The Private Higher School of Engineering and Technology

**ESPRIT**

TWIN : Web and Internet Technology

**Sprint 0 Report**

# Optimizing the Recruitment Process and HR Department

*Members : Fantastic five*

Mariem HABOURIA

Aziz SALMI

Iyed BEN FARHAT

Yassin NOURI

Hamouda Chekir

*Supervised By*

Ms. Badiaa Bouhdid

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Project Introduction

## 1.1 Background and Motivation

The recruitment process has become increasingly complex and data-driven. Traditional hiring methods are inefficient, leading to longer hiring times and poor candidate experiences. This project seeks to address these challenges by providing an intelligent Applicant Tracking System (ATS) that automates job postings, application management, interview scheduling, and candidate evaluation. By leveraging automation, AI, and real-time analytics, the system will improve the efficiency of recruitment operations, enhance decision-making, and provide a seamless experience for candidates, recruiters, and HR teams.

## 1.2 Study of the Existing Systems

Existing recruitment platforms in Tunisia, such as Tanitjobs, LinkedIn, and Keejob, offer various features to assist in job searches and hiring processes. However, each platform has its strengths and weaknesses. Below is a comparative analysis of these systems:

| **Feature** | **Tanitjobs** | **LinkedIn** | **Keejob** |
|---|---|---|---|
| **Customization** | Limited | Moderate | Limited |
| **Scalability** | Moderate | High | Moderate |
| **Integration Capabilities** | Basic | Extensive | Basic |
| **Cost** | Moderate | High | Low |
| **AI-Powered Features** | Basic filtering | Advanced | Keyword search |
| **User Interface** | Standard | User-Friendly | Standard |
| **GDPR Compliance** | Limited | Yes | Limited |
| **Candidate Evaluation** | Manual | Partially Automated | Manual |

Table 1.1: Comparative Analysis of Existing Recruitment Platforms in Tunisia

This table highlights the strengths and weaknesses of existing recruitment platforms in Tunisia.

## 1.3 Problem Statement

Despite the availability of several recruitment platforms in Tunisia, many existing solutions fail to meet the evolving needs of modern recruitment processes. Platforms like Tanitjobs, LinkedIn, and Keejob offer valuable features but often suffer from limitations such as lack of customization, scalability issues, and limited AI-driven automation. Therefore, there is a pressing need for a more adaptable, cost-effective, and intelligent ATS that can seamlessly integrate with third-party tools, offer advanced AI features, and improve the experience for both HR teams and candidates.

## 1.4 Proposed Solution

Next Hire is an AI-powered, scalable Applicant Tracking System (ATS) platform that automates key recruitment tasks, including job postings, application tracking, resume screening, and interview scheduling. The system leverages Natural Language Processing (NLP) for intelligent job matching and predictive analytics for informed decision-making. By integrating real-time recruitment analytics and reporting tools, HR teams can optimize their workflows and improve hiring outcomes. The solution is designed to be customizable, cost-effective, and fully compliant with GDPR and data privacy regulations, ensuring a secure and seamless experience for all users.

Below is the Next Hire logo:

Figure 1.1: Next Hire logo

Below is a comparative analysis of the proposed solution against existing platforms:

This solution aims to offer more customization, scalability, and advanced AI-powered features at a lower cost.

| **Feature** | **Existing Solutions** | **Our Solution** |
|---|---|---|
| **Customization** | Limited to Moderate | High |
| **Scalability** | Moderate to High | Very High |
| **Integration Capabilities** | Basic to Extensive | Extensive with third-party APIs |
| **Cost** | Moderate to High | Low |
| **AI-Powered Features** | Basic | Advanced NLP, AI screening |
| **User Interface** | Standard to User-Friendly | User-Friendly, Intuitive |
| **GDPR Compliance** | Limited to Yes | Yes |
| **Candidate Evaluation** | Manual to Partially Automated | Fully AI-driven evaluation |

Table 1.2: Comparative Analysis of Existing Recruitment Platforms and the Proposed System

## 1.5 Project Objective

This project aims to enhance the recruitment process by introducing AI-driven automation and optimization. By automating key tasks and providing intelligent insights, the system will improve the efficiency of recruitment workflows, enhance candidate experience, and support HR professionals with data-driven decision-making. The overall vision is to create a seamless, scalable, and innovative solution that addresses the evolving needs of the recruitment industry.
Key objectives include:

- Streamlining job posting and application tracking processes.
- Automating resume screening, interview scheduling, and candidate evaluation.
- Providing insightful recruitment analytics and reporting tools to optimize decision-making.
- Ensuring GDPR compliance and data security to protect user privacy.
- Enhancing the interview process with an AI-driven agent for real-time interaction.

This system integrates cutting-edge technologies to provide a seamless, user-friendly experience for all stakeholders in the recruitment process.

# Chapter 2

# Business Features Specifications

## 2.1 Project Specification and Analysis

### 2.1.1 Actors

The main actors of the system are as follows:

- **Admin:** Manages system settings, user accounts, and permissions.
- **HR Manager:** Handles job postings, application reviews, and interview scheduling.
- **Candidate:** Submits job applications, tracks application status, and participates in interviews.
- **Visitor:** Browses job offers and creates an account to become a candidate.

### 2.1.2 Functional Requirements

The functional requirements are categorized as follows:

#### User Management

- Allow candidates to create and manage profiles, including uploading resumes.
- Enable role-based access control for Admin, HR Manager, and Candidate roles.
- Provide options for updating or deleting user accounts.
- Manage user permissions and system access for Admins.

#### Job Management

- Post and manage job openings with detailed specifications.
- Integrate job postings with external platforms (e.g., LinkedIn, Glassdoor).
- Automatically remove or archive expired job postings.
- Handle automatic job posting across multiple platforms.

#### Application Management

- Receive and store job applications.
- Enable candidates to track the status of their applications.
- Provide notifications for application updates via email and SMS.
- Allow HR Managers to review, accept, or reject applications.
- Automate job application screening and ranking.

#### Interview Management

- Schedule interviews with automated calendar integration.
- Notify candidates and HR teams of scheduled interviews.
- Enable interviewers to share feedback and evaluation results.
- Provide the possibility to create a streaming access within the platform, either through an integrated custom solution or by leveraging external services.
- Automate scheduling processes.
- Integrate an AI agent to: Analyze the behavior of candidates, Assist with generating relevant interview questions and to Provide analytical insights on interviews.

#### Recruitment Analytics Management

- Generate recruitment analytics reports (e.g., time-to-hire, application-to-hire ratio).
- Provide dashboard widgets for key hiring metrics (e.g., job openings, application volume).
- Offer insights into candidate performance and assessment outcomes.

### Non-Functional Requirements

- **Performance:** The system should handle concurrent requests efficiently
- **Security:** Implement role-based access and data encryption.
- **Usability:** Provide an intuitive interface with responsive design.
- **Maintainability:** Ensure modular architecture, proper documentation, and automated testing.

## 2.2 User Stories (Product Backlog)

| **Epic** | **User Stories** |
|---|---|
| Automate HR Tasks | • As an HR Manager, I want to automate job postings across platforms so that I can save time.<br>• As an Admin, I want AI-driven resume screening so that I can identify top candidates quickly.<br>• As an HR Manager, I want automated interview scheduling so that the process is more efficient. |
| Enhance Candidate Experience | • As a Candidate, I want a user-friendly application portal so that I can apply for jobs easily.<br>• As a Candidate, I want to receive notifications about my application status so that I stay informed.<br>• As a Candidate, I want an AI chatbot to answer my queries instantly. |
| Data-Driven Decision Making | • As an HR Manager, I want a recruitment analytics dashboard to monitor hiring metrics effectively.<br>• As an Admin, I want candidate assessment reports to make better hiring decisions. |
| Improve HR Communication and Collaboration | • As an HR Manager, I want a centralized communication hub to coordinate with recruiters and candidates.<br>• As an HR Manager, I want a structured feedback system for interview evaluations to improve transparency. |
| Ensure Compliance and Security | • As an Admin, I want GDPR-compliant data handling to ensure privacy regulations are met.<br>• As an Admin, I want role-based access control to secure sensitive data. |

Table 2.1: Product Backlog

### 2.2.1 Deliverables

- Fully implemented Applicant Tracking System (ATS).
- HR dashboard with recruitment analytics.
- Training materials for HR teams.
- Optimized HR workflows documentation.
- Reports on recruitment efficiency and candidate satisfaction.

# Chapter 3

# System Design and Architecture

## 3.1 Overview

This project involves the development of a comprehensive full-stack JavaScript recruitment platform designed to streamline and enhance the recruitment process for companies and job seekers. The platform is built using modern technologies such as React.js, Express.js, GraphQL, and MongoDB, ensuring a responsive and scalable solution.

## 3.2 Architecture Overview

### 3.2.1 A. Physical Architecture

The physical architecture of the application consists of the following components:

- **Web Server:** Hosts the web application and manages incoming user requests.
- **Application Server:** Processes incoming requests and sends responses to the web server.
- **Database Server:** Stores all application data and manages data access and storage.

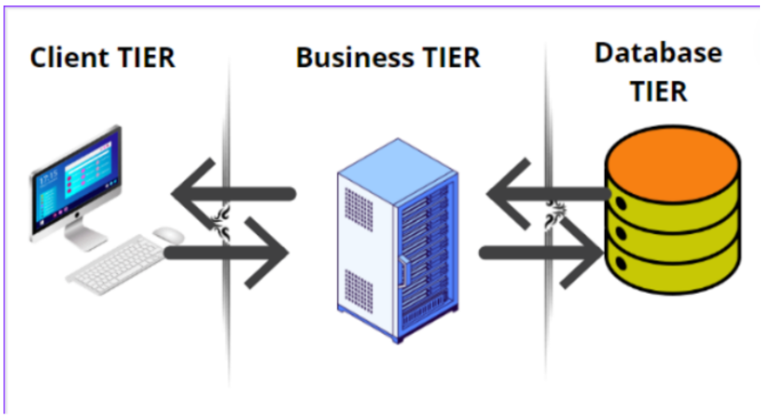

Figure 3.1: Physical Architecture of the System

### 3.2.2 B. Logical Architecture

The logical architecture of the application consists of the following components:

- **User Interface:** The front-end of the application, where users interact with the system.
- **Application Server:** Manages the application logic and processes user requests.
- **Database Server:** Stores all application data, including user profiles, project details, and transaction history.

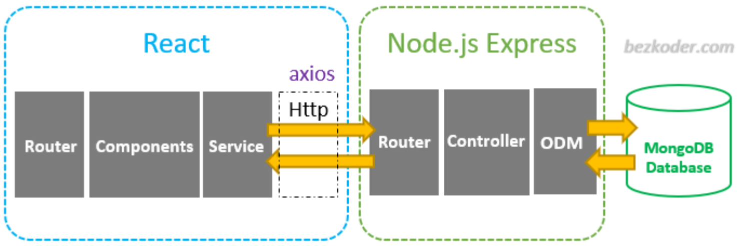

Figure 3.2: Logical Architecture of the System

## 3.3 Data Model

### 3.3.1 Global Use Case Diagram

The global use case diagram illustrates the interactions between the actors and the system's main functionalities.

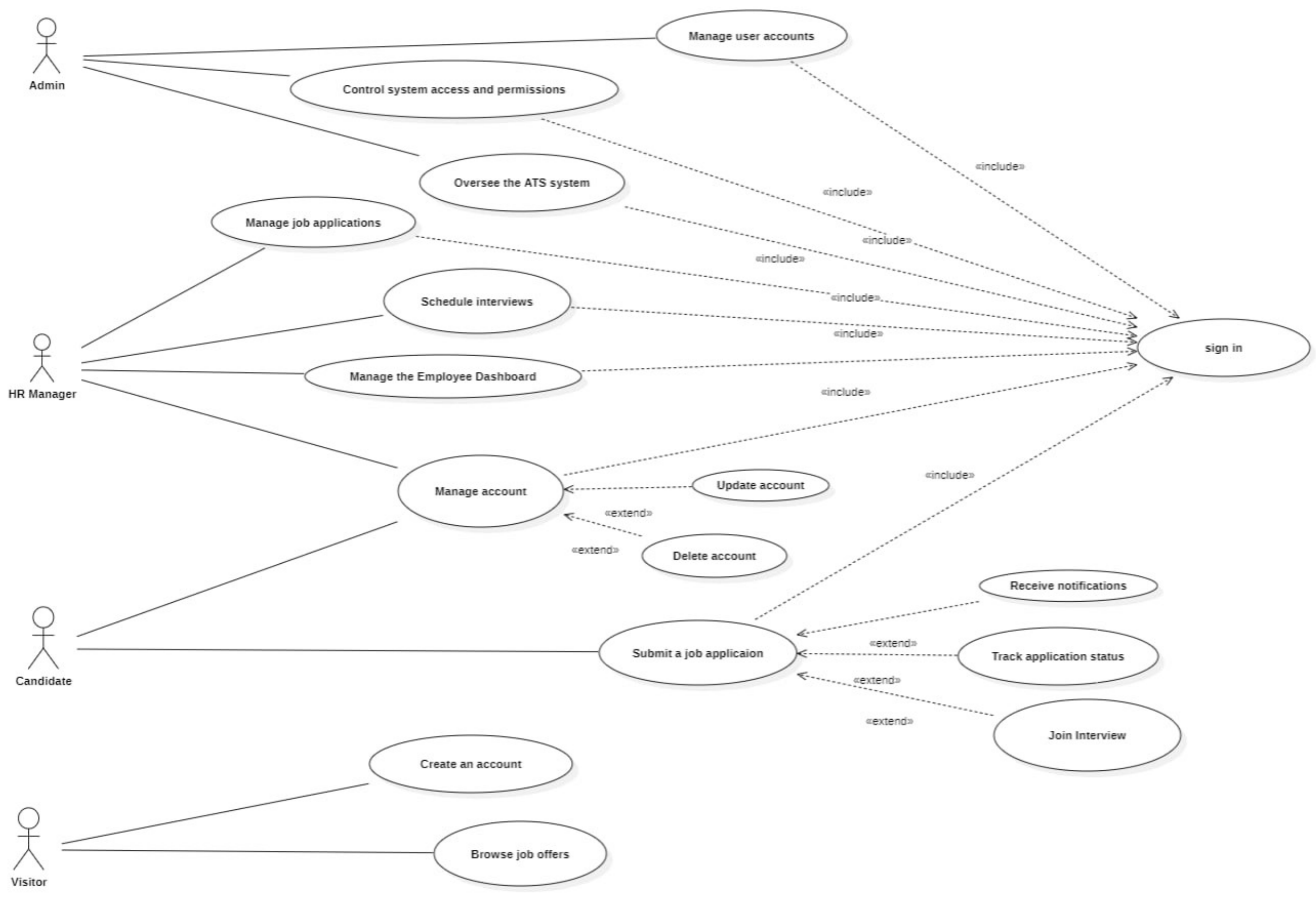

Figure 3.3: Global Use Case Diagram

### 3.3.2 Data Model

The data model outlines the structure of the system using a NoSQL approach, specifically MongoDB. It focuses on collections, documents, and their relationships.

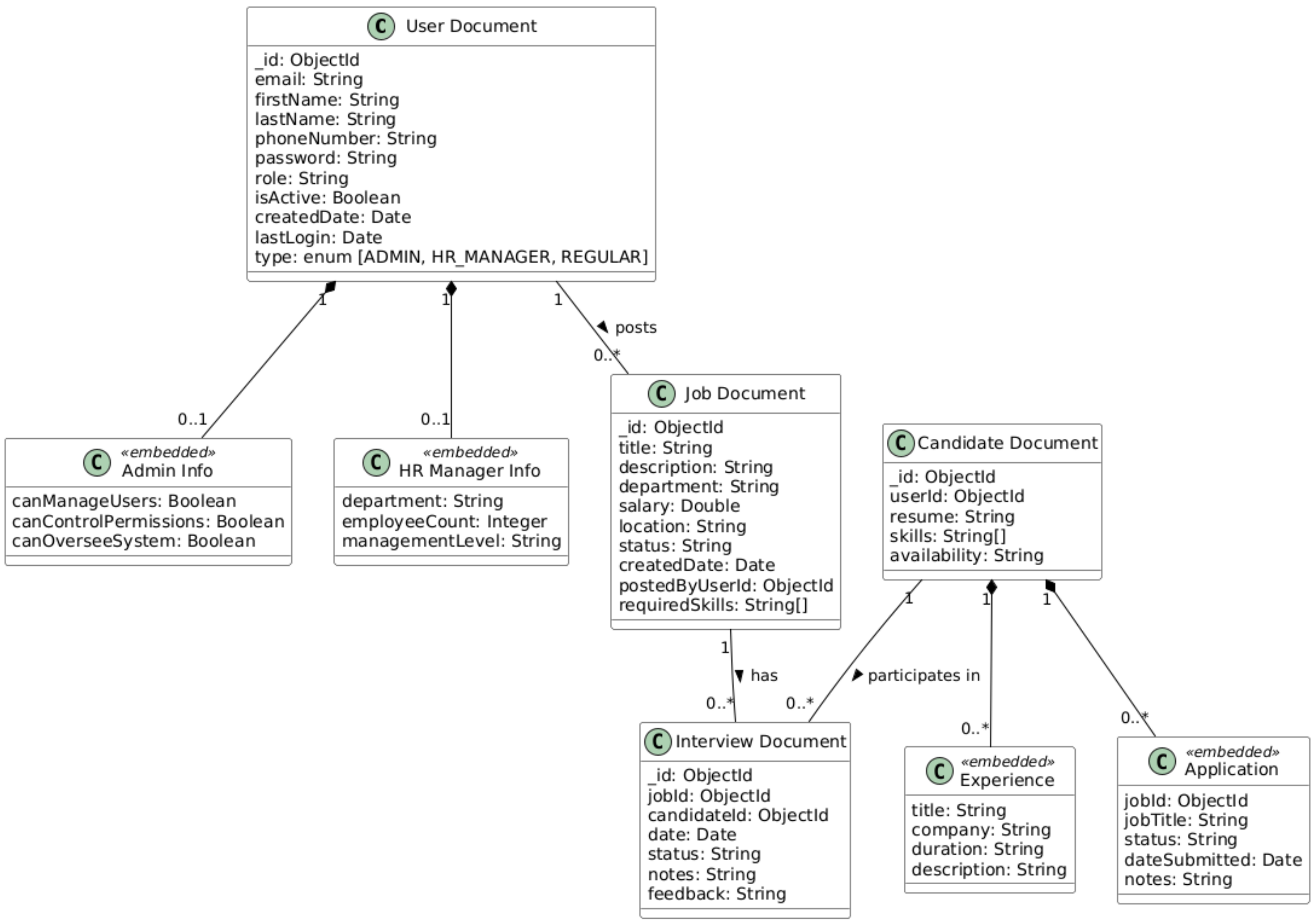

Figure 3.4: Data Model (NoSQL)

## 3.4 Technology Stack choice

### 3.4.1 Frontend

The frontend of the application is built using modern JavaScript technologies to ensure a fast, interactive, and responsive user experience.

- **React.js** - A powerful JavaScript library for building dynamic and interactive user interfaces.

Figure 3.5: React.js Logo

### 3.4.2 Backend

The backend is designed to be scalable, efficient, and capable of handling complex business logic. It integrates both JavaScript and Python technologies.

- **Python** - Used for AI-driven functionalities and API microservices.

Figure 3.6: Python Logo

- **Express.js** - A lightweight and flexible Node.js framework for building RESTful APIs.

Figure 3.7: Express.js Logo

- **GraphQL** - A query language for APIs that enables efficient data fetching.

Figure 3.8: GraphQL Logo

- **REST API / Swagger** - Provides API documentation and interactive testing.

Figure 3.9: Swagger Logo

### 3.4.3 Database

A NoSQL database is used to store and manage application data efficiently.

- **MongoDB** - A document-based NoSQL database that ensures high performance and scalability.

Figure 3.10: MongoDB Logo

### 3.4.4 Version Control and CI/CD Tools

The project follows DevOps best practices by integrating automated deployment, testing, and monitoring tools.

- **Git** - Distributed version control system for collaboration and source code management.

Figure 3.11: Git Logo

- **Docker** - Containerization for consistent deployment across environments.

Figure 3.12: Docker Logo

- **Jenkins / GitHub Actions** - CI/CD pipelines for automated testing and deployment.

Figure 3.13: Jenkins Logo

- **Kubernetes** - Orchestration for managing containerized applications.

Figure 3.14: Kubernetes Logo

- **Grafana** - Monitoring and logging for performance tracking.

Figure 3.15: Grafana Logo

# Chapter 4

# CI/CD and DevOps Strategy

## 4.1 Version Control Plan

### 4.1.1 Branching Strategy

We adopt the **Git Flow** branching model:

- **main**: Production-ready code.
- **develop**: Latest stable development version.
- **feature/{feature-name}**: Feature branches, merged into *develop*.
- **release/{version}**: Staging/testing before release.
- **hotfix/{fix-name}**: Critical fixes applied directly to *main*.

### 4.1.2 Repository Management

- Code is hosted on **GitHub**.
- Enforced **protected branches** for *main* and *develop*.
- **Pull Request (PR) Workflow**: Includes code review and automated CI checks before merging.
- **Git** is used for version control, ensuring collaboration and efficient tracking of changes.

## 4.2 CI/CD Pipeline Setup Plan

### 4.2.1 Continuous Integration (CI)

- **Jenkins** is used as the Continuous Integration server to automate the build process.
- **GitHub Actions** automates the workflow for testing and deployment.
- **JUnit** for unit testing Java-based applications.
- **SonarQube** for code quality and static analysis.

- **Linting, static code analysis, unit and integration tests**.
- **Docker-based builds** for deployment consistency.

### 4.2.2 Continuous Deployment (CD)

- Staging and production deployment through **Kubernetes (K8s)**.
- Deployment managed using **Jenkins**.
- Artifacts and deliverables managed using **Nexus Repository Manager**.
- **Docker Compose** and **Docker Volumes** for advanced container management.
- Canary releases for safer rollouts.

### 4.2.3 Infrastructure as Code (IaC)

- Automated infrastructure management using **Terraform** or **Ansible**.
- Defined environments: *Development, Staging, Production.*
- **Docker** is used for containerization, ensuring portability across different environments.

### 4.2.4 Monitoring and Logging

- System monitoring using **Grafana** and **Prometheus**.
- Real-time logging and alerting to detect system failures and anomalies.

## 4.3 Testing Strategy

- **Unit Testing**: **JUnit** (for Java-based components), Jest, Mocha (Frontend/Backend).
- **Integration Testing**: Cypress, Postman (API Testing).
- **End-to-End (E2E) Testing**: Selenium, Cypress.
- **Load Testing**: JMeter.
- **Security Testing**: OWASP ZAP (Vulnerability Scanning).
- Automated testing integrated into the CI/CD pipeline for continuous validation.

# Chapter 5

# Sprint Planning Timeline

## 5.1 Sprint Overview

The project will be developed over a total of **14 weeks**, divided into **5 sprints**:

- **Sprint 0 (Weeks 1-4)**: Project setup, CI/CD implementation, and infrastructure preparation.
- **Sprint 1 (Weeks 5-7)**: Core functionalities – authentication and user management.
- **Sprint 2 (Weeks 8-9)**: Job management and application processing.
- **Sprint 3 (Weeks 10-11)**: AI-powered recruitment and interview scheduling.
- **Sprint 4 (Weeks 12-14)**: Analytics, optimization, and final deployment.

## 5.2 Sprint Breakdown

### 5.2.1 Sprint 0: Project Initialization and Setup (Weeks 1-4)

- Define **product backlog** and key milestones.
- Establish **CI/CD pipeline**, repository structure, and cloud infrastructure.
- Set up **version control** with GitFlow.

### 5.2.2 Sprint 1: Core System and User Management (Weeks 5-7)

- Implement **user authentication** (role-based access control).
- Develop **Admin, HR Manager, Candidate, and Visitor** roles.
- Implement **profile management and settings**.
- Develop API endpoints for **user-related operations**.
- Perform **unit and integration tests** for authentication and user flows.

### 5.2.3 Sprint 2: Job Management and Application Processing (Weeks 8-9)

- Implement **job posting and application tracking** system.
- Automate **resume parsing** using NLP.
- Enable job recommendations based on candidate profiles.
- Integrate **notification system** (email/SMS alerts).
- Conduct **API integration** with external job platforms (LinkedIn, TanitJob).

### 5.2.4 Sprint 3: AI-Powered Recruitment and Interview Scheduling (Weeks 10-11)

- Develop **automated interview scheduling** with calendar integration.
- Implement AI-driven **candidate evaluation and behavior analysis**.
- Integrate **chatbot assistant** for candidate support.
- Optimize **security measures** (role-based access, data encryption).
- Conduct **end-to-end testing** for application processing and interviews.

### 5.2.5 Sprint 4: Analytics, Performance Optimization, and Deployment (Weeks 12-14)

- Implement **recruitment analytics dashboard**.
- Optimize system performance and conduct **load testing**.
- Finalize **security audits and penetration testing**.
- Deploy application to **staging and production** environments.
- Prepare **technical documentation and user training materials**.
- Conduct final **user acceptance testing (UAT)** and bug fixes.