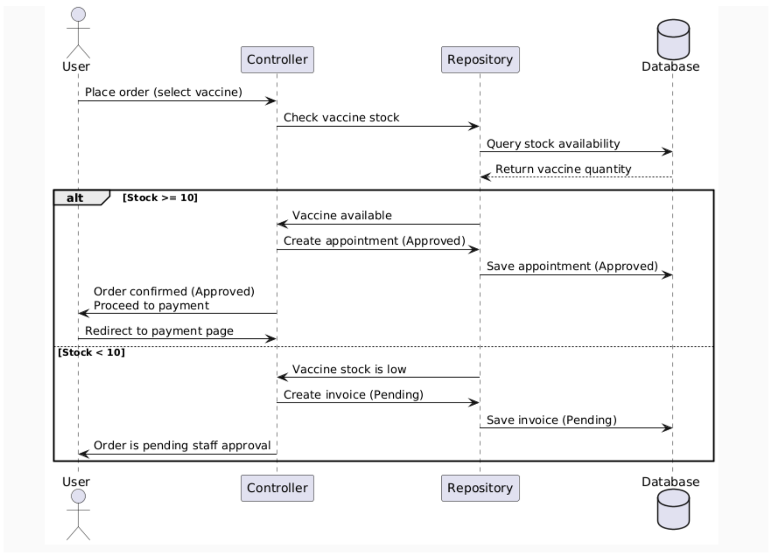
3.2.2 Sequence for admin create/ update/ delete holiday for the system.



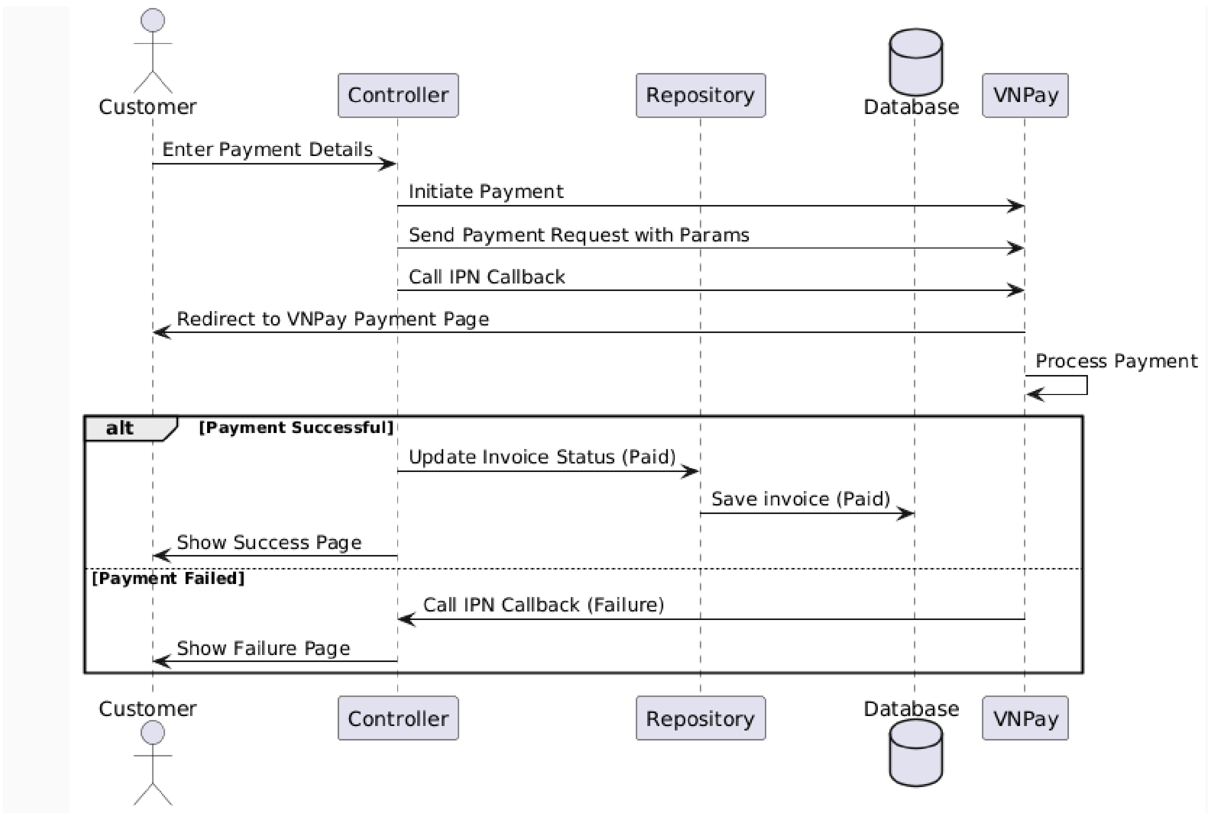
3.3 Customer handling process


OrderController
checkVaccineStock(vaccineID: int): int
createOrder(customerID: int, vaccineID: int): Order
Vaccine
vaccineID: int
name: string
stock: int
reduceStock(quantity: int): void
Customer
placeOrder(vaccineID: int): Order
makePayment(invoiceID: int): Payment
giveFeedback(appointmentID: int, content: string): Feedback
OrderRepository
queryStock(vaccineID: int): int
saveOrder(order: Order): void
Order
orderID: int
customerID: int
vaccineID: int
status: string
confirmOrder(): void
cancelOrder(): void
Appointment
appointmentID: int
customerID: int
vaccineID: int
status: string
approveAppointment(): void
FeedbackController
validateFeedback(appointmentID: int, content: string): boolean
saveFeedback(feedback: Feedback): void
Invoice
invoiceID: int
orderID: int
amount: float
status: string
markAsPaid(): void
PaymentController
initiatePayment(invoiceID: int): Payment
handlePaymentCallback(paymentID: int, status: string): void
Feedback
feedbackID: int
customerID: int
appointmentID: int
content: string
submitFeedback(): void
FeedbackRepository
insertFeedback(feedback: Feedback): void
Payment
paymentID: int
invoiceID: int
status: string
processPayment(): void
PaymentRepository
savePayment(payment: Payment): void
updateInvoiceStatus(invoiceID: int, status: string): void
1
many
1
many
1
many
1
many
1
1
1
1
1
1
1
many

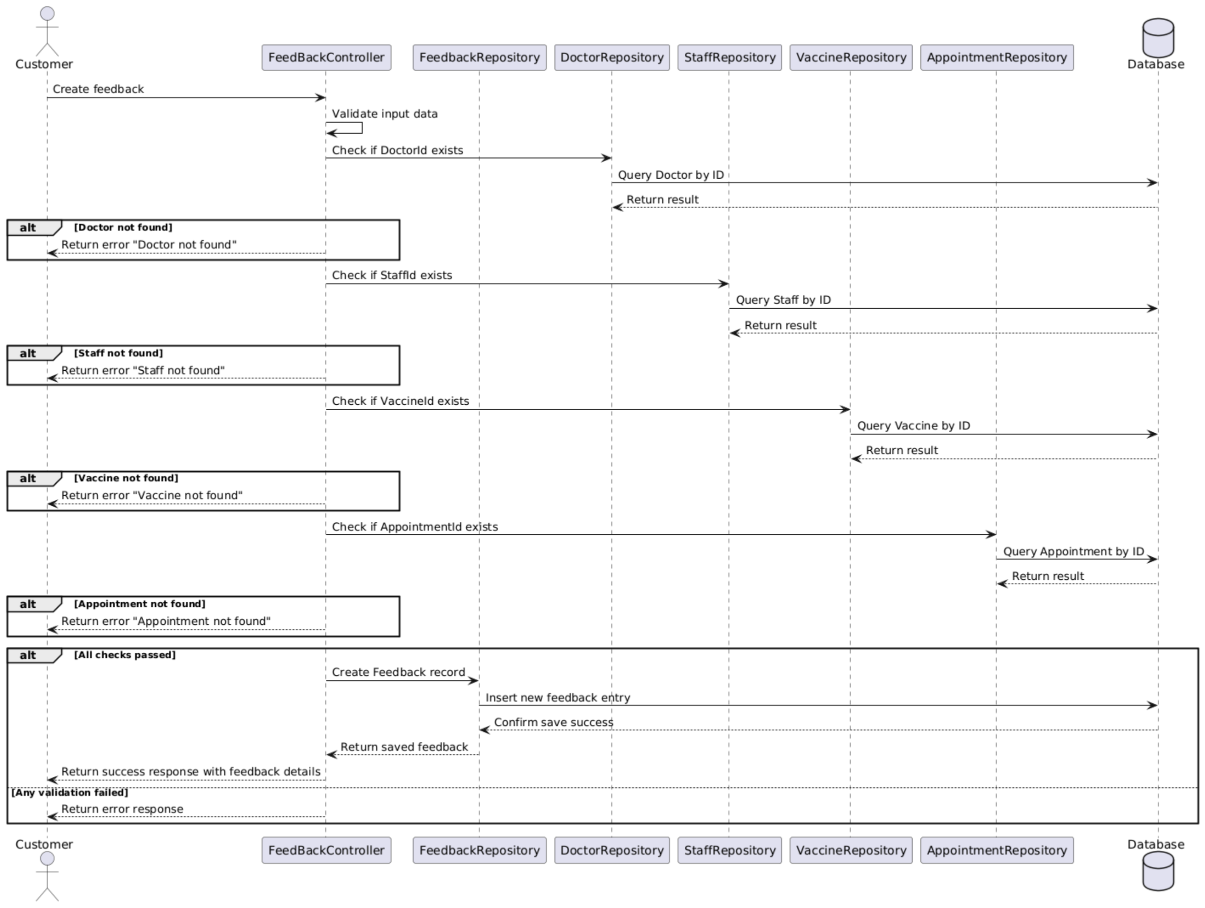
3.3.1 Senquence diagram for order of customer



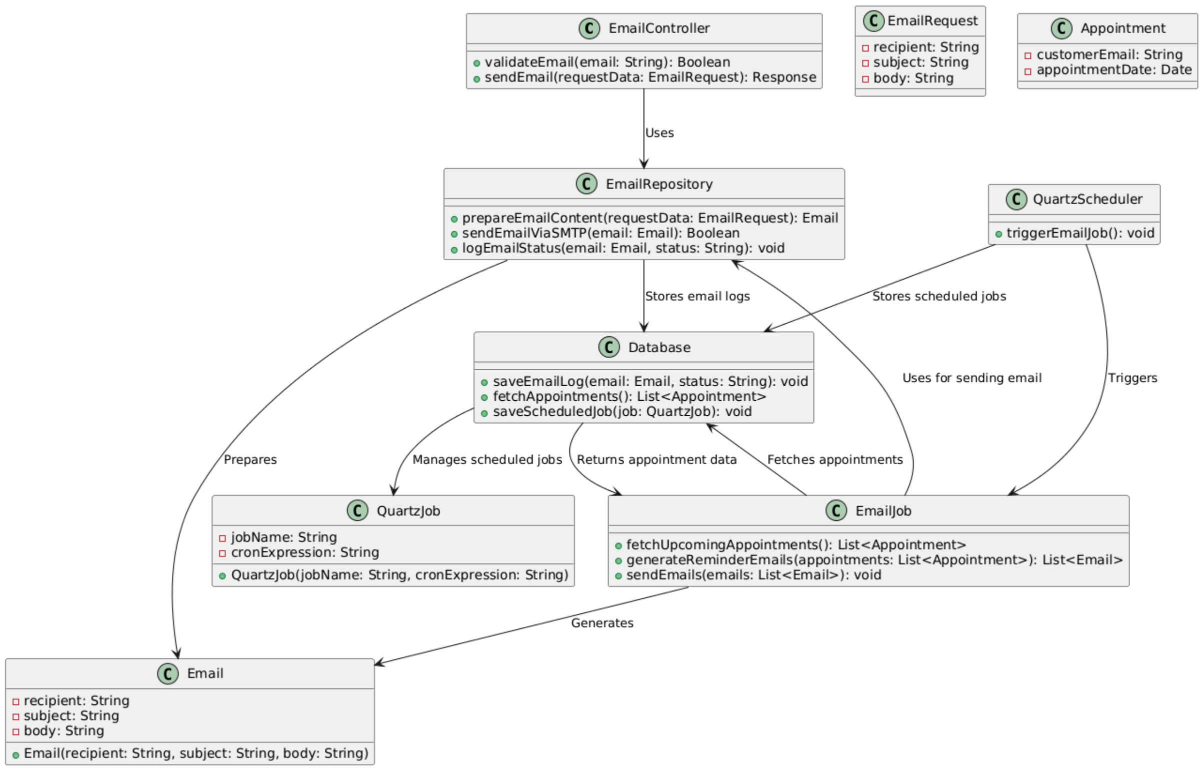
3.3.2 Sequence for customer paying invoice



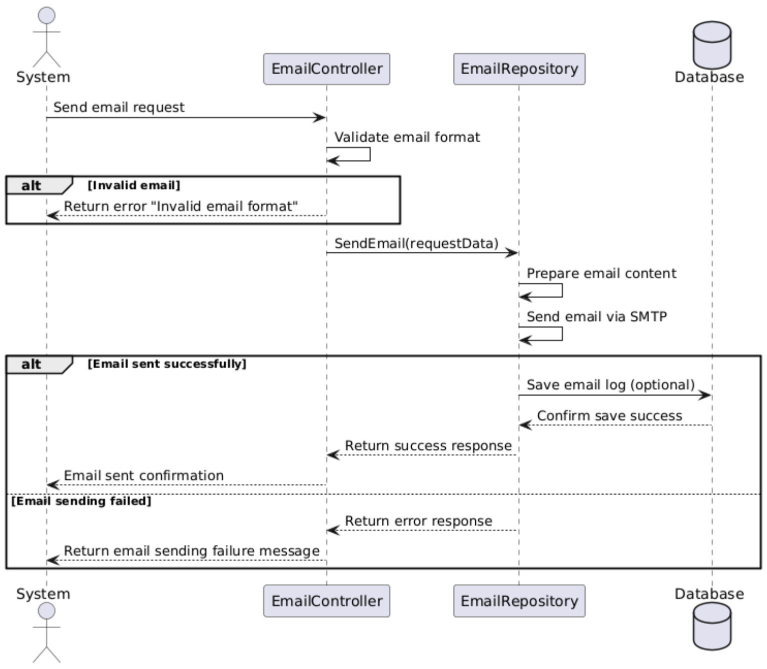
3.3.3 Sequence for customer providing feedback



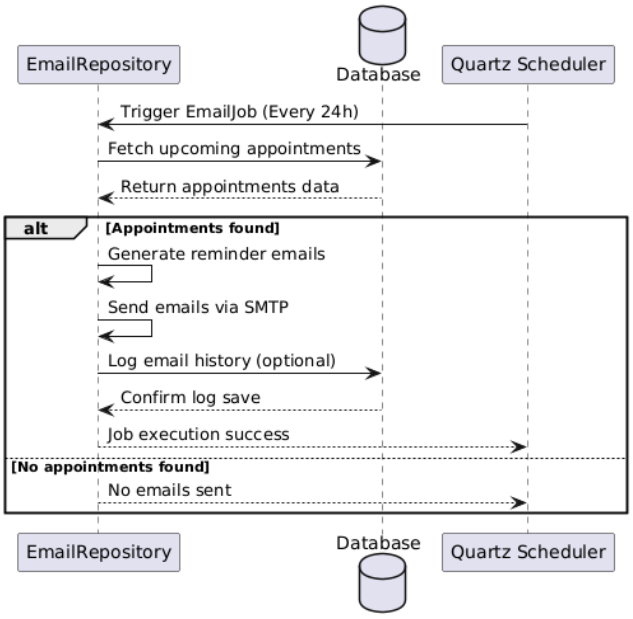
3.4 System handling process


EmailController
validateEmail(email: String): Boolean
sendEmail(requestData: EmailRequest): Response
EmailRequest
recipient: String
subject: String
body: String
Appointment
customerEmail: String
appointmentDate: Date
Uses
EmailRepository
prepareEmailContent(requestData: EmailRequest): Email
sendEmailViaSMTP(email: Email): Boolean
logEmailStatus(email: Email, status: String): void
QuartzScheduler
triggerEmailJob(): void
Stores email logs
Stores scheduled jobs
Database
saveEmailLog(email: Email, status: String): void
fetchAppointments(): List<Appointment>
saveScheduledJob(job: QuartzJob): void
Uses for sending email
Triggers
Prepares
Manages scheduled jobs
Returns appointment data
Fetches appointments
QuartzJob
jobName: String
cronExpression: String
QuartzJob(jobName: String, cronExpression: String)
EmailJob
fetchUpcomingAppointments(): List<Appointment>
generateReminderEmails(appointments: List<Appointment>): List<Email>
sendEmails(emails: List<Email>): void
Generates
Email
recipient: String
subject: String
body: String
Email(recipient: String, subject: String, body: String)

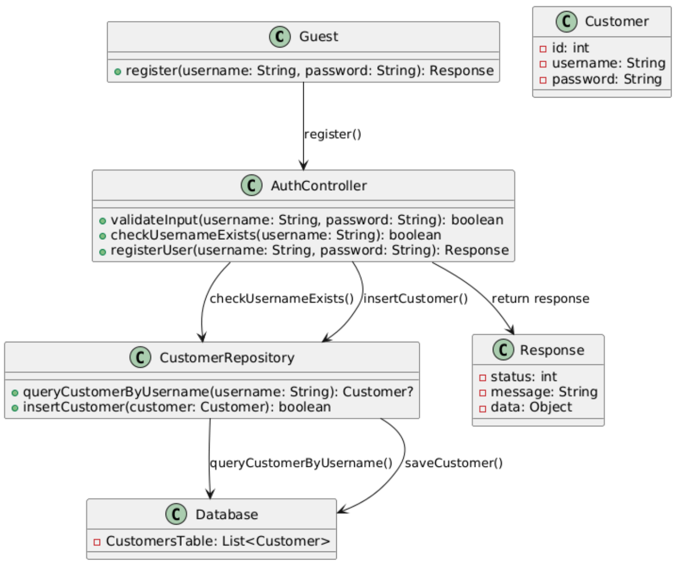
3.4.1. Sequence for system sending email to customer automatically



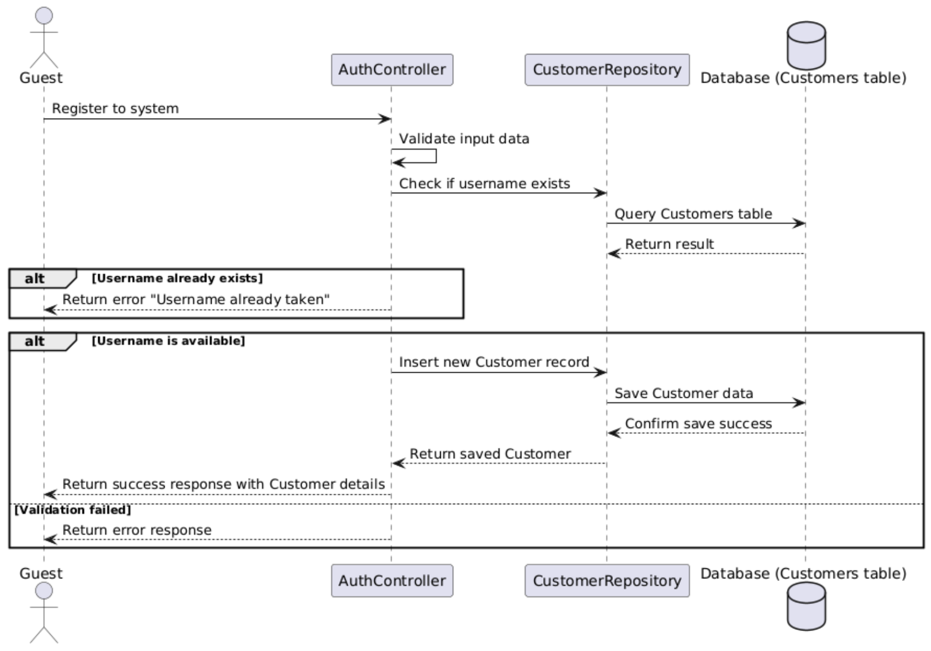
3.4.2. Sequence for system scheduling sending email with Quartz.NET



3.5. Guest handling process


Guest
register(username: String, password: String): Response
Customer
id: int
username: String
password: String
register()
AuthController
validateInput(username: String, password: String): boolean
checkUsernameExists(username: String): boolean
registerUser(username: String, password: String): Response
checkUsernameExists()
insertCustomer()
return response
CustomerRepository
queryCustomerByUsername(username: String): Customer?
insertCustomer(customer: Customer): boolean
Response
status: int
message: String
data: Object
queryCustomerByUsername()
saveCustomer()
Database
CustomersTable: List<Customer>

3.5.1. Sequence diagram for guest to create new account



V. Software Testing Documentation

1. Scope of Testing

The scope of testing for this project includes verifying the functionalities, performance, and security aspects of the vaccination management system. The testing will cover both functional and non-functional requirements to ensure the system operates as expected.

Target-of-Test Features:

- **Functional Requirements:**
    - User authentication (Login, Sign-up, Forgot Password)
    - Customer and Staff management
    - Appointment scheduling and management
    - Payment processing
    - Notification system (email automation)
    - Feedback and rating system
    - Holiday schedule management
- **Non-Functional Requirements:**
    - System performance and responsiveness
    - Security measures (user access control, data protection)
    - Usability and accessibility

2. Test Strategy

2.1 Testing Types

2.1.1. Functional Testing:

- Objective: Verify that all system functions work as intended

- Technique: Black-box testing, UI validation, API testing
- Completion Criteria: All functional test cases pass successfully

2.1.2. Performance Testing:

- Objective: Ensure the system operates efficiently under load
- Technique: Load testing, stress testing
- Completion Criteria: System meets response time and concurrency benchmarks

2.1.3. Security Testing:

- Objective: Validate system security measures
- Technique: Penetration testing, role-based access testing
- Completion Criteria: No major vulnerabilities found

2.1.4. Usability Testing:

- Objective: Ensure system is user-friendly and accessible
- Technique: UI/UX testing, user feedback evaluation
- Completion Criteria: Users can easily navigate the system

2.2 Test Levels

Test Level	Testing Types Applied
Unit Testing	Functional Testing
Integration Testing	Functional Testing, Security Testing
System Testing	Performance Testing, Usability Testing

2.3 Supporting Tools

- JMeter (Performance testing)
- JIRA (Bug tracking and test management)

3. Test Plan

3.1 Human Resources

Role	Responsibilities
Developers	Conduct unit testing and fix reported bugs
UX/UI Designers	Validate usability and design aspects
Security Analysis	Conduct penetration and security testing

3.2 Test Environment

Component	Details
Operating System	Windows 10
Browsers	Chrome, Edge, Safari, Firefox
Database	SQL Server
Backend	ASP.NET Core
Testing Tools	JMeter

3.3 Test Milestones

Milestone	Expected Completion
Unit Testting	Week 2
Integration Testing	Week 4
System Testing	Week 6
User Acceptance Testing (UAT)	Week 9

4. Test Reports

- Total cases: 50
- Success cases: 45
- Fail cases: 5

VI. Release Package & User Guides

1. Deliverable Package

No.	Deliverable Item	Description
1	Schedule/Task Tracking	Google docs SWD_ TỔNG HỢP
2	Project Backlog	Google docs, Excel
3	Source Codes	https://github.com/orgs/SWD-Vacine-project/repositories
4	Database Script(s)	SQL Server
5	Final Report Document	Final Project Report
6	Slide	Group 5_Slide
7	Test Cases Document	Group 5_TestCase

2. User Manual

2.1 Overview

The Vaccination Management App streamlines the vaccination process for children at a vaccination center, providing easy appointment booking, vaccination tracking, and feedback collection.

2.2 Workflow 1: Add children flow

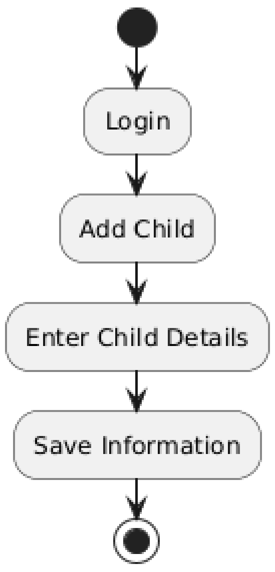
2.2.1. Introduction

This workflow allows customers to add their children's information to the system, enabling personalized vaccination scheduling and tracking.

2.2.2. Workflow Overview

1. The customer logs in to the system.
2. The customer selects the "Add Child" option.
3. The customer enters their child's details (e.g., name, date of birth, etc.).
4. The system saves the child's information in the database.

2.2.3. Workflow Diagram



2.3 Workflow 2: Purchase Vaccine & Make Payment

2.3.1. Workflow Introduction

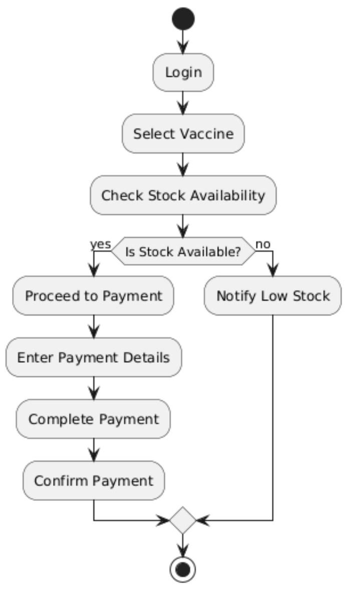
This workflow allows customers to purchase a vaccine for their child, select the appropriate vaccine, and complete the payment for the vaccination service.

2.3.2. Workflow Overview

1. The customer logs in to the system.
2. The customer selects the vaccine type for their child.
3. The system checks vaccine availability.
4. The customer proceeds to the payment page.
5. The customer enters payment details and completes the payment.

6. The system confirms the payment and updates the order status.

2.3.3. Workflow Diagram



2.3 Workflow 3: Edit Profile

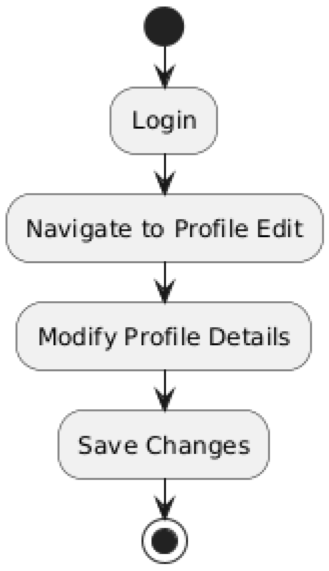
2.3.1. Introduction

This workflow allows customers to update their personal profile details in the system.

2.3.2. Overview

1. The customer logs in to the system.
2. The customer navigates to the profile edit section.
3. The customer modifies the desired fields (e.g., name, contact info).
4. The system saves the updated profile information.

2.3.3. Workflow Diagram


Login
Navigate to Profile Edit
Modify Profile Details
Save Changes