Implementation and Test Document (ITD)

Students & Companies (S&C) Platform

Shaurya Aditya Singh, Ergun Gani Çalışkan, Mohammadali Khaledi

Academic Year: 2024-2025

1 Introduction

1.1 Purpose

The StudentCompanies (S&C) platform is a web-based application designed to streamline and improve the process of connecting students seeking internships with companies offering them. SC leverages technology, including a sophisticated recommendation system, to make internship matching more efficient, personalized, and transparent. The platform aims to solve the following problems: Difficulty for students in finding relevant internships: Students often spend a significant amount of time searching through numerous job boards and company websites to find suitable internship opportunities. Lack of personalization: Traditional internship search methods often do not take into account a student's specific skills, interests, and academic background, resulting in irrelevant matches. Inefficient application processes: Applying for internships can be time-consuming, involving multiple platforms and inconsistent communication channels. Limited feedback and transparency: Students often receive little feedback on their applications, making it difficult to understand areas for improvement.

1.2 Scope

The S&C platform will include the following features and functionalities: Student Profiles: Students can create comprehensive profiles, including their academic background, skills, interests, and experience. They can also upload their CVs and portfolios. Company Profiles: Companies can create profiles to showcase their organization, internship programs, and company culture.

Internship Postings: Companies can post detailed internship descriptions, including required qualifications, responsibilities, duration, location, and compensation. Recommendation System: An intelligent recommendation system will analyze student profiles and internship postings to suggest relevant matches for both parties. Application Management: Students can easily apply for internships and track the status of their applications. Companies can manage incoming applications, shortlist candidates, and communicate with applicants directly through the platform. Feedback and Rating System: Students and companies can provide feedback and ratings on their internship experiences. This feedback can help improve future matches and enhance the platform's effectiveness. Communication Tools: S&C will offer messaging and notification features to facilitate communication between students and companies.

2 System Architecture

2.1 High-Level Design

he S&C platform will use a 3-tier architecture, a common and scalable model for web applications. The three tiers are:

Presentation Tier (Client): Responsible for handling user interactions and displaying the user interface. This tier will be implemented using a combination of HTML, CSS, and JavaScript frameworks. Application Tier (Application Server): Houses the business logic of the platform, processing user requests, managing data, and implementing the recommendation system. This tier can be built using languages like Python, Java, or Node.js and will interact with the database. Data Tier (Database Server): Stores all the platform's data, including user profiles, internship postings, applications, and feedback. This tier can utilize a relational database management system (RDBMS) like PostgreSQL or MySQL for structured data storage.

2.2 Deployment Model

The SC platform will be deployed on a cloud-based infrastructure, using services like AWS or Google Cloud Platform. This will ensure: Scalability: The ability to handle varying levels of user traffic and data storage needs. Availability: High uptime and resilience to failures. Security: Robust security measures to protect user data.

3 Module Breakdown and Implementation

3.1 Authentication and Authorization

- Users can sign up and log in using JWT-based authentication. - OAuth integration for Google and LinkedIn. - Role-based access control to ensure appropriate permissions for students, companies, and administrators.

3.2 User Dashboard

- Separate dashboards for students and companies. - Students can view recommended internships, while companies can manage applications. - Analytics and reporting features to track application trends and engagement metrics.

3.3 Matching Algorithm

- Uses AI-based ranking to suggest the best internships based on user profiles. - Dynamic scoring mechanism that considers user preferences, past applications, and feedback.

3.4 Application Process Flow

- Students apply for internships via a structured application process. - Companies review applications, filter candidates, and schedule interviews. - Status updates provided at each stage of the application process.

3.5 Notifications and Communication

- Email and push notifications for application updates. - Integrated chat system for student-company communication. - Automated reminders for upcoming interviews and deadlines.

4 Database Design

4.1 ER Diagram

The database schema includes tables for users, internships, applications, and matches.

4.2 Schema Overview

Table	Description
Users	Stores student and company information, including login credentials and profile details.
Internships	Contains internship details such as title, description, required skills, and application deadlines.
Applications	Tracks applications for internships, including student ID, internship ID, and application status.
Recommender System	Stores AI-based recommendations, linking students with relevant internship opportunities.

5 API Documentation

This section provides an overview of the API endpoints available in the Students & Companies (S&C) Platform.

5.1 Authentication Endpoints

Endpoint	Method	Description
/api/auth/register/	POST	Register a new user
/api/auth/login/	POST	User login
/api/auth/logout/	POST	User logout

5.2 Student Endpoints

Endpoint	Method	Description
/api/students/	GET	Retrieve list of students
/api/students/{pk}/	GET	Retrieve a specific student

5.3 Company Endpoints

Endpoint	Method	Description
/api/companies/	GET	Retrieve list of companies
/api/companies/{pk}/	GET	Retrieve a specific company

5.4 Internship Endpoints

Endpoint	Method	Description
/api/internships/	GET	Retrieve list of internships
/api/internships/{pk}/	GET	Retrieve a specific internship
/api/companies/internships/	GET	Retrieve internships by a company

5.5 Application Endpoints

Endpoint	Method	Description
/api/applications/	GET	Retrieve list of applications
/api/applications/{_id}/	GET	Retrieve a specific application
/api/internships/{_id}/	GET	Retrieve applications for an internship
/api/applications/status/	GET	Retrieve application status
/api/applications/details/	GET	Retrieve application details

5.6 Feedback and Complaint Endpoints

Endpoint	Method	Description
/api/feedback/	POST	Submit feedback
/api/feedback/{_id}/	GET	Retrieve specific feedback
/api/complaints/	POST	Submit a complaint
/api/complaints/	GET	Retrieve a specific complaint

5.7 University Endpoints

Endpoint	Method	Description
/api/universities/	GET	Retrieve list of universities
/api/universities/{pk}/	GET	Retrieve a specific university
/api/universities/students/	GET	Retrieve students under a university
/api/universities/search/	GET	Search universities
/api/departments/{pk}/	GET	Retrieve department details
/api/courses/{pk}/	GET	Retrieve course details

5.8 Interview and Recommendation Endpoints

Endpoint	Method	Description

/api/interviews/	GET	Retrieve list of interviews
/api/interviews/{_id}/	GET	Retrieve a specific interview
/api/applications/interviews/	GET	Retrieve interviews for an application
/api/recommendations/	GET	Retrieve AI-based internship recommendations

6 Testing Details

6.1 Unit Testing

Individual components are tested in isolation to verify their functionality.

6.1.1 Frontend Testing (Next.js)

Objective: Ensure components and functions perform correctly in isolation.
Tools: Jest, React Testing Library.
Test Cases:

- Component Rendering: Ensure components render correctly with various props.
- State Management: Test if the component's state updates correctly when triggered by user actions.
- Event Handlers: Test event handlers to ensure they trigger expected actions.
- API Calls: Mock and test API calls to check if components handle the responses correctly.
- Error Handling: Ensure the component handles edge cases and displays appropriate error messages.

6.1.2 Backend Testing (Django with pytest)

Objective: Test individual components of the backend to ensure they work correctly.
Tools: pytest, pytest-django, FactoryBoy (for data creation).
Test Cases:

- Model Tests: Test model methods and properties; ensure database constraints are enforced.

- View Tests: Test that views return correct status codes and data; test that permissions and authentication are correctly enforced.
- Serializer Tests: Test that serializers correctly validate incoming data and serialize/deserialize data accurately.
- Form Tests: Test form validation and data saving functionality.

6.2 Steps

1. Clone the repository: `git clone [https://github.com/Shaurya111001/CaliskanSinghK`
2. Install backend dependencies: `cd backend; pip install -r requirements.txt`
3. Install frontend dependencies: `cd frontend; npm install`
4. Configure database settings in Django.
5. Run database migrations: `python manage.py migrate`
6. Start backend server: `python manage.py runserver`
7. Start frontend server: `npm run dev`

7 Risk Analysis and Mitigation

- Security vulnerabilities mitigated via penetration testing.
- Data consistency ensured through ACID-compliant transactions.
- Downtime minimized with auto-scaling infrastructure and failover support.

8 Effort Spent

- Shaurya Aditya Singh: 10hr
- Ergun Gani Çalışkan: 10hr
- Mohammadali Khaledi: 10hr

9 Future Enhancements

- AI-powered interview scheduling.
- Enhanced real-time analytics for user engagement.
- Blockchain-based credential verification.

10 References

- Requirements Analysis and Specification Document (RASD)
- Design Document (DD)
- Unified Modeling Language (UML) Specifications