Table of Contents

[List of acronyms vi](#_Toc200699112)

[Acknowledgment vii](#_Toc200699113)

[Abstract viii](#_Toc200699114)

[CHAPTER ONE 1](#_Toc200699115)

[Introduction 1](#_Toc200699116)

[1.1 Background of the Project 1](#_Toc200699117)

[1.2 Statement of the Problem 2](#_Toc200699118)

[1.3 Objective of the Project 2](#_Toc200699119)

[1.3.1 General Objective 2](#_Toc200699120)

[1.3.2 Specific Objectives 2](#_Toc200699121)

[1.4 Methodology 3](#_Toc200699122)

[1.4.1 Data Collection 3](#_Toc200699123)

[1.4.2 System Development Process Model 4](#_Toc200699124)

[1.4.3 Selected Methodology: Agile 4](#_Toc200699125)

[1.4.4 Design Pattern 4](#_Toc200699126)

[1.4.5 Technology Stack 5](#_Toc200699127)

[1.4.6 Programming Languages 5](#_Toc200699128)

[1.5 Tools 6](#_Toc200699129)

[1.5.1 Hardware Tools 6](#_Toc200699130)

[1.5.2 Software Tools 6](#_Toc200699131)

[1.6 Scope of the Project 7](#_Toc200699132)

[1.7 Significance of the Project 8](#_Toc200699133)

[1.8 Feasibility Study 8](#_Toc200699134)

[1.8.1 Technical Feasibility 8](#_Toc200699135)

[1.8.2 Economic Feasibility 8](#_Toc200699136)

[1.8.3 Operational Feasibility 8](#_Toc200699137)

[1.8.4 Legal Feasibility 9](#_Toc200699138)

[1.9 Risk Assessment 9](#_Toc200699139)

[1.9.1 Risks and Mitigation 9](#_Toc200699140)

[1.9.2 Assumptions 9](#_Toc200699141)

[1.9.3 Constraints 9](#_Toc200699142)

[CHAPTER TWO 10](#_Toc200699143)

[Business area analysis and requirement definition 10](#_Toc200699144)

[2.1 Introduction 10](#_Toc200699145)

[2.2 Business Area Analysis 11](#_Toc200699146)

[2.2.1 Detailed Analysis 11](#_Toc200699147)

[2.2.2 Current System 11](#_Toc200699148)

[2.2.3 Players of the Existing System 11](#_Toc200699149)

[2.2.4 Proposed System 12](#_Toc200699150)

[2.2.5 Forms and Reports Used 12](#_Toc200699151)

[2.3 Requirement Gathering 13](#_Toc200699152)

[2.3.1 Requirement Gathering Techniques 13](#_Toc200699153)

[2.4 Method of Communication 13](#_Toc200699154)

[2.4.1 Communication Techniques 13](#_Toc200699155)

[2.5 Requirement Definition 13](#_Toc200699156)

[2.5.1 Functional Requirements 13](#_Toc200699157)

[2.5.2 Non-Functional Requirements 15](#_Toc200699158)

[2.5.3 Collaboration Modeling 16](#_Toc200699159)

[2.6 System Modeling 17](#_Toc200699160)

[2.6.1 Use Case Diagram 17](#_Toc200699161)

[2.6.1.1 Essential Use Case Modeling 17](#_Toc200699162)

[2.6.1.2 Actor Description 18](#_Toc200699163)

[2.6.1.3 Essential Use Case Description 20](#_Toc200699164)

[2.6.2 Sequence diagram 27](#_Toc200699165)

[2.6.3 State chart Diagram 35](#_Toc200699166)

[2.6.4 Activity Diagram 36](#_Toc200699167)

[2.6.5 Class Diagram 43](#_Toc200699168)

[2.6.6 Multiplicity of Relationships between Classes 45](#_Toc200699169)

[2.6.7 User interface prototyping 46](#_Toc200699170)

[CHAPTER THREE 49](#_Toc200699171)

[System Design 49](#_Toc200699172)

[3.1 Introduction 49](#_Toc200699173)

[3.2 Purpose of the system 49](#_Toc200699174)

[3.3 System Design 49](#_Toc200699175)

[3.4 Current Software Architecture 50](#_Toc200699176)

[3.5 Proposed Software Architecture 51](#_Toc200699177)

[3.5.1 Subsystem Decomposition 53](#_Toc200699178)

[3.5.2 Component diagram 56](#_Toc200699179)

[3.5.3 Deployment diagram 56](#_Toc200699180)

[3.5.4 Persistent Data Management 57](#_Toc200699181)

[3.5.5 Detailed Database Design 60](#_Toc200699182)

[3.5.6 Access Control and Security 61](#_Toc200699183)

[3.5.7 Global Software Control 62](#_Toc200699184)

[3.5.8 Boundary Conditions 63](#_Toc200699185)

[References 65](#_Toc200699186)

List of Tables

[Table 1.1 Programming Languages 5](#_Toc200656815)

[Table 1.2 Software tools 6](#_Toc200656816)

[Table 1.3 risk and mitigation 9](#_Toc200656817)

[Table 2.1Actor Description 18](#_Toc200656818)

[Table 2.2 Use case description for registration 20](#_Toc200656819)

[Table 2.3 Use case description for login 21](#_Toc200656820)

[Table 2.4 Use case description for View Available Internships 22](#_Toc200656821)

[Table 2.5 Use case description for Apply for Internship 23](#_Toc200656822)

[Table 2.6 Use case description for Post Internship Opportunity 23](#_Toc200656823)

[Table 2.7 Use case description for Approve Student Application 24](#_Toc200656824)

[Table 2.8 Use case description for Submit Final Internship Report 25](#_Toc200656825)

[Table 2.9 Use case description for Track Internship Progress 26](#_Toc200656826)

[Table 2.10 Use case description for Notification System 27](#_Toc200656827)

[Table 2.11 Multiplicity of Relationships between Classes 46](#_Toc200656828)

[Table 3.1 How the New Architecture Solves Current Limitations: 52](#_Toc200656829)

[Table 3.2 Architecture Comparison Summary 53](#_Toc200656830)

[Table 3.3 List of Subsystems & Their Descriptions 54](#_Toc200656831)

[Table 3.4 Users Table 59](#_Toc200656832)

[Table 3.5 Student Table 59](#_Toc200656833)

[Table 3.6 Companies Table 59](#_Toc200656834)

[Table 3.7 Relational Tables 60](#_Toc200656835)

[Table 3.8 Extreme Conditions 64](#_Toc200656836)

List of Figures

[Figure 1 Use case for WBIPS 19](#_Toc201345148)

[Figure 5 Sequence Diagram for Registration 27](#_Toc201345149)

[Figure 6 Sequence Diagram for Login 28](#_Toc201345150)

[Figure 7 Sequence Diagram for View Internship Listing 29](#_Toc201345151)

[Figure 8 Sequence Diagram for Apply for Internship 30](#_Toc201345152)

[Figure 9 Sequence Diagram for Post Internship Opportunity 31](#_Toc201345153)

[Figure 10 Sequence Diagram for Approve Student Application 32](#_Toc201345154)

[Figure 11 Sequence Diagram for Track Internship Progress 33](#_Toc201345155)

[Figure 12 Sequence Diagram for Submit Final Report 34](#_Toc201345156)

[Figure 13 State chart diagram 35](#_Toc201345157)

[Figure 14 Activity Diagram for Registration 36](#_Toc201345158)

[Figure 15 Activity Diagram for Login 37](#_Toc201345159)

[Figure 16 Activity Diagram for Logout 38](#_Toc201345160)

[Figure 17 Activity Diagram for Apply for Internship 39](#_Toc201345161)

[Figure 18 Activity Diagram for Post Internship Opportunity 40](#_Toc201345162)

[Figure 19 Activity Diagram for Approve Student Application 41](#_Toc201345163)

[Figure 20 Activity Diagram for Admin 42](#_Toc201345164)

[Figure 21 Class Diagram 44](#_Toc201345165)

[Figure 22 User interface prototyping 47](#_Toc201345166)

[Figure 23 ER Diagram 48](#_Toc201345167)

[Figure 24 Subsystem Decomposition 55](#_Toc201345168)

[Figure 25 Component Diagram 56](file:///C:\Users\Admin\Desktop\Nati\IT\Internship.docx#_Toc201345169)

[Figure 26 Deployment Diagram 57](#_Toc201345170)

# List of acronyms

1. HR - Human Resources
2. MVC - Model-View-Controller
3. RAD - Rapid Application Development
4. SQL - Structured Query Language
5. HTML – Hyper Text Markup Language
6. CSS - Cascading Style Sheets
7. JS - JavaScript
8. XAMPP - Cross-Platform (X), Apache, MySQL, PHP, Perl
9. API - Application Programming Interface
10. UI/UX - User Interface/User Experience
11. ER - Entity-Relationship
12. SSL - Secure Sockets Layer
13. TLS - Transport Layer Security
14. JWT - JSON Web Token
15. XSS - Cross-Site Scripting
16. ACID - Atomicity, Consistency, Isolation, Durability
17. PK - Primary Key
18. FK - Foreign Key
19. RBAC - Role-Based Access Control
20. WBIPS - Web-Based Internship Placement System

# 

# Acknowledgment

We would like to sincerely thank our advisor, Mr. Samuel Shibabw for his guidance, encouragement, and valuable insights throughout the development of this project documentation.

Our gratitude also goes to the Department of Computer Science at St. Mary University for providing the academic framework and support necessary for this research.

Lastly, we appreciate the motivation and collaboration of our team, which played a vital role in completing this phase of the project.

# 

# Abstract

The Web-Based Internship Placement System is designed to improve the internship management process between students, companies, and university coordinators. The system aims to provide a centralized digital platform that allows students to apply for internships online, enables companies to post available positions, and gives coordinators the tools to monitor, approve, and evaluate internship activities.

This document focuses on the design and documentation phase of the system. It includes detailed business analysis, requirement gathering, use case modeling, class diagrams, and database design using object-oriented principles. The system is structured to address the inefficiencies of the current manual internship assignment process, such as lack of tracking, communication issues, and time delays.

Key design tools used include UML diagrams (use case, class, sequence, ERD), Agile methodology for future development, and low-fidelity UI prototypes. The documentation serves as a foundation for implementation in the next phase of the project.

# CHAPTER ONE

## Introduction

## 1.1 Background of the Project

Internships have become an essential part of higher education, helping students to transition smoothly from academic life to the professional world. In Ethiopia, universities require students to complete internships as part of their graduation requirements. The goal is to enable students to apply theoretical knowledge in real-life settings, gain professional experience, and build industry connections before they formally enter the job market.

Despite the importance of internships, the process of internship placement in many Ethiopian universities remains largely manual. Traditional methods, such as paper-based notice boards, word-of-mouth communication, and random emailing, are still widely used. These outdated practices cause inefficiencies, delays, and sometimes even unfair placement of students. Students often face confusion, lack of information, or end up securing internships unrelated to their field of study due to the absence of an organized system.

With the growth of digital technologies, web-based systems are now being used worldwide to manage student internships more efficiently. A web-based internship placement system can centralize information, simplify the application process, automate placement tracking, and allow real-time communication between students, universities, and companies.

The proposed project, titled “Development of a Web-Based Internship Placement System for Ethiopian University Students,” aims to address the challenges in the current system by designing a digital platform tailored for Ethiopian universities. The system will enable students to browse available internships, apply online, and allow coordinators to manage placements easily, thereby modernizing the entire process.

## 1.2 Statement of the Problem

The existing manual internship placement system in Ethiopian universities presents significant challenges. First, students lack a centralized platform to access and apply for internships. They often have to consult various departments or external sources, resulting in missed opportunities or confusion.

Second, communication between universities and students regarding internships is delayed and inconsistent. This causes students to apply under pressure or miss deadlines, and in many cases, accept internships that do not align with their academic background.

Third, the application process itself is inefficient and paper-based. Students must submit physical forms, await feedback without updates, and there’s no systematic way for universities to track or manage applications.

Moreover, universities face difficulty monitoring internship progress, evaluating outcomes, and maintaining records due to the absence of a digital platform. Students in rural areas face an additional barrier, as they lack regular access to notice boards or university offices.

These problems demonstrate the need for a digital, web-based system that streamlines the entire internship process for students, companies, and universities alike.

## 1.3 Objective of the Project

### 1.3.1 General Objective

To develop a web-based internship placement system that simplifies the internship process and enhances the connection between students, universities, and companies in Ethiopia.

### 1.3.2 Specific Objectives

* To analyze the current internship placement challenges faced by Ethiopian universities.
* To identify the system’s functional and non-functional requirements.
* To design a user-friendly interface tailored for students, administrators, and companies.
* To develop a secure and scalable web application using appropriate technologies.
* To implement features that support user registration, internship listings, applications, and approval workflows.
* To provide an admin dashboard for monitoring student placements and managing internship data.
* To integrate a notification system for timely updates and feedback.
* To ensure user data protection through secure authentication and authorization mechanisms.
* To conduct system testing and deploy a working prototype for pilot use.

## 1.4 Methodology

### 1.4.1 Data Collection

To better understand the needs of students, companies, and universities, and to design a system that meets these needs, the following data collection methods were used:

* Survey/Questionnaires:

Distributed to a sample group of students across different departments to gather insights on the challenges they face during internship placements.

* Interviews:

Conducted with internship coordinators, administrative staff, and company HR managers to understand the requirements from an administrative perspective.

* Observation:

Observed the current manual placement processes at selected universities to identify weaknesses and areas that could be automated.

* Document Analysis:

Reviewed existing guidelines, templates, and reports related to internship programs to understand current documentation processes.

### 1.4.2 System Development Process Model

There are several system development methodologies, including:

* Waterfall Model – A linear, step-by-step development process
* V-Model – A verification and validation-based approach
* RAD (Rapid Application Development) – Focuses on quick development through prototyping
* Agile Model – An iterative and flexible approach that supports continuous feedback and improvement

### 1.4.3 Selected Methodology: Agile

The Agile Development Model is selected for this project. Agile emphasizes iterative development, active user collaboration, and flexibility in response to changes. It allows continuous testing, quick adaptation to feedback, and short development cycles (sprints).

Reasons for Choosing Agile:

* Requirements may evolve during development as user feedback is received
* Encourages close collaboration with stakeholders (students, universities, and companies)
* Ensures early and frequent delivery of working modules
* Allows for continuous improvement and better risk management

### 1.4.4 Design Pattern

The Model-View-Controller (MVC) design pattern will be adopted.

• Model: Manages the data and business logic.

• View: Handles the display and user interface.

• Controller: Handles user input and updates the model or view accordingly.

MVC is chosen because it separates concerns, making the application more manageable, scalable, and easier to maintain, especially when adding new features or making changes.

### 1.4.5 Technology Stack

The system is primarily planned to use PHP and SQL. However, Node.js and React.js may also be used, depending on project needs and performance requirements. This decision will be finalized during the development phase based on feasibility, team expertise, and system complexity.

### 1.4.6 Programming Languages

Table 1.1 Programming Languages

|  |  |  |
| --- | --- | --- |
| Component | Language | Reason |
| Frontend | HTML, CSS, JavaScript (light framework if needed) | For interactive and responsive user interfaces |
| Backend | PHP | Widely used for web applications, easy integration with SQL databases and suitable for server-side scripting. |
| Database | SQL | Reliable, structured data storage with powerful querying capability |

## 1.5 Tools

### 1.5.1 Hardware Tools

• Personal Computers (for development and testing)

• Smartphones (for testing mobile responsiveness)

### 1.5.2 Software Tools

Table 1.2 Software tools

|  |  |
| --- | --- |
| Tool | Purpose |
| Visual Studio Code | Code editor for PHP, HTML, CSS and JavaScript |
| XAMPP | Local server environment |
| MySQL Workbench | Database design and management |
| Postman | Testing APIs if needed |
| Git and GitHub | Version control and collaboration |
| Figma | UI/UX design and prototyping |
| Draw.io | Diagram creation (ER diagrams, usecase diagrams, class diagrams, etc…) |
| Google chrome | Browser for system testing and debugging |

## 1.6 Scope and Limitations of the Project

### 1.6.1 Scope of the Project

This project focuses on building a web-based internship placement system specifically for use within Ethiopian universities. The system will support students in finding and applying for internships, enable companies to post internship opportunities, and allow university coordinators to manage the entire placement process. It will be accessible through desktop and mobile web browsers, operating during academic and internship periods in Ethiopia.

### 1.6.2 Limitations of the Project

Although the proposed Web-Based Internship Placement System is designed to enhance efficiency and transparency in internship management, it is expected to face the following limitations during and after implementation:

* Limited Internet Access: The system will rely on stable internet connectivity, which may not be consistently available to all users, especially in rural areas.
* Mobile Optimization: While designed to be responsive, the system may not perform equally across all mobile devices and screen sizes without further optimization.
* Scalability Constraints: In its initial deployment, the system will not be tested for large-scale use across multiple universities, which may limit its scalability until further enhancements are made.
* Security Limitations: The system will include basic authentication and authorization, but advanced security features such as two-factor authentication or intrusion detection will not be implemented in the first release.
* Company Verification: The verification process for company accounts will initially be manual, which may introduce delays or risks without integration with official business registries.
* Training Requirements: Users such as coordinators and company representatives will require orientation or training to use the system effectively, which may affect adoption speed.

## 1.7 Significance of the Project

This project has value across multiple stakeholders:

* Students will benefit from a centralized, digital platform where they can explore and apply for internships with ease and receive timely updates.
* Universities will enjoy improved oversight, easier record-keeping, and the ability to manage placements efficiently.
* Companies gain structured access to student applicants and can manage internship opportunities digitally.
* Government/Educational Bodies can use the platform for better reporting and strategic decision-making on internship outcomes.
* Future Researchers may extend or adapt the system for other academic or employment matchmaking purposes.

## 1.8 Feasibility Study

Feasibility studies help to evaluate whether the project can be successfully completed under current conditions.

### 1.8.1 Technical Feasibility

All selected hardware and software tools are available and sufficient. The project team has the required skills in Node.js, React.js, MongoDB, and software design to carry out the project successfully.

### 1.8.2 Economic Feasibility

Development costs are minimal because most tools are open-source. Hosting and operational costs are manageable within a student project budget. Overall, the project is economically viable.

### 1.8.3 Operational Feasibility

Once deployed, the system will require low maintenance. It solves current placement problems, increases efficiency, and will be easily adopted by students and administrators.

### 1.8.4 Legal Feasibility

There are no known legal barriers for the development and deployment of the system. Data privacy policies will be followed to ensure users’ information is protected.

## 1.9 Risk Assessment

### 1.9.1 Risks and Mitigation

|  |  |
| --- | --- |
| Risk | Mitigation |
| Internet Access Problems | Design lightweight pages that load even with slow connections |
| Low User Adoption | Provide user training and awareness programs |
| Technical Challenges | Conduct thorough testing and security checks |
| Security Risks | Implement user authentication and data encryption mechanisms |

Table 1.3 risk and mitigation

### 1.9.2 Assumptions

* Internet access will be available for the majority of users.
* Universities will support the adoption of the system.
* Students and companies will actively participate in using the platform.

### 1.9.3 Constraints

* Limited timeframe for development and deployment.
* Limited financial resources for professional hosting services.
* Dependency on external companies to post internships regularly.

# CHAPTER TWO

# Business area analysis and requirement definition

## 2.1 Introduction

This chapter focuses on understanding the existing internship placement environment, identifying its key players, and defining the requirements for the proposed system. It includes a detailed analysis of the current process, stakeholder interactions, data collection methods, and clearly outlines the system’s functional and non-functional requirements.

## 2.2 Business Area Analysis

### 2.2.1 Detailed Analysis

Internship placement in Ethiopian universities is currently handled manually. The process involves departments preparing placement letters, students searching for companies on their own, and internship coordinators keeping physical records. The absence of a centralized digital system leads to delays, disorganization, and poor tracking of student internship performance.

### 2.2.2 Current System

* Manual Processes: Students receive internship details from notice boards, word of mouth, or department emails.
* Paper-Based Applications: Application letters and forms are submitted physically to companies.
* No Central Tracking: Universities do not have a real-time system to monitor application statuses or internship progress.
* Communication Delays: Students often receive updates too late to act effectively.

### 2.2.3 Players of the Existing System

* Students: Seek internship opportunities, apply, and submit reports after completion.
* University Coordinators: Assign, monitor, and evaluate student internships.
* Companies: Accept or reject internship applications and provide practical experience.
* Department Heads: Review placements and ensure alignment with academic goals.

Business Rules

1. Only registered users can access the system.
2. Students must use valid university info during registration.
3. Companies must be approved before posting internships.
4. Students can apply only to relevant and active internships.
5. One student can’t apply to the same internship twice.
6. Coordinators and companies review and approve applications.
7. Students must submit weekly progress and a final report.
8. System roles determine dashboard access (Student, Company, and Coordinator).
9. Admins have full control over all operations.
10. Applications and reports cannot be edited after submission.

### 2.2.4 Proposed System

The proposed system is a centralized, web-based platform that streamlines internship placements. It enables students to apply online, companies to post opportunities and manage applications, and universities to track all placement activities in real time. It includes secure login for each user role, an admin dashboard, notification features, and internship history tracking.

### 2.2.5 Forms and Reports Used

* Internship Application Form
* Company Acceptance Letter
* Internship Evaluation Form
* Student Weekly Reports
* Final Internship Completion Report
* Internship Placement Summary Report

## 2.3 Requirement Gathering

### 2.3.1 Requirement Gathering Techniques

To collect comprehensive and accurate requirements for the system, the following techniques were used:

* Surveys: Distributed to students from various departments to understand common challenges.
* Interviews: Conducted with internship coordinators and company HR personnel.
* Observation: Monitored current manual placement processes in selected universities.
* Document Review: Examined existing forms, templates, and placement policies.

## 2.4 Method of Communication

### 2.4.1 Communication Techniques

During the requirement analysis phase, the following communication techniques were employed:

* Face-to-Face Meetings: Held with coordinators and staff for deeper insight into the process.
* Email Communication: Used to reach out to companies and follow up with participants.
* Phone Calls and Messaging: For quick clarification and reminders.
* Focus Group Discussions: Conducted with students to identify shared concerns and expectations.

## 2.5 Requirement Definition

### 2.5.1 Functional Requirements

1. **User Registration and Login**
   * Users (students, companies, coordinators) can create accounts and log in securely.
   * The system should verify email and role before granting access.
2. **Internship Listings**
   * Companies can post internships with title, description, deadline, and required field.
   * Students can browse and filter internships based on their field of study.
3. **Application Management**
   * Students can apply for available internships and upload required documents.
   * The system should prevent duplicate applications.
4. **Approval Workflow**
   * Coordinators and companies can review, approve, or reject applications.
   * Notification is sent to the student upon decision.
5. **Progress Tracking**
   * Students must submit weekly internship progress reports.
   * Coordinators can view reports and send feedback or reminders.
6. **Final Report Submission**
   * Students submit a final internship report after completion.
   * Coordinators can review and evaluate the report.
7. **Role-Based Dashboards**
   * Students, companies, and coordinators access different interfaces based on their roles.
   * Admin dashboard provides full access to manage users and data.
8. **Notification System**
   * The system sends alerts for deadlines, approvals, and missing reports.
   * Email or in-app notifications must be supported.
9. **Security and Data Protection**
   * User data must be securely stored with encryption.
   * Only authorized roles can access specific data.

### 2.5.2 Non-Functional Requirements

1. **Performance**

The system should respond to user actions within two seconds to ensure a smooth and stress-free experience.

Example: When a student applies for an internship or checks the status of their application, the page should load quickly without delays.

1. **Scalability**

As more students and companies use the platform, it must continue to perform well—even with 10,000+ users active at the same time.

Example: During peak times like the start of a new semester, the system should support thousands of students browsing internships or submitting documents without slowing down.

1. **Security**

The platform must protect student and company data using SSL encryption, secure logins, and role-based access control.

Example: A student should only be able to view and edit their own profile and applications, while university staff can review and approve internship placements.

1. **Availability**

The system should have at least 99% uptime during active periods to make sure students, staff, and employers can access it when they need to.

Example: A company should be able to post a new internship opportunity even during evenings or weekends without facing downtime.

1. **Usability**

The interface should be simple and user-friendly so students, many of whom are first-time users, can navigate it easily without needing special training.

Example: A dashboard with clear steps like "Find Internships," "Apply," and "Track Application" helps students know exactly what to do next.

1. **Maintainability**

The system should follow clean coding practices, like the Model-View-Controller (MVC) architecture, to make future updates and bug fixes easier.

Example: If a new feature needs to be added, like interview scheduling, it can be done quickly without disrupting other parts of the system.

1. **Portability**

The platform must work well across all modern browsers and mobile devices, ensuring accessibility from anywhere.

Example: A student should be able to upload their CV or check internship updates from their phone while commuting.

1. **Backup**

Daily automated backups should be in place to protect all data and allow recovery in case of unexpected issues.

Example: If there’s a system failure, no applications or records should be lost, and everything should be restorable from the latest backup.

### 2.5.3 Collaboration Modeling

Collaboration modeling defines how different actors and system components interact with each other to complete tasks.

Collaboration Diagram Key Elements:

* Actors: Students, Coordinators, Companies
* Objects: System modules (Authentication, Listings, Notifications, Reporting, Dashboard)

Message Flow:

* Student → System: Request internship list
* Company → System: Post internship
* System → Coordinator: Notify of new application
* Coordinator → System: Approve application
* System → Student: Send approval status
* Student → System: Submit report
* System → Coordinator: Store and present report

## 2.6 System Modeling

## 2.6.1 Use Case Diagram

The Use Case Diagram illustrates the interactions between the system and its primary actors: Students, Company Representatives, and University Coordinators. It highlights the main functions each user can perform.

### 2.6.1.1 Essential Use Case Modeling

Main Actors:

* Student
* University Coordinator
* Company Representative

Essential Use Cases:

* Register account
* Login
* View available internships
* Apply for internship
* Post internship (Company)
* Review and approve student application (Coordinator/Company)
* Track internship progress
* Submit final internship report

### 2.6.1.2 Actor Description

|  |  |
| --- | --- |
| Actor | Role |
| Student | Registers, browses internships, applies and submits reports |
| Company Representative | Posts internships, reviews applicants and provides feedback |
| University Coordinator | Manages student applications, verifies placements and evaluates reports |

Table 2.1Actor Description

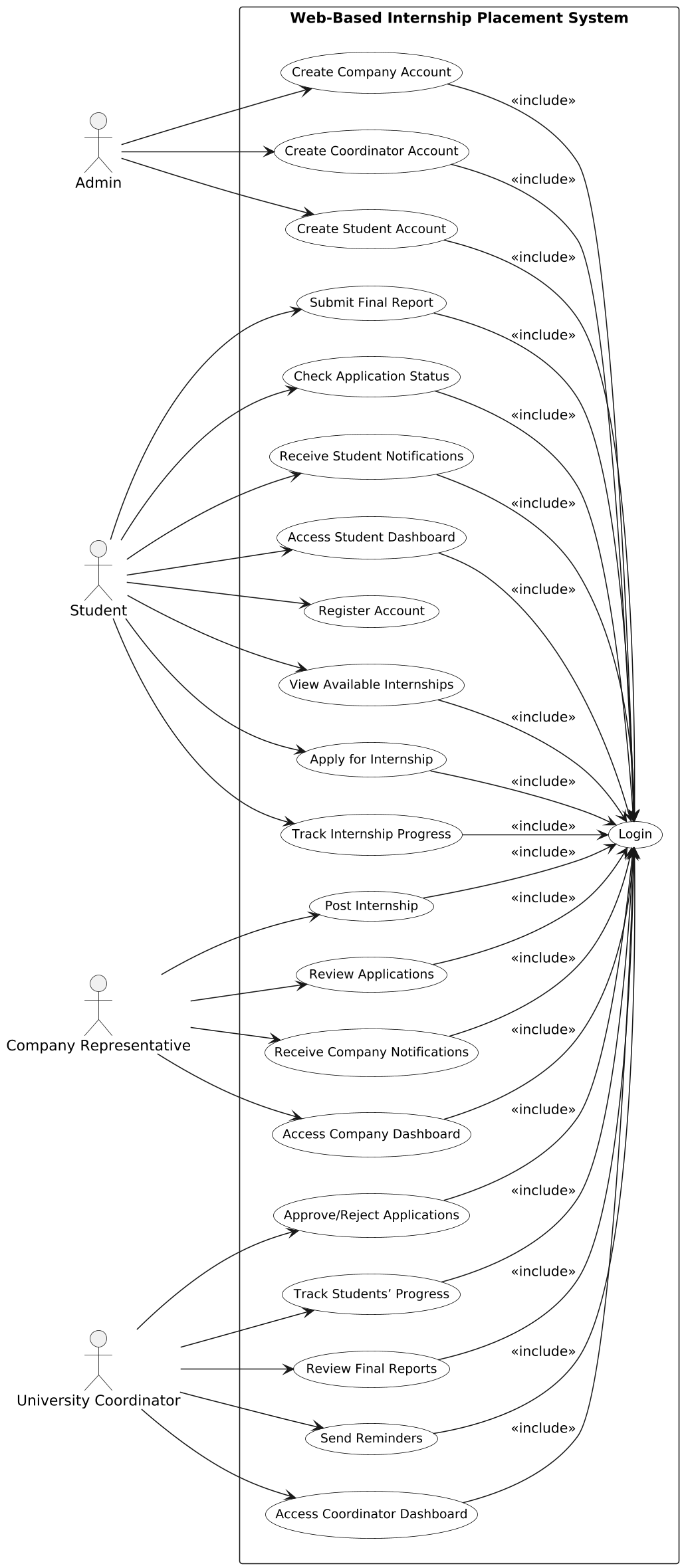


Figure 1 Use case for WBIPS

### 2.6.1.3 Essential Use Case Description

|  |  |
| --- | --- |
| **Use case Name** | Student Registration |
| **Use case ID** | UC-01 |
| **Participating Actors** | Student |
| **Pre-conditions** | The student is not registered in the system |
| **Normal Flow** | 1. . Student accesses registration page 2. Enters name, email, password, etc. 3. Submits form 4. System validates and stores data   Use case ends |
| **Alternative flow of events** | * If email already exists, show error and block registration |
| **Include** | Form validation |
| **Extends** | None |
| **Post conditions** | Student account is created and ready for login |

Table 2.2 Use case description for registration

|  |  |
| --- | --- |
| **Use case Name** | User Login |
| **Use case ID** | UC-02 |
| **Participating Actors** | Student, Company, Coordinator |
| **Pre-conditions** | User account must be registered |
| **Normal Flow** | 1. User opens login page 2. Enters email and password 3. System verifies credentials 4. Redirects to respective dashboard   Use case ends |
| **Alternative flow of events** | * If credentials are incorrect, display error |
| **Include** | Session initialization |
| **Extends** | Redirect to role-based dashboard |
| **Post conditions** | User accesses their dashboard |

Table 2.3 Use case description for login

|  |  |
| --- | --- |
| **Use case Name** | View Available Internships |
| **Use case ID** | UC-03 |
| **Participating Actors** | Student |
| **Pre-conditions** | Student must be logged into the system |
| **Normal Flow** | 1. Student logs in 2. Navigates to internship listings 3. System retrieves and displays active internships 4. Student applies filters (field, location, duration) if needed 5. Views details of selected internships   Use case ends |
| **Alternative flow of events** | * If no internships are available, system displays “No listings found” message |
| **Include** | Filter/search function |
| **Extends** | Apply for Internship (if student chooses to apply from the listing page) |
| **Post conditions** | Student is able to view relevant and active internship opportunities |

Table 2.4 Use case description for View Available Internships

|  |  |
| --- | --- |
| **Use case Name** | Apply for Internship |
| **Use case ID** | UC-04 |
| **Participating Actors** | Student |
| **Pre-conditions** | Student must be logged into the system |
| **Normal Flow** | 1. Student selects an internship 2. Fills application 3. Uploads documents 4. Submits form   Use case ends |
| **Alternative flow of events** | * If required documents are missing, show validation error |
| **Include** | Upload CV and documents |
| **Extends** | Notify company/coordinator |
| **Post conditions** | Application is stored and status is pending |

Table 2.5 Use case description for Apply for Internship

|  |  |
| --- | --- |
| **Use case Name** | Post Internship Opportunity |
| **Use case ID** | UC-05 |
| **Participating Actors** | Company Representative |
| **Pre-conditions** | Company must be logged in |
| **Normal Flow** | 1. Company accesses posting form 2. Enters internship details 3. Submits posting 4. System saves and publishes   Use case ends |
| **Alternative flow of events** | * If fields are missing, system returns form with error messages |
| **Include** | Data validation |
| **Extends** | Notify system |
| **Post conditions** | Internship opportunity is visible to students |

Table 2.6 Use case description for Post Internship Opportunity

|  |  |
| --- | --- |
| **Use case Name** | Approve Student Application |
| **Use case ID** | UC-06 |
| **Participating Actors** | University Coordinator/ Company |
| **Pre-conditions** | * Coordinator/ Company must be logged in * Application must be submitted |
| **Normal Flow** | 1. Coordinator/Company views pending applications 2. Reviews details 3. Approves or rejects   Use case ends |
| **Alternative flow of events** | * Coordinator/Company skips action → application stays pending |
| **Include** | Status update |
| **Extends** | Notify Student |
| **Post conditions** | Application status is updated |

Table 2.7 Use case description for Approve Student Application

|  |  |
| --- | --- |
| **Use case Name** | Submit Final Internship Report |
| **Use case ID** | UC-07 |
| **Participating Actors** | Student |
| **Pre-conditions** | Internship must be completed |
| **Normal Flow** | 1. Student accesses report submission 2. Uploads document 3. Submits   Use case ends |
| **Alternative flow of events** | * File format unsupported → show error |
| **Include** | File upload |
| **Extends** | Notify coordinator |
| **Post conditions** | Final report is stored for coordinator review |

Table 2.8 Use case description for Submit Final Internship Report

|  |  |
| --- | --- |
| **Use case Name** | Track Internship Progress |
| **Use case ID** | UC-08 |
| **Participating Actors** | University Coordinator |
| **Pre-conditions** | * Students must be actively placed in internships * Coordinator must be logged in |
| **Normal Flow** | 1. Coordinator logs into dashboard 2. Views list of currently placed students 3. Accesses progress updates (e.g., weekly reports) 4. Sends reminders or feedback if needed   Use case ends |
| **Alternative flow of events** | * If no reports are submitted, system flags student as “non-compliant” and notifies the coordinator |
| **Include** | * Weekly report viewing * Student communication module |
| **Extends** | Notify student to submit missing reports |
| **Post conditions** | Coordinator has up-to-date insight into internship activities and can follow up if necessary |

Table 2.9 Use case description for Track Internship Progress

|  |  |
| --- | --- |
| **Use case Name** | Notification System |
| **Use case ID** | UC-09 |
| **Participating Actors** | Student, Company Representative, Coordinator |
| **Pre-conditions** | User is logged in |
| **Normal Flow** | 1. System triggers notification on events (e.g. application, approval, reminders) 2. Notification is sent to target user (in-app or email) 3. User views and responds   Use case ends |
| **Alternative flow of events** | * If offline, user sees the notification on next login |
| **Include** | Alert dispatch |
| **Extends** | All use cases with state changes (Apply, Approve, Report) |
| **Post conditions** | User is informed of system actions or required input |

Table 2.10 Use case description for Notification System

## 2.6.2 Sequence diagram

Sequence diagram is primarily used to model the interactions between the actors and the objects in a system and the interactions between the objects themselves. As the name implies, a sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.

Sequence Diagram for Registration

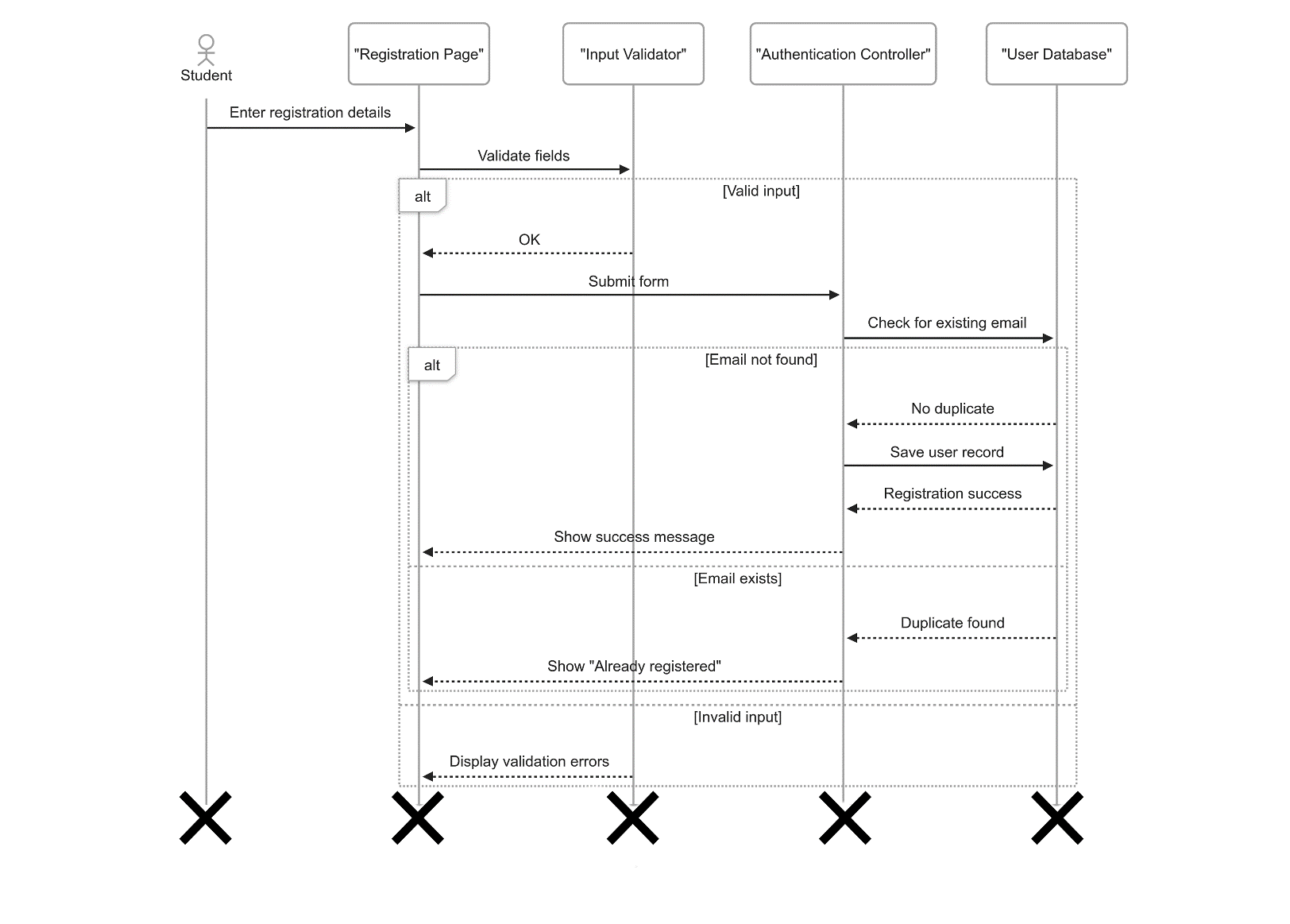


Figure 2 Sequence Diagram for Registration

Sequence Diagram for Login

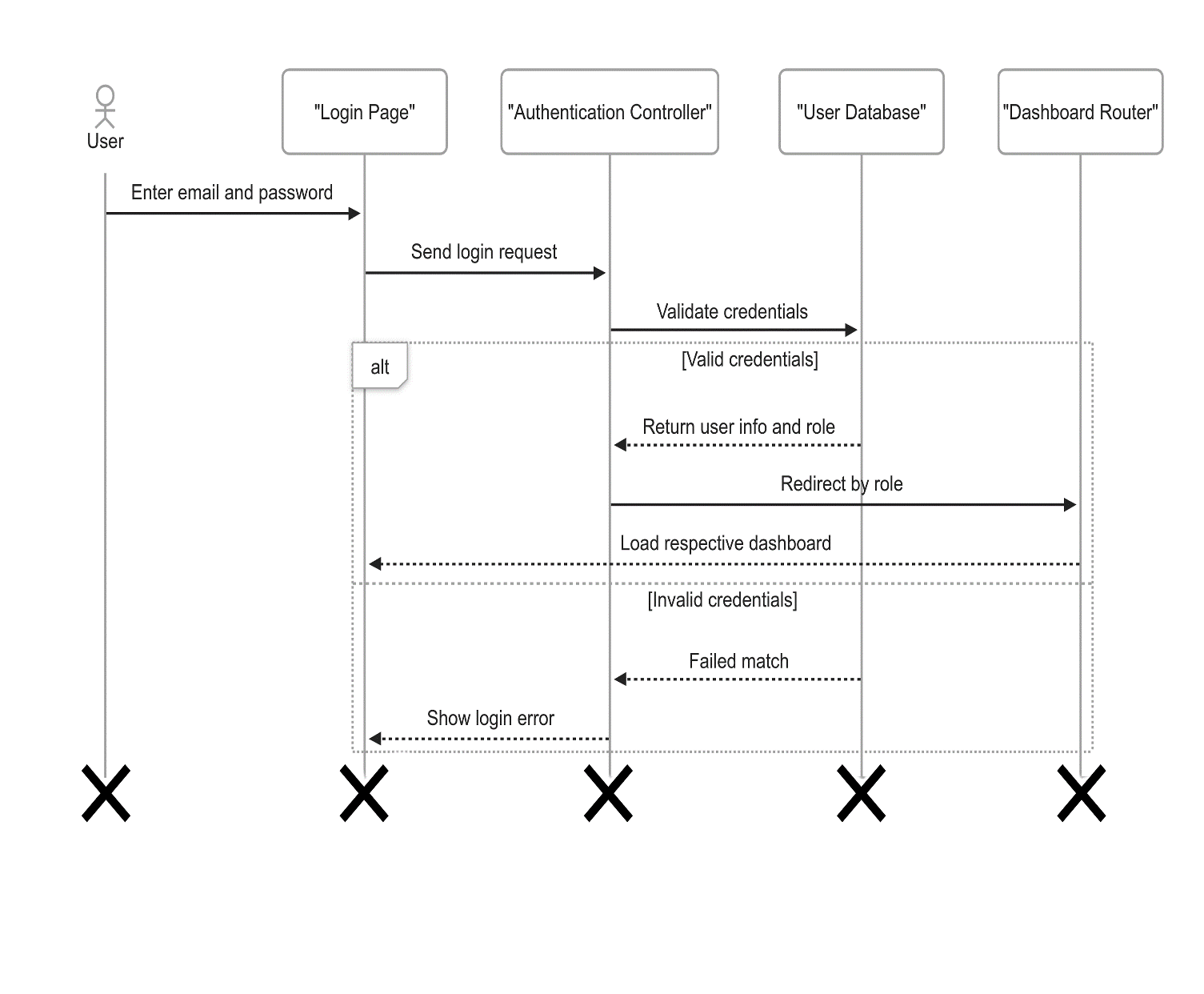


Figure 3 Sequence Diagram for Login

Sequence Diagram for View Internship Listing

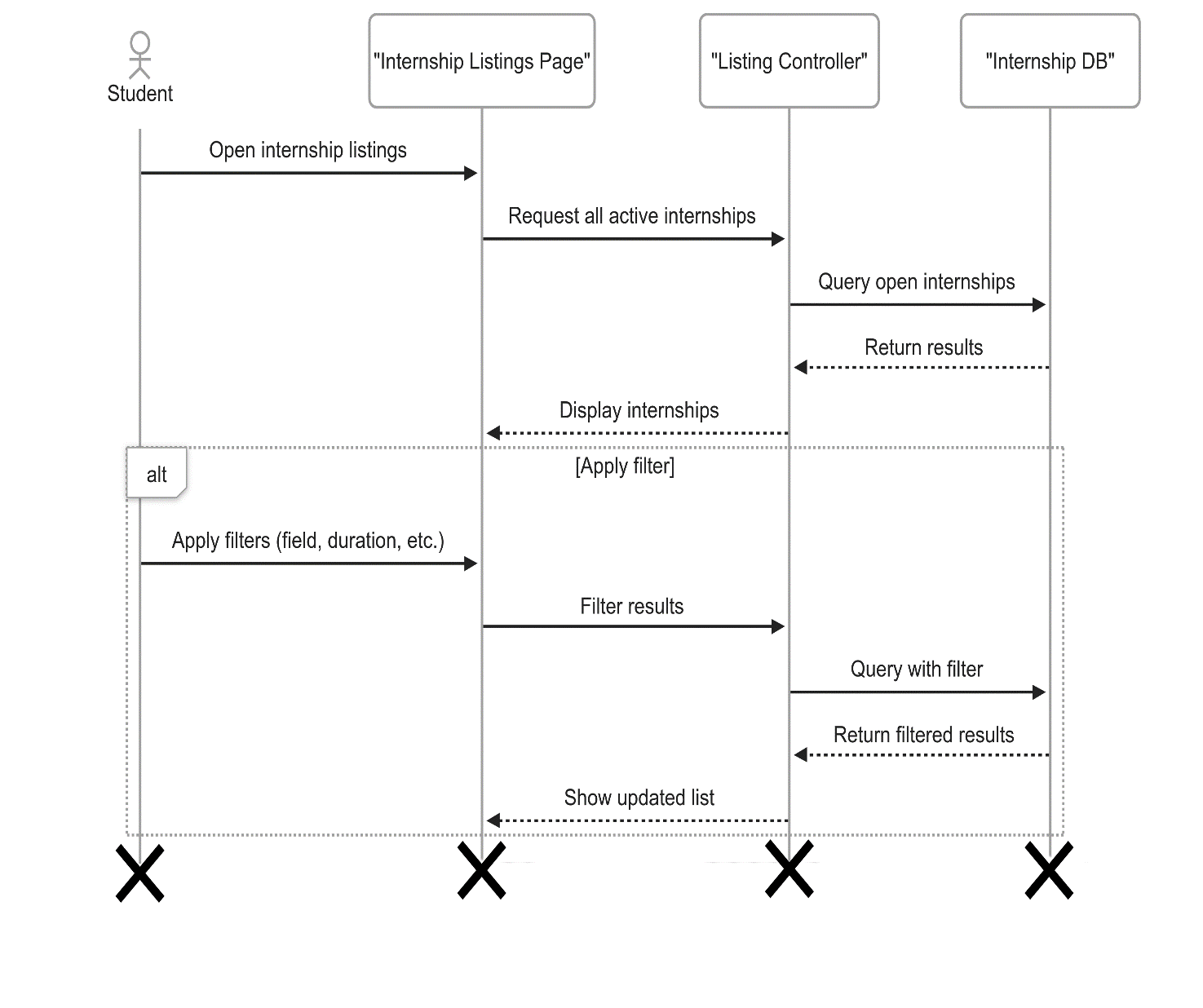


Figure 4 Sequence Diagram for View Internship Listing

Sequence Diagram for Apply for Internship

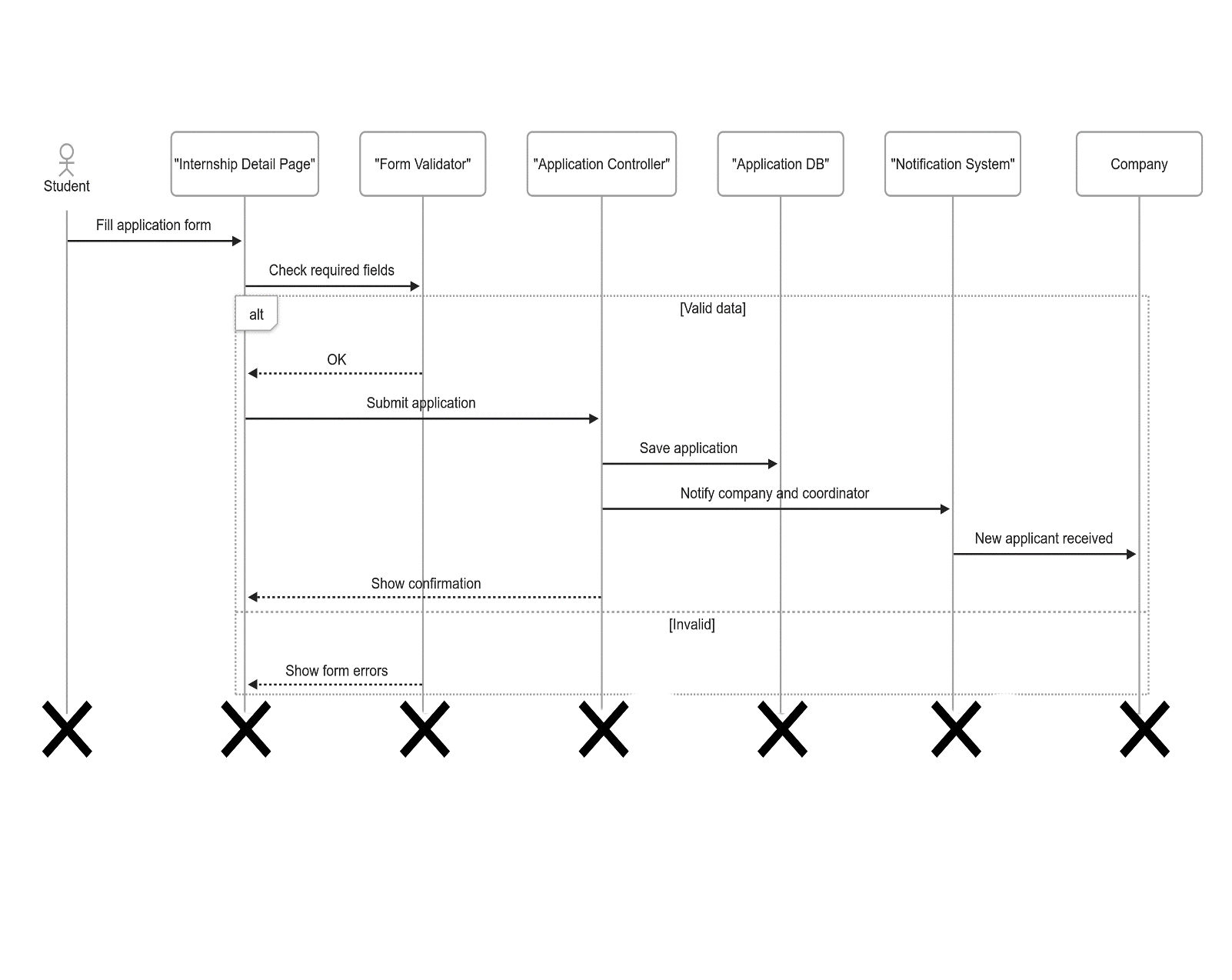


Figure 5 Sequence Diagram for Apply for Internship

Sequence Diagram for Post Internship Opportunity

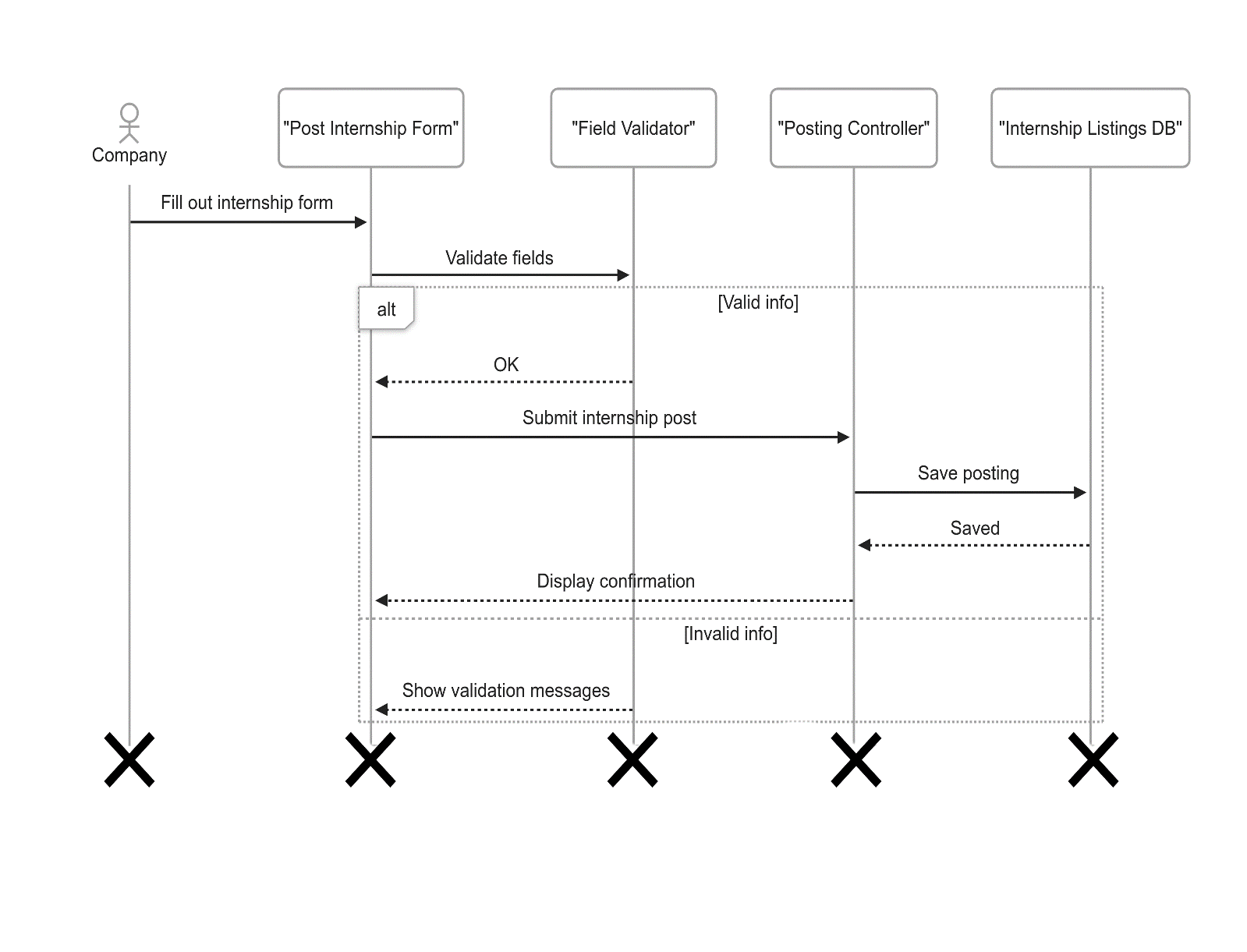


Figure 6 Sequence Diagram for Post Internship Opportunity

Sequence Diagram for Approve Student Application

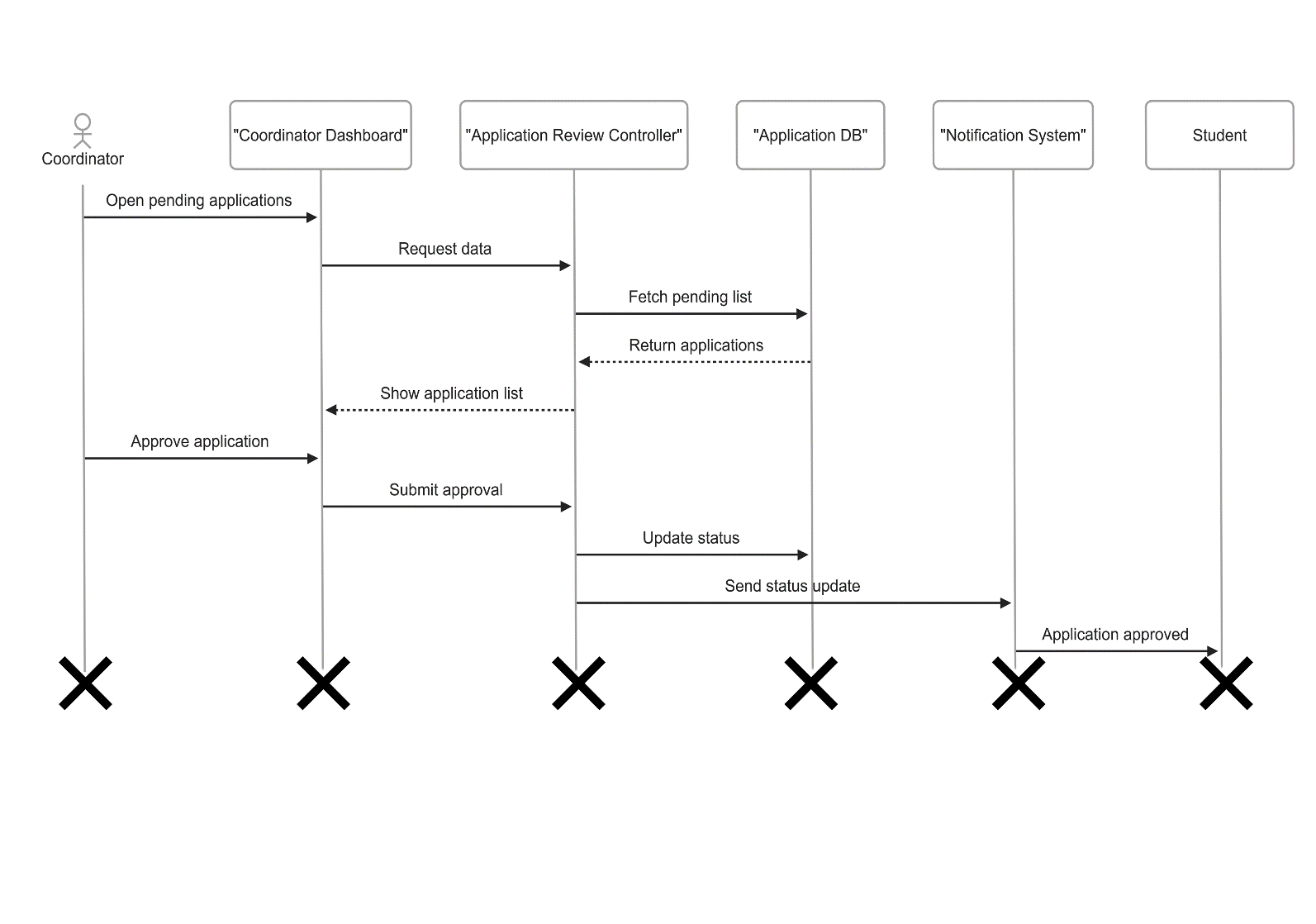


Figure 7 Sequence Diagram for Approve Student Application

Sequence Diagram for Track Internship Progress

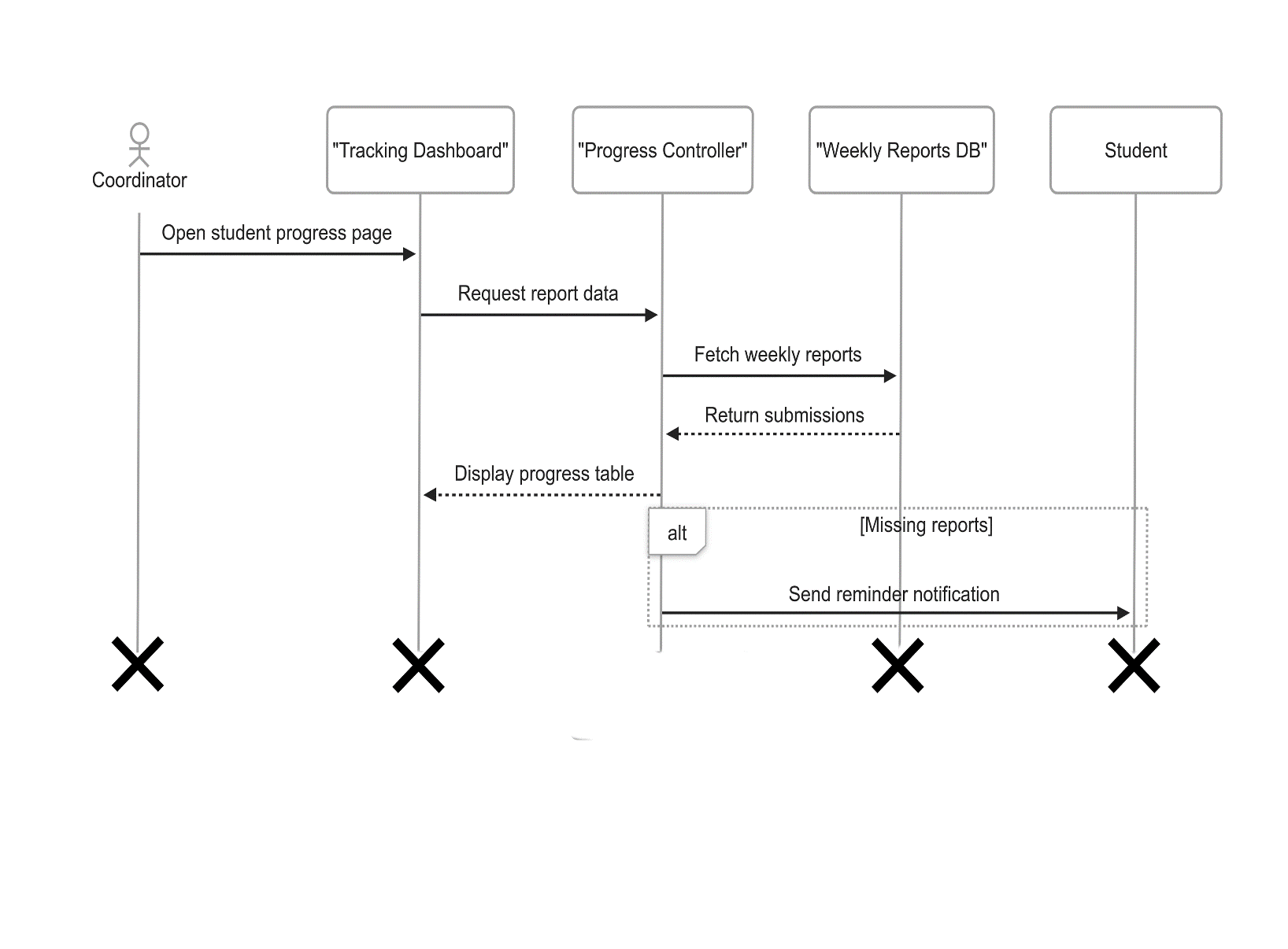


Figure 8 Sequence Diagram for Track Internship Progress

Sequence Diagram for Submit Final Report

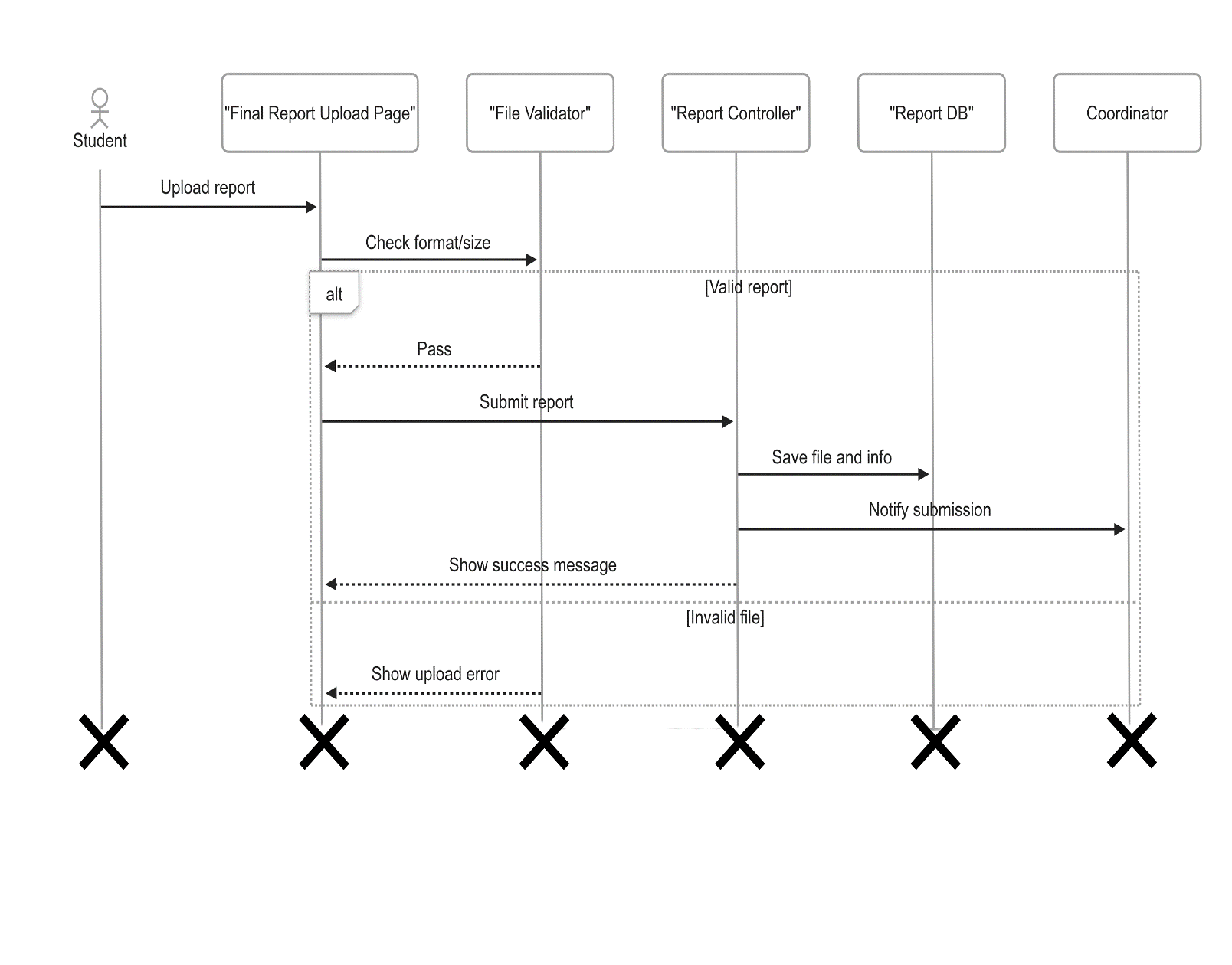


Figure 9 Sequence Diagram for Submit Final Report

## 2.6.3 State chart Diagram

A state chart diagram is a type of diagram used to describe the behavior of a system in response to internal or external events. It shows the different states of an object and the transitions between those states based on the events that occur.

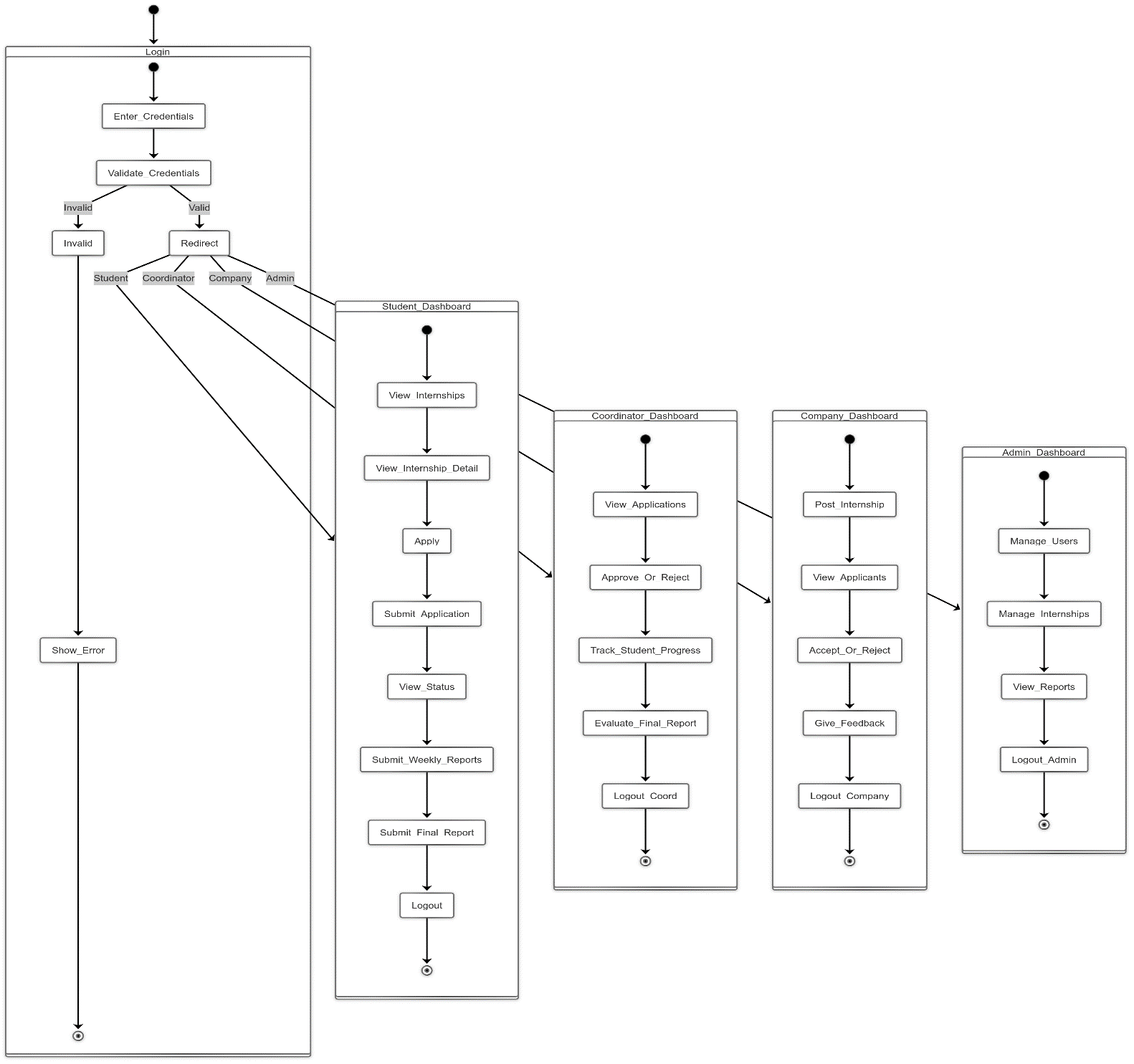


Figure 10 State chart diagram

## 2.6.4 Activity Diagram

An activity diagram is another type of diagram used to model the behavior of a system. It is a graphical representation of the steps involved in a process or workflow. Activity diagrams are used to model the flow of activities within a system, including decisions, loops, and concurrent activities.

Activity Diagram for Register



Figure 11 Activity Diagram for Registration

Activity Diagram for Login

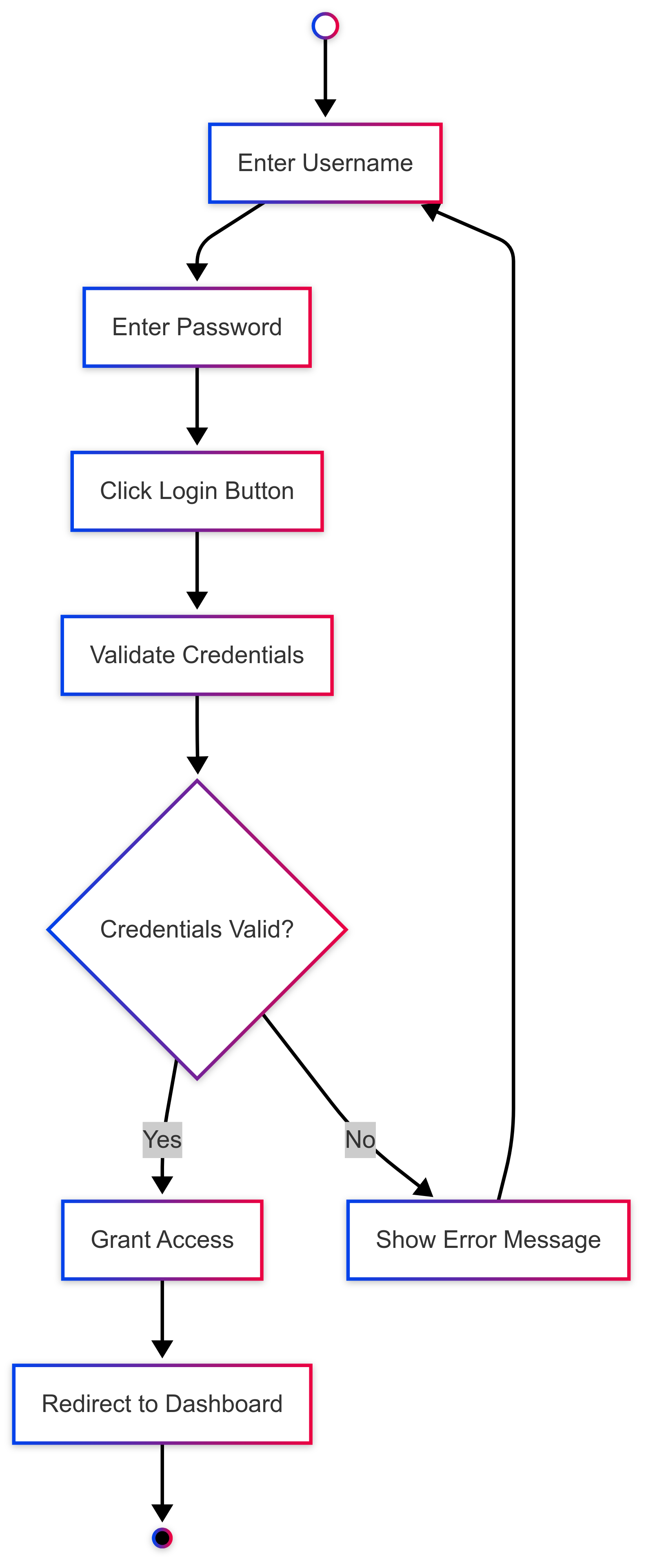


Figure 12 Activity Diagram for Login

Activity Diagram for Logout

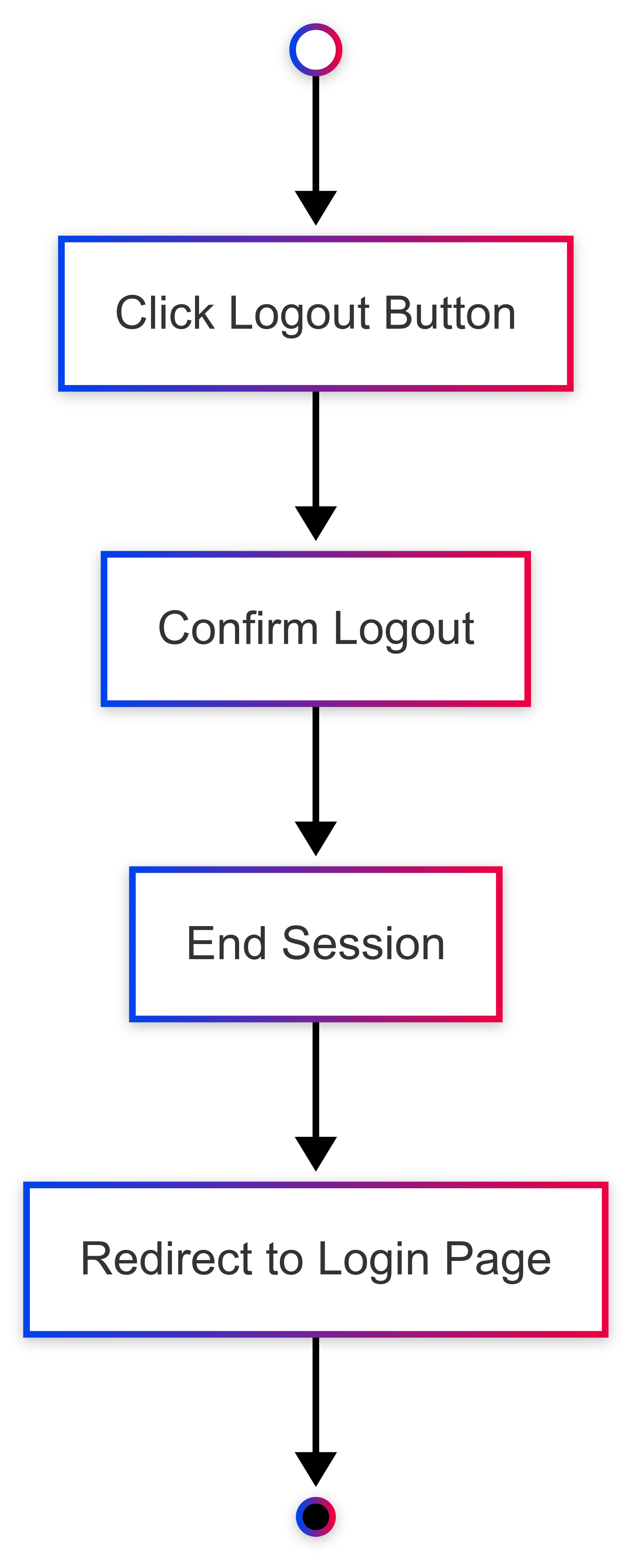


Figure 13 Activity Diagram for Logout

Activity Diagram for Apply for Internship

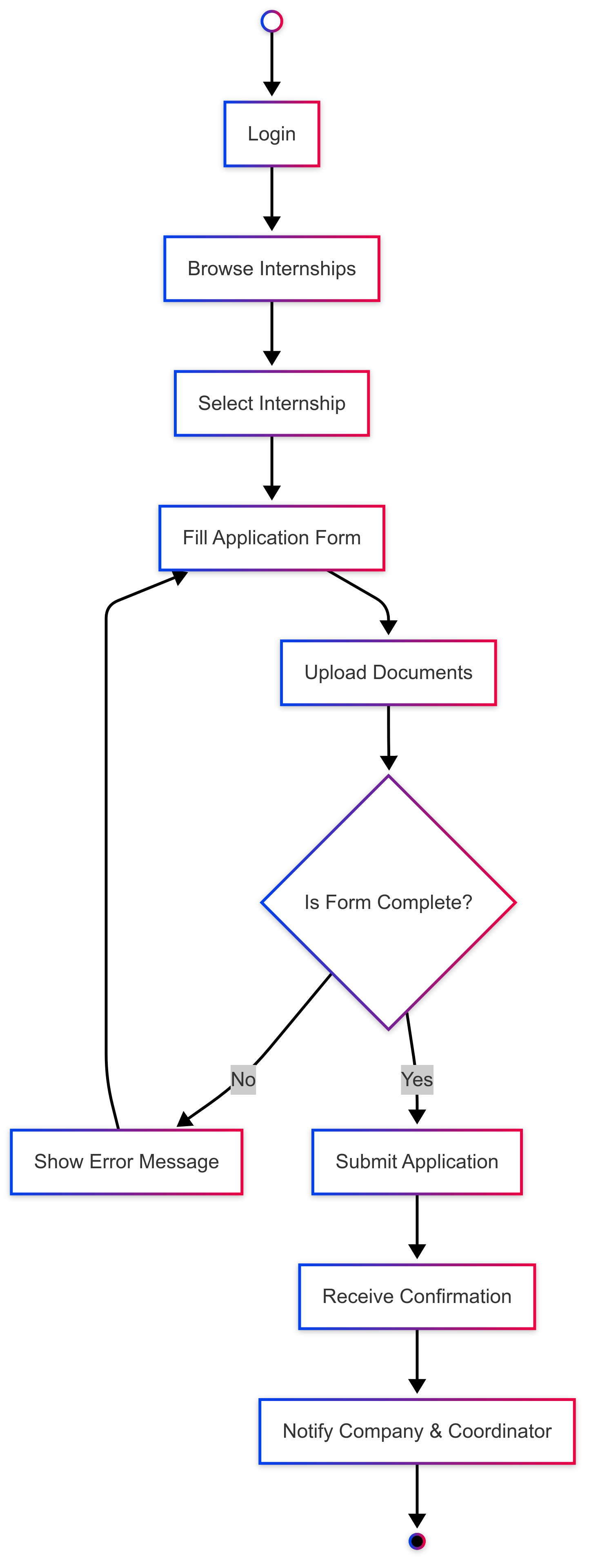


Figure 14 Activity Diagram for Apply for Internship

Activity Diagram for Post Internship Opportunity

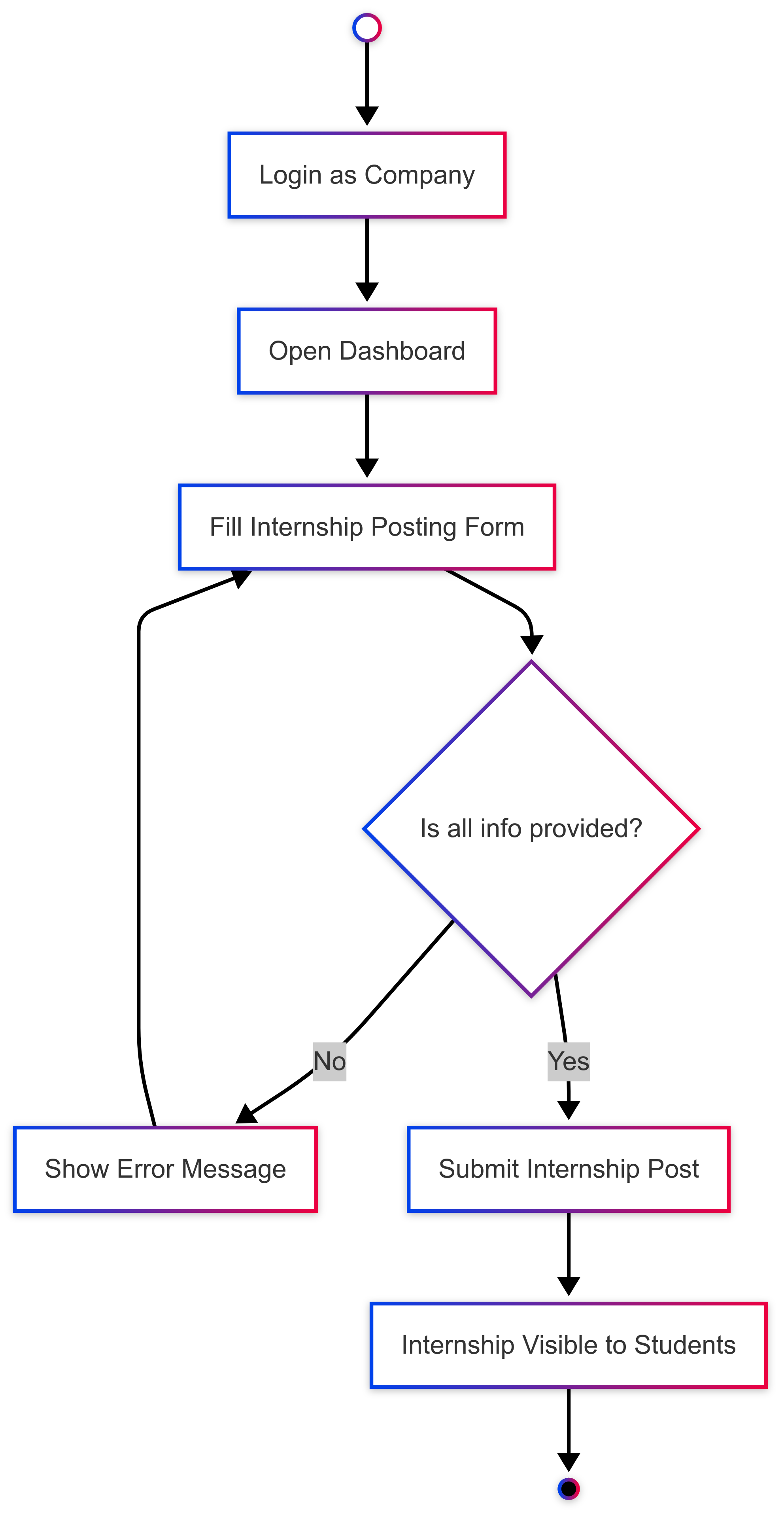


Figure 15 Activity Diagram for Post Internship Opportunity

Activity Diagram for Approve Student Application

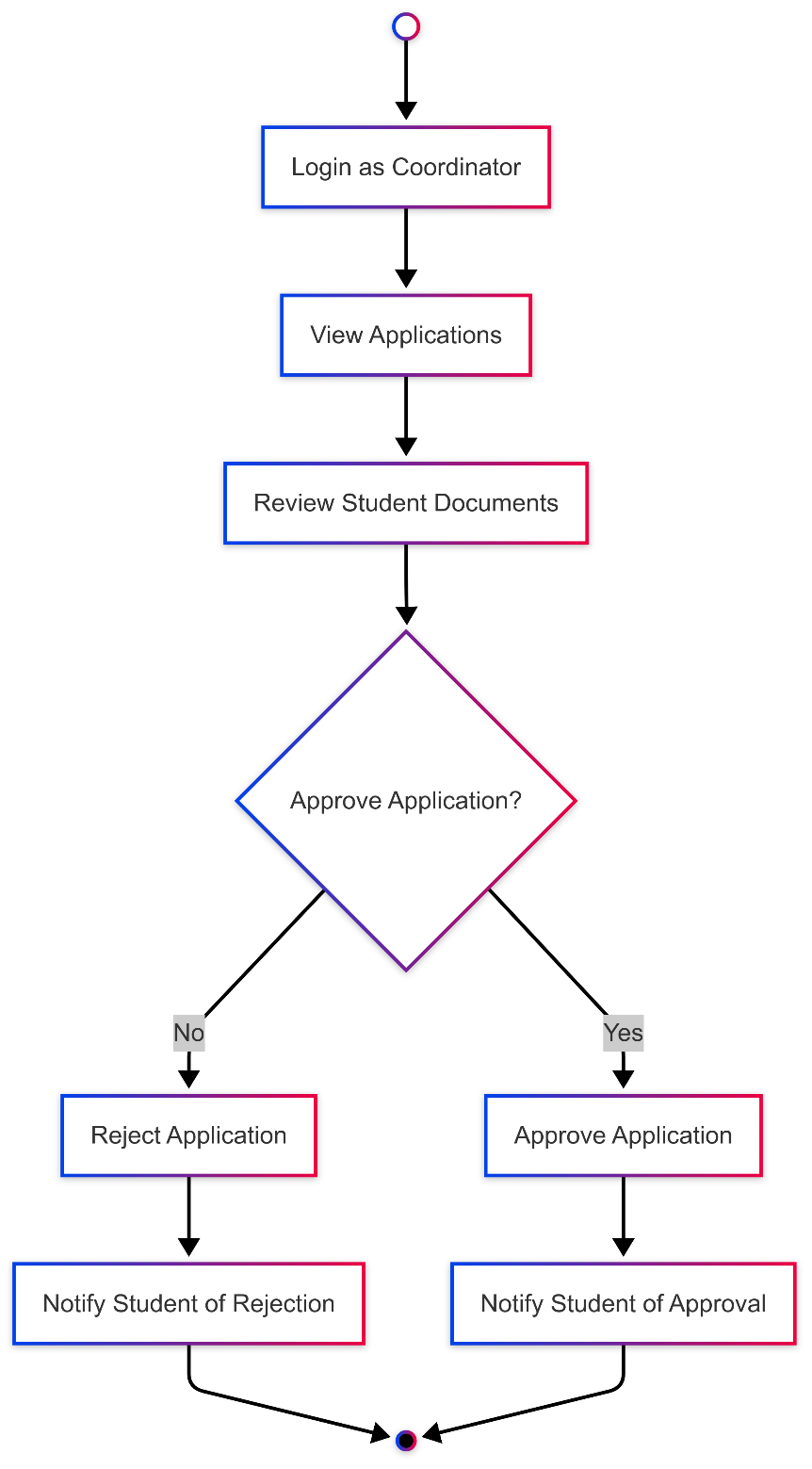


Figure 16 Activity Diagram for Approve Student Application

Activity Diagram for Admin

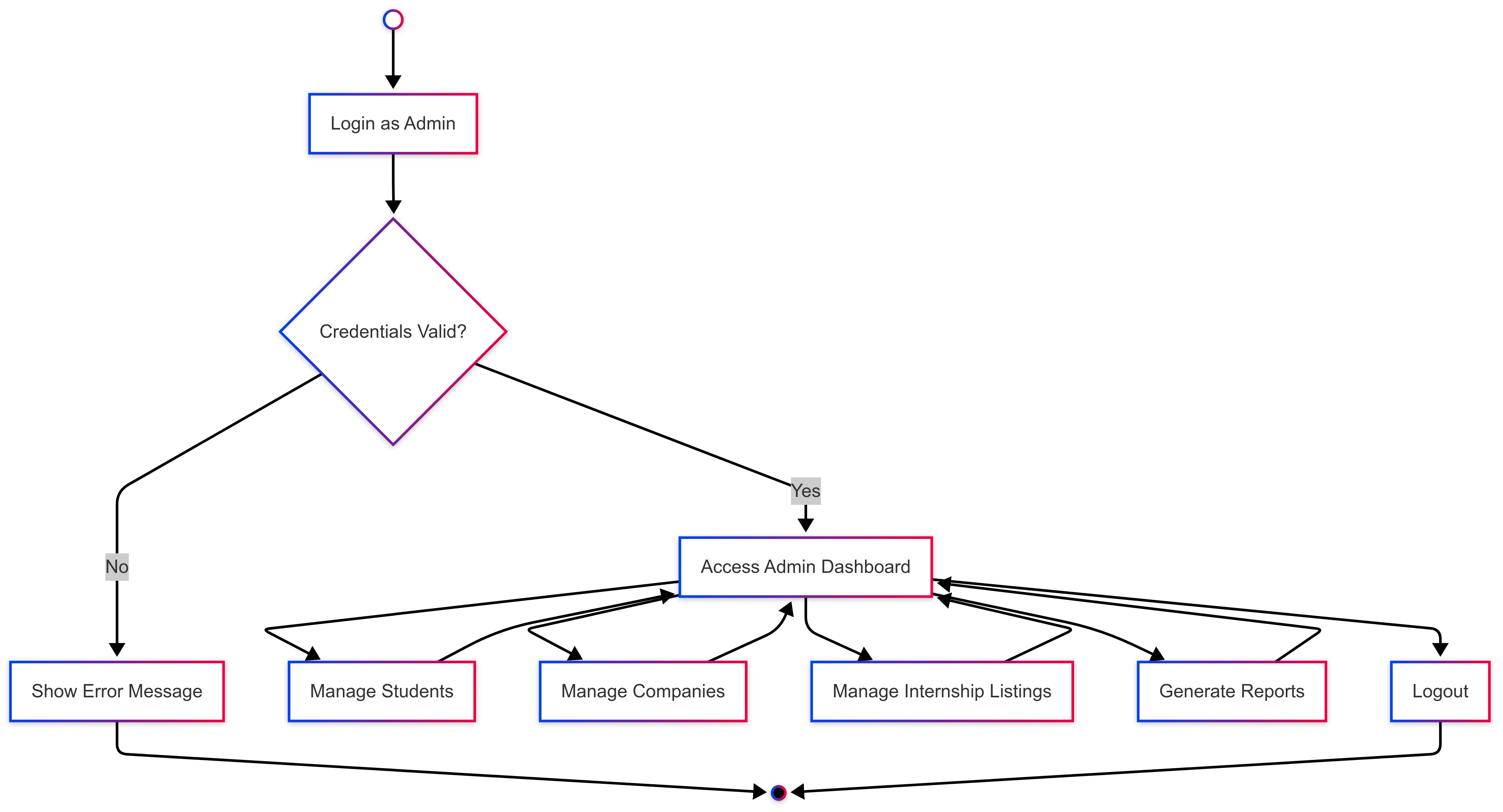


Figure 17 Activity Diagram for Admin

## 2.6.5 Class Diagram

A Class Diagram models the system structure, including data attributes and methods.

Key Classes:

* User (Attributes: userID, name, email, role | Methods: login(), logout())
* Student (Attributes: studentID, department | Methods: apply(), submitReport())
* Company (Attributes: companyID, name, field | Methods: postInternship(), reviewApplicants())
* Coordinator (Attributes: coordinatorID, faculty | Methods: approveApplication(), viewReports())
* Internship (Attributes: internshipID, title, company, status | Methods: updateStatus())
* Application (Attributes: appID, studentID, internshipID, status | Methods: submit(), cancel())

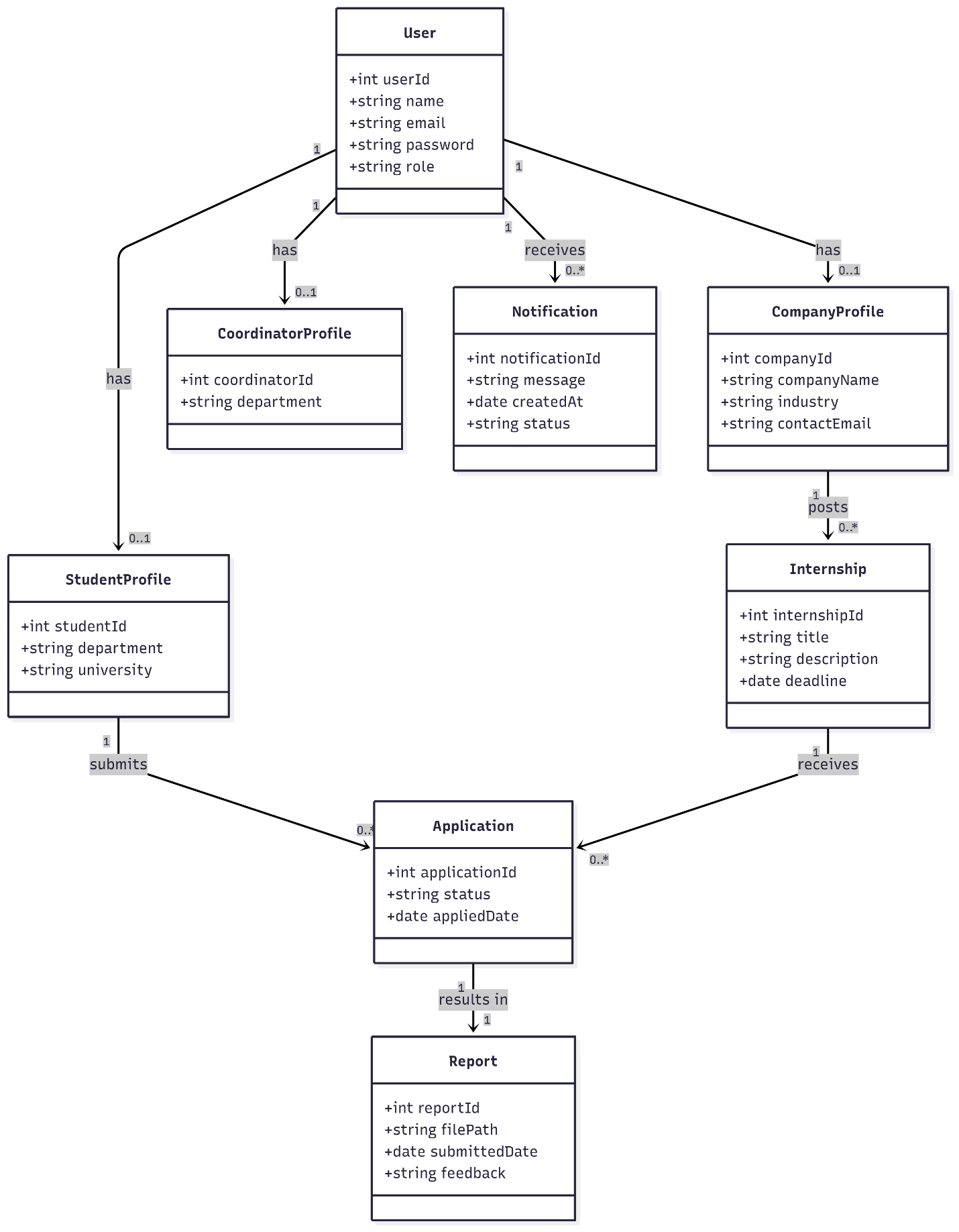


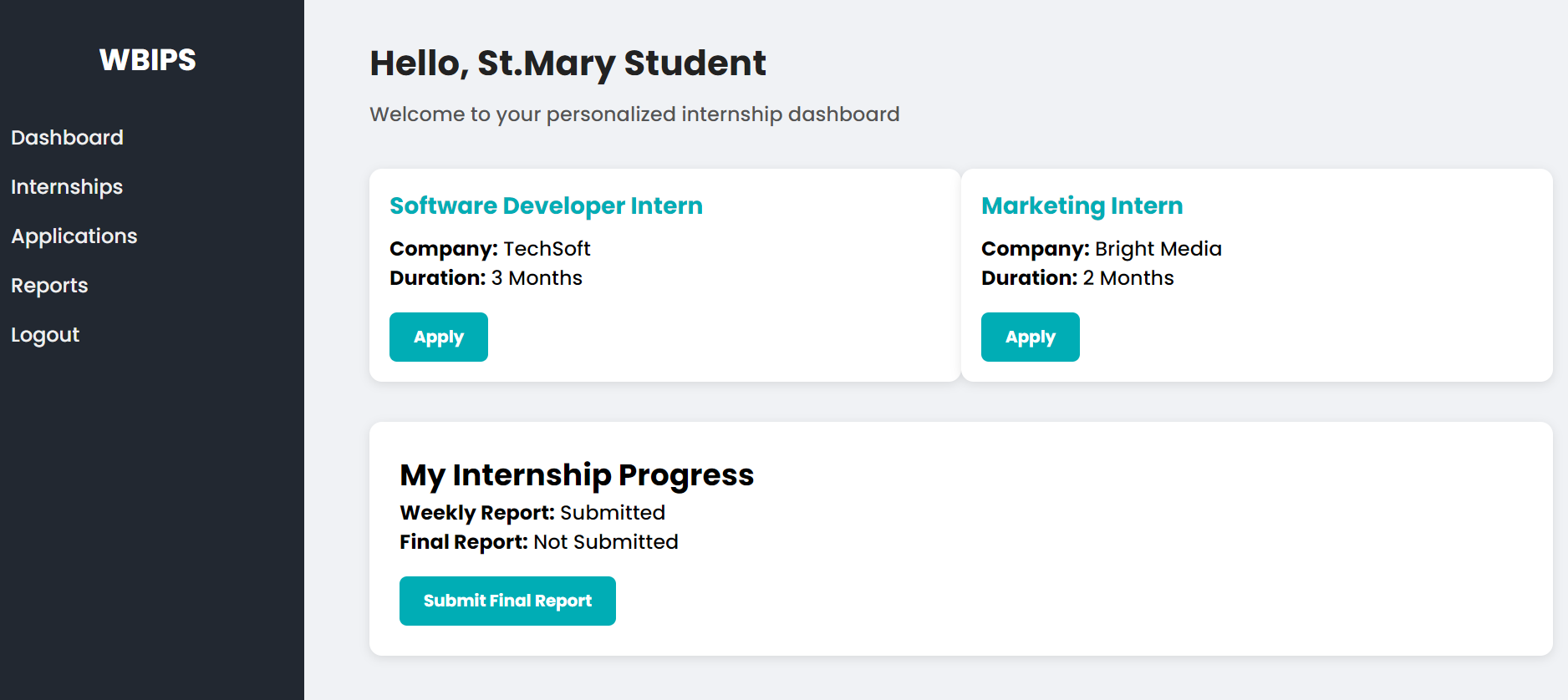
Figure 18 Class Diagram

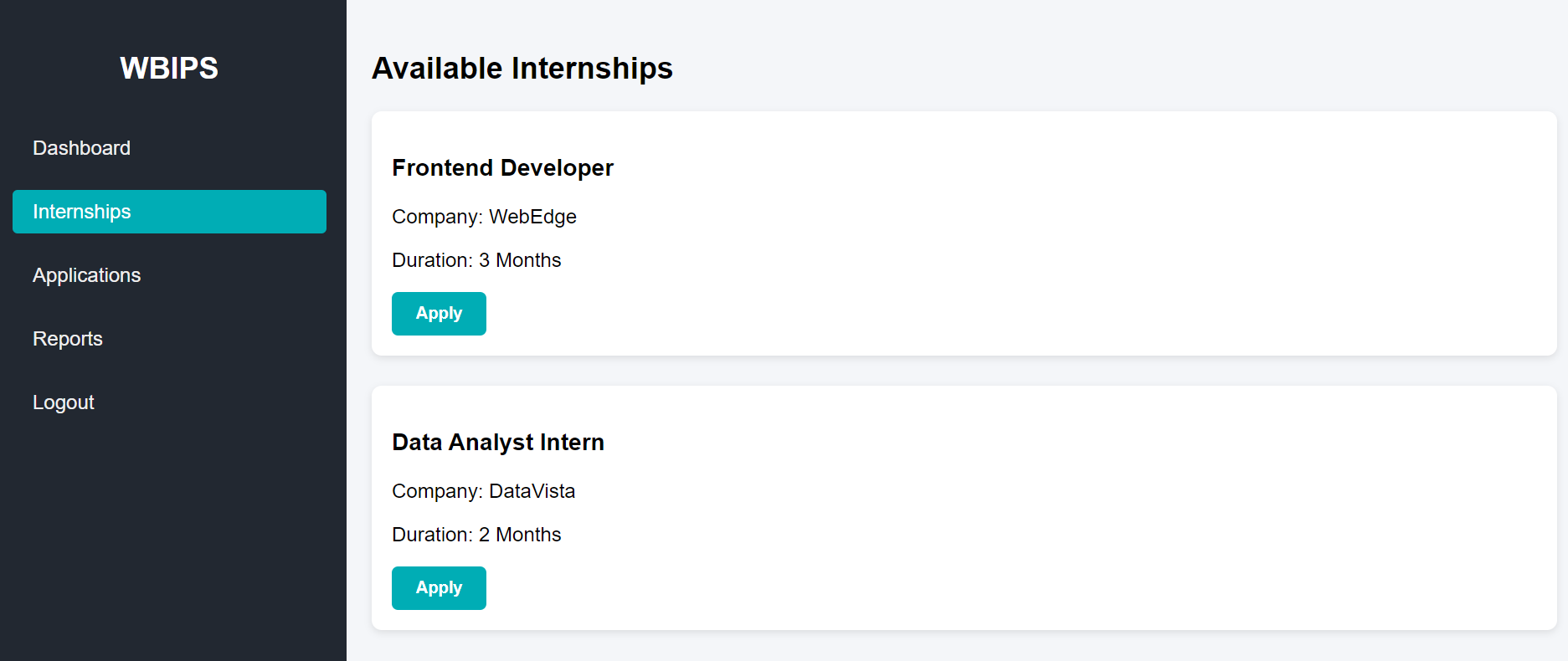
## 2.6.6 Multiplicity of Relationships between Classes

|  |  |  |  |
| --- | --- | --- | --- |
| Class A | Class B | Relationship Description | Multiplicity |
| Student | Application | A Student can submit many applications | 1 user 🡪0..\* applications |
| Company | Internship | A company can post multiple internships | 1 company 🡪0..\* internships |
| Internship | Application | Each internship can receive many applications | 1 internship 🡪0..\* applications |
| Application | Internship | Each application is for exactly one internship | 0..\* applications 🡪1 internship |
| Company | Application | A company can view applications submitted for its internships | 1 company 🡪0..\* applications |
| Student | Report | A student can submit multiple reports | 1 user 🡪0..\* reports |
| Coordinator | Application | A coordinator can review multiple applications | 1 coordinator 🡪0..\* applications |
| Coordinator | Report | A coordinator can evaluate many reports | 1 coordinator 🡪0..\* reports |
| Internship | Company | Each internship is posted by exactly one company | 0..\* internships 🡪1 company |
| User | Notification | Each user can receive many notifications | 1 user 🡪0..\* notifications |
| Application | Student | Each application is submitted by exactly one student | 0..\* applications 🡪1 student |

Table 2.11 Multiplicity of Relationships between Classes

## 2.6.7 User interface prototyping





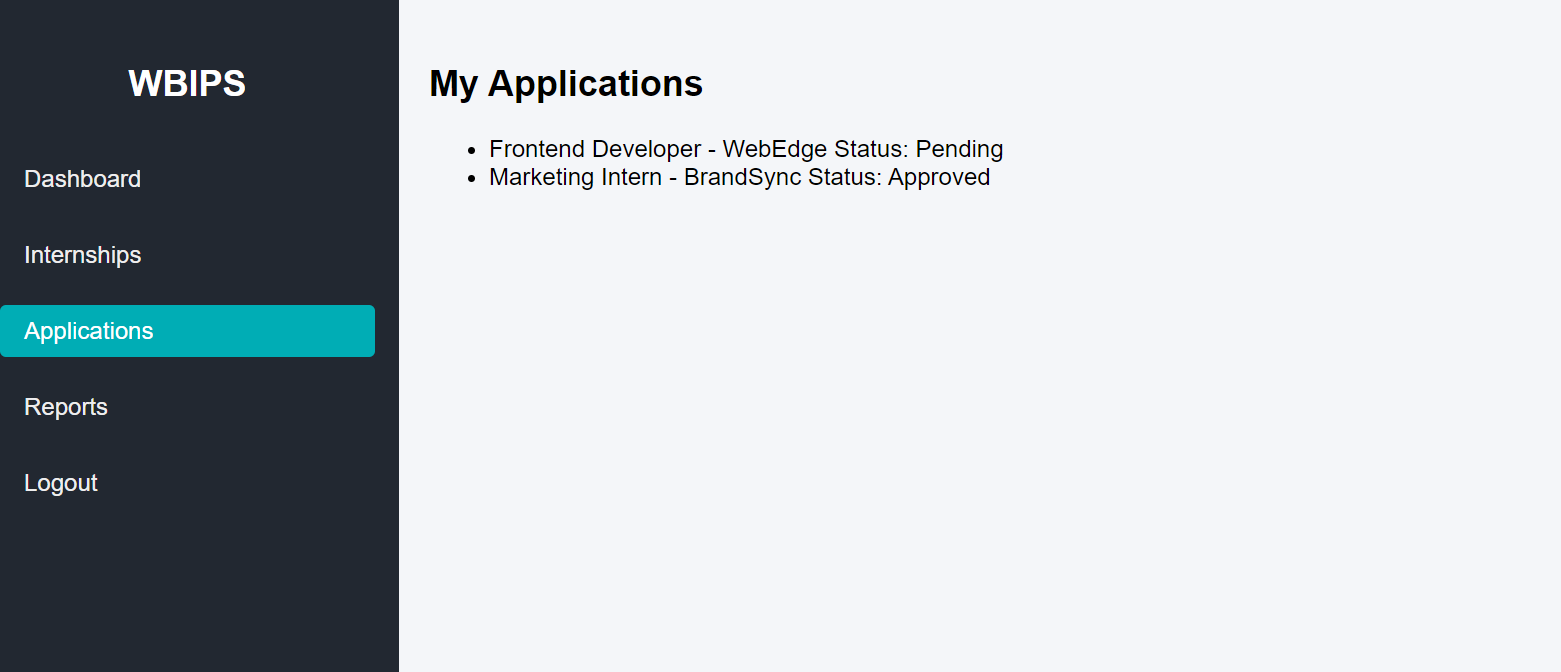


Figure 19 User interface prototyping

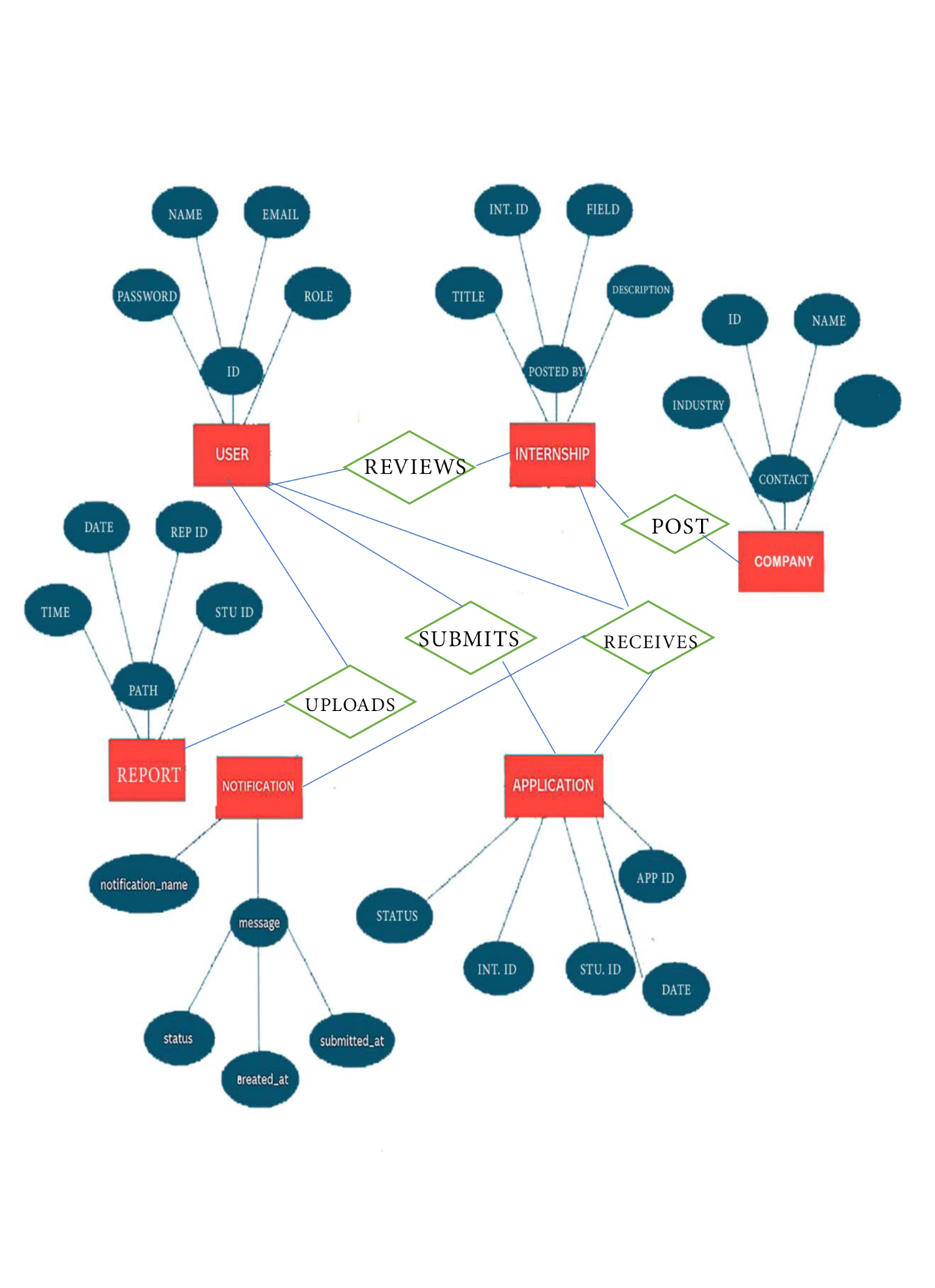


Figure 20 ER Diagram

# CHAPTER THREE

# System Design

## 3.1 Introduction

System design is the blueprint stage of software development where abstract requirements are transformed into structured architecture and technical specifications. It defines the system’s components, their interactions, data flows, and deployment strategies. This chapter outlines the design of a centralized internship placement platform tailored for Ethiopian universities prioritizing efficiency, scalability, and security across all user roles: students, coordinators, and companies.

## 3.2 Purpose of the system

The purpose of the proposed system is:

**To provide a user-friendly, efficient, and secure digital platform that connects students, university coordinators, and companies for seamless internship placement, monitoring, and reporting.**

It eliminates manual inefficiencies, ensures real-time communication, and enables coordinated management of the entire internship lifecycle.

## 3.3 System Design

The system is designed to achieve the following goals:

✅ Usability: Clean and intuitive interfaces for students, companies, and coordinators.

🔒 Security: Role-based access control, secure authentication, and encrypted data storage.

⚡ Performance: Optimized for fast data retrieval and low-latency communication.

🧱 Scalability: Able to support multiple universities and thousands of users concurrently.

🔄 Maintainability: Modular codebase and clear separation of concerns.

💾 Data Integrity: Ensures consistency of internship data, applications, and reports.

## 3.4 Current Software Architecture

In most Ethiopian universities today, the internship placement process is primarily manual or conducted through informal digital communication tools, such as email, printed letters, and basic Excel sheets. This traditional architecture lacks any integrated software framework or centralized system.

🔷 Current Architecture Characteristics:

* No unified platform for students, coordinators, or companies.
* Communication occurs via email, phone, or physical visits.
* Student applications are submitted manually or as documents attached to emails.
* Companies offer internships via calls, posters, or informal outreach.
* Internship progress is tracked via weekly logs, often submitted as Word/PDF documents offline or emailed.
* Final reports are submitted in hard copy or via email, with no centralized repository.

🔻 Limitations of the Current Architecture:

1. Lack of Centralization: Data is scattered across emails, paper documents, and personal storage.
2. No Automation: Manual work is prone to delays, errors, and inefficiency.
3. No Real-Time Updates: Stakeholders cannot track student progress or application status instantly.
4. Security Risks: Sensitive data is exchanged over unprotected channels without encryption or access control.
5. No Analytics: Universities cannot generate insights or reports about internship trends, placement rates, or student performance.
6. Inconsistent Communication: Reminders and follow-ups are manual, resulting in missed deadlines.

🔎 Why Change is Necessary:

Given the increasing scale of university enrollment and the need for efficient workforce preparation, this outdated system cannot support modern internship workflows. A new architecture is essential to handle:

* + High volume of students
  + Professional company collaboration
  + Real-time communication
  + Long-term internship tracking and reporting

## 3.5 Proposed Software Architecture

The proposed architecture introduces a modern, cloud-based, centralized web system built on a three-tier architecture that cleanly separates user interface, business logic, and data storage.

🔷 Key Components of the Proposed Architecture:

1. Presentation Layer (Frontend)
   * Built with HTML/CSS/JavaScript (or frameworks like React)
   * Provides intuitive dashboards for students, coordinators, and companies
   * Enables real-time interaction and dynamic content updates
2. Application Layer (Backend/Server)
   * Handles all business logic, authentication, request processing, and system workflows
   * Built using PHP or a similar web framework
   * Implements role-based access and notification handling
3. Data Layer (Database)
   * Centralized relational database
   * Stores user data, internship listings, applications, reports, and logs
   * Includes backup, recovery, and data integrity enforcement mechanisms

✅ How the New Architecture Solves Current Limitations:

Table 3.1 How the New Architecture Solves Current Limitations:

|  |  |
| --- | --- |
| **Problem in Current System** | **Solution in Proposed System** |
| Scattered manual processes | Centralized web-based platform for all stakeholders |
| No real-time communication | Built-in notification and status tracking |
| Security vulnerabilities | Encrypted connections, hashed passwords, role-based access |
| Lack of visibility | Dashboards and progress tracking for all actors |
| Missed deadlines and weak reminders | Automated alerts and reminders via email or in-app messages |
| Difficulty tracking applications | All applications are stored, searchable, and filterable in system |

✅ Architecture Comparison Summary:

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Current** | **Proposed** |
| User Access | Manual, unverified | Authenticated, role-based |
| Data Management | Scattered files, emails | Centralized relational database |
| Communication | Delayed, informal | Real-time notifications, alerts |
| Progress Tracking | Manual weekly logs | Structured digital reporting |
| Application Handling | Paper/email submission | In-system application & status tracking |
| Scalability | Not scalable | Cloud-ready, scalable backend |
| Security | Low (no encryption) | Encrypted, secure access, audit logs |

Table 3.2 Architecture Comparison Summary

### 3.5.1 Subsystem Decomposition

To manage complexity and support modular development, the Web-Based Internship Placement System (WBIPS) is decomposed into logical subsystems. Each subsystem is responsible for a well-defined set of functionalities and interacts with others through clearly defined interfaces.

|  |  |
| --- | --- |
| **Subsystem** | **Purpose** |
| Authentication Module | Handles user registration, login, role management, and access control |
| Internship Management | Allows companies to post internships and students to view listings |
| Application Module | Enables students to apply for internships and companies to review them |
| Report Management | Allows students to submit weekly/final reports and coordinators to review |
| Notification System | Sends automated alerts to users (e.g., approvals, reminders, updates) |
| Admin Control Panel | Provides system-wide control and user/data management for administrators |

Table 3.3 List of Subsystems & Their Descriptions

🔁 Subsystem Interactions and Dependencies

* + The Authentication Module is required before any other subsystem is accessible. It is the foundation for security and access management.
  + The Internship Management and Application Module are tightly coupled, as internships are the targets of applications.
  + The Report Management subsystem relies on the Application Module to identify approved students who are actively interning.
  + The Notification System is event-driven and loosely coupled with all other modules, triggered by changes in application status, report submission, or deadlines.
  + The Admin Control Panel accesses and oversees all modules to ensure platform integrity and data governance.

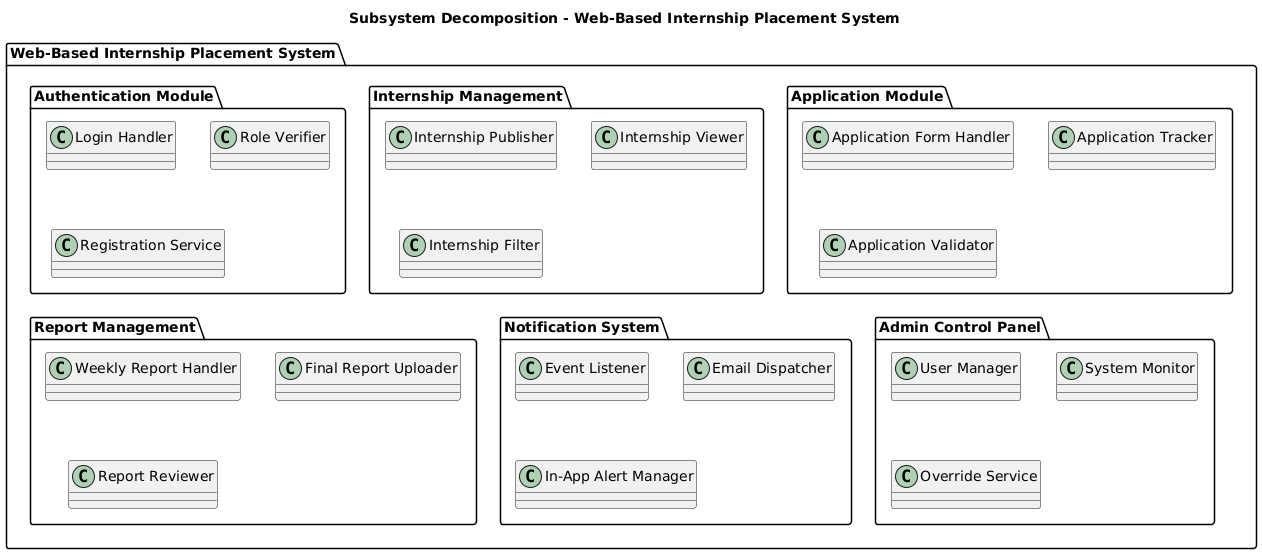


Figure 21 Subsystem Decomposition

### 3.5.2 Component diagram

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system.

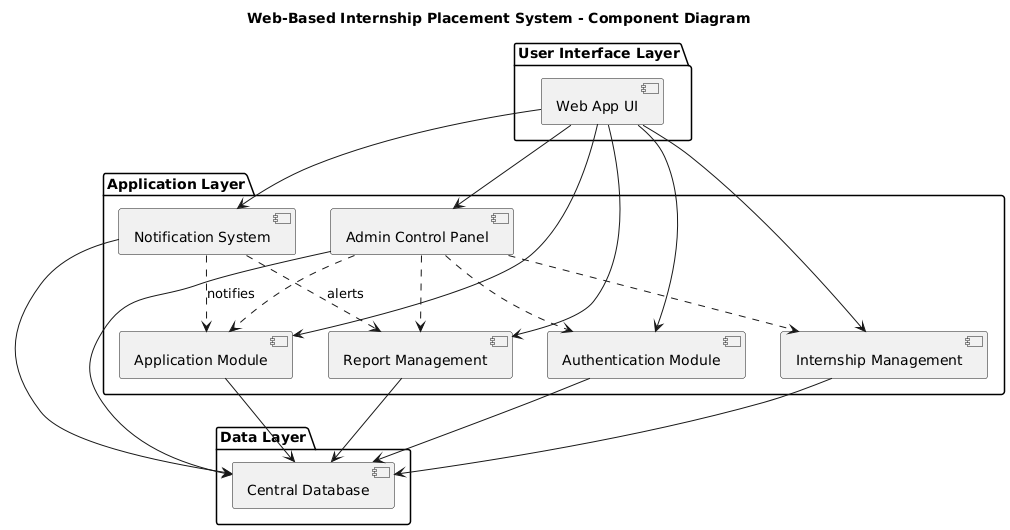


Figure Component Diagram

### 3.5.3 Deployment diagram

Deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).



Figure 23 Deployment Diagram

### 3.5.4 Persistent Data Management

The system manages a variety of persistent data, which must be securely stored, consistently accessed, and efficiently queried.

🔹 Types of Persistently Stored Data

* + User Information: Name, email, role, department, company, password (hashed)
  + Internship Listings: Title, description, field, deadline, company ID
  + Applications: Internship ID, student ID, status, timestamp
  + Reports: Weekly and final internship reports (PDF/DOC), comments, submission dates
  + Notifications: Alert messages for status updates, reminders, and deadlines
  + Admin Logs: User actions, system changes

🔹 Chosen Database Technology

* + Relational DBMS: PostgreSQL (or MySQL)
  + Rationale:
  + Strong ACID compliance
  + Robust indexing and query optimization
  + Supports complex relationships and integrity constraints

🔹 Data Storage Mechanisms

* + Structured data is stored in relational tables
  + Files (PDFs, DOCs) are stored in the filesystem/cloud and linked via DB metadata
  + All text is stored as VARCHAR or TEXT; documents as file paths

🔹 Backup, Recovery & Integrity

* + Data integrity enforced using:
  + Primary/foreign keys
  + NOT NULL & UNIQUE constraints
  + Role-based access validation

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| user\_ id | INT | PK, AUTO\_INCREMENT |
| name | VARCHAR(100) | NOT NULL |
| email | VARCHAR(100) | UNIQUE, NOT NULL |
| password | VARCHAR(255) | NOT NULL |
| role | ENUM('student', 'company', 'coordinator',' admin | Not NULL |
| status | ENUM('active', 'inactive') | DEFAULT 'active' |

Table 3.4 Users Table

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| student\_ id | INT | PK, FK -> users(user­ id) |
| university | VARCHAR(100) | NOT NULL |
| department | VARCHAR(100) | NOT NULL |

Table 3.5 Student Table

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Company id | INT | PK, FK -> users(user id) |
| Company name | VARCHAR(100) | NOT NULL |
| Industry field | VARCHAR(100) | NOT NULL |
| approved | BOOLEAN | DEFAULT FALSE |

Table 3.6 Companies Table

### 3.5.5 Detailed Database Design

#### 3.5.5.1 Relational Tables

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Primary Key | Fields | Foreign Keys |
| Users | User id (PK) | name, email, password, role, department, company name | — |
| Internships | Internship id (PK) | title, description, field, duration, posted by, deadline | Posted by → users(user id) |
| Applications | Application id (PK) | Student id, internship id, status, submitted at | Student id → users(user id), internship id → internships |
| Reports | Report id (PK) | Student id, file path, type (weekly/final), submitted at | Student id → users(user id) |
| Notifications | Notification id (PK) | User id, message, status, created at | User id → users(user id) |

Table 3.7 Relational Tables

#### 3.5.5.2 Object-Oriented Data Modeling and Normalization

Unlike traditional relational database normalization (1NF–3NF), object-oriented systems apply normalization through class design, encapsulation, and composition.

The system avoids redundancy and ensures data integrity using the following OOP practices:

* + Encapsulation: Attributes are grouped within their appropriate classes, e.g., User, Internship, and Application.
  + Composition: Role-specific data such as Student Profile, Company Profile, and Coordinator Profile are associated with the core User class.
  + Association: Relationships between objects are modeled using associations instead of foreign keys.
  + Avoiding Inheritance Misuse: Instead of making Company, Student, or Coordinator subclasses of User, a single User class is used with a role attribute.

### 3.5.6 Access Control and Security

The Web-Based Internship Placement System (WBIPS) implements layered security measures to ensure that data is protected and only authorized users can access system resources.

🔒 User Authentication

* + Role-Based Login: Only registered users (students, company reps, coordinators, and admins) can access the platform.
  + Password Hashing: Passwords are never stored in plain text. They are hashed using industry-standard algorithms (e.g., bcrypt).
  + Session Management: Users are assigned tokens (JWT or session-based) after login. These tokens expire after a set time of inactivity.

🛡User Authorization

* + Role-Based Access Control (RBAC):
  + Students: Can apply, upload reports, and track status.
  + Companies: Can post internships, review applications.
  + Coordinators: Can approve, track, and evaluate.
  + Admins: Can manage users and oversee platform data.
  + Every endpoint checks user role before granting access.

🔐 Data Privacy and Encryption

* + SSL/TLS Encryption: All data in transit is encrypted using HTTPS.
  + Input Validation & Sanitization: Prevents injection attacks (SQL/XSS).
  + Secure File Uploads: Limits file size and type, stores files in protected directories with randomized names.
  + Audit Logging: Tracks all key system actions (logins, uploads, approvals).

📂 Data Protection

* + Access to personal or confidential information (e.g. reports) is restricted by user role.
  + Users can only view their own data unless granted permissions (e.g., coordinators view student reports).

### 3.5.7 Global Software Control

The global software control structure of WBIPS is event-driven and modular, ensuring smooth coordination between subsystems.

🔄 Control Flow Overview

* + The system follows a user-initiated request-response model.
  + Users (students, coordinators, companies) initiate actions through the frontend.
  + Requests are routed through a central controller (backend/API) and dispatched to appropriate subsystems.
  + Responses are returned via JSON and rendered dynamically on the client side.

⚙ Subsystem Synchronization

* + Subsystems like Application Handling, Internship Management, and Notifications are connected via internal services.

For example:

* When a student submits an application, it triggers a notification to the coordinator.
* When a report is uploaded, the system checks deadlines and updates the tracker.
  + Notifications and updates are queued and processed asynchronously using a task/job queue system (e.g., Celery or Node-based queues).

### 3.5.8 Boundary Conditions

The system is designed to remain stable and secure even under edge or extreme-case scenarios.

🚀 Startup

* Performs system health checks (database, file storage, authentication service)
* Loads environment variables and initializes cache/middleware

🛑 Shutdown

Graceful shutdown ensures:

* + All active sessions are closed
  + Pending uploads are completed or rolled back
  + Logs are saved before termination

⚠ Error Handling

System uses centralized error handlers for:

* 404 (Page not found)
* 401 (Unauthorized access)
* 500 (Internal server errors)

Errors are logged and presented to users with friendly messages.

Input validation is done both client-side and server-side to avoid malformed data entry.

💥 Extreme Conditions

|  |  |
| --- | --- |
| Scenario | System Response |
| High user traffic | Load balancer + auto-scaling of backend instances |
| File upload over size limit | Reject with proper error message and upload limits |
| Multiple concurrent applications | Queued submissions; prevent duplication using unique keys |
| Attempt to access unauthorized data | Block and log the attempt; notify admin if needed |
| Database unresponsive | Retry mechanism + temporary fallback message |
| Deadline-sensitive reports | Auto-reminders and deadline validation before submission |

Table 3.8 Extreme Conditions

# References

1. Sommerville, I. (2016). Software engineering (10th ed.). Pearson.

2. Larman, C. (2004). Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development (3rd ed.). Prentice Hall.

3. Pressman, R. S. (2014). Software engineering: A practitioner’s approach (8th ed.). McGraw-Hill Education.

4. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.

5. Elmasri, R., & Navathe, S. B. (2017). Fundamentals of database systems (7th ed.). Pearson.

6. Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The unified modeling language user guide (2nd ed.). Addison-Wesley.

7. International Organization for Standardization. (2011). ISO/IEC 25010:2011 – Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – System and software quality models.

8. Object Management Group. (2017). UML 2.5 specification. https://www.omg.org/spec/UML