جـــامـعـة الـــشرق الأوسـط
MIDDLE EAST UNIVERSITY

Faculty of Information Technology

Computer Science Department

Computer Science

JobHive: Empowering Students and Graduates to Find Jobs and Internships
Graduation Project (1/2) Report

Prepared by:

Student ID	Students Name
202110400	Motasem Haroon
202110434	Ghaith Al-Bilbesi
202110440	Waleed Al-lawie
202110737	Wadea Al-halaiqa

Supervised by:

Dr. Madallah Al Majali

To obtain

BSc in Computer Science

Semester / Academic Year

Group No.: CS-24-1-**1**-**serial**

Middle East University

Declaration

We hereby acknowledge that the work presented in this document report and the ideas based upon are the group members own unless stated otherwise and properly cited in text and referenced at the end of the document.

Student ID	Students Name	Signature	Date
202110400	Motasem Haroon		
202110434	Ghaith Al-Bilbesi		
202110440	Waleed Al-lawie		
202110737	Wadea Al-halaiqa		

Faculty of Information Technology
Graduation Project (1/2) Report

Supervisor Approval موافقة المشرف

APPROVAL FOR SUBMISSION

I certify that this project report entitled "**JobHive: EMPOWERING STUDENTS AND GRADUATES TO FIND JOBS AND INTERNSHIPS**" was prepared by **Ghaith Al Bilbesi, Motasem Haroon** has met the required standard for submission in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science at MEU.

Approved by

Signature: ……..……..…….....

Supervisor: Dr…..…….……

Date: ……..……

Acknowledgements الشكر والعرفان
ACKNOWLEDGEMENTS I would like to thank everyone who had contributed to the successful completion of this project starting from the guidance of our supervisor Dr. …… and the team work with my colleagues who worked with this project Dr. ……., Mrs/Miss…….

Abstract (English) المستخلص (إنجليزى)

JobHive: Empowering Students and Fresh Graduates to Secure Career Opportunities

The transition from academia to the professional world is often challenging for students and fresh graduates due to a lack of accessible, centralized platforms that connect them to relevant job opportunities. JobHive aims to address this issue by developing a web portal tailored specifically for students and recent graduates seeking internships, part-time, and full-time employment.

The proposed platform will integrate user-friendly design with advanced features such as personalized job recommendations, resume-building tools, and employer-student interaction modules. Additionally, it will include filters for location, skills, and industry preferences, ensuring a seamless job search experience.

To achieve this, the project will employ modern web development frameworks and technologies, including React.js for the front-end and Node.js for the back-end. User data and preferences will be managed securely using a scalable database infrastructure. A prototype will be tested with a sample group of users to gather feedback and refine the platform's functionality.

If successful, JobHive will serve as a valuable tool in bridging the gap between academia and the workforce, empowering students and graduates to launch their careers effectively. The platform's impact could extend beyond individual users, benefiting educational institutions and employers by facilitating efficient talent acquisition and placement.

المستخلص (عربى) Abstract (Arabic)

عنوان المشروع

المستخلص

يشكل الانتقال من الأوساط الأكاديمية إلى عالم العمل تحدياً كبيراً للطلاب والخريجين الجدد بسبب عدم وجود منصة متكاملة تقدم فرص عمل مناسبة يعمل مشروع Jobhive على تطوير بوابة إلكترونية مخصصة للطلاب والخريجين الجدد الباحثين عن فرص التدريب والوظائف بدوام جزئي ودائم ويهدف إلى معالجة هذه المشكلة من خلال

سيتمتع النظام المقترح بتصميم سهل الاستخدام ويتضمن ميزات متقدمة مثل التوصيات الوظيفية المخصصة، وأدوات بناء السيرة الذاتية ووحدة حوار بين صاحب العمل والطالب. كما سيتضمن أيضاً وظائف التصفية بناءً على الموقع الجغرافي والمهارات وتفضيلات الصناعة لضمان سلاسة البحث عن الوظائف. ولتحقيق ذلك، سيستخدم المشروع أحدث تقنيات تطوير الويب مثل React.js للواجهة الأمامية وNode.js للواجهة الخلفية. ستتم إدارة بيانات المستخدم وتفضيلاته بشكل آمن باستخدام بنية قاعدة بيانات قابلة للتطوير. سيتم اختبار النماذج الأولية مع مجموعات المستخدمين للحصول على تعليقاتهم وتحسين الوظائف. إذا نجح المشروع، فسيكون Jobhive أداة قيّمة لسد الفجوة بين التعليم وسوق العمل، مما يمكّن الطلاب والخريجين من بدء حياتهم المهنية بكفاءة. كما يمكن أن تؤثر أيضاً على المؤسسات التعليمية وأصحاب العمل من خلال تسهيل عملية التوظيف والاختيار.

Table of Contents

List of Figures 10

List of Tables 12

Chapter 1: Introduction 13

1.1 Background and Context 13

1.2 Description of the Challenge/Problem/Opportunity 13

1.3 Description of the Suggested Solution 14

1.4 Literature Review (related work) 14

1.5 Technology and tools to be used 15

1.6 Organization of the Report 15

Chapter 2: Project Plan 18

2.1 Project Objective 18

2.2 Project Scope 18

2.3 Software Process Model 20

2.4 Project Schedule 22

2.5 Gantt Chart 23

Chapter 3: Requirements and Analysis 24

Research Methodology 24

3.1 Functional Requirements 27

3.2 Non-Functional Requirements 29

Chapter 4: Architecture and Design 30

Chapter 4: 30

4.1 Architecture 30

4.2 Use Case Diagram 35

4.3 Use Cases Descriptions/Flow of Events 36

4.4 Activity Diagram..... 46

4.5 Class Diagram..... 57

4.6 Entity Relationship Diagram..... 58

List of Figures

Figure 2.3.1 Methodology..... 21

Figure 2.4.1 Project Schedule..... 22

Figure 2.5.1 Team Structure and Roles..... 23

Figure 4.1.1 Architecture Diagram..... 30

Figure 4.1.1 Use case diagram..... 35

Figure 4.3.1 Job Seeker Registration Activity Diagram..... 46

Figure 4.4.2 Register (Employer) Activity Diagram..... 47

Figure 4.4.3 Login Activity Diagram..... 48

Figure 4.4.4 Job Search Activity Diagram..... 49

Figure 4.4.5 Apply for a Job (Activity Diagram)..... 50

Figure 4.4.6 Posting Jobs (Employer) - Activity Diagram..... 51

Figure 4.4.7 Review Applications (Employer) - Activity Diagram..... 52

Figure 4.4.8 Messaging Activity Diagram..... 53

Figure 4.4.9 Track Application Status (Job Seeker)..... 54

Figure 4.4.10 Manage Users (Admin)..... 55

Figure 4.4.11 Track Issues (Admin) 56

Figure 4.5.1 Class Diagram..... 57

Figure 4.6.1 Entity Relationship Diagram 58

List of Tables

Table 2.4.1 Project Schedule Table 22

Table 0.1 Comparing vendor Solution 26

Table 3.1.1 Functional Requirments..... 28

Table 3.2.1 Non-Functional Requirements 29

Table 4.3.1 Login Use Case Description 36

Table 4.3.2 Register Use Case Description..... 38

Table 4.3.3 Job Search Use Case Description 39

Table 4.3.4 Track Application Status Use Case Description..... 40

Table 4.3.5 Messaging Use Case Description..... 41

Table 4.3.6 Posting Jobs Use Case Description..... 42

Table 4.3.7 Manage profile Use Case Description..... 43

Table 4.3.8 Manage users Use Case Description..... 44

Table 4.3.9 Tracking Issues Use Case Description 45

Chapter 1: Introduction

This chapter illustrates the description of the challenges in assisting students and fresh graduates in finding career opportunities, the proposed solution to address these challenges, related work, and the technologies and tools to be used in implementing the JobHive platform.

1.1 Background and Context

Students and recent graduates face significant challenges transitioning from academia to the professional world. Many struggle to access relevant job opportunities tailored to their skills and preferences. Existing platforms often fail to address the specific needs of this demographic, such as personalized job recommendations, resume-building tools, and integration with academic institutions. This gap presents an opportunity to develop a platform specifically designed to bridge this divide, helping students and graduates connect with suitable internships, part-time, and full-time jobs.

1.2 Description of the Challenge/Problem/Opportunity

The critical issue is the lack of a dedicated platform that caters to the unique needs of students and fresh graduates in the job market. Existing systems are too generalized, leaving users overwhelmed with irrelevant options or underwhelmed by the lack of meaningful opportunities. Employers also struggle to identify and recruit fresh talent efficiently. This disconnect highlights the need for a specialized system that simplifies the process for both job seekers and employers.

1.3 Description of the Suggested Solution

JobHive is a web-based platform tailored to address these challenges. It will provide an intuitive interface with features like personalized job recommendations, resume-building tools, and employer-student engagement modules. Users will be able to filter opportunities based on location, industry, and skills, ensuring a streamlined job search experience. Employers will benefit from tools designed to identify and engage with promising candidates effectively. The platform's community-driven approach will foster meaningful connections between job seekers and employers.

1.4 Literature Review (related work)

Existing platforms like LinkedIn, Glassdoor, and Indeed provide general job search functionalities but lack specific features for students and entry-level job seekers. Research indicates that platforms integrated with universities or focused on early-career individuals are more effective in addressing this demographic's needs. For instance, some systems allow employers to post internships and entry-level jobs while providing career guidance to users. However, these systems often have limitations, such as subscription fees or limited access to opportunities. JobHive aims to improve upon these by offering a free, modern, and easy-to-navigate platform with tailored functionalities.

1.5 Technology and tools to be used

- **Frontend:** React.js for a dynamic and responsive user interface.
- **Backend:** Node.js for a scalable server-side architecture.
- **Database:** MongoDB for flexible and efficient data management.
- **Tools:**
  - Visual Studio Code for development
  - GitHub for version control
  - Visual Paradigm for design modelling
- **Deployment:** Railway for continuous deployment from the GitHub repository, ensuring the platform remains up-to-date.

1.6 Organization of the Report

- **Chapter 1: Introduction**

  This chapter sets the stage for the report, presenting the problem statement around the lack of a specialized job platform for students and fresh graduates. It provides a thorough description of the problem, outlining the challenges faced in finding suitable career opportunities. The suggested solution, JobHive, is introduced as a web-based platform tailored to the needs of students and fresh graduates. This chapter also reviews related literature and details the technologies and tools utilized in the development of JobHive.

- **Chapter 2: Project Plan**

  Project Plan The project plan chapter outlines the objectives, scope, and the software process model of the JOBHIVE project. It details the strategic plan for achieving the project's goals, defining the limits of the project scope, and describing the Waterfall methodology adopted for structured development. The project schedule, including Gantt charts, and the roles and responsibilities within the team are also presented, providing a comprehensive roadmap for the project's execution.

- **Chapter 3: Requirements and Analysis**

  Chapter 3 delves into the functional and non-functional requirements that frame the development of JOBHIVE. It lists the essential features and specifications that the system must meet, such as user authentication, resource cataloging, and search functionality. The chapter also defines performance, security, and usability requirements, ensuring the system's reliability and efficiency.

- **Chapter 4: Architecture and Design**

  Architecture and Design The architecture and design chapter describes the structural blueprint of JOBHIVE, highlighting the layered approach consisting of the presentation, application, and data layers. It includes detailed design components like use case diagrams, sequence diagrams, activity diagrams, class diagrams, and the entity-relationship diagram, illustrating the system's design considerations and database schema.

- **Chapter 5: System Implementation**

System Implementation In this chapter, the implementation details are discussed, covering the development environment, database implementation, user interface construction, and functionality implementation. The integration of external services/APIs, security measures, and performance optimization techniques are also elaborated upon, showcasing the practical application of the proposed design and architecture.

- **Chapter 6: System Testing and Installation**

System Testing and Installation Chapter 6 addresses the testing and installation processes of the JOBHIVE. It describes the various testing strategies employed, such as unit, integration, system, and user acceptance testing, as well as security and performance testing. The installation process is detailed to ensure the correct deployment of the system, and a thorough validation of requirements is conducted to confirm that the system fulfills the established criteria.

- **Chapter 7: Conclusion and Future Work**

The final chapter wraps up the report with a summary of the project's achievements and an evaluation of how the project goals were met. It acknowledges the limitations and challenges encountered during the project and proposes recommendations for future enhancements. The concluding remarks reflect on the project's contribution to academic resource

Chapter 2: Project Plan

2.1 Project Objective

1. Develop a **responsive web application** that assists students and fresh graduates in finding internships, part-time, and full-time jobs.
2. Implement personalized features, such as **job recommendations**, **resume-building tools**, and **interaction modules** to connect users with employers.
3. Create a user-friendly, **accessible platform** optimized for both desktop and mobile devices.
4. Enable employers to identify and engage with fresh talent efficiently through targeted tools.
5. Build a robust and scalable system that fosters career growth and opportunities.

2.2 Project Scope

The JobHive platform is envisioned as a state-of-the-art, responsive web application designed to bridge the gap between students, fresh graduates, and potential employers. This project aims to create a dynamic job portal that provides personalized job recommendations, resume-building tools, and employer-student interaction features. Users will be able to search for opportunities by industry, location, and job type, ensuring a streamlined and efficient job-hunting experience. The platform will offer an intuitive interface compatible across devices and focus on accessibility and ease of use to cater to diverse user needs.

We operate under the assumption that our users—students, graduates, and employers—are familiar with basic computer and internet operations, have stable internet access, and are eager to leverage the platform for job-seeking or recruitment purposes. Employers are expected to provide accurate job postings and actively engage with the platform. The system will comply with industry standards for data security and usability.

The success of the JobHive platform depends on a scalable server infrastructure to handle high traffic, a reliable database system to store job listings and user data, and a responsive design to ensure compatibility with various devices. Additionally, we aim to foster collaboration with universities and businesses for outreach and promotion, along with continuous technical support for system maintenance and updates.

Our deliverables include a fully functional web-based job portal with a secure authentication system, a robust database of job listings and candidate profiles, and advanced filtering and search options. Other key features include resume-building tools, employer-candidate communication modules, a feedback system for platform improvements, and comprehensive documentation for users and administrators. Regular system updates and enhancements based on user feedback and evolving technology trends will ensure JobHive remains a valuable resource for its target audience.

2.3 Software Process Model

The **Waterfall Model** has been selected for the development of the JobHive platform due to its structured and sequential approach. This model ensures that each phase of the project is completed before moving on to the next, providing a clear and organized workflow.

Phases of the Waterfall Model

Planning

- The project commenced with an initial planning stage, where the scope, resources, and timeline were defined.

Requirements and Analysis

- This stage involved extensive interaction with stakeholders to gather comprehensive functional and non-functional requirements.

Design

- System and software design were the focus of this phase, resulting in detailed design documents that guided subsequent development.

Implementation

- The coding phase transformed design documents into a working software product.

Testing

- In this critical phase, the system underwent rigorous testing to identify and correct defects.

Deployment

- Upon successful testing, the software was deployed to the production environment.

Maintenance

- The final phase involves making necessary updates and improvements to ensure the system's long-term efficiency and reliability.

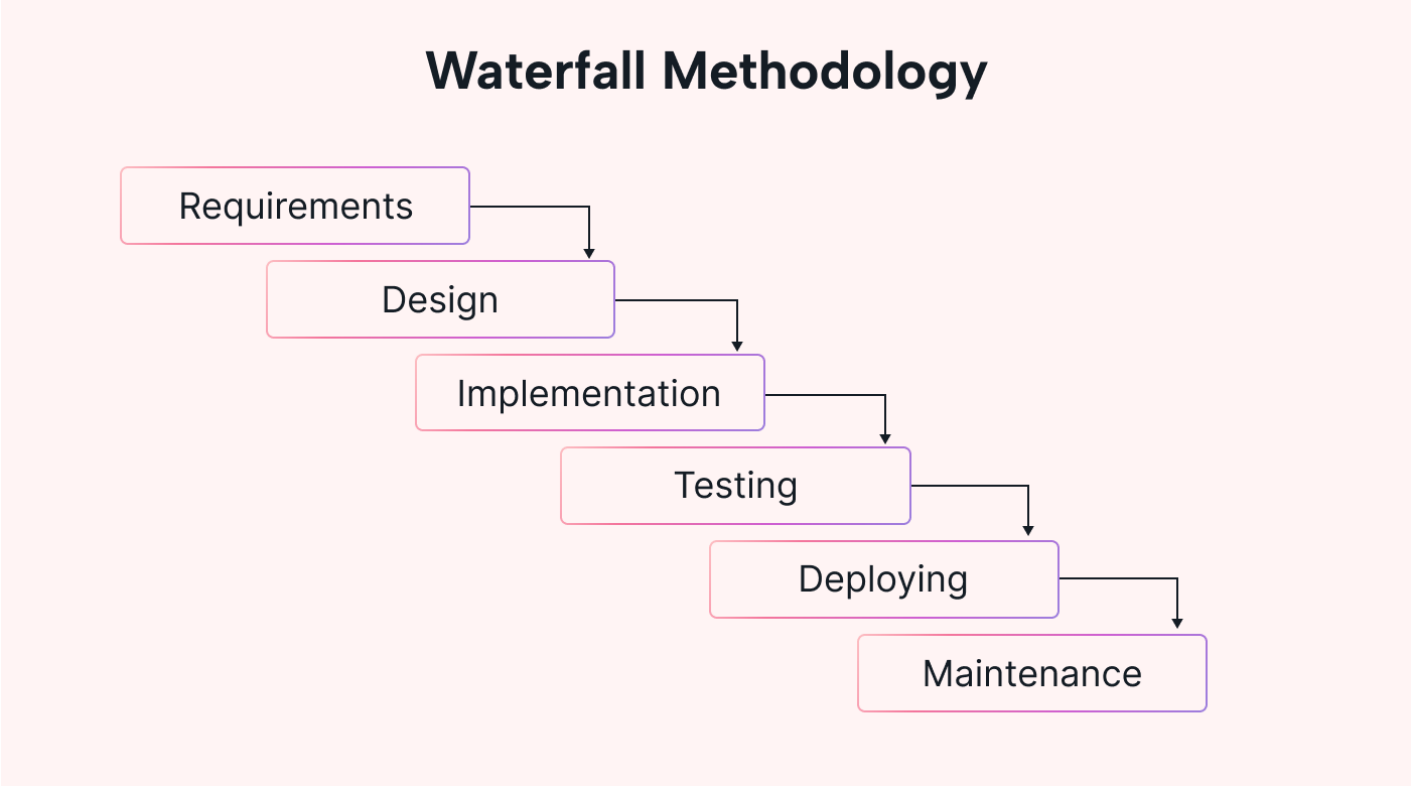

Figure 2.3.1 Methodology

2.4 Project Schedule

Task No.	Task Name	Start Date	Finish Date	Resources
1	JobHive Development	Thu 11/7/24	Thu 1/16/25	
1.1	Start	Thu 11/7/24	Thu 11/7/24	
2	Project Planning	Fri 11/8/24	Thu 11/21/24	Ghaith
2.1	Project Statement	Fri 11/8/24	Fri 11/14/24	Ghaith
2.2	Project Objectives	Fri 11/15/24	Thu 11/21/24	Ghaith
3	Requirements Analysis	Fri 11/22/24	Tue 12/3/24	Motasem
3.1	Functional Requirements	Fri 11/22/24	Wed 11/27/24	Motasem
3.2	Non-functional Requirements	Thu 11/28/24	Tue 12/3/24	Motasem
4	Architecture & Design	Wed 12/4/24	Mon 12/23/24	Wadea, Waleed
4.1	Use Case Diagrams	Wed 12/4/24	Mon 12/9/24	Wadea, Waleed
4.2	Use Case Flow-of-Events	Tue 12/10/24	Thu 12/12/24	Wadea, Waleed
4.3	Class Diagram	Fri 12/13/24	Tue 12/17/24	Wadea, Waleed
4.4	Entity Relationship Diagram	Wed 12/18/24	Thu 12/19/24	Wadea, Waleed
5	End of Phase 1	Thu 1/16/25	Thu 1/16/25	

Table 2.4.1 Project Schedule Table

Figure 2.4.1 Project Schedule

2.5 Gantt Chart

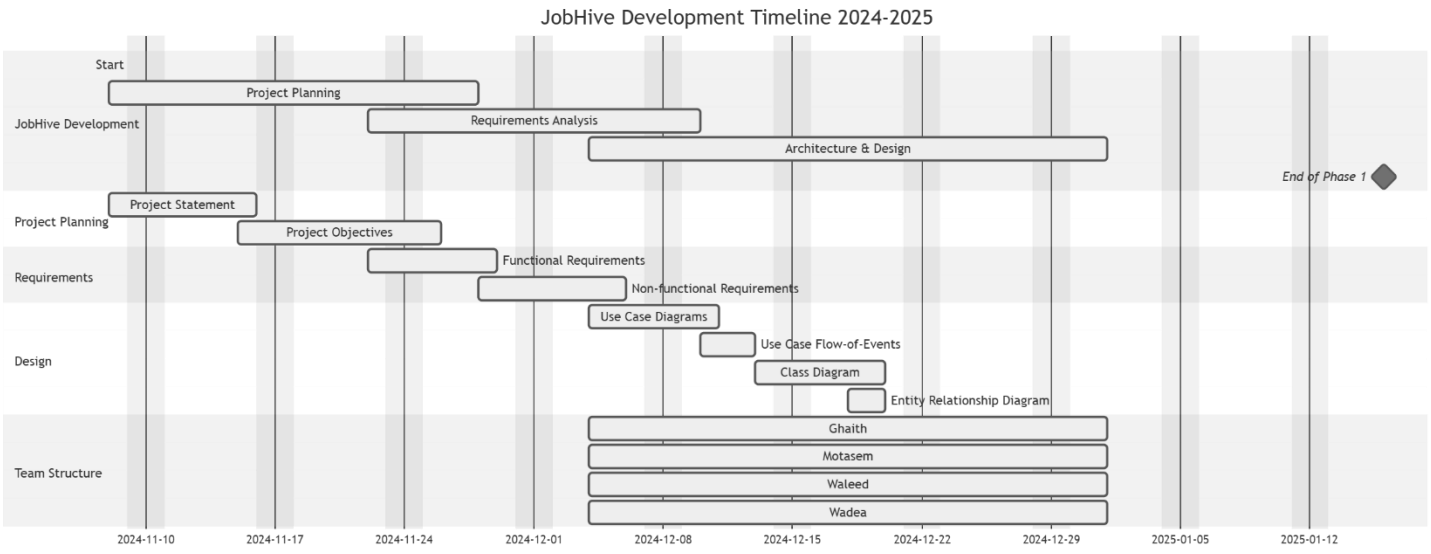

Figure 2.5.1 Team Structure and Roles

Chapter 3: Requirements and Analysis

Research Methodology

Definition:
Researching vendor solutions is a systematic approach to gathering information from existing platforms, products, and industry-leading services to identify best practices, assess capabilities, and discover gaps that can be improved upon in the proposed system.

Methodology:
In the JobHive project, vendor research was conducted by analyzing existing job portals and employment platforms, including:

- LinkedIn

- Indeed

- Glassdoor

- Handshake

- WayUp

The research focused on:

- **Feature Comparison:** Identifying core functionalities, usability, and effectiveness of job portals.

- **User Reviews & Market Analysis:** Understanding user feedback, complaints, and common issues.

- **Performance Benchmarking:** Assessing platform performance, scalability, and efficiency.

- **Security and Compliance Analysis:** Ensuring data privacy, compliance with GDPR, and role-based security.

- **Accessibility & UX/UI Design:** Reviewing how user-friendly and mobile-responsive the platforms are.

Findings:

- Many platforms lack personalized job recommendations tailored specifically to students and fresh graduates.

- Employer-student communication is minimal, making networking harder.

- Some platforms charge high fees for job postings, limiting opportunities for smaller companies.

- Internships and part-time jobs are not well-categorized.

- Resume-building tools are often unavailable or require third-party integration.

Impact on JobHive:

- Implementing a student-centric recommendation algorithm.

- Offering free job postings to encourage more employers to list openings.

- Integrating direct messaging & Q&A forums for employer-student engagement.

- Developing a custom resume-building tool for users to create strong applications.

- Enhancing search filters to help users find part-time, internship, and full-time roles effectively.

Category	LinkedIn	Indeed	Glassdoor	Handshake	WayUp	JobHive (Proposed)
User Registration & Authentication	✓	✓	✓	✓	✓	✓ Multi-factor Authentication
Profile Management	✓	✓	✓	✓	✓	✓ Student-centric Profiles
Job Listings & Search	✓	✓	✓	✓	✓	✓ Advanced Filters
Application Management	✓	✓	✓	✓	✓	✓ AI-based Matching
Direct Messaging	✓	✗	✗	✗	✗	✓ Employer-Student Chat
Internship & Part-Time Filtering	✗	✓	✗	✓	✓	✓ Specialized Student Filtering
Career Resources & Guidance	✓	✓	✓	✓	✓	✓ Student-focused Guidance
Security & Compliance	✓	✓	✓	✓	✓	✓ GDPR & AI Monitoring
Accessibility & Mobile Optimization	✓	✓	✓	✓	✓	✓ Mobile-First Design
Resume Builder	✗	✗	✗	✗	✗	✓ Integrated Resume Builder

Table 0.1 Comparing vendor Solution

3.1 Functional Requirements

FR-ID	Title	Description	Actors	Priority	Notes
FR-001	User Registration	Users must create an account with email verification.	Users	High	Supports multi-factor authentication.
FR-001.1	Resume Builder	The system provides users with a Resume Builder feature.	Users	High	AI-driven resume suggestions.
FR-002	User Login	Users log in with email and password, with optional two-factor.	Users	High	OAuth support recommended.
FR-003	Profile Management	Users can update personal details, education, and experience.	Users	High	Profile completion progress indicator.
FR-005	Posting Jobs	Employers can post job openings with descriptions.	Employers	High	Allow multimedia attachments.
FR-006	Job Searching	Users filter jobs by skills, location, industry, and type.	Users	High	Real-time search functionality.

FR-006.1	Filtering Jobs	Filter search results by criteria like location, skills, and industry.	Users	High	Advanced filtering options.
FR-006.2	Saving Jobs	Allow users to save jobs for later review.	Users	High	Favorites list for saved jobs.
FR-007	Job Application	Users apply to jobs and track status.	Users	High	Employers can shortlist applications.
FR-008	Review Applications	Employers review applications and interact with candidates.	Employers	High	AI-powered candidate ranking.
FR-011	Messaging System	Direct messaging between employers and candidates.	Users, Employers	High	Encrypted communication.
FR-013	Dashboard View	Admins manage users, jobs, and analytics.	Admin	High	Role-based access control.
FR-014	Report and Issue Tracking	Users report fraudulent job postings.	Users	High	AI-assisted fraud detection.

Table 3.1.1 Functional Requirments

3.2 Non-Functional Requirements

NFR-ID	Title	Description	Priority	Notes
NFR-001	Performance	The system must handle at least 1,000 concurrent users.	High	Response time < 2 seconds.
NFR-002	Scalability	The platform scales dynamically based on demand.	High	Cloud-based auto-scaling.
NFR-003	Security	User data is encrypted and complies with GDPR.	High	AES-256 encryption.
NFR-004	Usability	The UI is mobile-friendly and intuitive.	High	WCAG-compliant accessibility.
NFR-005	Availability	System uptime is 99.9%.	High	Automated failover mechanisms.
NFR-006	Maintainability	Allows easy updates and debugging.	Medium	Modular microservices architecture.
NFR-007	Compliance	Adheres to employment and privacy laws.	High	Legal audits bi-annually.

Table 3.2.1 Non-Functional Requirements

Chapter 4: Architecture and Design

4.1 Architecture

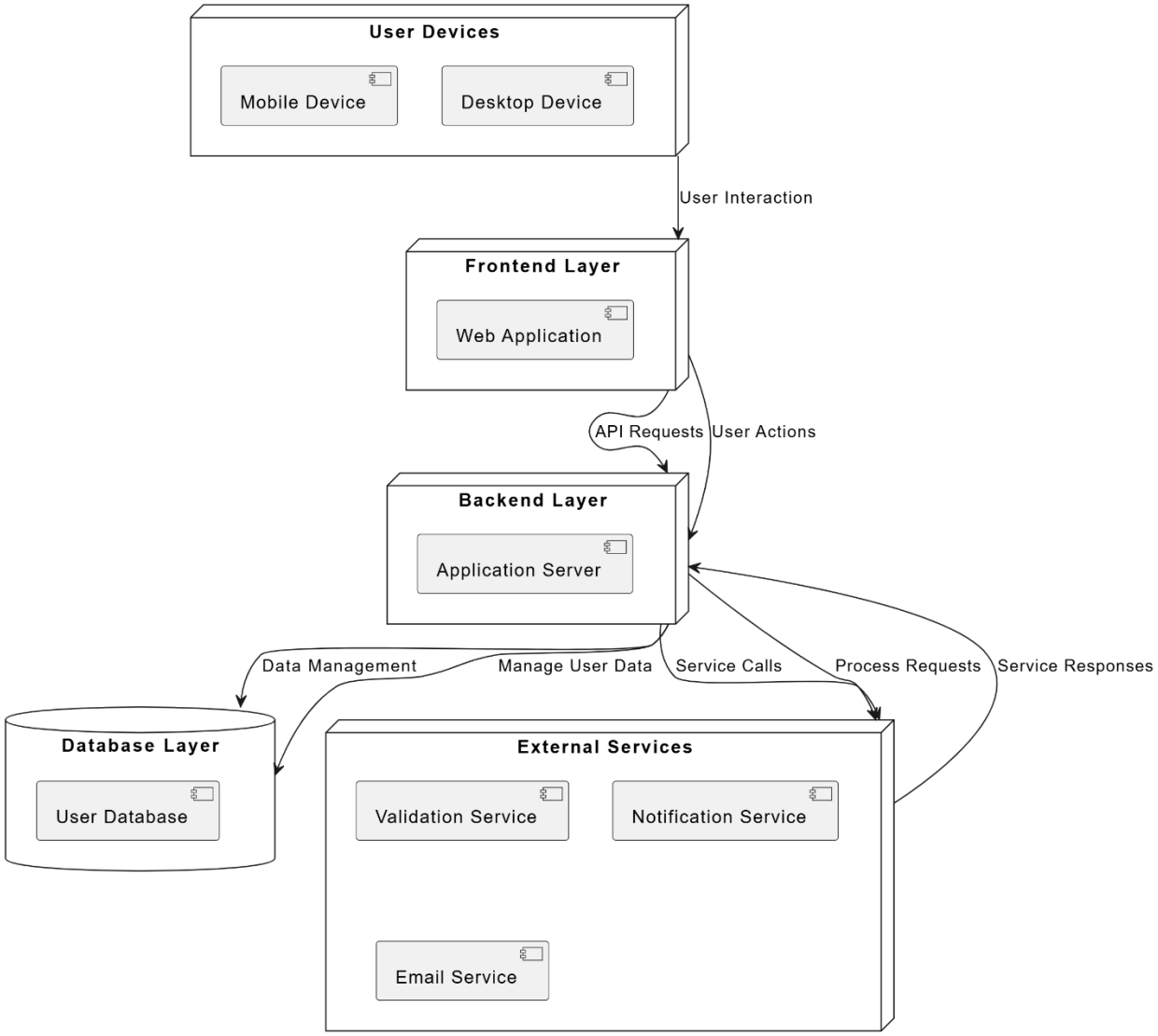

Figure 3.2.1 Architecture Diagram

High-Level Architecture of JobHive

User Devices Layer

- **Description**: This layer represents the end-user devices, such as mobile phones and desktops, through which users interact with the JobHive platform.
- **Components**:
  - **Mobile Devices**: Allow users to access the web application through responsive design features.
  - **Desktop Devices**: Provide full access to the web platform via modern browsers.
- **Functionality**: Facilitates user interaction with the frontend layer through browsers.

Frontend Layer (Presentation Layer)

- **Description**: Acts as the user-system interface, providing a responsive and dynamic experience. It is built with **ReactJS** to ensure a seamless and interactive user interface.
- **Components**:
  - **Web Application**: Built using ReactJS, it dynamically renders user interfaces for tasks such as searching jobs, posting jobs, and managing profiles.
  - **Client-Side Logic**: Implements interactivity using JavaScript, enabling real-time updates without page reloads.
- **Functionality**: Sends user actions to the backend as API requests and renders responses dynamically.

Backend Layer (Application and Business Logic)

- **Description**: The backend, powered by **NodeJS**, processes requests from the frontend, applies business rules, and interacts with the database and external services.
- **Components**:
  - **Application Server**: Manages user sessions, processes requests, and executes the platform's business logic.
  - **Business Logic**: Executes algorithms for matching job seekers with job postings, handling applications, and verifying resumes.
  - **Service Integration**: Manages API calls to external services such as the Resume Validation API, Notification Service, and Email Service.
- **Functionality**: Centralizes and secures all user data processing and system logic.

Database Layer (Data Storage and Management)

- **Description**: This layer stores all data related to the system, including user profiles, job postings, applications, and notifications, using **MongoDB**.
- **Components**:
  - **User Database**: Stores structured data, such as user details, resumes, and job postings.
  - **Database Management**: Ensures efficient and scalable data retrieval and storage.
- **Functionality**: Handles all database operations, ensuring secure and quick access to data by the backend layer.

External Services

- **Description**: Enhances the functionality of the platform by integrating with third-party services.
- **Components**:
    - **Resume Validation Service**: Validates uploaded resumes to ensure they are ATS-compliant and correctly structured.
    - **Notification Service**: Delivers real-time notifications and updates to users about job postings and application statuses.
    - **Email Verification Service**: Ensures the authenticity of user accounts during registration.
- **Functionality**: Processes external API calls initiated by the backend, providing additional functionality without adding complexity to the core system.

Data Flow

1. **User Interaction**: Users interact with the platform via the frontend (web application), submitting requests such as registration, job search, or resume uploads.
2. **Request Handling**: The frontend sends these requests to the backend as API calls.
3. **Backend Processing**:
    - Processes the request based on business logic.
    - Interacts with the database to retrieve or store data.
    - Makes service calls to external APIs (e.g., resume validation, notifications).
4. **Response Delivery**: The backend sends responses to the frontend, which dynamically updates the user interface for the user.

Architecture Type

- The architecture of JobHive follows a **3-Tier Architecture**, which ensures:
    - **Enhanced Security**: The database is accessed only through the backend, preventing direct access from users.
    - **Scalability**: Each layer (frontend, backend, database) can be scaled independently.
    - **Performance**: Load balancing and modular design ensure high availability and reliability.
    - **Ease of Maintenance**: Changes in one layer do not impact the others, allowing modular updates.

4.2 Use Case Diagram

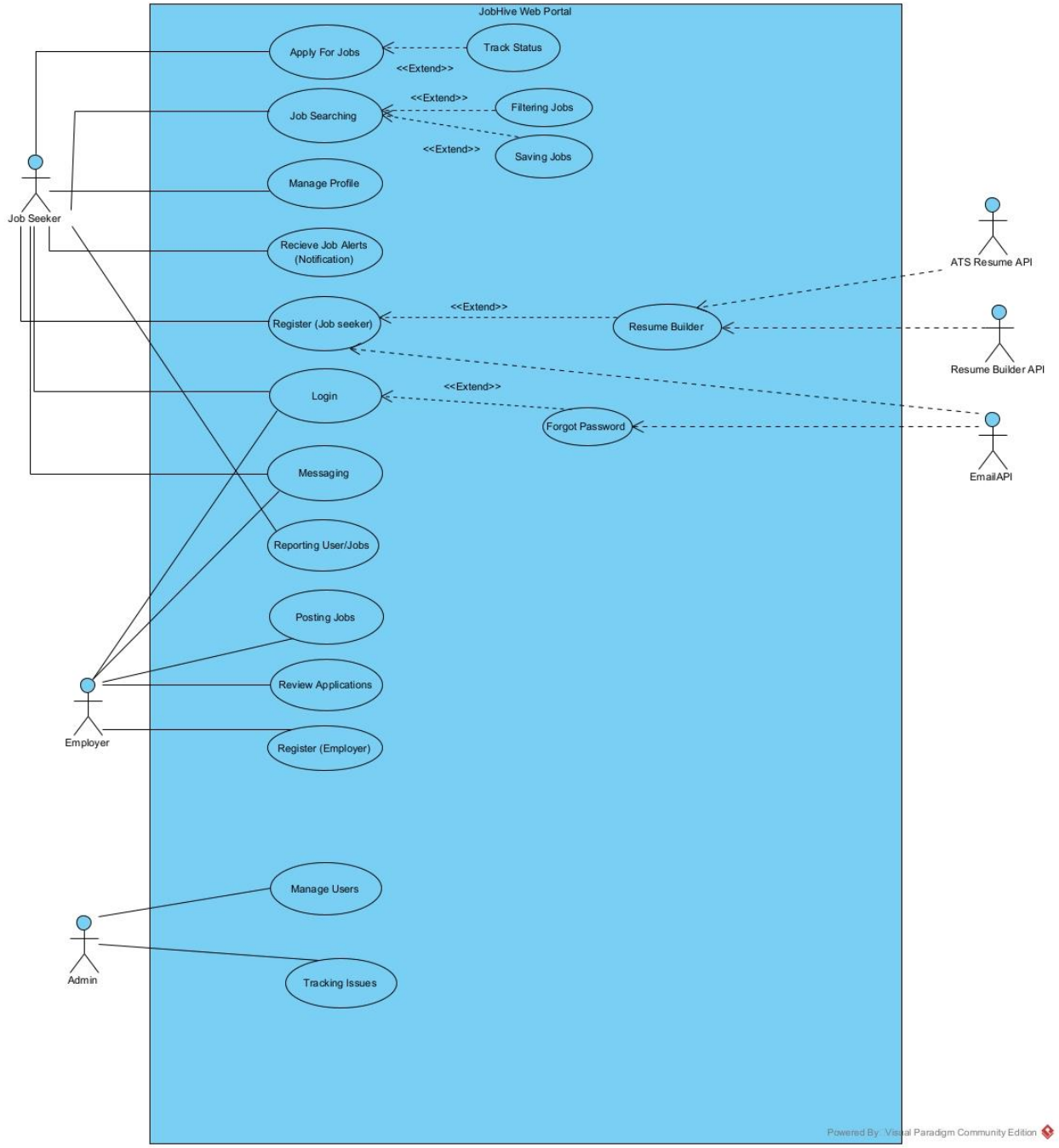

Figure 4.1.1 Use case diagram

4.3 Use Cases Descriptions/Flow of Events

Use Case name:	Login
Use Case ID:	1 | High
Description:	This use case describes how the actor logs into the System.
Primary Actor:	Job Seeker, Employer
Secondary Actor:	Registration system.
Pre-condition:	The user must register (use case must be performed first).
Main flow of events:	User/admin: 1. The system prompts the actor to enter their email and password. 2. The actor enters valid credentials and clicks the "Login" button. 3. The system validates the credentials against the database. 4. The system grants access and redirects the actor to their dashboard. **1. Sub-flow (Forgot Password):** 2. The actor selects the "Forgot Password" option. 3. The system prompts the actor to enter their email. 4. The system sends a password reset link to the actor's email. 5. The actor resets their password and logs in.
Alternative Flow of events:	**A1**: If credentials are invalid: • The system displays an error message: "Invalid email or password." • The actor is prompted to re-enter their credentials.
Post-condition:	The actor gains access to the system.

Table 4.3.1 Login Use Case Description

Use Case name:	Register(Job Seeker)
Use Case ID:	2 | High
Description:	This use case describes how a job seeker creates a new account.
Primary Actor:	Job Seeker
Secondary Actor:	Registration System, Resume Validation API, Resume Builder API
Pre-condition:	The job seeker must have a valid email address.
Main flow of events:	1. The actor clicks on (Register) option from the home page. 2. The system prompts the actor to enter his/her Username, E-mail and password and confirm-password. 3. The system prompts the job seeker to upload a resume or create one using the Resume Builder. 4. If the resume is uploaded, it is validated via the Resume Validation API. 5. The system creates a new user account and stores the resume in the database 6. The system sends a verification email to the job seeker. 7. The job seeker verifies their email address.
Alternative Flow of events:	**A1**: If the resume validation fails: 1. The system displays a message: "Invalid resume format. Please upload a valid file." 2. The job seeker is prompted to re-upload the resume.
Post-condition:	A new account is created, and the resume is either uploaded or built.

Use Case name:	Register (Employer)
Use Case ID:	2b| High
Description:	This use case describes how an employer creates a new account.
Primary Actor:	Employer
Secondary Actor:	Registration system.
Pre-condition:	The employer must have a valid email address.
Main flow of events:	1. The employer selects the "Register" option. 2. The system displays a registration form. 3. The employer enters required details (company name, email, and password). 4. The system validates the entered details. 5. The system creates a new employer account in the database. 6. The system sends a verification email to the employer. 7. The employer verifies their email address.
Alternative Flow of events:	**A1**: If the email is already registered: 1. The system displays an error message: "Email is already registered." 2. The employer is prompted to use a different email or log in.
Post-condition:	A new employer account is created, and the email is verified.

Table 4.3.2 Register Use Case Description

Use Case name:	Job Search
Use Case ID:	3 | High
Description:	This use case describes how a user searches for job postings.
Primary Actor:	Job Seeker
Secondary Actor:	Job Database
Pre-condition:	The user must be logged into the system.
Main flow of events:	1. The actor navigates to the job search page. 2. The actor enters search criteria (e.g., job title, location, skills). 3. The system retrieves matching job postings from the database. 4. The system displays the job postings in a list. 5. The actor selects a job to view detailed information.
Alternative Flow of events:	**A1**: If no results are found: 1. The system displays a message: "No jobs found matching your criteria." 2. The actor modifies the search criteria.
Post-condition:	The system displays a list of relevant job postings.

Table 4.3.3 Job Search Use Case Description

Use Case name:	Track Application Status
Use Case ID:	4 | Medium
Description:	This use case describes how a job seeker tracks their application status.
Primary Actor:	Job Seeker
Secondary Actor:	Job Database
Pre-condition:	The user must have applied for at least one job.
Main flow of events:	1. The job seeker logs into their account. 2. The user navigates to the "My Applications" section. 3. The system retrieves the list of applications from the database. 4. The system displays the status (e.g., Submitted, Under • Review, Accepted, Rejected) for each application.
Alternative Flow of events:	**A1**: If no applications exist: 1. The system displays a message: "No applications found."
Post-condition:	The system displays the current status of the job application.

Table 4.3.4 Track Application Status Use Case Description

Use Case name:	Messaging
Use Case ID:	5 | Medium
Description:	This use case describes how users communicate via the platform.
Primary Actor:	Job Seeker, Employer
Secondary Actor:	Messaging Service
Pre-condition:	The user must be logged in.
Main flow of events:	1. The user selects a contact from the messaging interface. 2. The user types a message and clicks "Send." 3. The system delivers the message to the recipient. 4. The recipient is notified of the new message.
Alternative Flow of events:	**A1**: If the recipient is unavailable: • The system queues the message for later delivery.
Post-condition:	The message is sent successfully.

Table 4.3.5 Messaging Use Case Description

Use Case name:	Posting Jobs
Use Case ID:	6 | High
Description:	This use case describes how employers post jobs.
Primary Actor:	Employer
Secondary Actor:	Job Database
Pre-condition:	The employer must be logged in.
Main flow of events:	1. The employer fills out job details (e.g., title, description, location). 2. The system validates the input. 3. The job is saved in the database. 4. A confirmation message is displayed to the employer.
Alternative Flow of events:	
Post-condition:	The job is posted and available for job seekers to view.

Table 4.3.6 Posting Jobs Use Case Description

Use Case name:	Manage profile
Use Case ID:	7 | High
Description:	This use case describes how employers review job applications.
Primary Actor:	Employer
Secondary Actor:	
Pre-condition:	Logged in, Must have at least one job posting.
Main flow of events:	1. The employer selects a job posting. 2. The system retrieves the list of applications for that job. 3. The employer reviews the applications (e.g., resumes, cover letters). 4. The employer updates the application status (e.g., • Accepted, Rejected).
Alternative Flow of events:	**A1**: If no applications exist: 1. The system displays a message: "No applications received for this job."
Post-condition:	The employer reviews and manages applications.

Table 4.3.7 Manage profile Use Case Description

Use Case name:	Manage users
Use Case ID:	1 | High
Description:	This use case describes how the admin manages user accounts.
Primary Actor:	Admin.
Secondary Actor:	User.
Pre-condition:	Logged in as an admin.
Main flow of events:	1. Admin logs in and navigates to the "Manage Users" section. 2. Admin views a list of users and selects a user to modify or delete. 3. The system applies the requested changes to the user account. 4. A confirmation message is displayed to the admin.
Alternative Flow of events:	**A1**: If the user does not exist: 1. The system displays an error message: "User not found."
Post-condition:	User accounts are updated or removed as needed.

Table 4.3.8 Manage users Use Case Description

Use Case name:	Tracking Issues
Use Case ID:	1 | Medium
Description:	This use case describes how admins monitor and address reported issues.
Primary Actor:	Admin
Secondary Actor:	Issue Database.
Pre-condition:	Admin must be logged in.
Main flow of events:	1. Admin navigates to the "Track Issues" section. 2. The system retrieves a list of reported issues. 3. Admin selects an issue to address. 4. The system applies the admin's resolution and updates the issue status.
Alternative Flow of events:	**A1**: If no issues are found: The system displays a message: "No issues to track."
Post-condition:	Reported issues are resolved.

Table 4.3.9 Tracking Issues Use Case Description

4.4 Activity Diagram

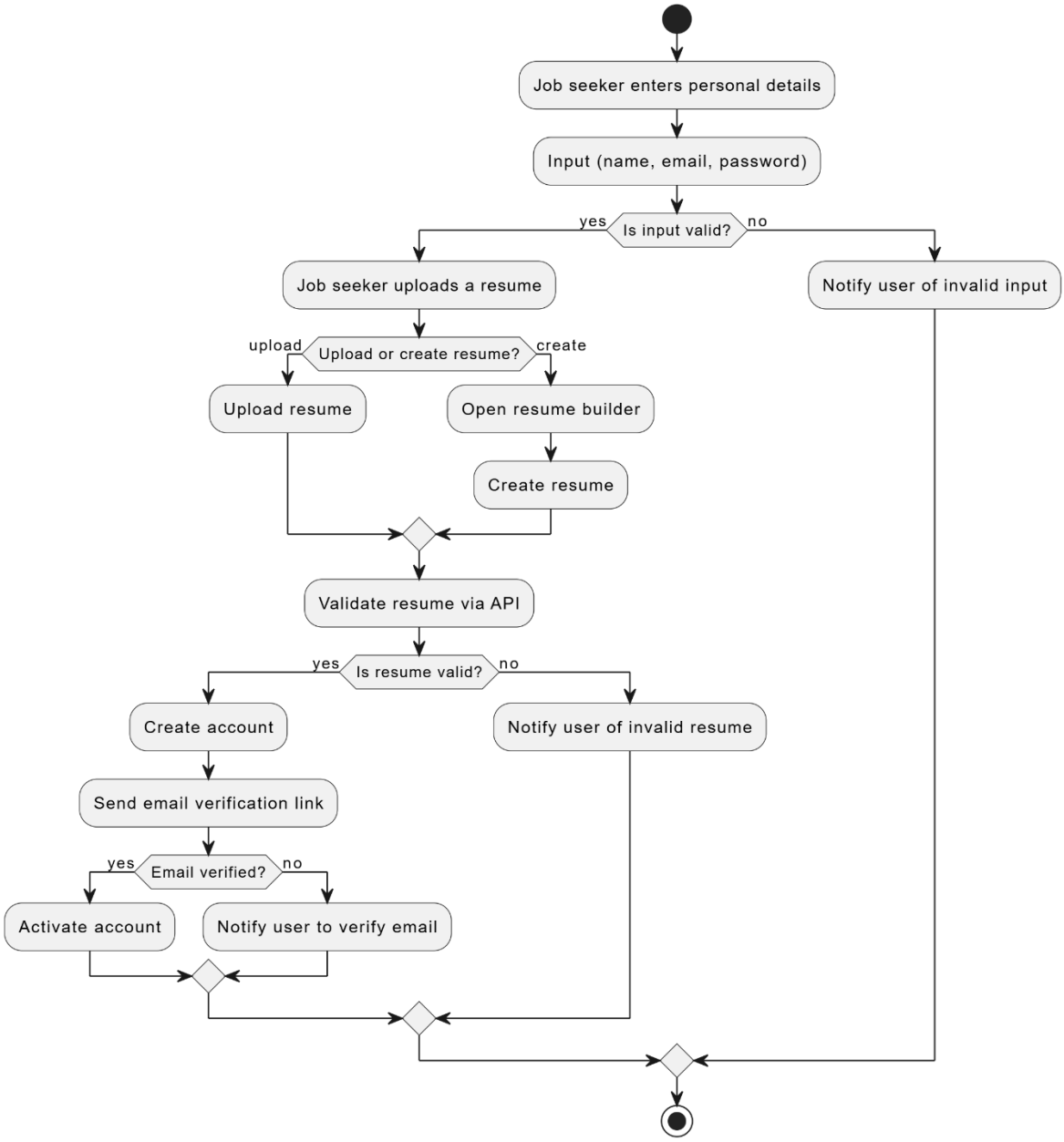

Figure 4.3.1 Job Seeker Registration Activity Diagram

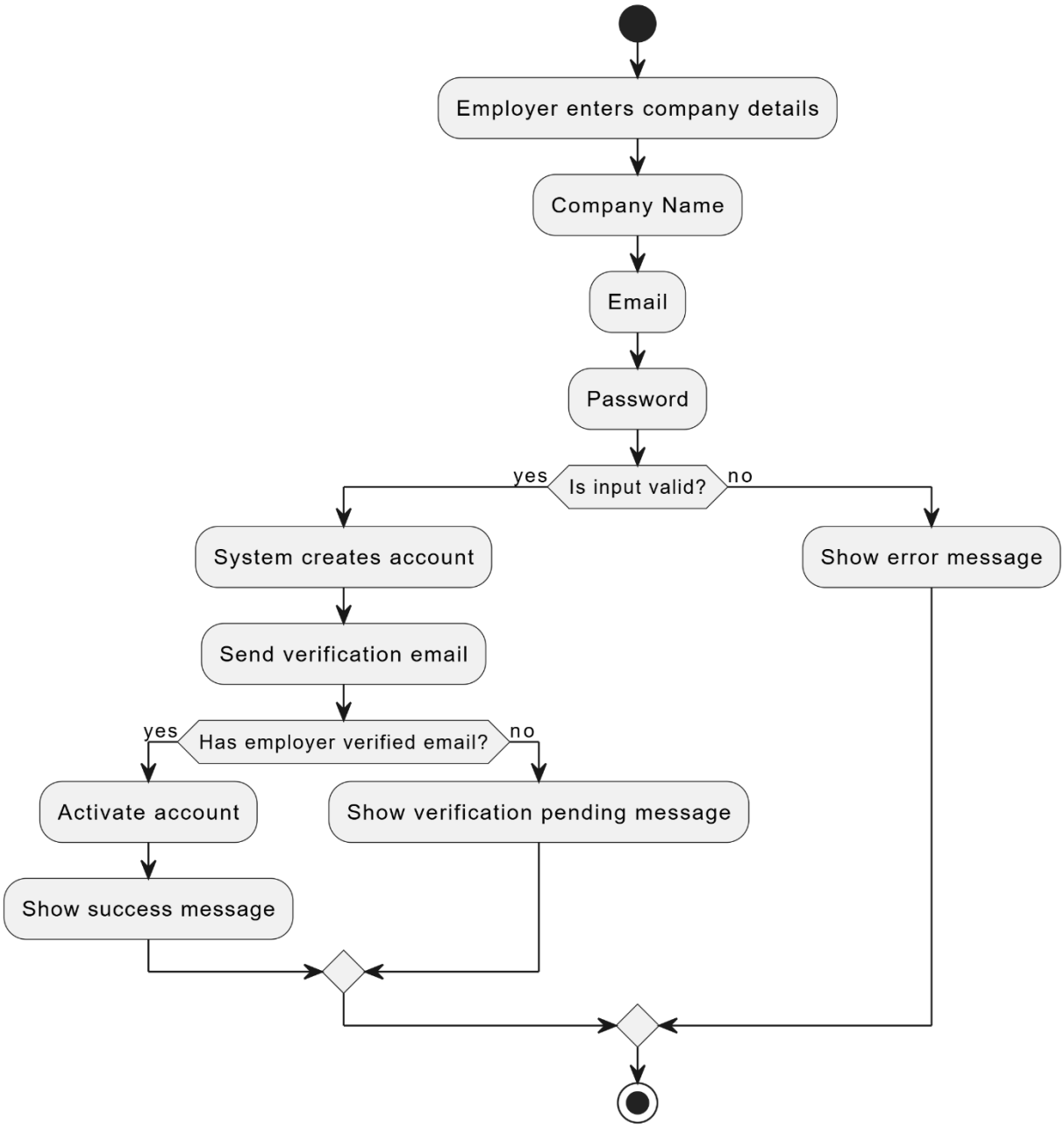

Figure 4.4.2 Register (Employer) Activity Diagram

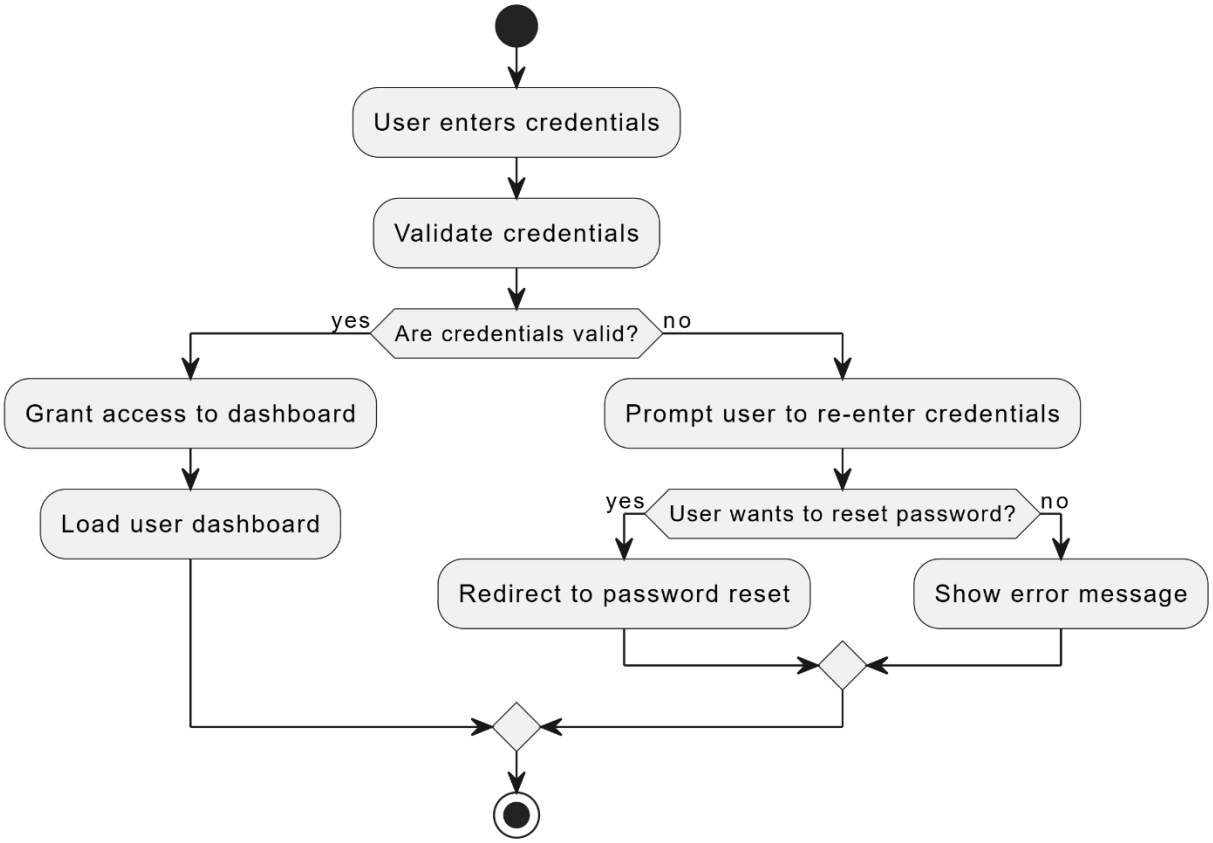

Figure 4.4.3 Login Activity Diagram

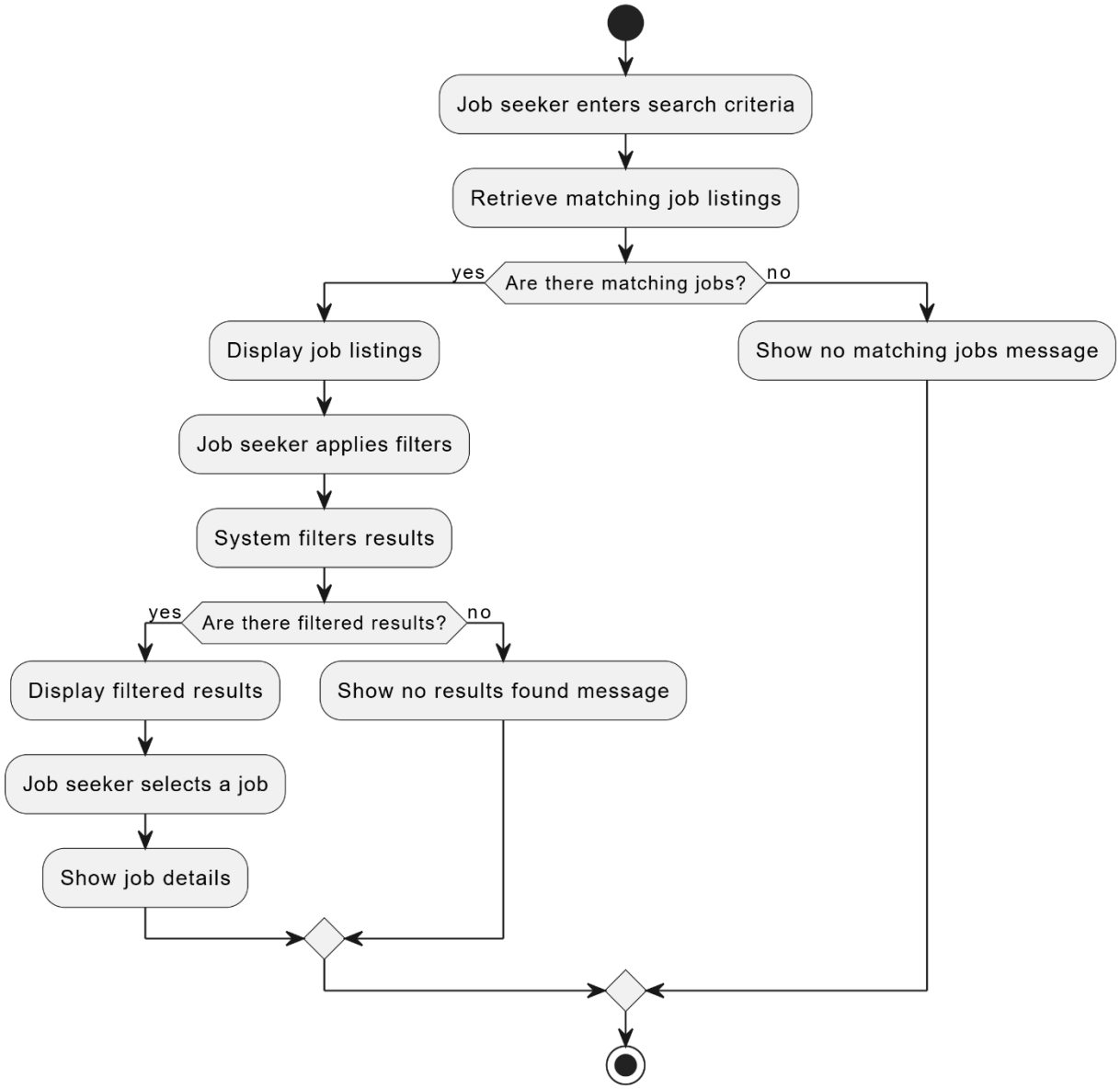

Figure 4.4.4 Job Search Activity Diagram

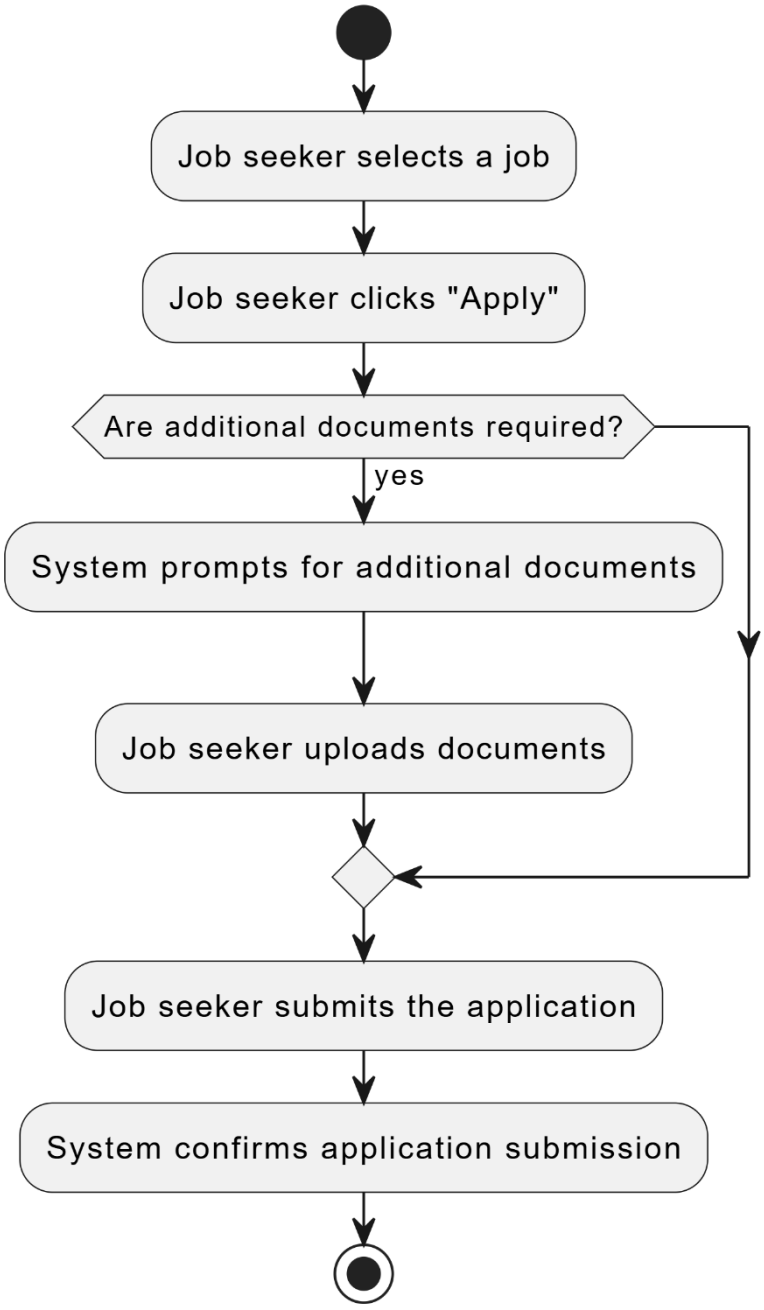

Figure 4.4.5 Apply for a Job (Activity Diagram)

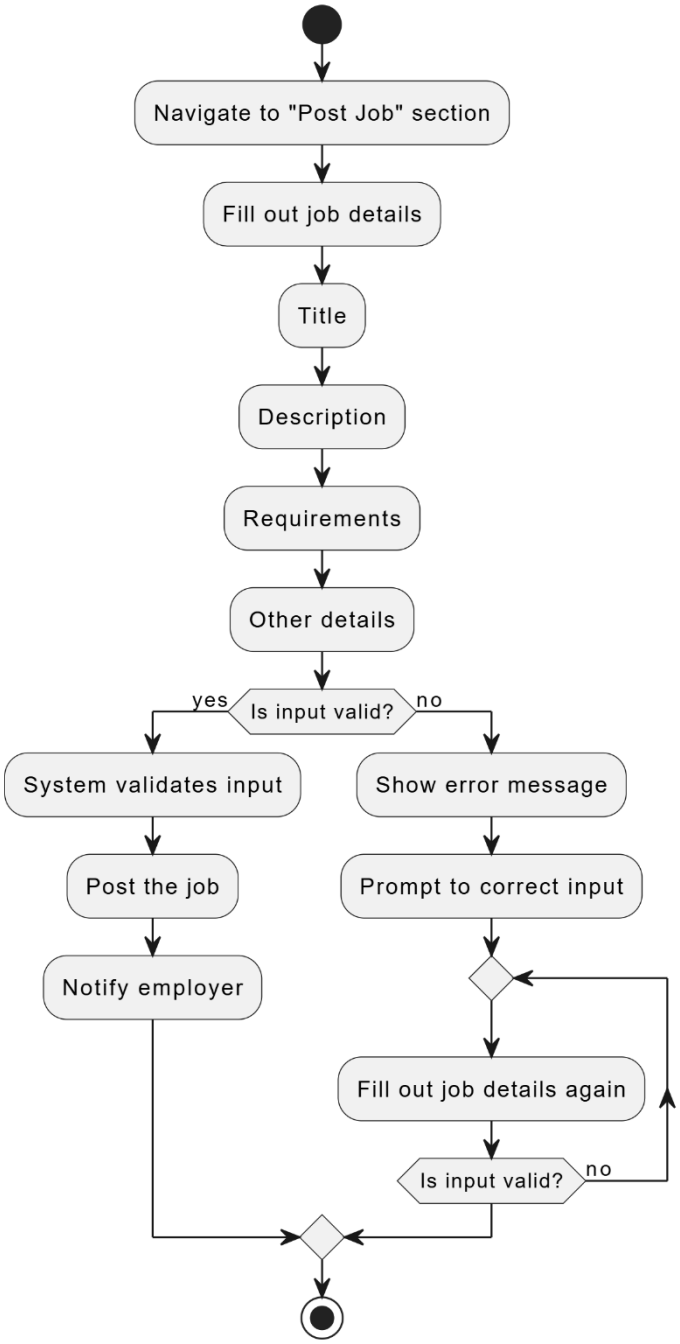

Figure 4.4.6 Posting Jobs (Employer) - Activity Diagram

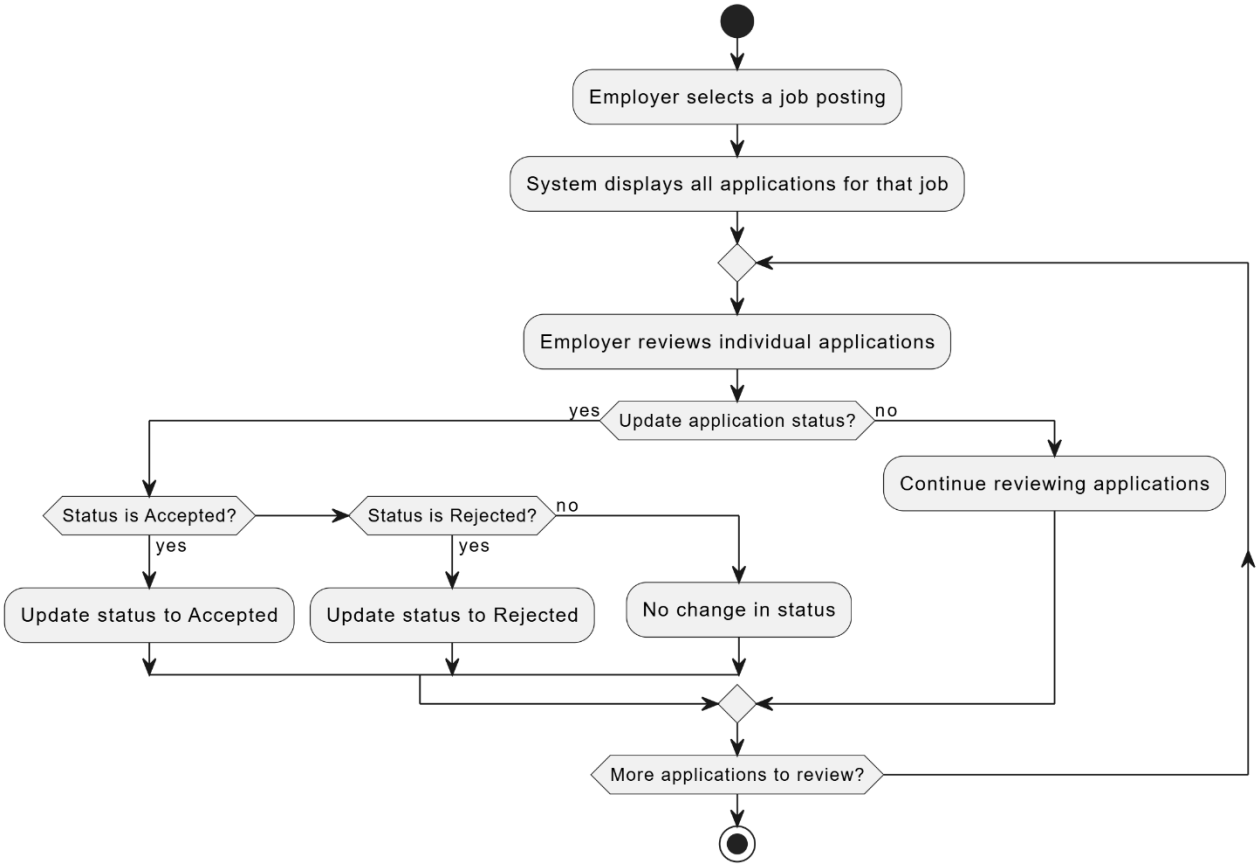

Figure 4.4.7 Review Applications (Employer) - Activity Diagram

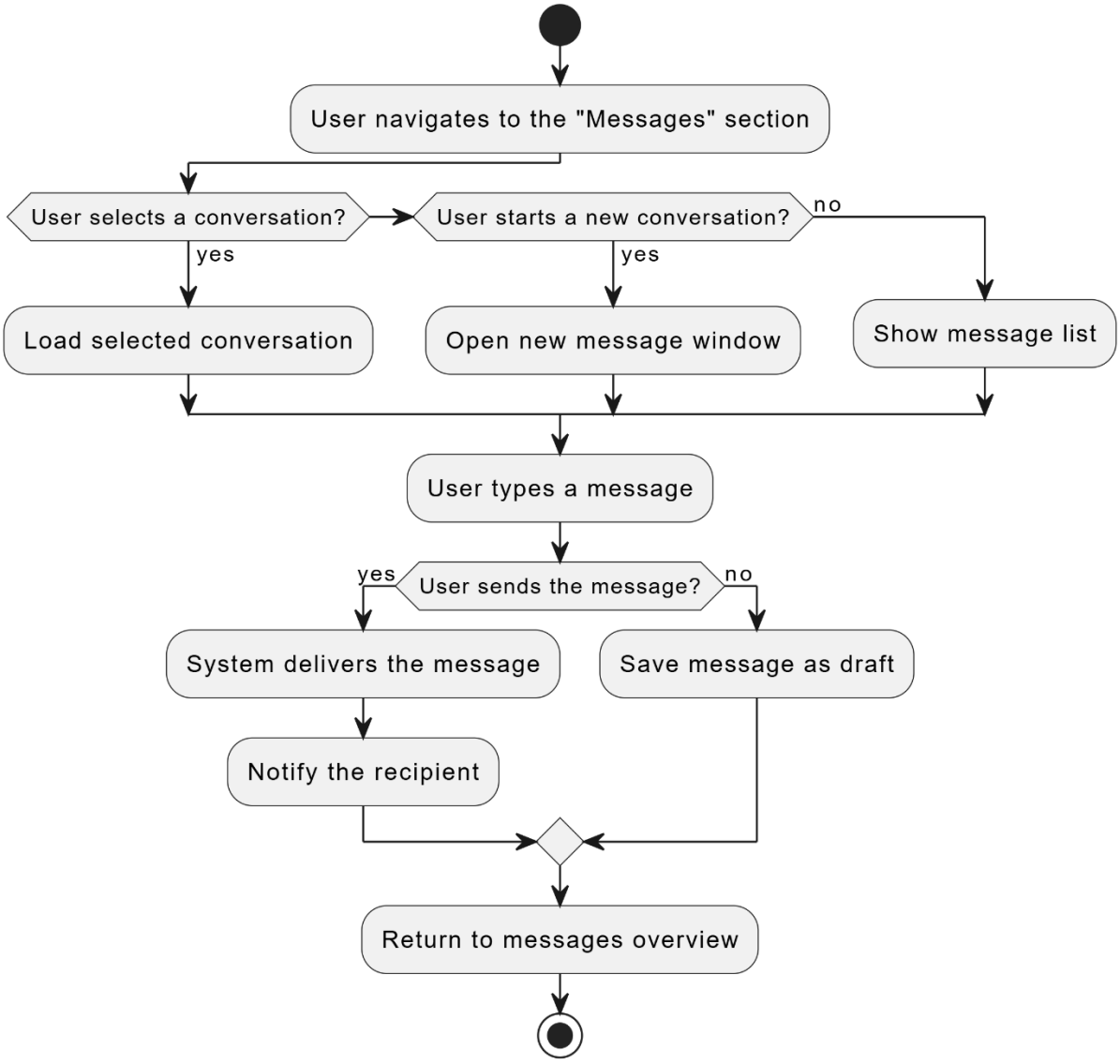

Figure 4.4.8 Messaging Activity Diagram

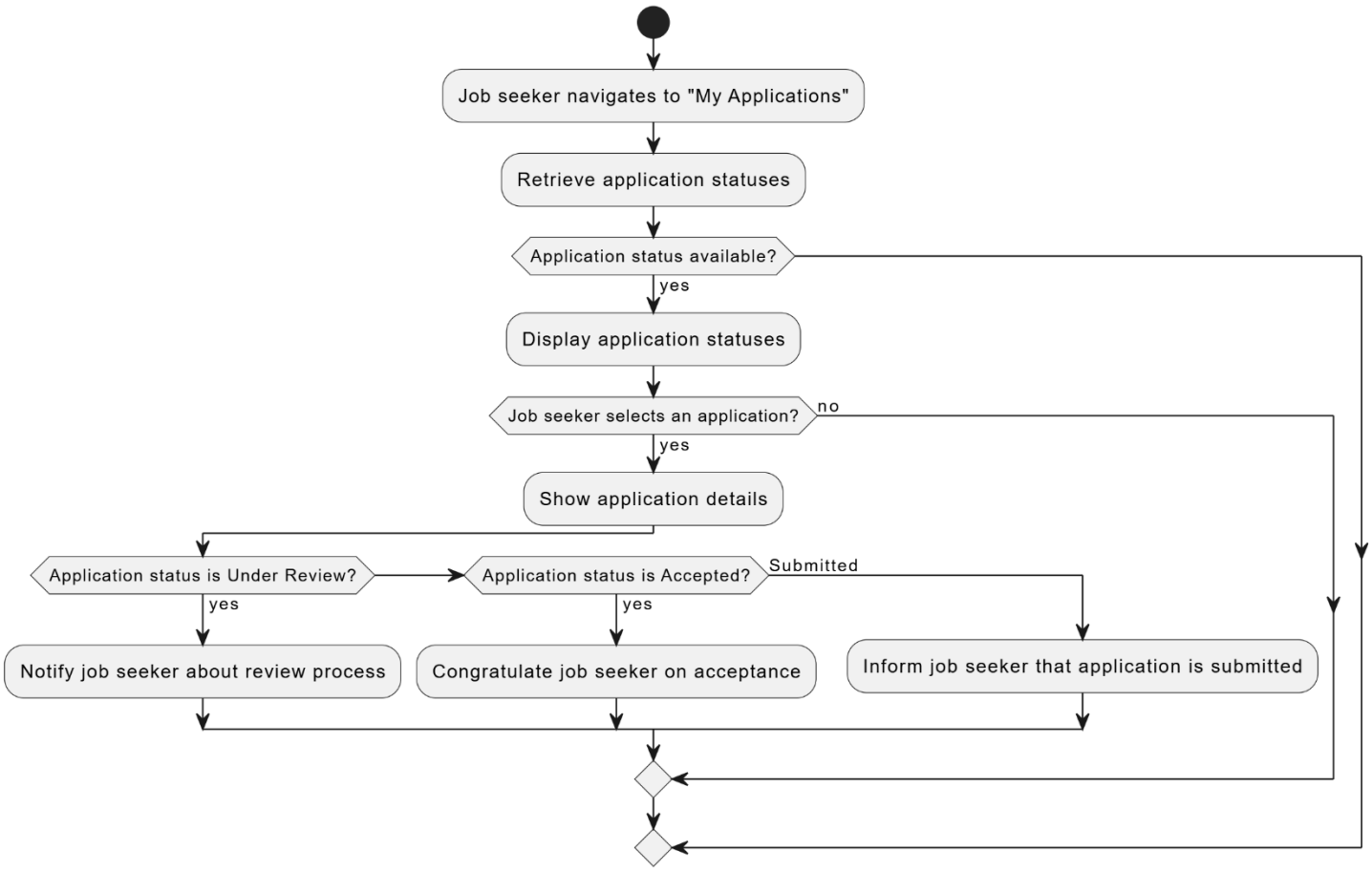

Figure 4.4.9 Track Application Status (Job Seeker)

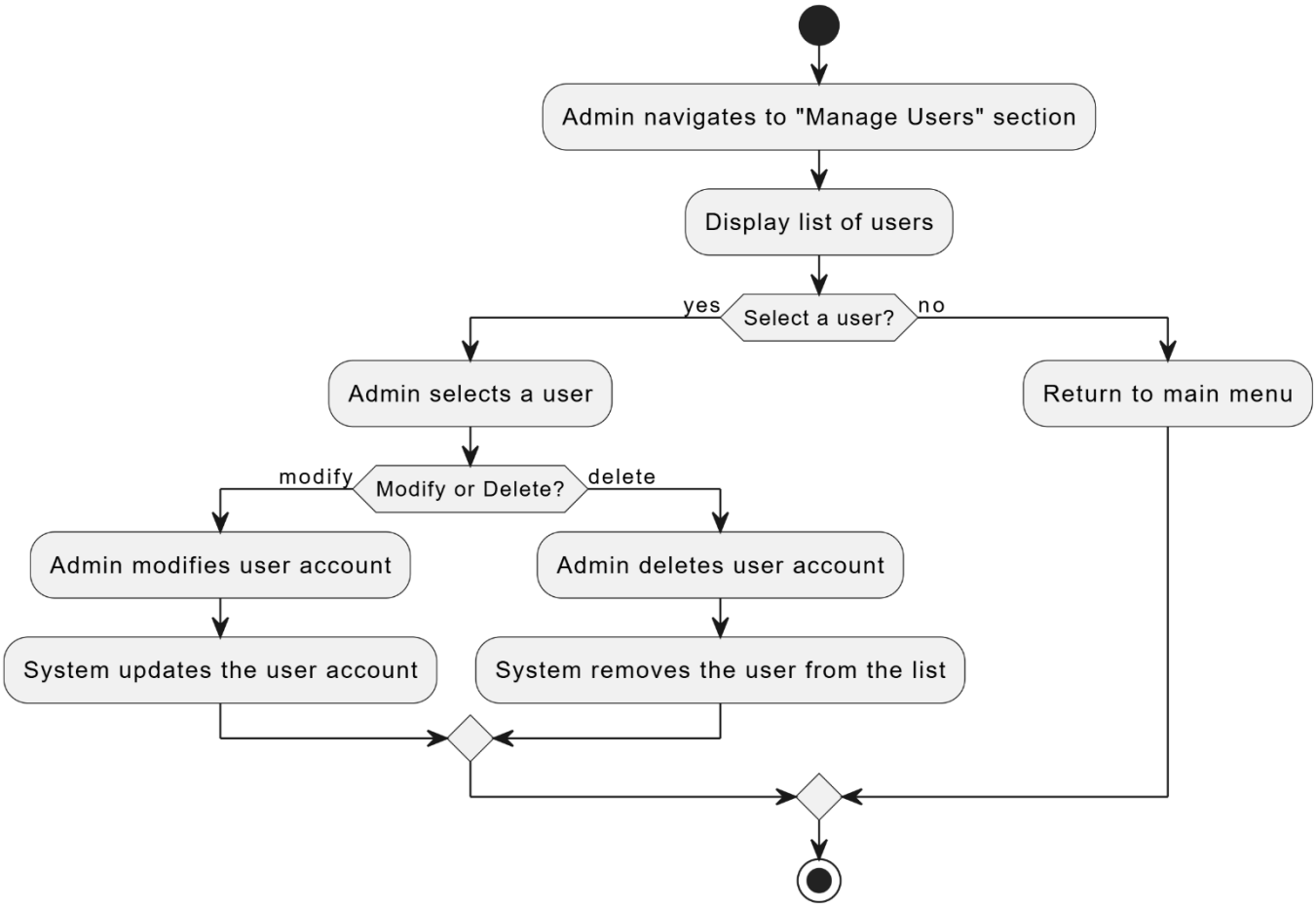

Figure 4.4.10 Manage Users (Admin)

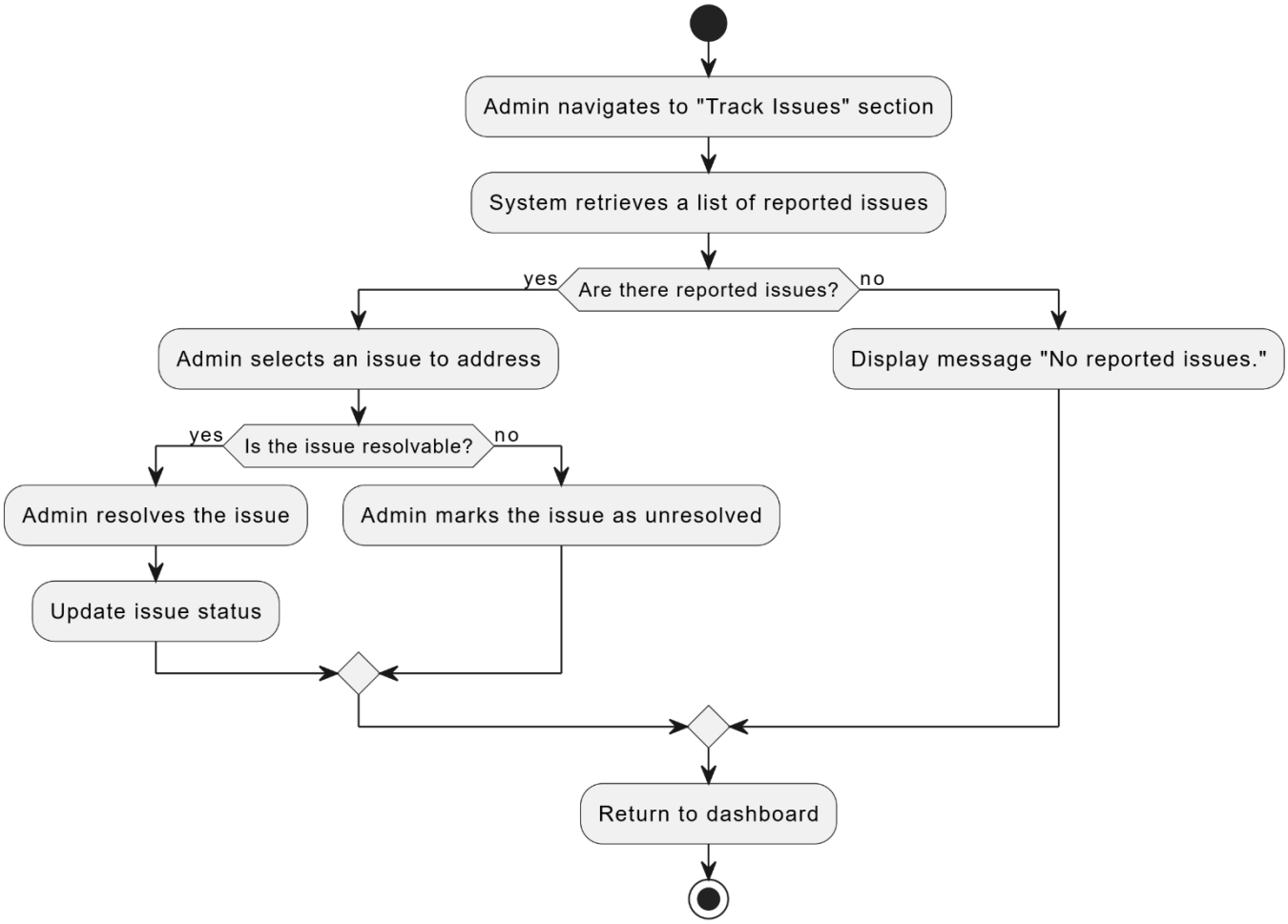

Figure 4.4.11 Track Issues (Admin)

4.5 Class Diagram

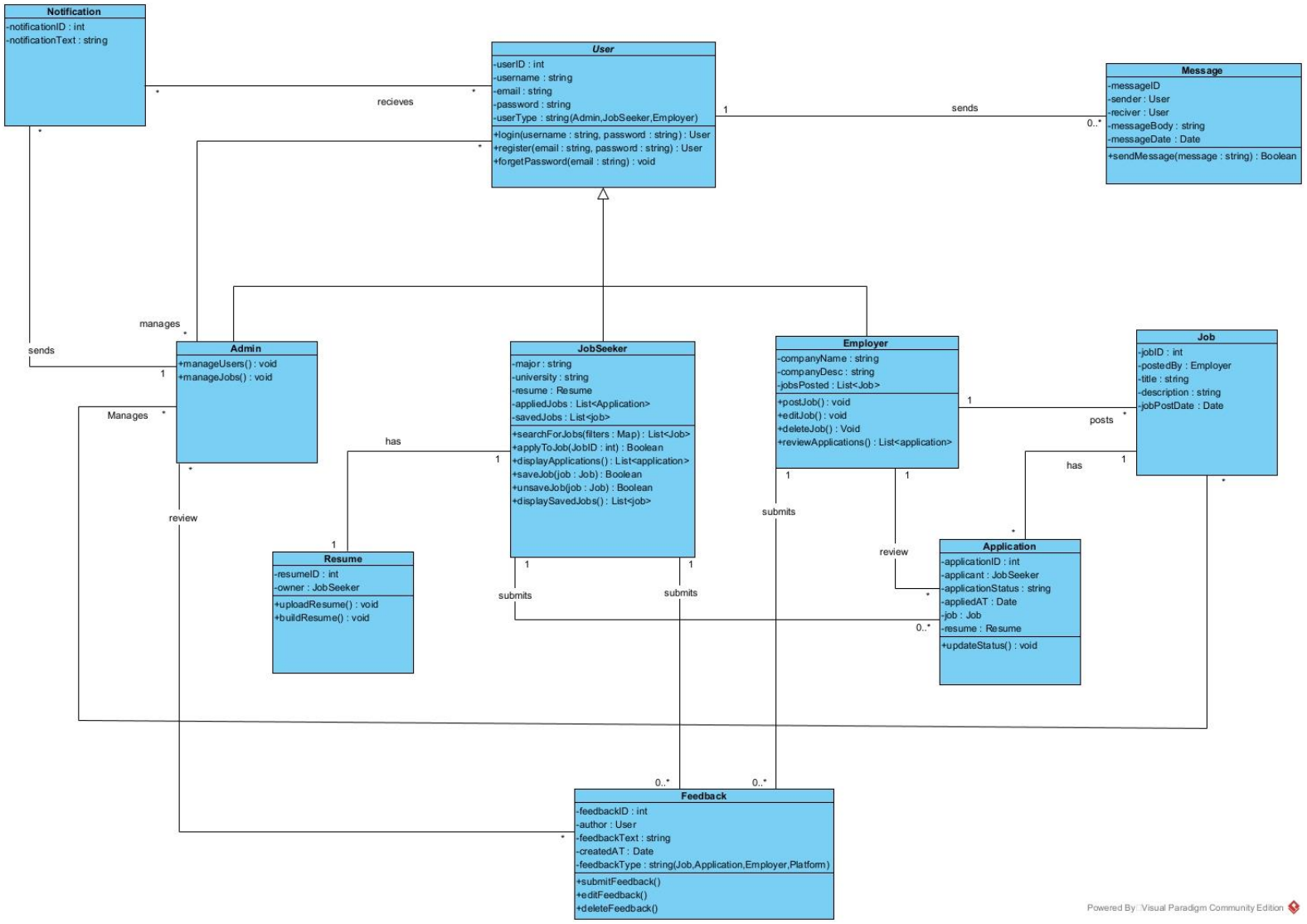

Figure 4.5.1 Class Diagram

4.6 Entity Relationship Diagram

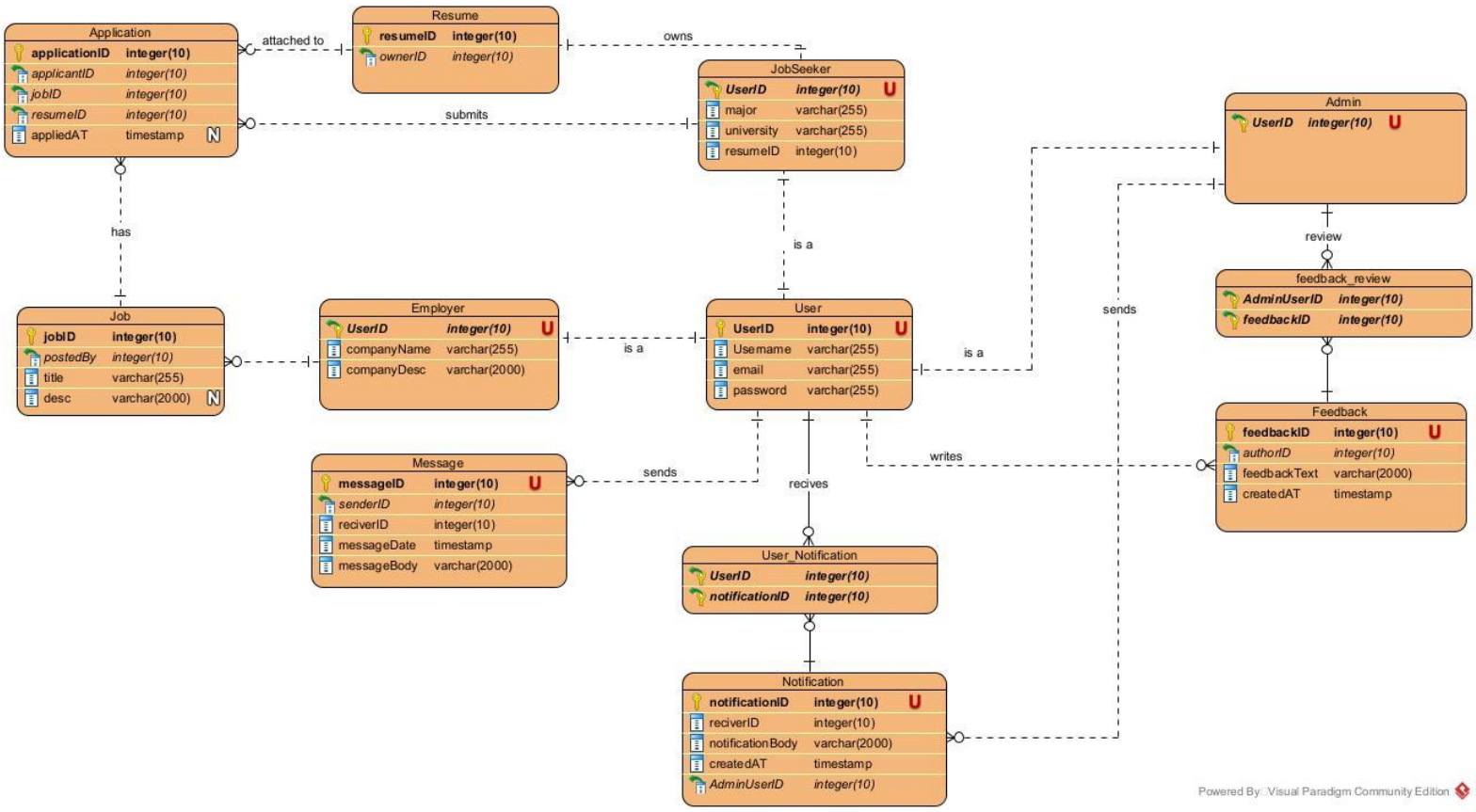

Figure 4.6.1 Entity Relationship Diagram