National University of Computer and Emerging Sciences



**Subject: Software Engineering**

**NESCON EVENT MANAGEMENT SYSTEM**

**By:**

Muhammad Mubeen (21I-0794)

Maaz Masood 21I2551  
Syed Ali 22I

Submitted to: *Mr. Naveed Khursheed*

### 

### **Introduction**

#### **Purpose**

The purpose of this SRS document is to provide a detailed description of the requirements for the **NASCON Event Management System**. It covers the initial phase of development focusing on the **User Registration and Login Module**, which is essential for accessing the system's core functionalities. This document outlines the scope, functional and non-functional requirements, and design considerations to ensure a successful implementation of the module.

#### **Document Conventions**

* **Bold** is used for section titles and emphasis.
* Monospace font is used for code, technical terms, and commands.
* The priorities for requirements are categorized as:  
  + **High**: Critical features necessary for system operation.
  + **Medium**: Features that enhance user experience but are not critical.
  + **Low**: Optional features that can be deferred for future versions.

#### **Intended Audience and Reading Suggestions**

This document is intended for:

* **Developers:** To understand the system requirements and design.
* **Project Managers:** To plan and track the progress of the project.
* **Testers:** To design test cases and validate system features.
* **University Administration and Event Organizers:** To review and ensure that the software meets organizational needs.

It is recommended to read the overview sections first to understand the system scope and purpose. Technical readers should then focus on the detailed requirements and design sections.

#### **Product Scope**

The **NASCON Event Management System** is a centralized platform for managing events such as hackathons, concerts, food stalls, and sports competitions. The system aims to streamline event registration, participant management, and scheduling. In this iteration, we will focus on the **User Registration and Login Module** to provide authenticated access for students, outsiders, and administrators.

#### **References**

* **Assignment 1 Document:** Describing the overall project scope and envisioned features.
* **IEEE Software Requirements Specification Template:** Used to structure this document.
* **GitHub Repository:** (Add the repository link when available)
* **Trello Board:** (Add the Trello board link when available)

### **Overall Description**

#### **Product Perspective**

The NASCON Event Management System is a new, self-contained product designed for the efficient management of events held by the university. It is intended to replace the manual and fragmented methods currently in use. The system will integrate various user roles (students, outsiders, organizers, admins) for seamless coordination.

#### **Product Functions**

The key functions of the system include:

* **User Registration and Login:** Allowing students, outsiders, and admins to create accounts and access the system.
* **Event Registration:** Registering for events and viewing event details.
* **User Management:** Managing user profiles and authentication.

#### **User Classes and Characteristics**

* **Students:** Primary users who can view and register for events.
* **Outsiders:** External users who can access limited features for event registration.
* **Organizers:** Responsible for managing events and participants.
* **Admins:** Have full access to the system for maintenance and monitoring.

#### **Operating Environment**

* **Hardware:** Desktop and laptop computers with standard peripherals.
* **Software:** The system will be deployed on a web-based platform, accessible through common browsers (e.g., Chrome).
* **Operating System:** Windows.
* **Database:** MySQL for backend data storage.

#### **Design and Implementation Constraints**

* **Compliance:** Must adhere to university IT policies and data protection regulations.
* **Language:** The initial implementation will use JavaScript and Express.js for the front end.
* **Database:** The backend will use My**SQL**.
* **Security:** User authentication must follow standard encryption practices.

#### **User Documentation**

The following user documentation will be provided:

* User manual for registration and login processes.
* Online help and FAQs for common issues.
* Training sessions for administrators.

#### **Assumptions and Dependencies**

* Internet connectivity is available for accessing the web-based system.
* Users have basic knowledge of computers and web browsing.
* The system will rely on a centralized server for data storage.
* Dependencies include integration with the university's student management system for verification.

### **External Interface Requirements**

#### **User Interfaces**

* **Login Screen:** Provides input fields for username and password along with a "Forgot Password" option.
* **Registration Form:** Includes fields for personal details such as name, email, role, and password.
* **Profile Management:** Users can update their information and change their passwords.
* **Error Handling:** Appropriate error messages will be displayed for invalid inputs.

#### **Hardware Interfaces**

* Compatible with standard input devices (keyboard, mouse) and display devices.
* Requires a stable internet connection.

#### **Software Interfaces**

* **Backend Database:** The system will use MySQL for backend data storage. It will manage user credentials, event information, and participant records. All data retrievals and updates will be handled through secure SQL queries.
* **Email System Integration:** The system will integrate with the university's official email system (e.g., Outlook) to send confirmation emails, password reset notifications, and event reminders. Communication will use **SMTP protocols**.
* **Web Browsers:** The system will be compatible with popular browsers like **Google Chrome (v95+)**. Browser compatibility testing will ensure proper functionality.
* **Development Framework:** The system will utilize **ASP.NET Core( or Express.js)** for the frontend and server-side logic.
* **API Integration:** Future iterations may include RESTful APIs for seamless integration with other university systems, such as student information systems or payment gateways.
* **Data Exchange:** JSON format will be used for any data exchange between the frontend and backend to maintain consistency.
* **Security:** All communication will be secured using **HTTPS** and data encryption standards (e.g., AES-256) for sensitive information.

#### **Communications Interfaces**

* **Communication Protocols:** The system will use **HTTPS** to ensure secure communication between the client and server. **TLS 1.2** or higher will be implemented for encryption.
* **Email Notifications:** The system will use **SMTP protocols** for sending emails, including registration confirmations, password reset requests, and event updates. Message formatting will follow standard **MIME** formats.
* **Web Browser Communication:** Communication with the client side will be based on **HTTP requests** for data retrieval and submission. The system will handle JSON-formatted requests and responses for seamless integration.
* **Network Server Communication:** The system will be hosted on a network server that supports **TCP/IP** protocols, ensuring reliable data transmission. The server will be configured to handle concurrent connections.
* **Data Transfer and Synchronization:** Asynchronous communication techniques will be used to maintain data synchronization without affecting user experience. The average data transfer rate is expected to be adequate for handling up to 100 simultaneous users without significant latency.
* **Security Measures:** Data encryption will be implemented for sensitive information like passwords. Authentication tokens will be used to maintain session security.

### **System Features**

#### **Feature 1: User Registration and Login**

* **Description and Priority:** Facilitates authenticated access to the system. Priority: **High**
* **Stimulus/Response Sequences:**
  + User submits registration form → System validates → Sends a confirmation email.
  + User enters login credentials → System authenticates → Grants access to the dashboard.
* **Functional Requirements:**
  + REQ-1: The system must validate user input for registration.
  + REQ-2: The system must authenticate users upon login.
  + REQ-3: The system should send a confirmation email upon successful registration.

#### **Feature 2: Event Registration**

* **Description and Priority:** Allows users to register for events like hackathons, concerts, and sports competitions. Priority: **High**
* **Stimulus/Response Sequences:**
  + User selects an event → Views event details → Registers for the event → Receives a confirmation message.
  + Organizer schedules an event → System updates the event list → Users can see the updated events.
* **Functional Requirements:**
  + REQ-4: The system should display available events for registration.
  + REQ-5: The system must validate user eligibility before registering for an event.
  + REQ-6: The system should confirm successful registration through an on-screen message and an email notification.

#### **Feature 3: Profile Management**

* **Description and Priority:** Enables users to view and update their profiles. Priority: **Medium**
* **Stimulus/Response Sequences:**
  + User accesses profile page → Views existing information → Edits and updates details → Receives a confirmation of changes.
* **Functional Requirements:**
  + REQ-7: The system should allow users to view their profile details.
  + REQ-8: The system should enable users to update their personal information.
  + REQ-9: The system should validate changes and prompt users for confirmation before updating.

### **Other Nonfunctional Requirements**

#### **Performance Requirements**

* The system should handle up to **100 concurrent users** without significant degradation in performance.
* Response time for user requests (e.g., login, registration) should be **less than 2 seconds**.
* The system should support batch processing of event registrations and data updates without affecting active user sessions.

#### **Safety Requirements**

* The system should maintain data integrity to prevent accidental loss or corruption of user data.
* Backup and recovery procedures should be in place to protect against data loss.
* System administrators must be trained to handle data recovery and conflict resolution in case of errors.

#### **Security Requirements**

* User passwords must be stored using **strong encryption techniques** (e.g., AES-256).
* The system should implement **multi-factor authentication (MFA)** for administrative access.
* Data transmitted between the client and server must be encrypted using **HTTPS/TLS**.
* User roles and access levels should be defined to restrict unauthorized access.

#### **Software Quality Attributes**

* **Reliability:** The system should have a **99.9% uptime** to ensure availability during event seasons.
* **Maintainability:** The system code should be modular and follow industry-standard coding practices for easy maintenance.
* **Scalability:** The system should accommodate future expansion to include more event types and user roles.
* **Usability:** The system interface should be user-friendly and intuitive, requiring minimal training.

#### **Business Rules**

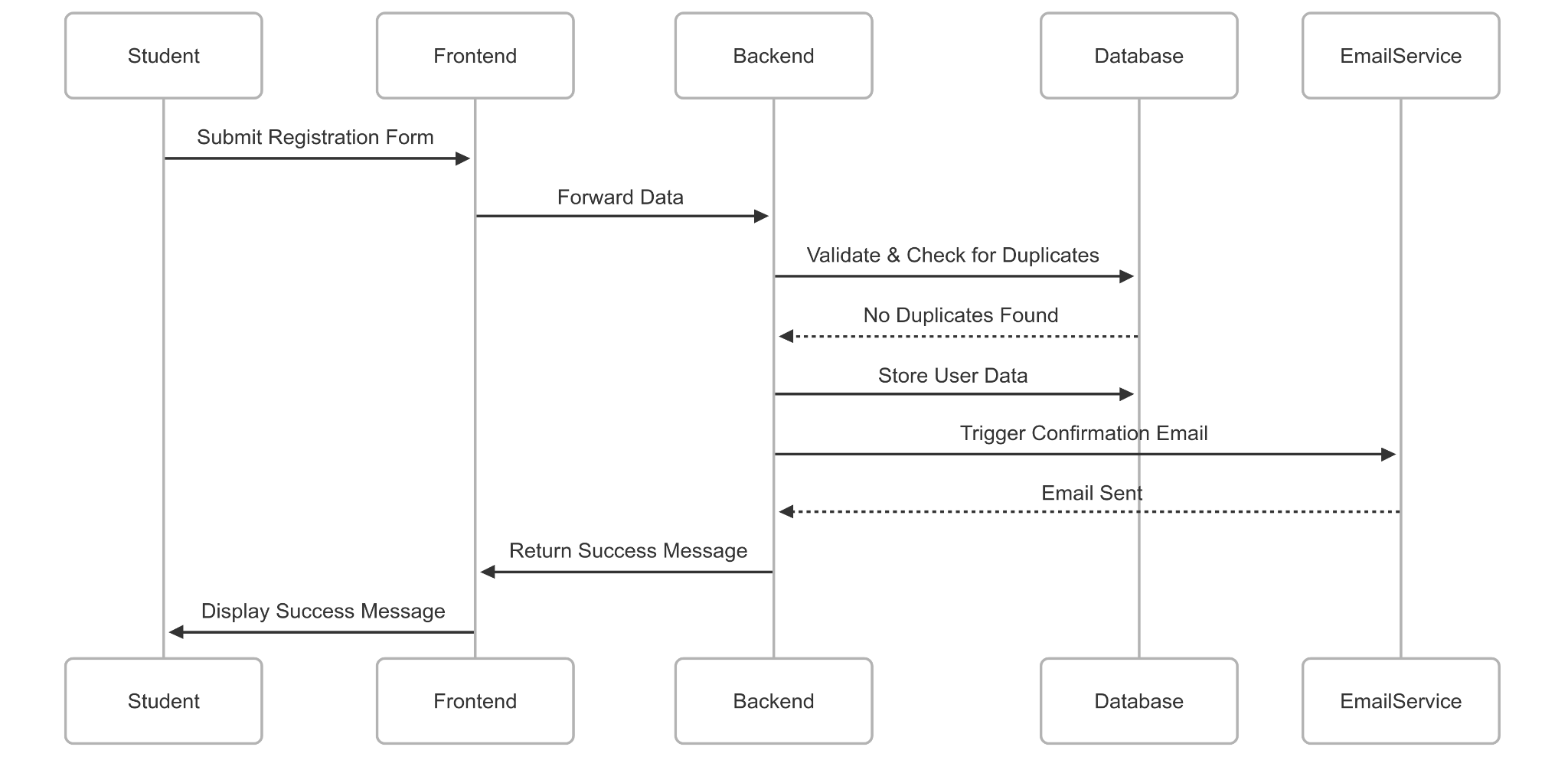
* Only **registered users** can access the system’s main features like event registration and profile management.
* **Organizers** can only manage the events they create.
* **Admins** have the right to create, update, and delete events, users, and roles.
* Event registrations must be closed **24 hours before the event's start time**.

### **Sprint 1 Backlog**

**Sprint Duration:** 2 weeks  
 **Module(s):** User Registration, Login, and Profile Management

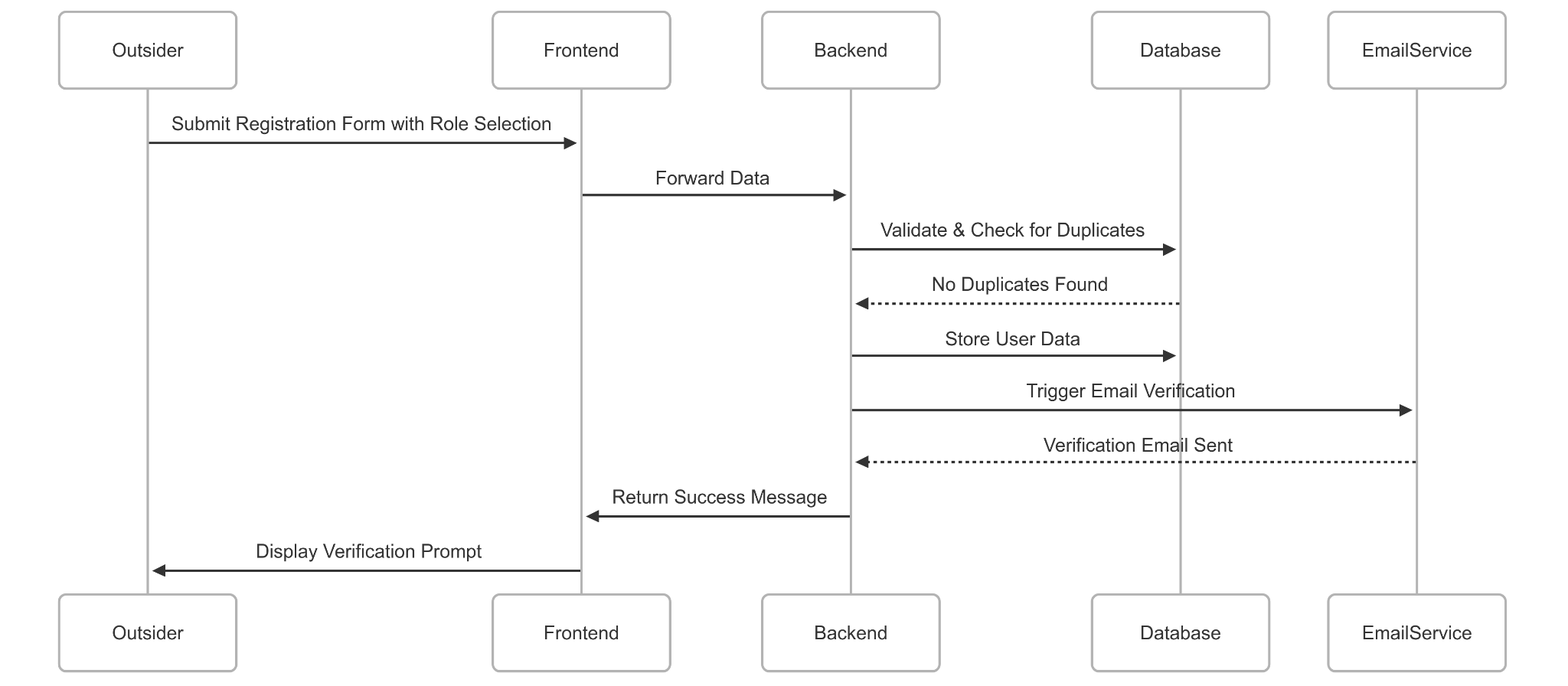
#### **User Story 1: Student Registration**

* **As a** student
* **I want to** register for an account
* **So that** I can participate in NASCON events
* **Sub-Tasks:**
  + Design registration form UI
  + Validate input fields (email, name, password)
  + Store user details in database
  + Send confirmation email after registration



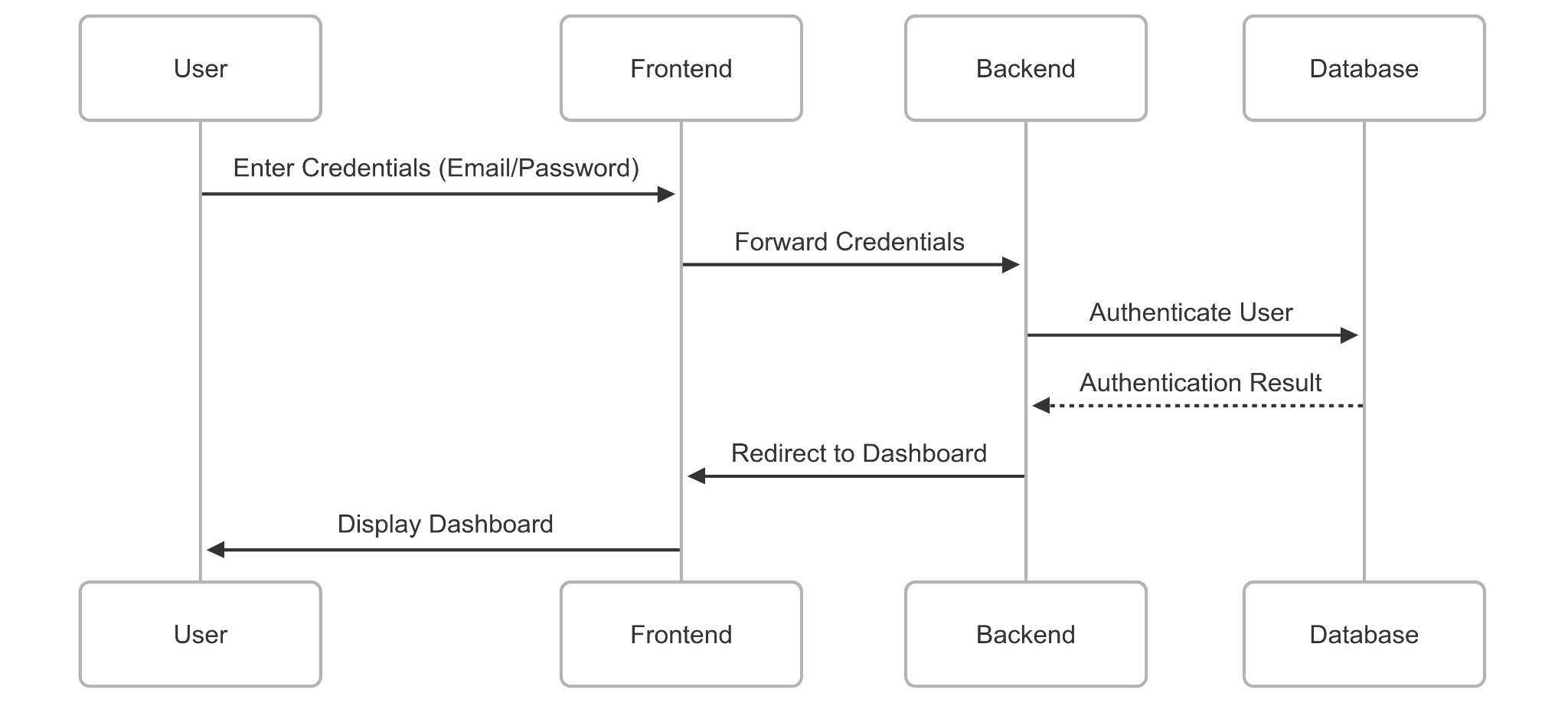
#### **User Story 2: Outsider Registration**

* **As an** outsider
* **I want to** register for an account
* **So that** I can join NASCON events
* **Sub-Tasks:**
  + Add role selection in registration form
  + Implement email verification for outsiders



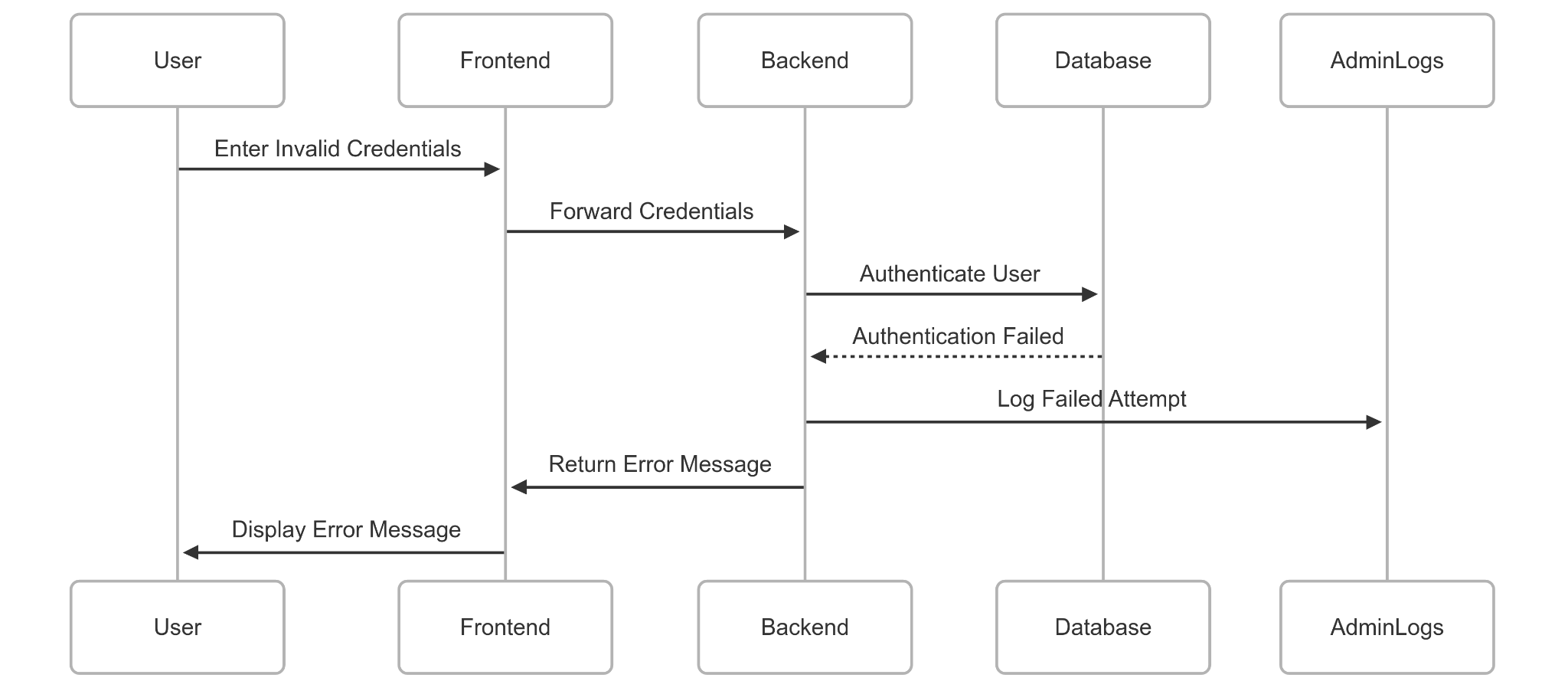
#### **User Story 3: User Login**

* **As a** registered user
* **I want to** log in to my account
* **So that** I can manage my events
* **Sub-Tasks:**
  + Create login form UI
  + Implement password encryption
  + Authenticate credentials
  + Redirect user to dashboard



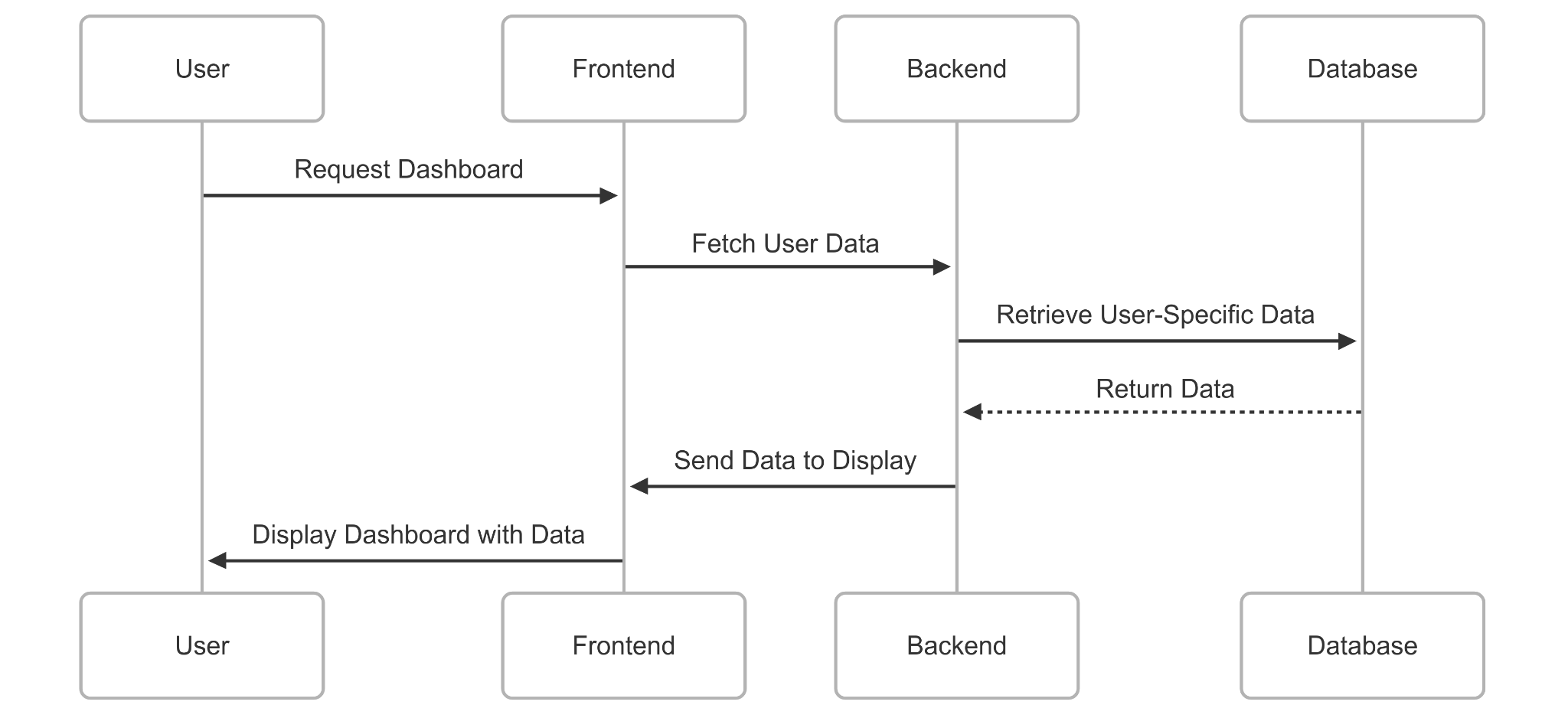
#### **User Story 4: Invalid Login Feedback**

* **As a** user
* **I want to** be notified of invalid login attempts
* **So that** I know what went wrong
* **Sub-Tasks:**
  + Show error messages for wrong credentials
  + Log failed attempts for admin reference



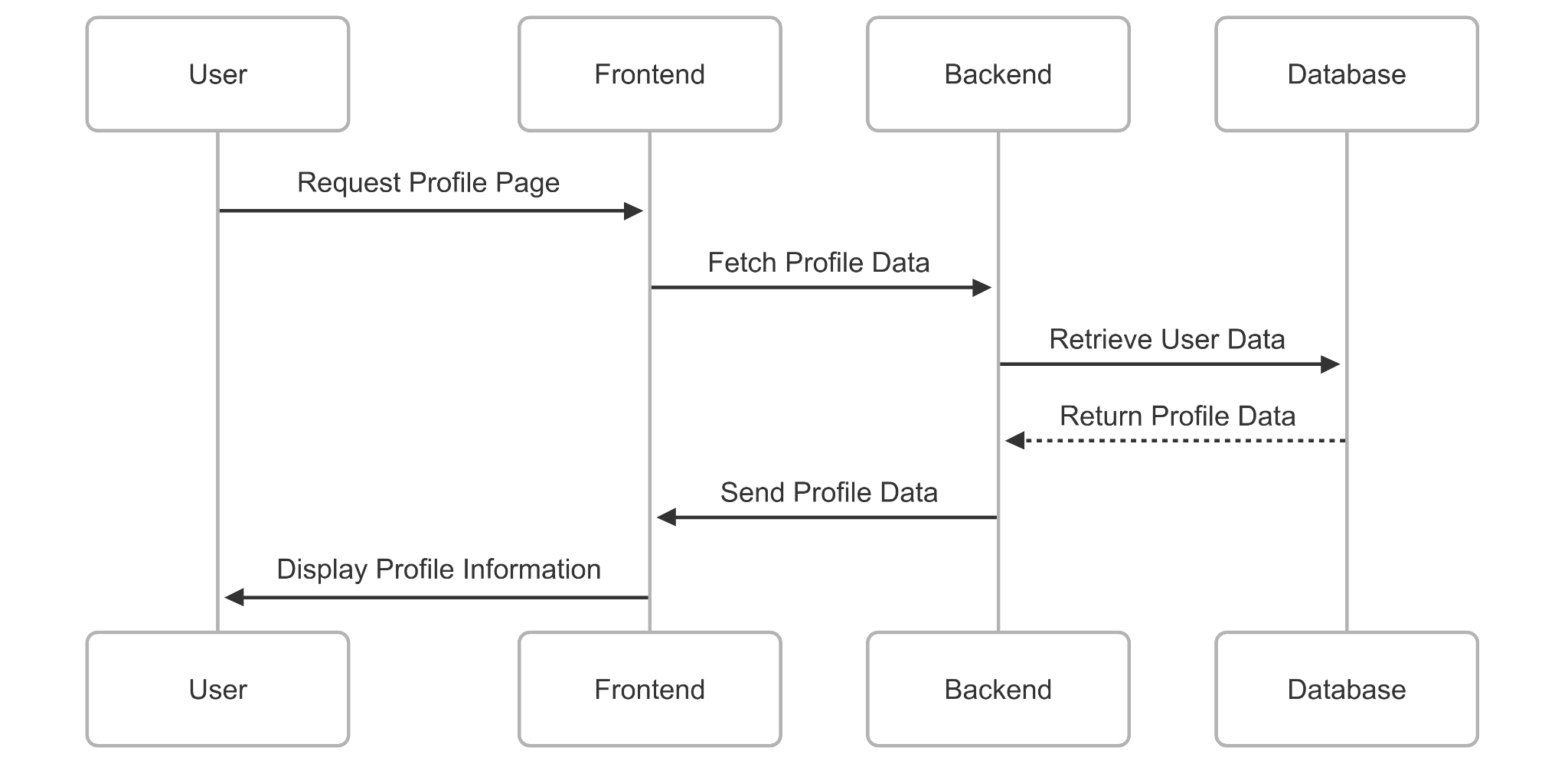
**User Story 5: Dashboard Access**

* **As a** logged-in user
* **I want to** view my dashboard
* **So that** I can manage my registrations and profile
* **Sub-Tasks:**
  + Design dashboard layout
  + Fetch and display user-specific data



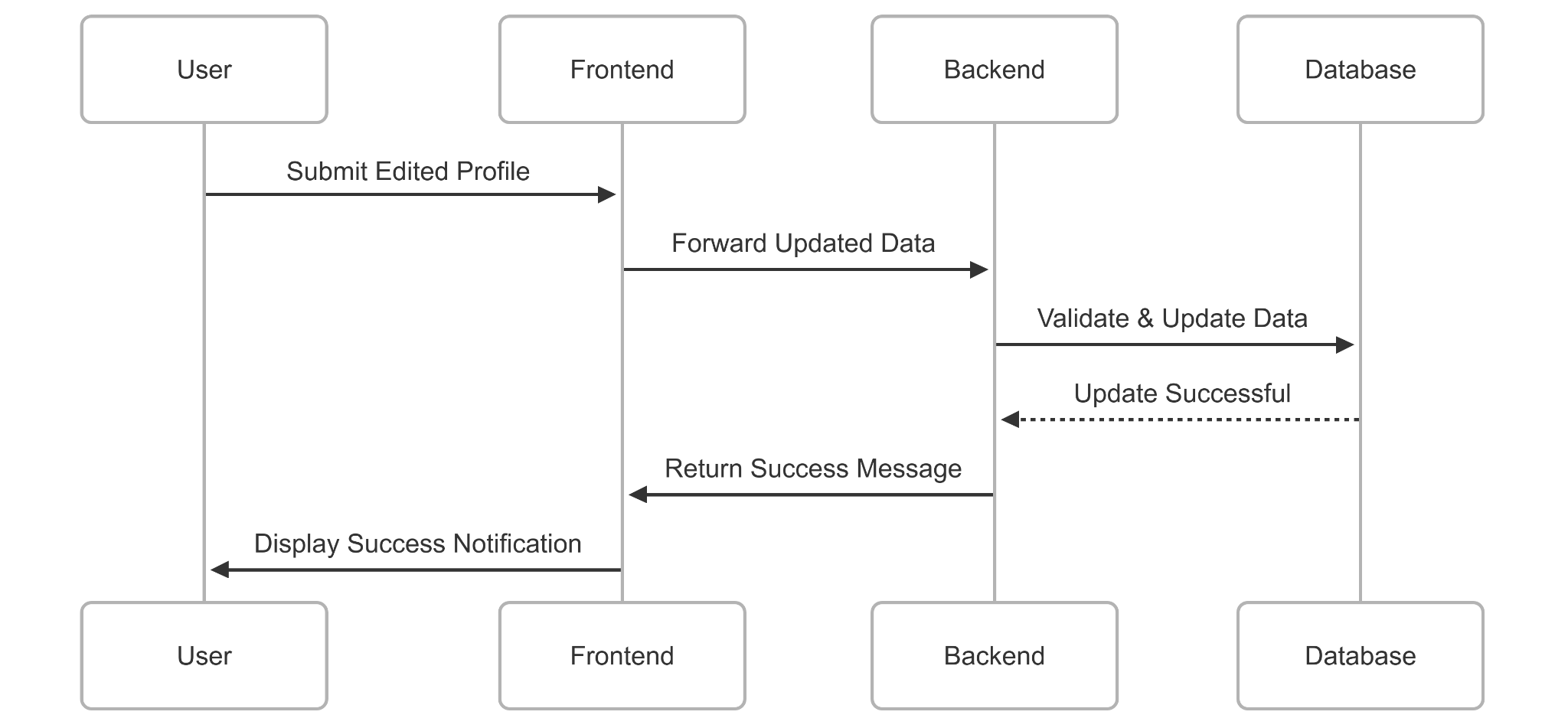
#### **User Story 6: View Profile**

* **As a** user
* **I want to** view my profile information
* **So that** I can confirm my registered details
* **Sub-Tasks:**
  + Create profile viewing page
  + Retrieve user data from DB



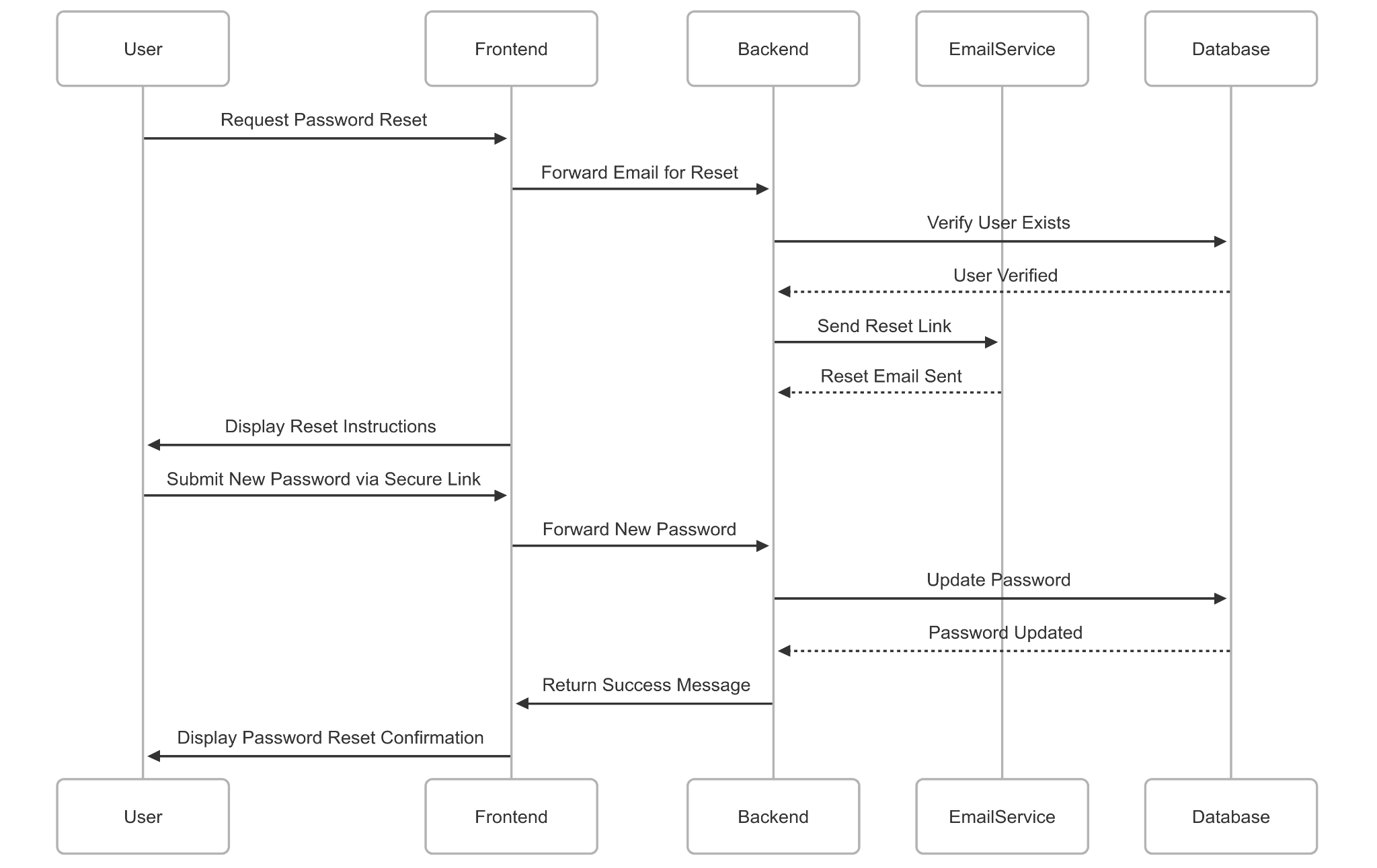
#### **User Story 7: Edit Profile**

* **As a** user
* **I want to** edit my profile
* **So that** I can keep my details updated
* **Sub-Tasks:**
  + Implement editable fields
  + Validate inputs
  + Update database
  + Show success notification



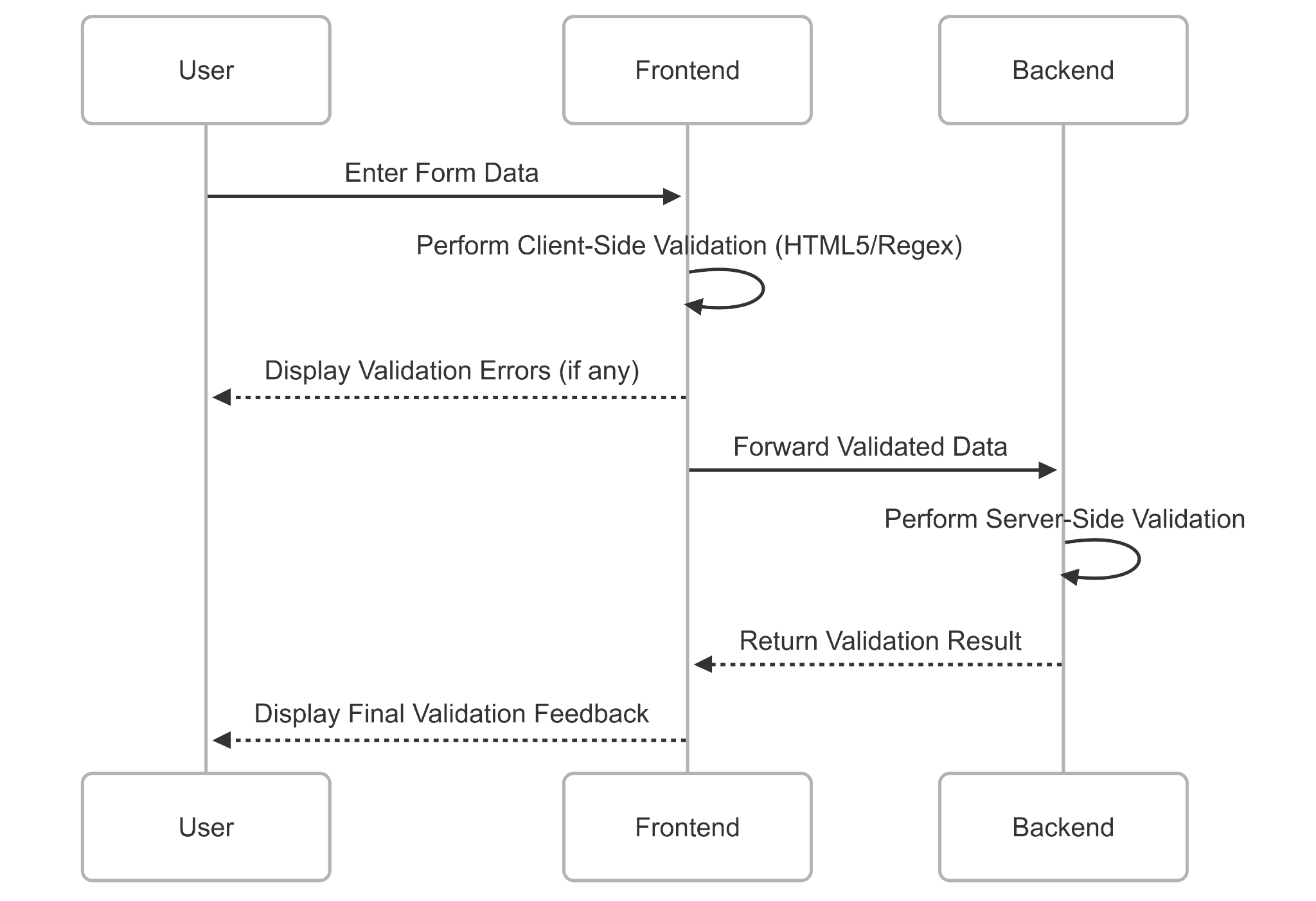
#### **User Story 8: Password Recovery**

* **As a** user
* **I want to** reset my password
* **So that** I can regain account access
* **Sub-Tasks:**
  + Create "Forgot Password" UI
  + Send password reset email
  + Allow password reset through secure link



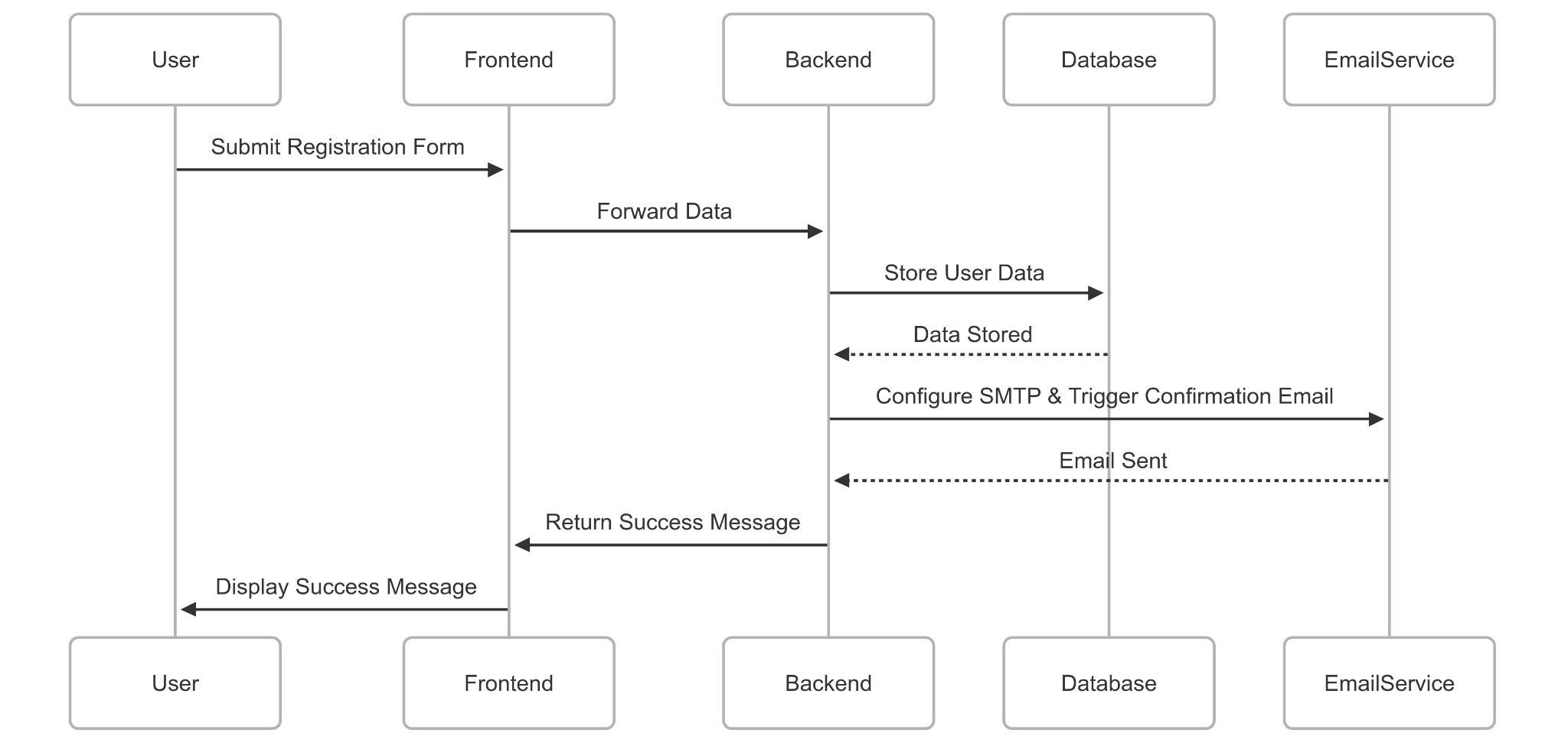
#### **User Story 9: Input Field Validation**

* **As a** user
* **I want to** be prompted to enter correct information
* **So that** I don't make mistakes during form filling
* **Sub-Tasks:**
  + Apply client-side validations (e.g., HTML5)
  + Use regex for email/password checks



#### **User Story 10: Confirmation Email System**

* **As a** newly registered user
* **I want to** receive a confirmation email
* **So that** I know my registration was successful
* **Sub-Tasks:**
  + Configure SMTP service
  + Create email template
  + Trigger email after DB insertion



### **Software Testing**

#### **Equivalence Class Partitioning**

| **Field** | **Valid Classes** | **Invalid Classes** | **Remarks** |
| --- | --- | --- | --- |
| **Email** | Valid email format (abc@domain.com) | Missing '@', missing domain, empty input | Use regex for validation |
| **Password** | 8–16 characters, includes letters and numbers | Less than 8 chars, no number/letter, empty | Must be encrypted |
| **Username** | Alphabetic string (min 3 chars) | Numeric only, special chars, < 3 chars | Shown in profile/dashboard |
| **Phone Number** | 11-digit numeric string | Non-numeric, <11 or >11 digits | Optional but validated |
| **Role** | Student, Outsider, Admin | Blank, any other string | Dropdown enforced |

#### **Weak and Strong Equivalence Class Testing**

* **Weak Equivalence Class Testing (WECT):**
  + One representative value is selected from each equivalence class.
  + Approximate number of test cases: **20** (1 for each valid/invalid class across 5 fields).
* **Strong Equivalence Class Testing (SECT):**
  + All combinations of valid/invalid classes are considered.
  + Approximate number of test cases: **32–40** (based on multiplying valid/invalid classes per field).

#### **Boundary Value Analysis**

| **Field** | **Valid Boundary Values** | **Invalid Boundary Values** | **Remarks** |
| --- | --- | --- | --- |
| **Email** | Valid formats only | Empty, near-invalid formats | No numerical range; focus on structural rules |
| **Password** | 8, 9, 16 characters | 7, 17 characters | Password must be between 8–16 characters |
| **Username** | 3, 4 characters | 2, 1 characters | Min length = 3 |
| **Phone No.** | 11 digits | 10, 12 digits | Must be exactly 11 digits |

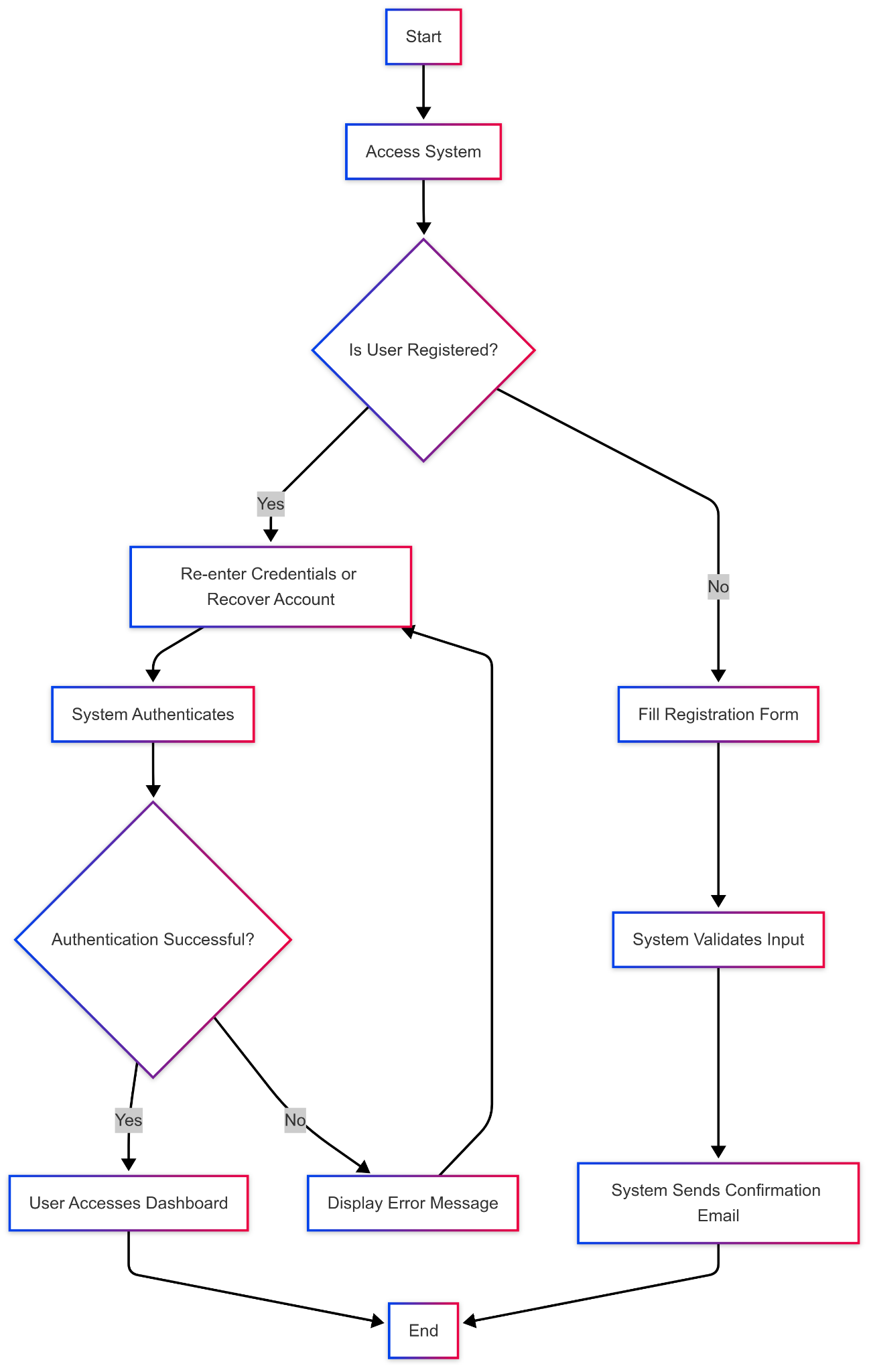
### **Diagrams**

#### **Use Case Diagram**

* This diagram will illustrate the interactions between the system and its users, including Students, Outsiders, Organizers, and Admins.
* Use cases will cover user registration, login, event registration, and profile management.

#### **Activity Diagram**

* The activity diagram will depict the flow of actions involved in user registration, event registration, and login processes.
* It will showcase decision points, parallel activities, and synchronization.



#### **Sequence Diagram**

* The sequence diagram will illustrate the communication between users and the system during login, event registration, and profile updates.
* It will represent the order of interactions between the user interface, authentication system, and database.

