POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Design Document

Computer Science and Engineering

Author(s): **Giovanni Vaccarino**
Vittorio Palladino
Nicolò Vacis

Contents

1 Introduction

1.1. Scope

The S&C platform is a web-based solution intended to support:

- **Students** in finding internships suited to their skills and aspirations, allowing them to submit CVs and receive tailored recommendations.
- **Companies** in listing internship opportunities and identifying candidates based on skills and profiles.
- **Universities** in overseeing internships, tracking progress, addressing complaints, and ensuring the quality of internships.

The platform enables interaction among these groups, handling tasks like matching candidates with internships, interview scheduling, feedback collection, and assessing internship effectiveness. The platform is built to scale across universities, supporting thousands of users.

1.2. Definitions, Acronyms, Abbreviations

1.2.1. Definitions

- **Slug**: A human-readable and URL-friendly string (typically lowercase ASCII characters) that uniquely identifies a specific resource. It is commonly used in URLs but may also serve as an identifier for resources, like internship listings or profiles, within the S&C platform.

1.2.2. Acronyms

- **S&C**: Students&Companies platform
- **API**: Application Programming Interface
- **CV**: Curriculum Vitae
- **GDPR**: General Data Protection Regulation
- **UUID**: Universal Unique Identifier
- **DB**: Database

- **DBMS**: Database Management System
- **REST**: Representational State Transfer
- **SPA**: Single Page Application
- **CDN**: Content Delivery Network

1.2.3. Abbreviations

- **e.g.**: For example
- **repo**: Repository
- **ID**: Identifier

1.3. Revision History

Version	Date	Description
1.0	January 7, 2024	Initial release

1.4. Reference Documents

- Specification document: "Assignment RDD AY 2023-2024"
- UML official specification: `https://www.omg.org/spec/UML/`
- Requirements Analysis and Specification Document: "RASD"

1.5. Document Structure

The document is organized into the following sections:

- **Section 1: Introduction** - Provides an overview of the problem and the system's scope. It also includes definitions, acronyms, abbreviations, and a revision history to track document versions and modifications.
- **Section 2: Architectural Design** - Describes the system architecture, starting with a high-level overview of design choices, followed by detailed component and deployment views, including server, client, and data components (DB schemas). This section also presents sequence diagrams to illustrate the system's runtime view and includes information on component interfaces, design styles, and architectural patterns.
- **Section 3: User Interface Design** - Outlines the design of the user interface, providing an overview of the platform's visual layout and user interactions.
- **Section 4: Requirements Traceability** - Maps the requirements from the RASD to corresponding design elements in this document.

- **Section 5: Implementation, Integration, and Test Plan** - Details the implementation order of the system's subcomponents, integration steps, and testing procedures to ensure system reliability.

2 | Architectural Design

2.1. Overview

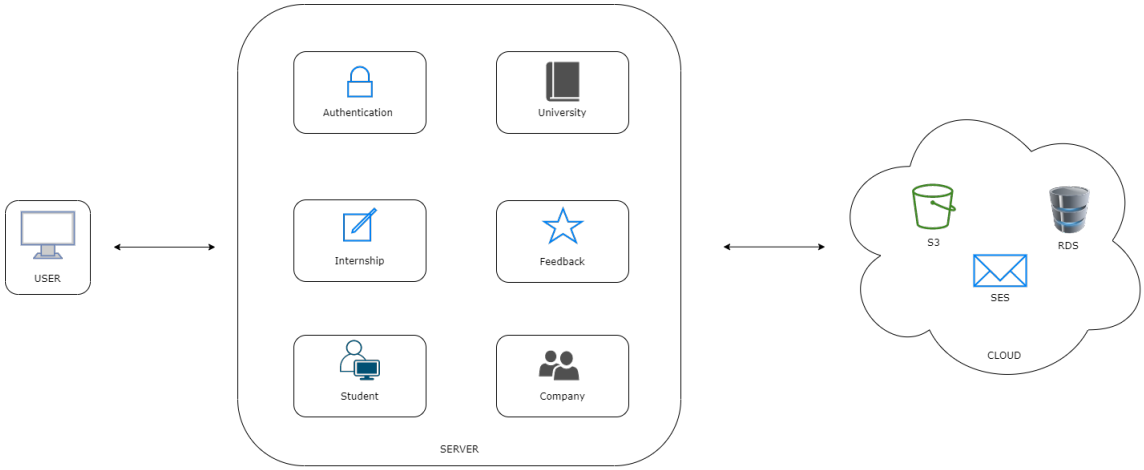


Figure 2.1: Overview diagram

The Students&Companies (S&C) platform is designed as a client-server application, where the term *client* refers to the users of the system: students, companies, and universities. We have chosen a *fat client* approach to provide a highly interactive, app-like experience, enabling responsive and rich functionality directly in the browser. This approach also reduces server load by offloading some processing to the client side, enhancing scalability.

Users interact with the system through a Single Page Application (SPA), a desktop web application that serves as the primary user interface for accessing platform features. The SPA allows for seamless navigation and interaction. By managing application state on the client side, the SPA minimizes page reloads, enhancing usability and responsiveness.

On the backend, we have followed a *monolithic server* approach. This choice simplifies the overall system architecture by consolidating functionality into a single codebase, which can facilitate easier development, deployment, and maintenance. Although monolithic, the server is structured into multiple controllers and several business logic parts, each responsible for specific areas, such as authentication, internship management, profile handling, and recommendations. This modular organization within the monolith ensures that each component is cohesive and manageable while maintaining performance and consistency.

To support database, storage, and email services, we are using a cloud provider infrastructure. Specifically, we utilize Amazon Web Services (AWS) with Amazon RDS for the database, Amazon S3 for file storage, and Amazon SES for email notifications.

2.2. Component view

2.2.1. Client Components

The principal component of the client side is the Web-App, which the user will interact with.

2.2.2. Server Components

In the server side, resides the principal components such as the controller used to manage the HTTP request sent by the user.

- **Internship**: Handles the logic related to managing internships, including creating, updating, and tracking internship opportunities. It allows administrators and companies to post internship positions and lets students apply, track, and view their application statuses.
- **Student**: Manages all student-related data, including profile creation, academic details, and progress tracking. It allows students to update their profiles and view their application history, while administrators can monitor student progress and make updates as needed.
- **Match**: Manages all the matches between the students and the companies' internship, finding matching between the suitable skills. Allows sending invitation to the student, and accepting those ones.
- **Feedback**: Implements the system for collecting and managing feedback from students, companies, and administrators. This component allows participants to submit feedback on internship experiences, and administrators can review and analyze this feedback to improve the program.
- **Company**: Responsible for managing company accounts and profiles, as well as facilitating company interactions with students and internships. It allows companies to create profiles, post internship opportunities, and review student applications, making the hiring and feedback process streamlined and accessible.
- **University**: Responsible for managing university interaction with the platform. Such as tracking students' internship, reading their feedback and leaving a feedback about the platform.
- **Authentication**: The Authentication handles all processes related to user authentication, such as logging in, logging out, session validation, and password management.

2.2.3. Data Component

We use a relational database hosted on AWS to securely manage and organize structured data for the application. AWS's managed RDS service offers high availability, automated backups, and scalability, ensuring reliable data storage and quick access for all server com-

ponents. This setup allows us to maintain relational integrity, enforce data consistency, and easily scale as our data requirements grow.

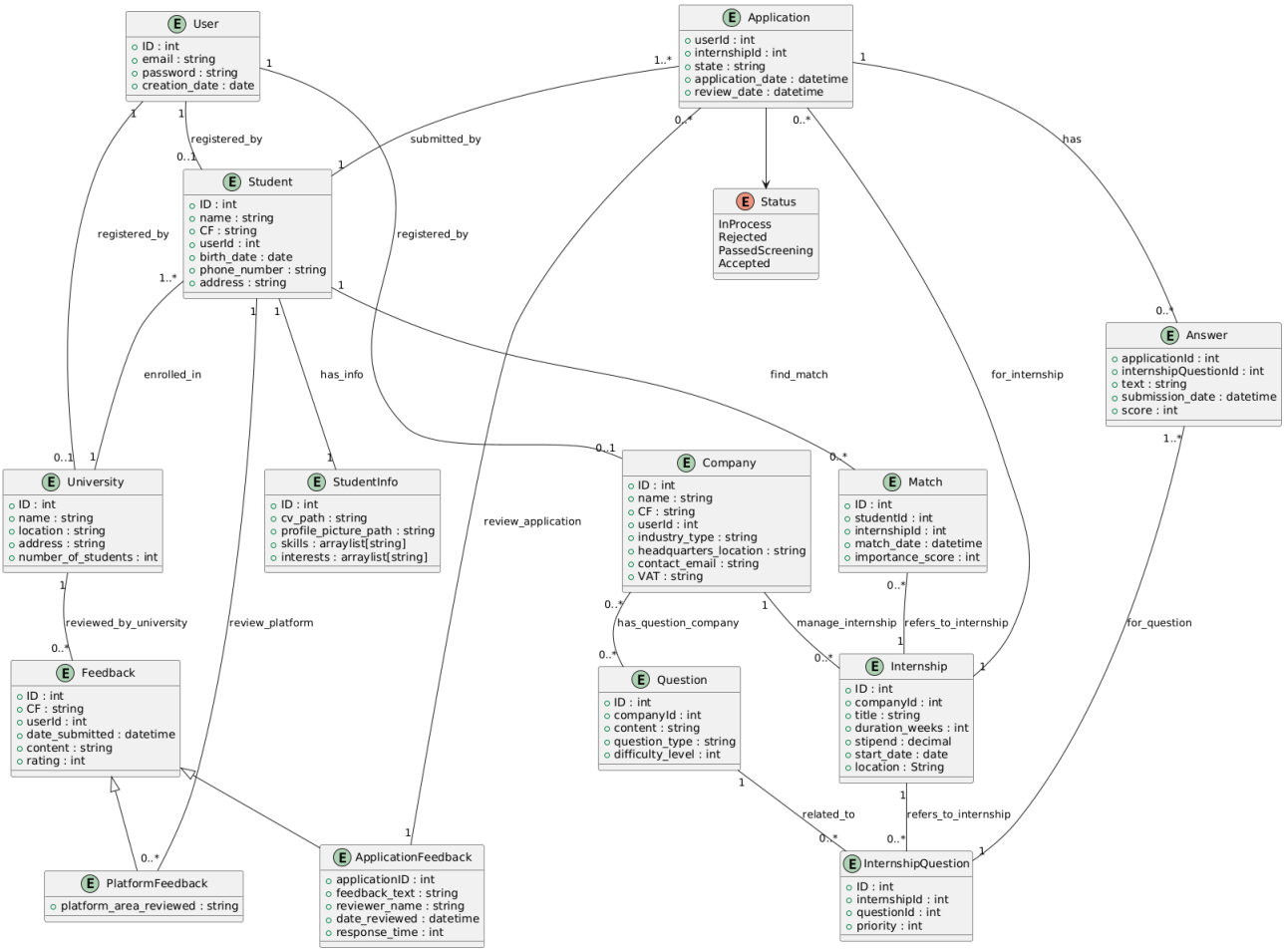


Figure 2.2: Class diagram referring to the data components

2.3. Deployment view

The frontend is a SPA implemented with the React framework. The website can be visited from any modern web browser, such as Google Chrome, Mozilla Firefox or Microsoft Edge.

Clients can be any desktop device that can run the previous browsers, such as computers with Windows, Linux or macOS. The website is served by a CDN like Amazon CloudFront. Using a CDN greatly enhances performance by caching static assets on globally distributed servers, ensuring users access content from locations closest to them for faster load times and reduced latency.

The backend of the platform is powered by an ASP.NET Core server, deployed within a Kubernetes cluster that orchestrates and manages the containerized environment. This setup ensures that the backend is scalable, resilient, and highly available, with Kubernetes

automating the deployment, scaling, and management of application pods. NGINX is used within the cluster as an Ingress Controller, handling incoming traffic, SSL termination, and advanced routing rules. Kubernetes performs horizontal scaling by automatically deploying new pods during traffic surges to maintain seamless performance.

The ASP.NET Core application communicates with AWS services like RDS for database and S3 for storage. Continuous integration and deployment pipelines facilitate automated updates, pushing new Docker images to the cluster and enabling rolling updates with minimal downtime.

This integration of Kubernetes and NGINX ensures that the backend infrastructure is robust and adaptable, capable of handling large volumes of requests while providing efficient load distribution and reliability.

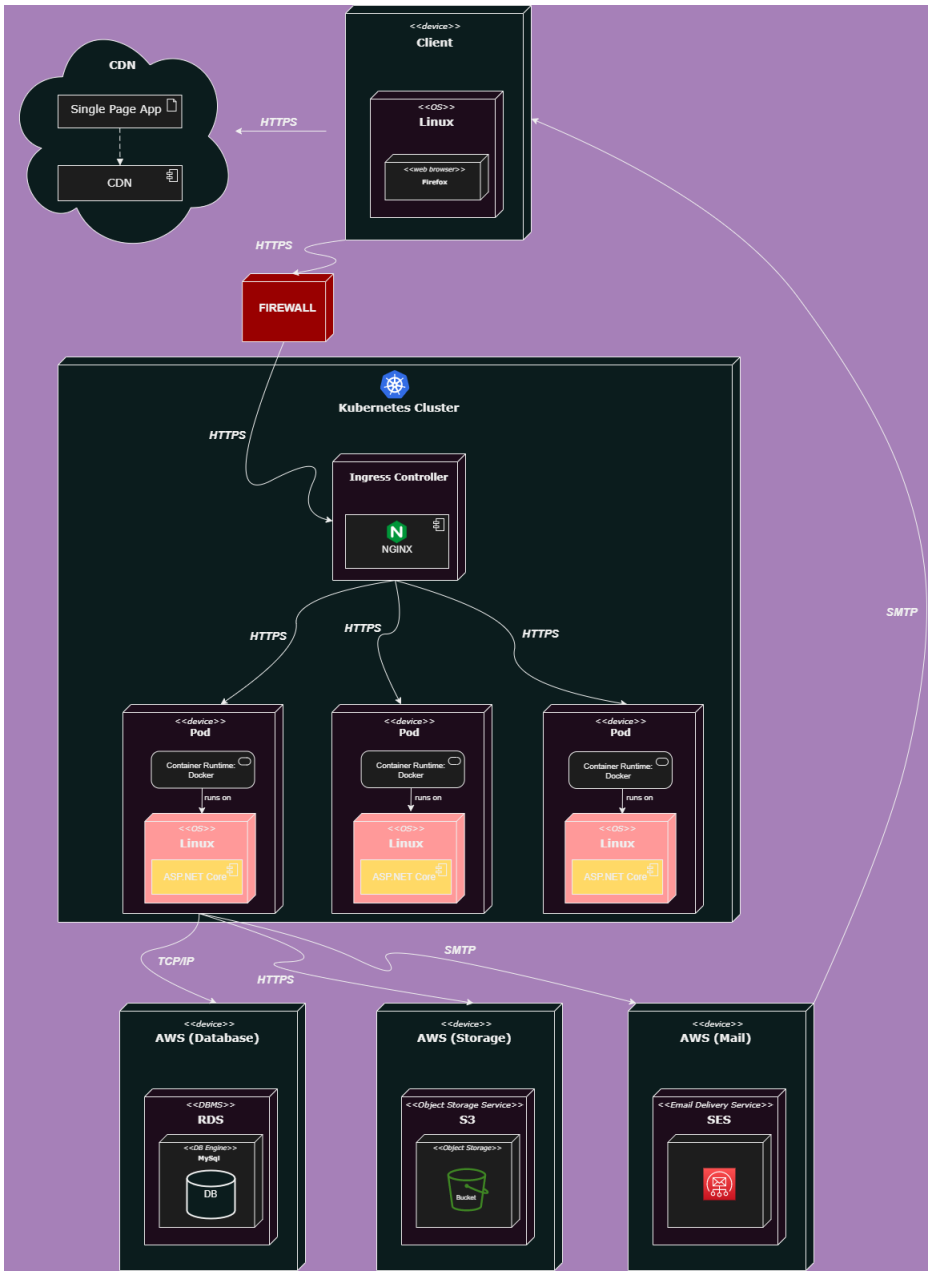


Figure 2.3: Deployment diagram

2.4. Runtime view

The following sequence diagrams illustrate the interactions between system components.

Each diagram corresponds to and represents the implementation of its respective use case, as described in the RASD document, in the same order.

In each sequence diagram it is omitted the part specified in the diagram 0.1, that illustrates the case of unauthorized access and the tentative to gain a new valid authentication token.

The following function (2.2) for simplicity is represented in another diagram, but it is

meant to be intended as an action that happens before all the other diagrams.

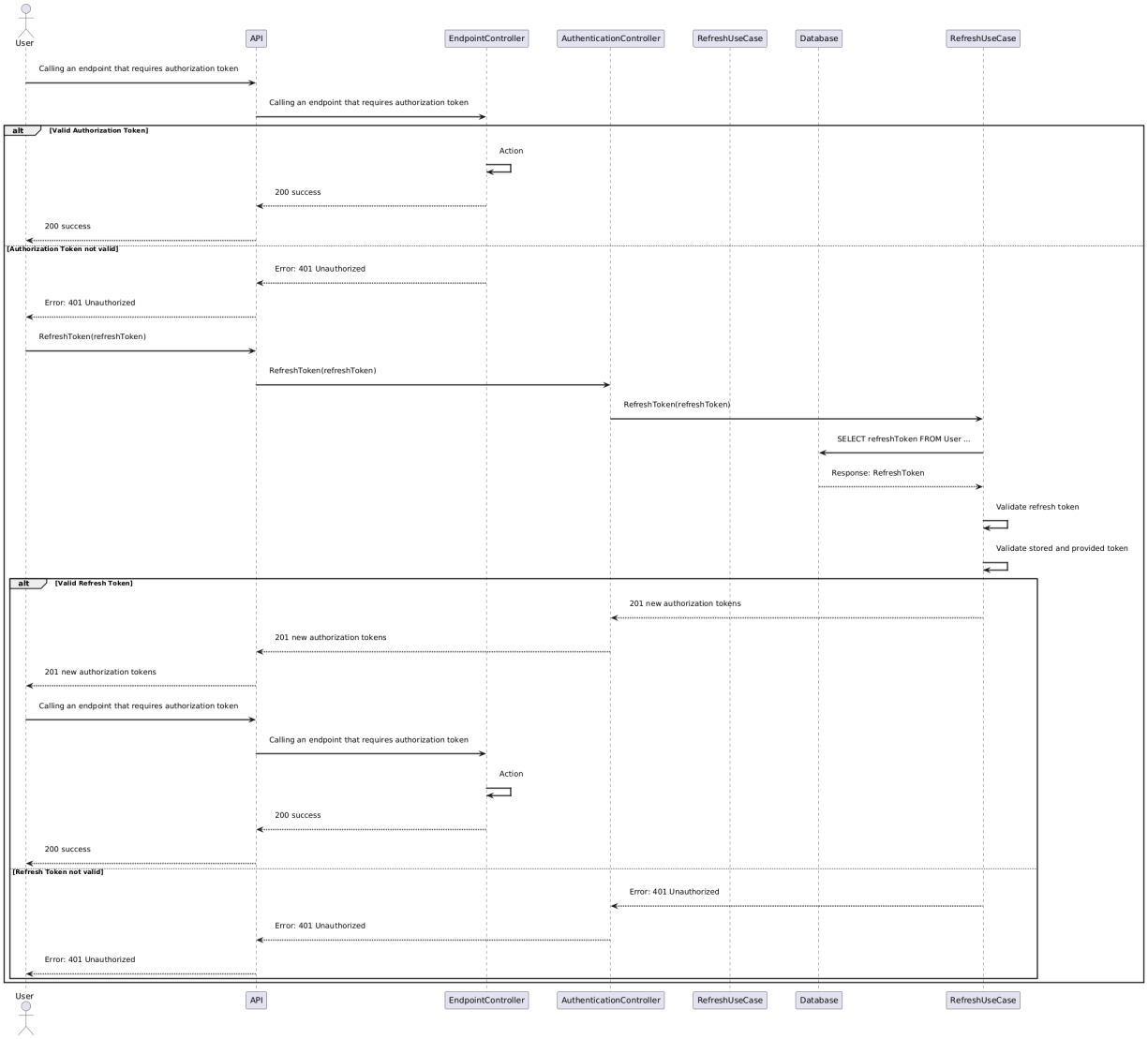


Figure 2.4: Sequence diagram 0.1 : Authorization Flow

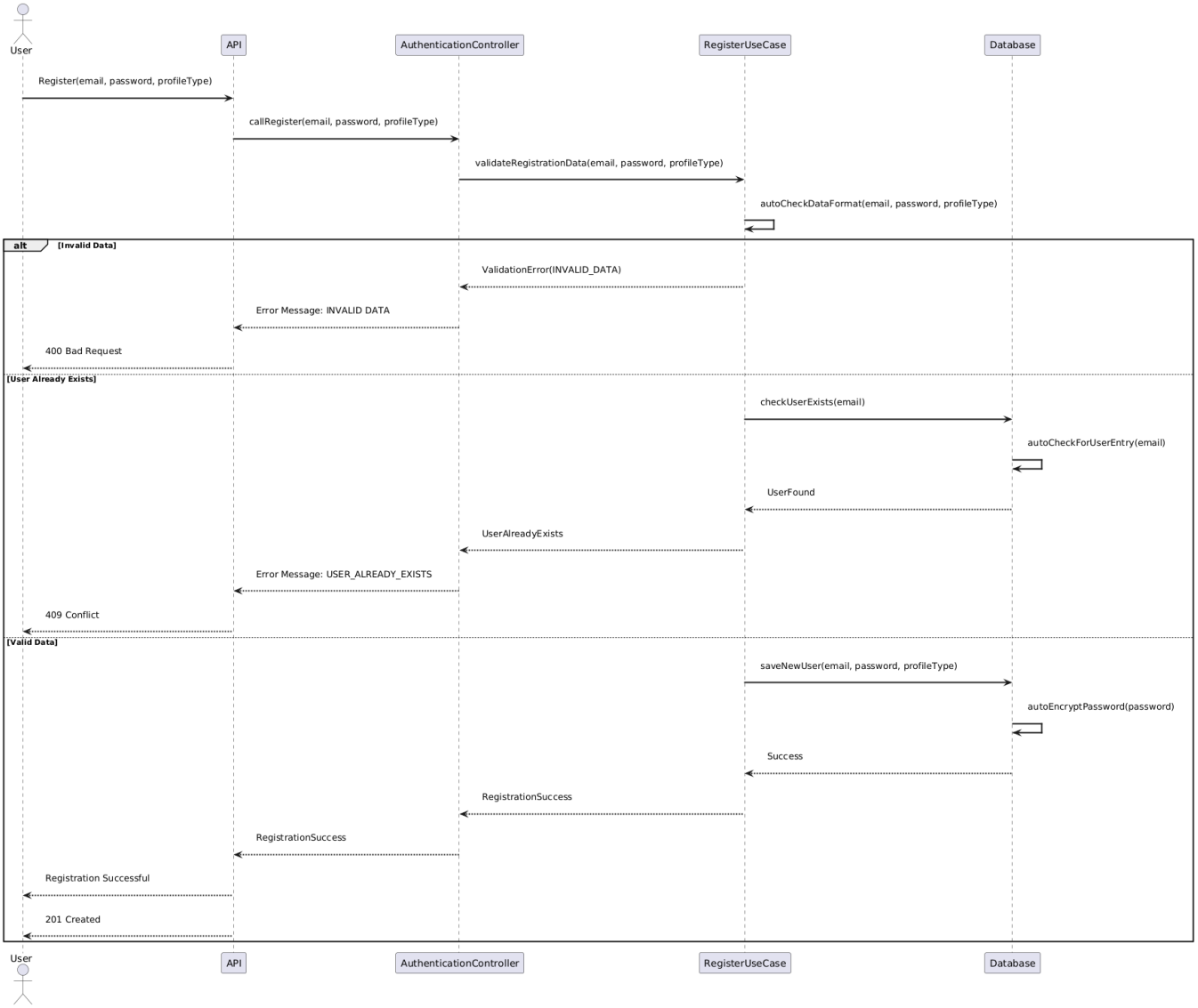


Figure 2.5: User (Student, Company or University) Registration

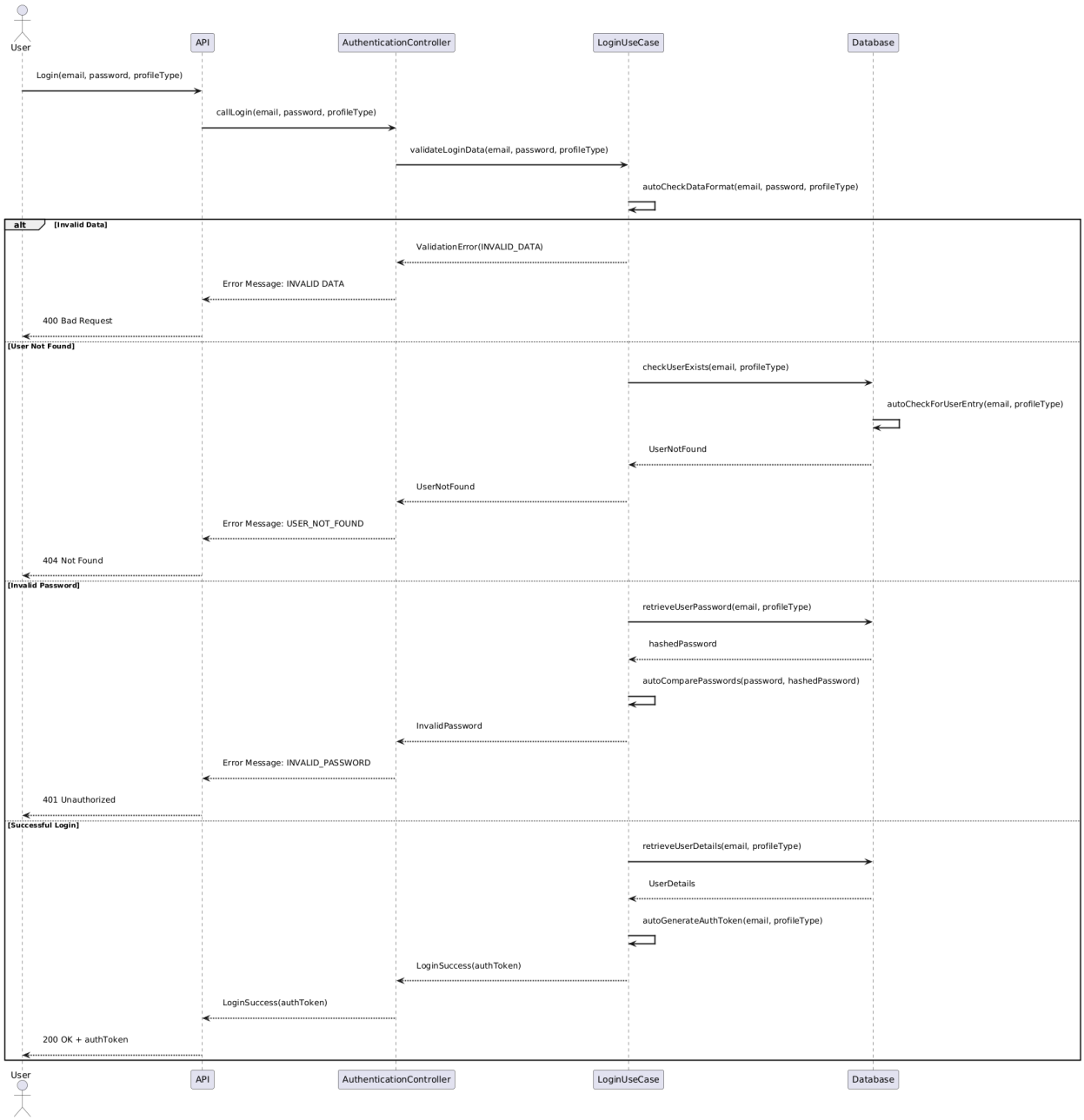


Figure 2.6: User (Student, Company or University) Login

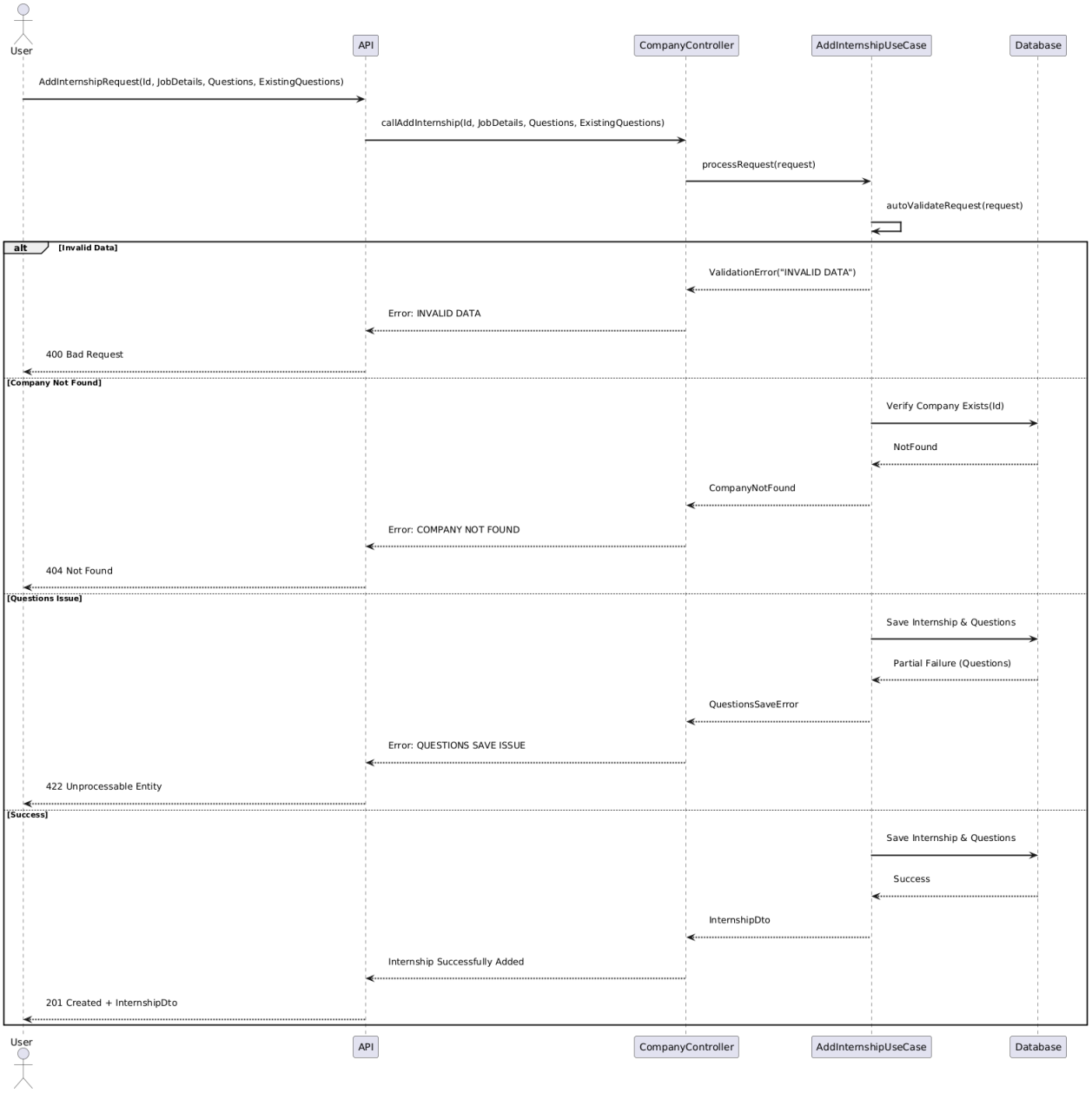


Figure 2.7: Internship post creation

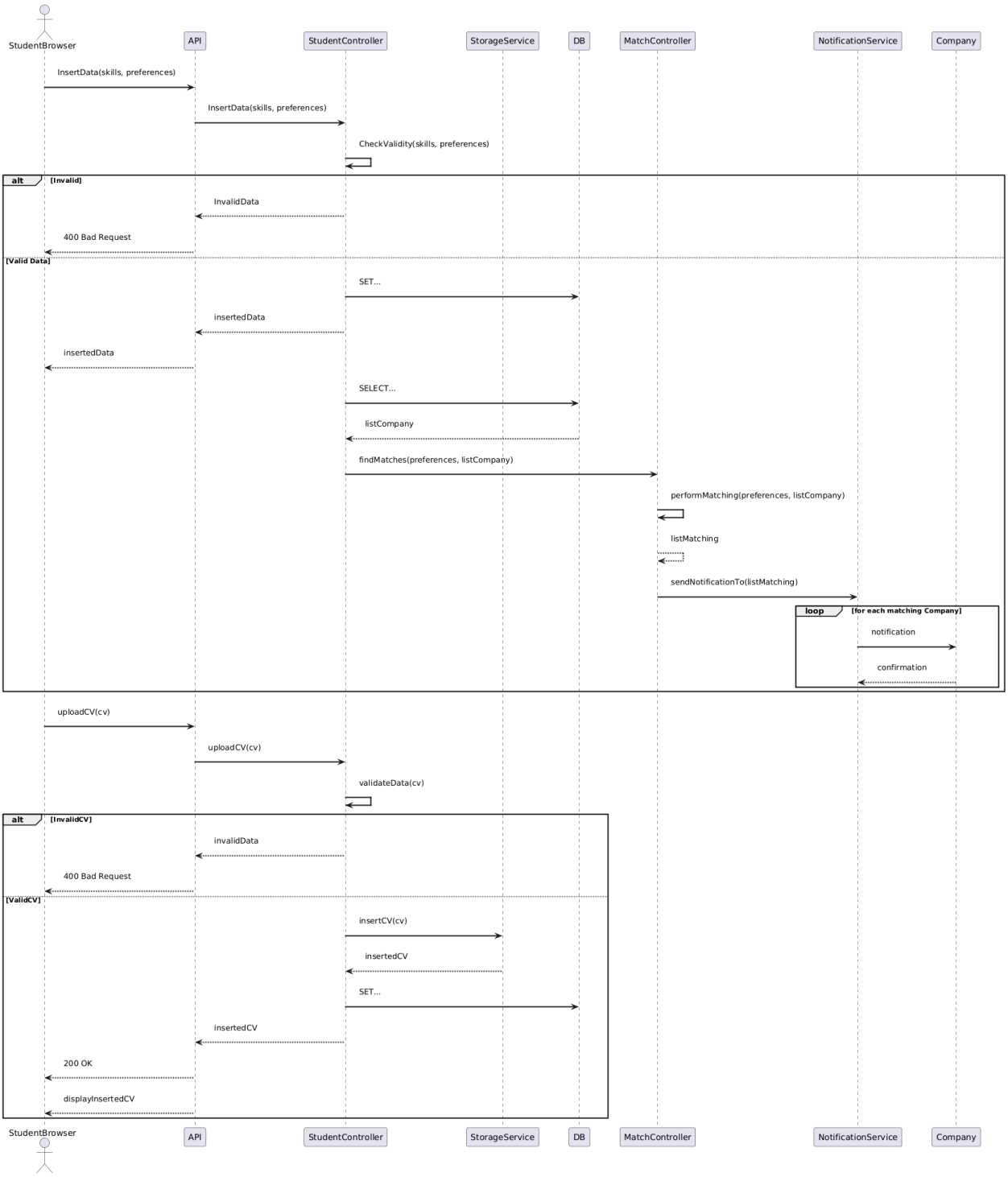


Figure 2.8: Student inserts CV and data

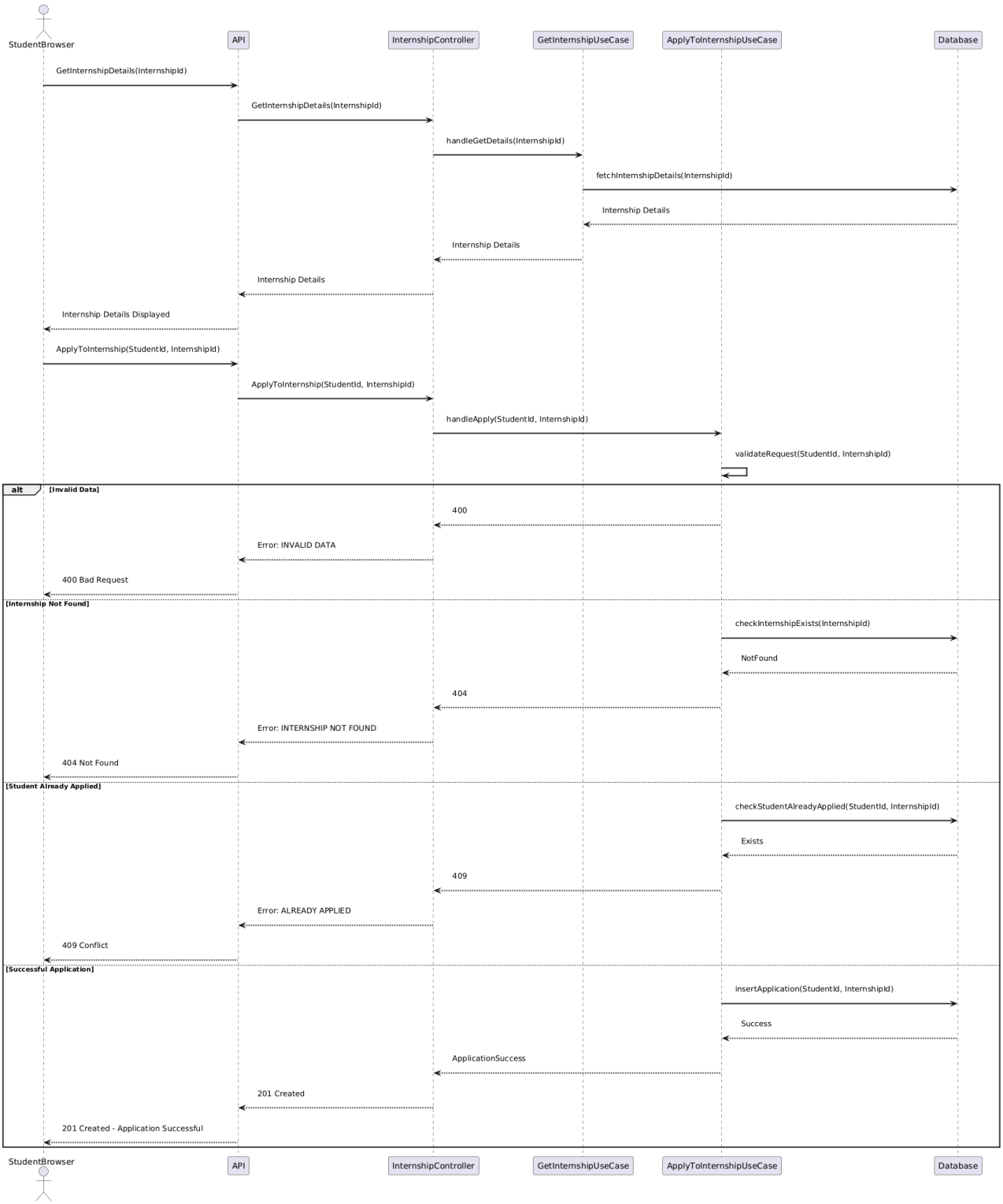


Figure 2.9: Student submit an application to an internship

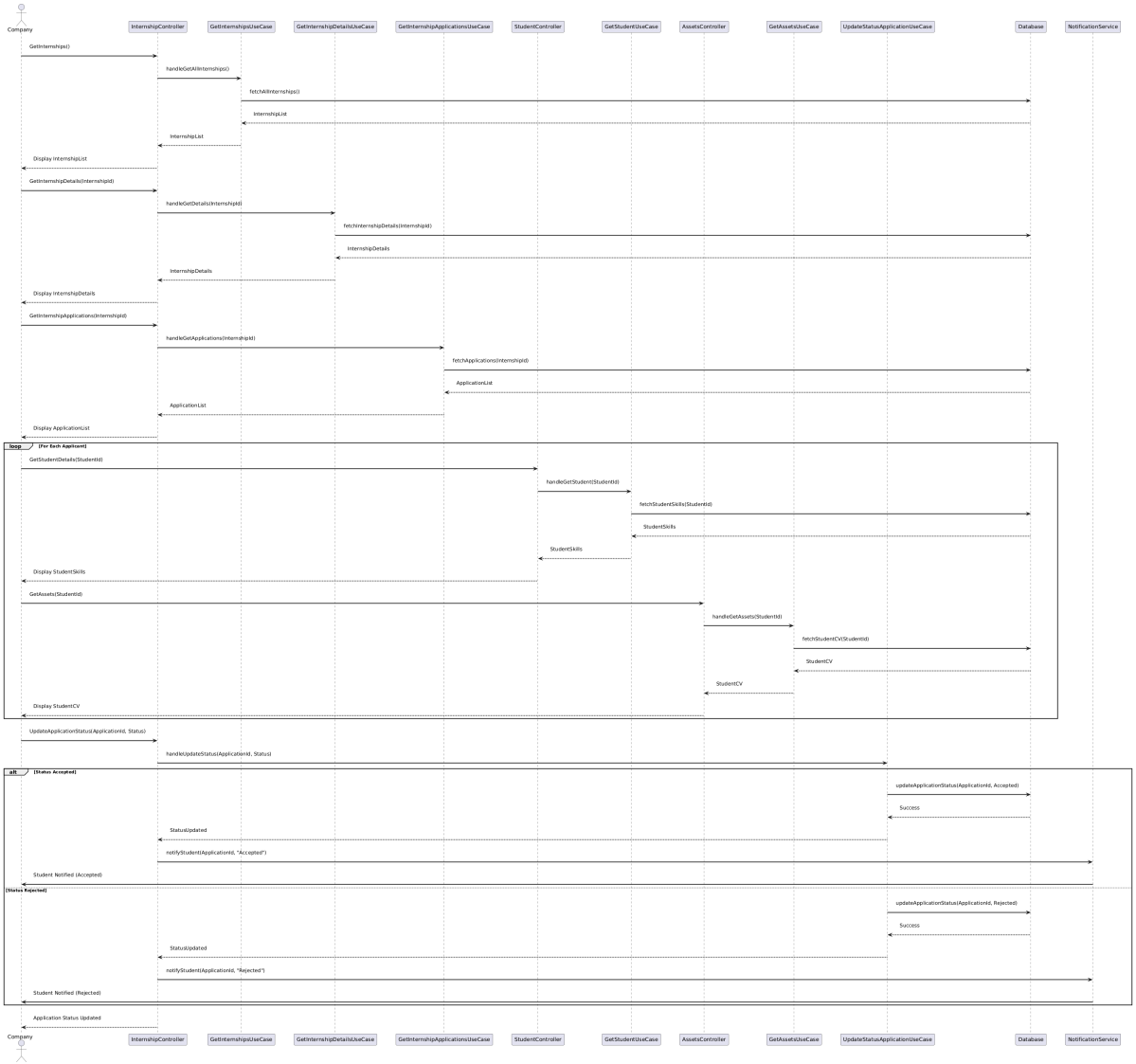


Figure 2.10: Company management of internship applications

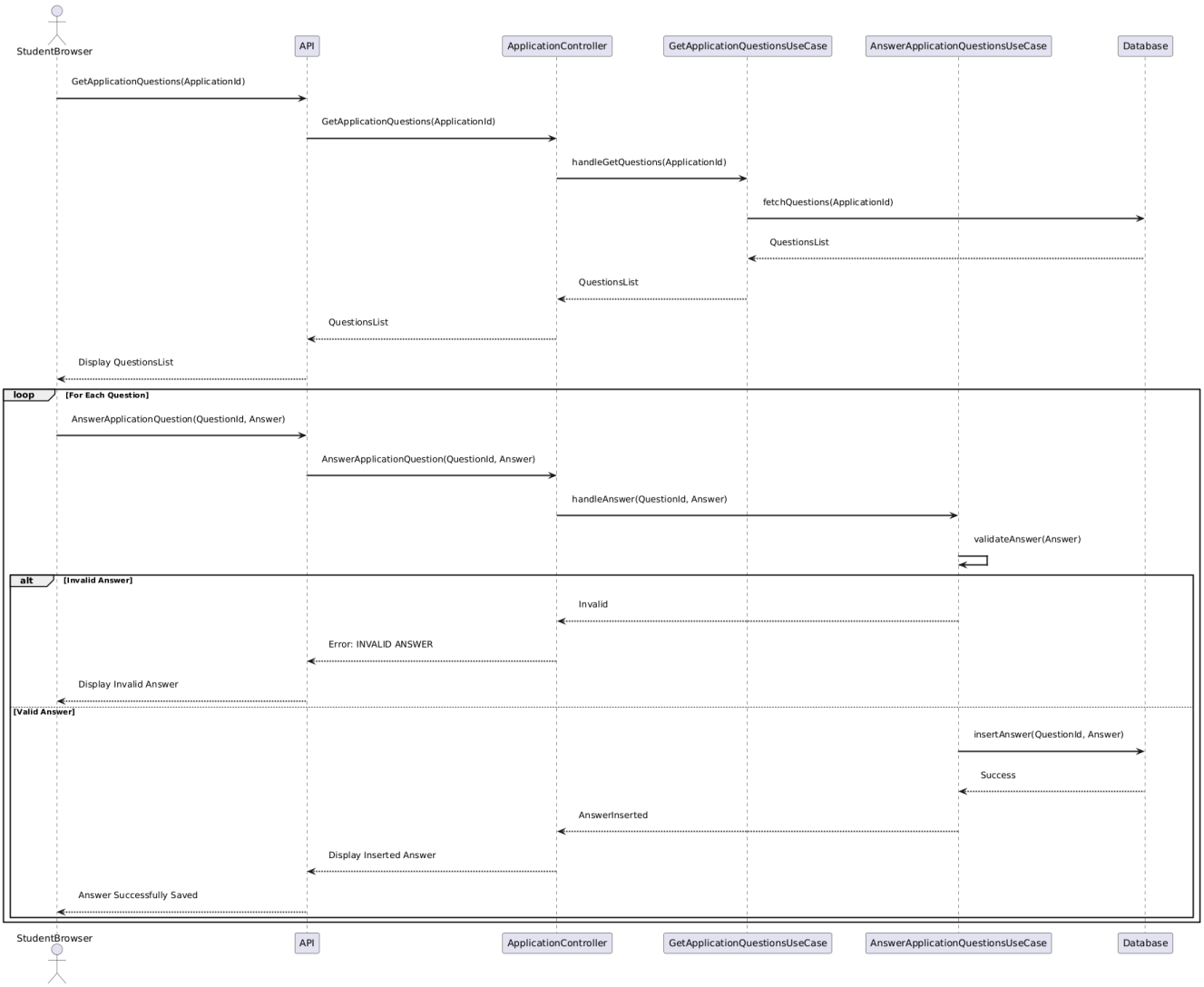


Figure 2.11: Student answers questions

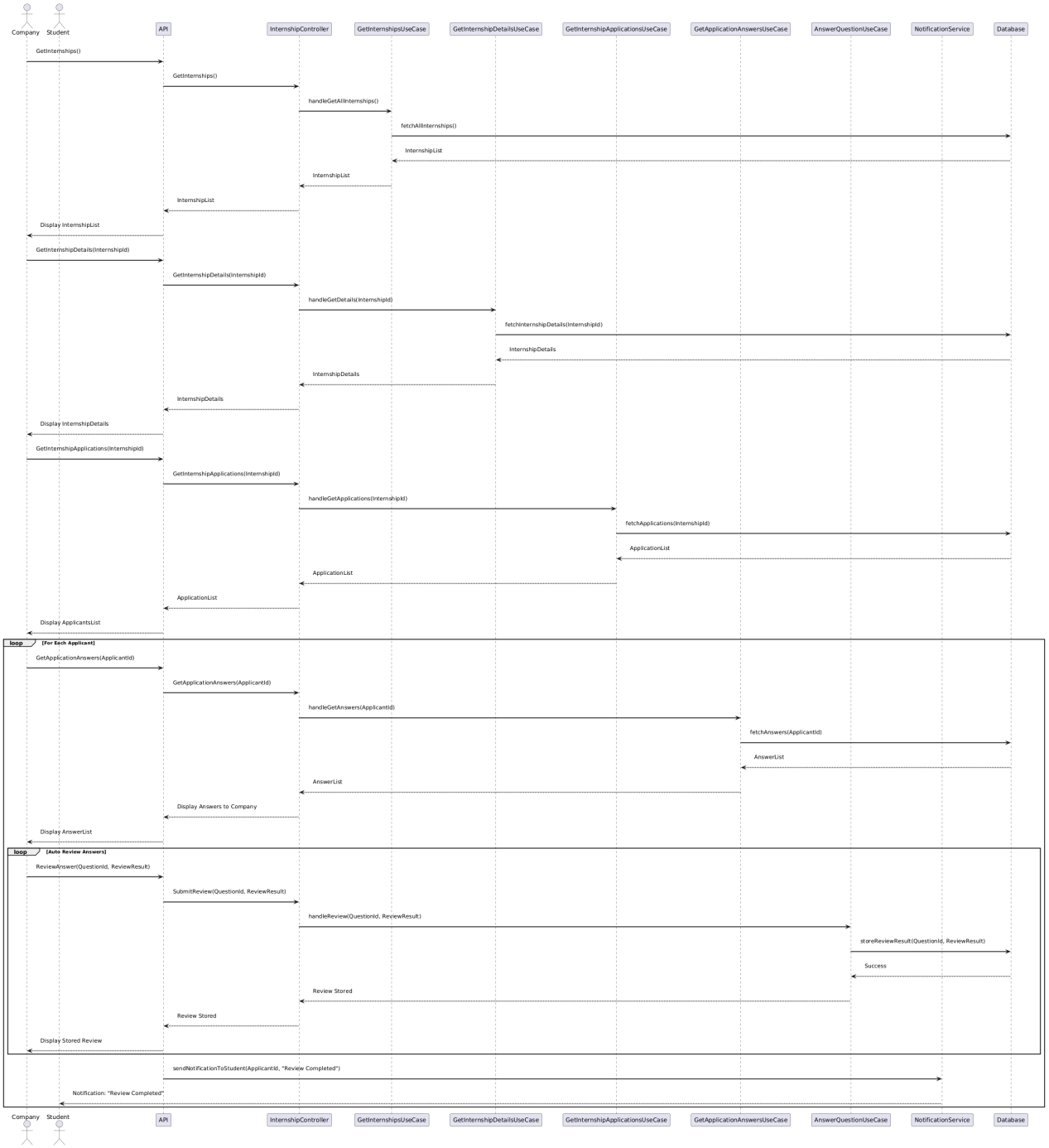


Figure 2.12: Company reviews online assessment questions

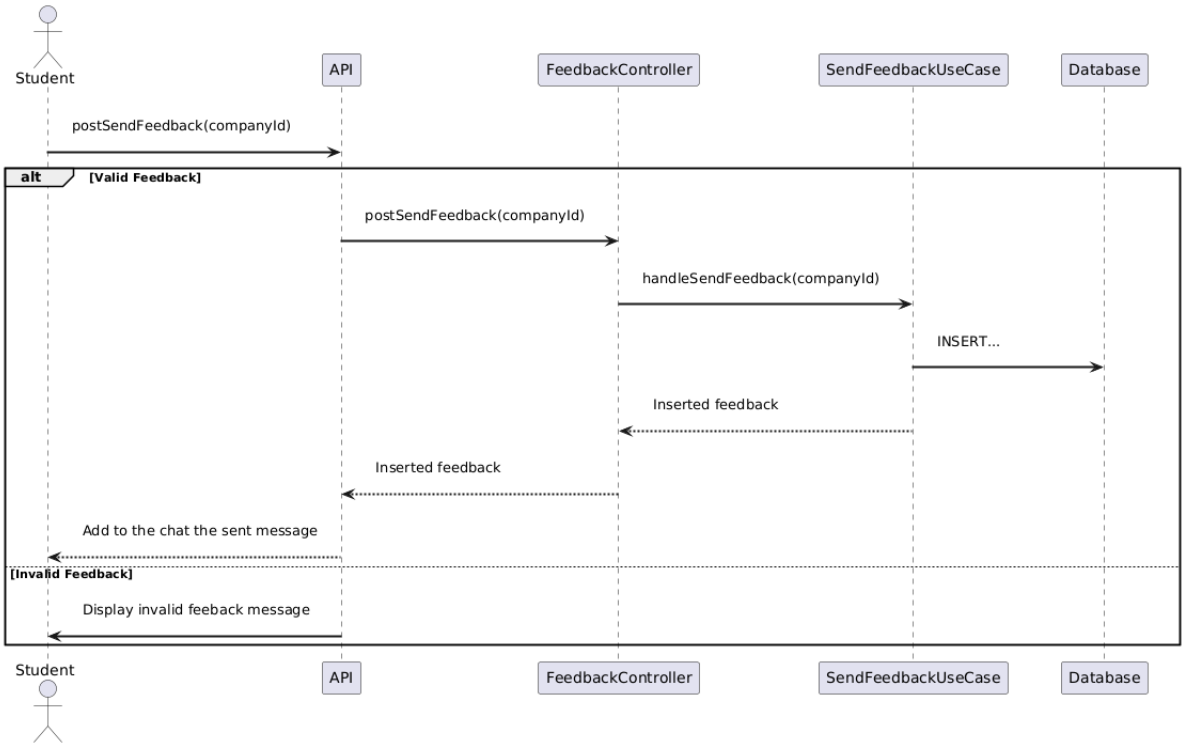


Figure 2.13: Student(Company) sends feedback to the platform.

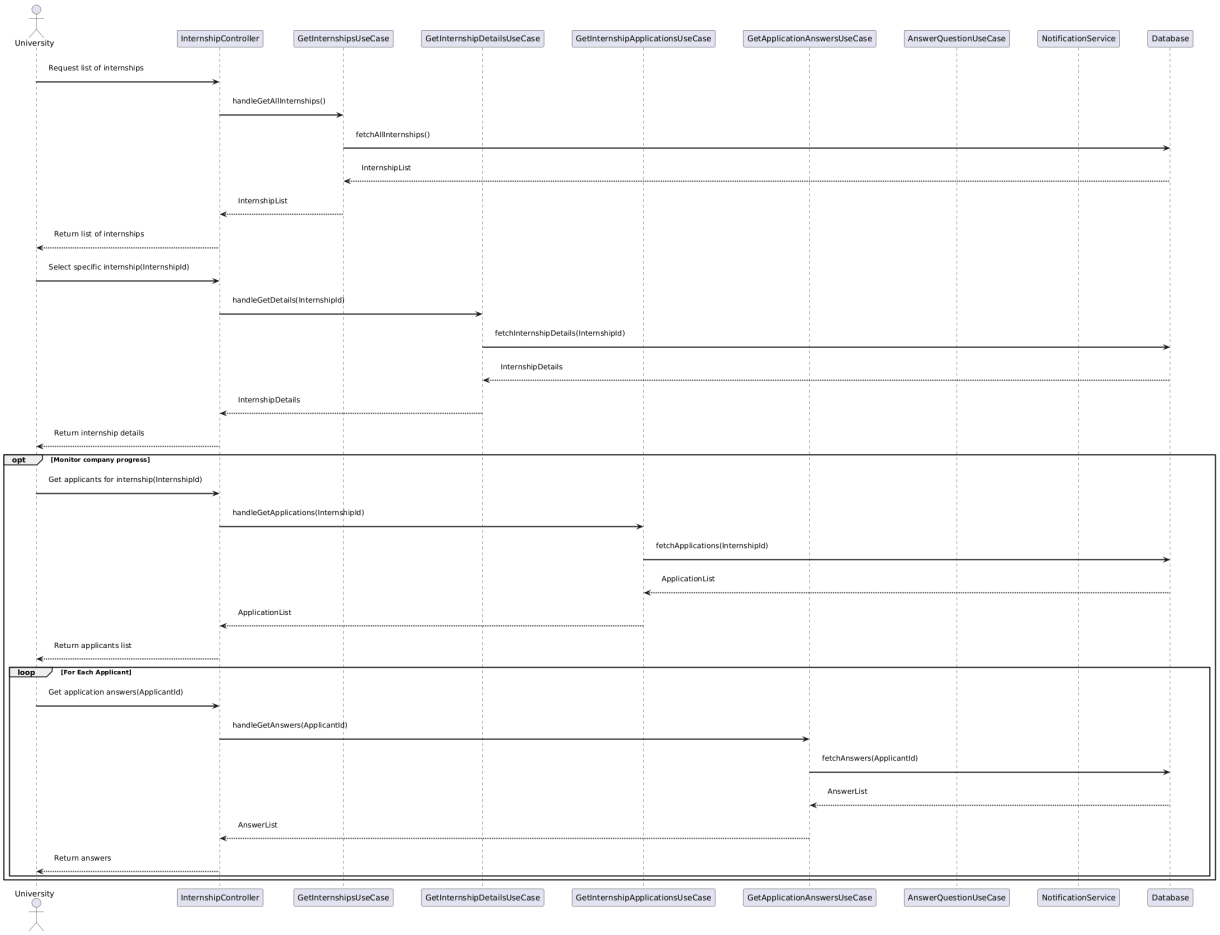


Figure 2.14: University monitors the progress of internships

2.5. Component interfaces

In this section we use the word **"Dto"** referring to a datatype defined in the backend containing all the required attribute. The input taken from the frontend or the output coming from the backend are mapped into a dto to double check the correctness of the values.

- **Authentication** Handles all processes related to user authentication, such as logging in, logging out, token refreshment, and password management.
  - **register(dto: RegisterDto): TokenResponse** – Registers a new user (student, company or university) by saving the provided data and returns a *TokenResponseDto*.
  - **login(email: string, password: string): TokenResponseDto** – Authenticates a user (student or company) by validating credentials. If successful, returns a *TokenResponseDto*.
  - **logout(): void** – Logs out the user, ending their session.
  - **refreshToken(refreshToken: String): TokenResponse** – Extends the session duration by generating a new *authToken* if the refresh token is valid.
  - **resetPasswordRequest(email: String): void** – Sends a password reset link to the provided email, allowing users to initiate the password reset process.
  - **resetPassword(token: String, newPassword: String): Bool** – Validates the reset token sent to the user's email and updates the password if the token is valid. Returns *true* upon success.
  - **verifyEmail(verifyEmail: VerifyEmailDto): VerifyResponse** - verify the email validity.
  - **sendVerificationEmail(sendVerify: SendVerificationDto): void** - end the verification email after the registration.
- **Internship**
  - **getInternship() : List<InternshipDto>** - Allows platform to retrieve all the internship to fill the homepage.
  - **createInternship(companyId: ID, internshipData: Internship): ID** – Allows a company to create a new internship posting, returning the ID of the new internship.
  - **applyToInternship(studentId: int, internshiptId: int): ApplicationDto** - Send the student application to a specific internship.
  - **getInternshipApplicants(internshipId: int, companyId: int): List<StudentDto>** - Retrieves the student that had applied for an internship of a company.
  - **updateStatusApplication(applicationId: int, updatedStatus: updatedStatusApplicationDto, companyId: int): ApplicationDto** – Up-

dates an existing application changing its status.

  - **answerApplicationQuestions(applicationId: int, answer: AnswerQuestionsDto, studentId: int** - Add the student's answers to the application.
  - **deleteInternship(internshipId: int): void** – Deletes an internship listing, making it inactive or removing it from the view.

- **Student** The Student Controller manages requests related to student profiles, such as updates and viewing application history.
  - **getStudent(studentId: int): void** – Get the student profile.
  - **updateStudentProfile(studentId: ID, updatedData: UpdateStudentDto): void** – Allows a student to update their profile with new information, such as contact details or academic information.
  - **loadCvStudent(dto: LoadCvDto, studentId: int): CvDto**
  - **listStudentApplications(studentId: int): List<Application>** – Returns the history of internship applications made by the student.

- **Match** The match controller manages finding and retrieving the matches between the company's internship suitable skills and the students' one.
  - **GetStudentMatches(studentId: int): List<MatchDto>** - Retrieve the student matches with the intenship of various companies.
  - **GetCompanyMathes(companyId: int): List<MatchDto** - Retrieve the company matches between the suitable skills of the students and of their internships.
  - **inviteStudent(matchId: int, companyId: int): void** - Company invite a student who is matching with one of its internship.
  - **acceptMatch(matchId: int, studentId: int): void** - Student accept one of the invite sent by the company.

- **Feedback** The Feedback Controller manages operations for collecting and handling feedback within the system.
  - **submitFeedback(userId: ID, internshipId: ID, feedbackData: Feedback): void** – Allows a user (student or company) to submit feedback on an internship or application experience.
  - **getFeedbackForInternship(internshipId: ID): List<Feedback>** – Retrieves all feedback associated with a specific internship, useful for administrators to monitor feedback trends.
  - **getFeedbackForStudent(studentId: ID): List<Feedback>** – Retrieves feedback given to or by a student, allowing administrators to track individual experiences.

- **Company** The Company Controller is responsible for managing company accounts and interaction with internship postings and applications.
  - **getProfile(comapanyId: int): CompanyDto** – Creates a new company profile and returns the company ID.
  - **updateCompanyProfile(companyId: int, updatedData: CompanyDto): void** – Allows a company to update its profile information, such as contact details or industry.
  - **getInternships(companyId: int): List<Internship>** – Lists all internships posted by the company.
  - **addInternship(companyId: int, updateData: InternshipDto): void** - Add an internship.
  - **updateInternship(companyId: int, internshipId: int, updateData: InternshipUpdateDto): void** - Update a current internship of a company.
  - **getQuestions(companyId: int)** - Get the questions created by the companies.
  - **addQuestion(companyId: int, updateQuestion: UpdateQuestionDto** - Add a new question.
  - **acceptApplication(companyId: int, applicationId: int): void** – Allows a company to accept a student application for an internship.
  - **rejectApplication(companyId: int, applicationId: int): void** – Allows a company to reject a student application for an internship.
- **University** The university controller is used to manage the university's interactions with students' internships and feedback.
  - **getStudentInternshipRecords(studentId: ID): List<Internship>** – Retrieves a list of internships a student has participated in, including completed and current ones.
  - **collectFeedback(studentId: ID, internshipId: ID): Feedback** – Collects feedback from students regarding their internship experience for quality assurance and improvement.
  - **assignInternshipAdvisor(studentId: ID, advisorId: ID): void** – Assigns an academic advisor to a student's internship to provide guidance and support.
  - **listAllFeedback(internshipId: ID): List<Feedback>** – Returns all feedback related to a specific internship to assess overall satisfaction and outcomes.
  - **getInternshipApprovalStatus(studentId: ID, internshipId: ID): ApprovalStatus** – Retrieves the current approval status for a student's internship application.

– **updateStudentFeedback(studentId: ID, feedbackId: ID, updated-Feedback: Feedback): void** – Updates feedback provided by a student if changes are needed.

2.6. Selected architectural styles and patterns

Monolithic server: We chose a monolithic approach for its simplicity, faster development, and easier management. It minimizes the complexities of inter-service communication and ensures consistent control, which is ideal for smaller teams or projects with limited resources. This approach also offers better performance and simpler security management, making it a practical choice for our project's scope

Fat client: Fat clients allow offering a wide variety of functionalities independent from the central server, as well as to move part of the business logic off of the server and into the clients. The main advantages it offers are greater decoupling of frontend and backend as well as a better interactive experience, especially in conditions where the client is on an unstable network. Additionally, with the adoption of a single-page applications, there are many cross-platform frameworks which allow to reuse code partially or entirely on multiple platforms. The single-page web application can also be served by a dedicated static web server, as all its interactivity is implemented client-side, further reducing the burden on the server by delegating its serving to a CDN, which is highly optimized for this specific use case.

REST API: An architectural style defined on top of HTTP centered around the definition of a standardized set of stateless operations. Its main advantages are its simplicity, its use of widely adopted standards which facilitate adoption, and its ease of scalability given by its stateless nature. It allows to have a single API interface against which a heterogeneous set of clients can make requests.

Component-based architecture: In the frontend, we have focused on a component-based architecture. This approach structures the application as a collection of reusable and self-contained components, each managing its logic, rendering, and state. By adopting this architecture, we ensure that the code is modular, making it easier to maintain, update. Reusability is a key advantage, as components can be composed and reused across different sections of the app, leading to more efficient development and consistent user interfaces.

2.7. Other design decisions

Relational DBMS: We chose to integrate a Database Management System (DBMS) into our project to ensure efficient data management, storage, and retrieval. A DBMS provides a structured and scalable way to handle large volumes of data while maintaining data integrity, consistency, and security. It allows for multi-user access, concurrent data manipulation, and facilitates complex queries that enhance the project's functionality and performance. The choice of a DBMS also supports data backup and recovery, reducing the risk of data loss and ensuring system reliability. Overall, it is an essential component for building a robust, reliable, and scalable solution

3 | User interface design

3.1. Student Flows

3.1.1. Student Registration/Login and First Job Application Flow

- Students register by providing email, password, and selecting the profile type (`Student`).
- Students fill their personal info before applying to any job.
- Students browse job postings.
- Applications are confirmed upon submission, and students can track their status.

3.1.2. Matches Flow

- Students can review invitations from companies and explore job suggestions provided by the application tool.
- Students have the option to select a suitable job from the suggested matches and proceed with the application process.

3.1.3. Application Status and Feedback Flow

- Jobs requiring assessments allow students to submit responses to:
  - Open-ended questions.
  - Multiple-choice questions.
  - True/False questions.
- Students monitor the status of applications.
- Students can exchange feedback on Ongoing Jobs.
- Students can exchange feedback about the platform.

3.2. Company Flows

3.2.1. Company Registration/Login and Job Creation Flow

- Companies register by providing email, password, and selecting the profile type (`Company`).
- Companies fill their personal info before creating any job.
- Companies create jobs by providing:
  - Title, location, deadline, description, job type, and required skills.
- Additional questions (open-ended, multiple choice, true/false) can be added to applications.
- A list of posted jobs is available for management and tracking.

3.2.2. Candidate Matching and Invitation Flow

- Matches suggest suitable candidates for listed jobs.
- Invitations are sent to candidates.

3.2.3. Application Review Flow

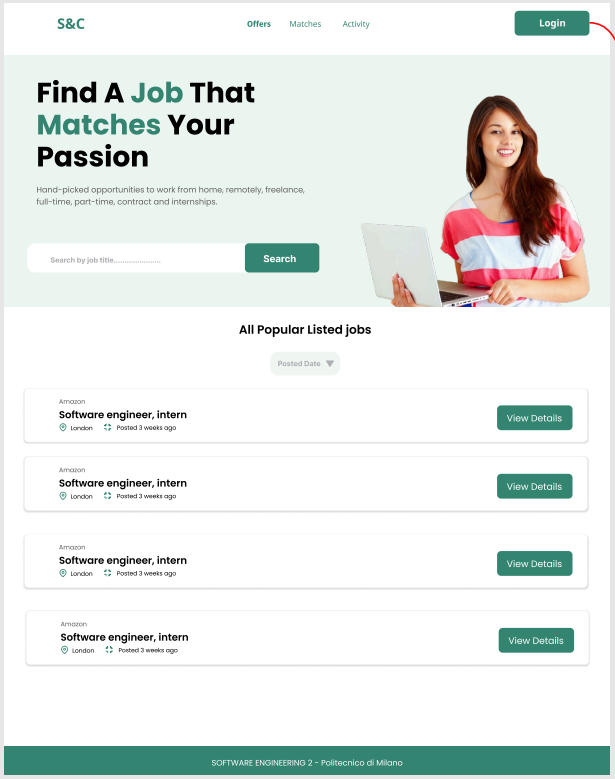
- Companies review applications.
- Applicants' details, including CV and skills, are displayed alongside submission dates and informations depending on the status.
- Applications can be accepted or rejected.
- Companies can add feedback on Ongoing jobs.

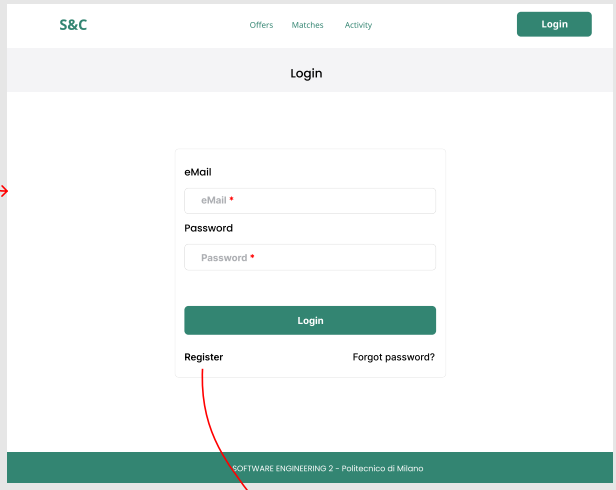
3.3. University Flows

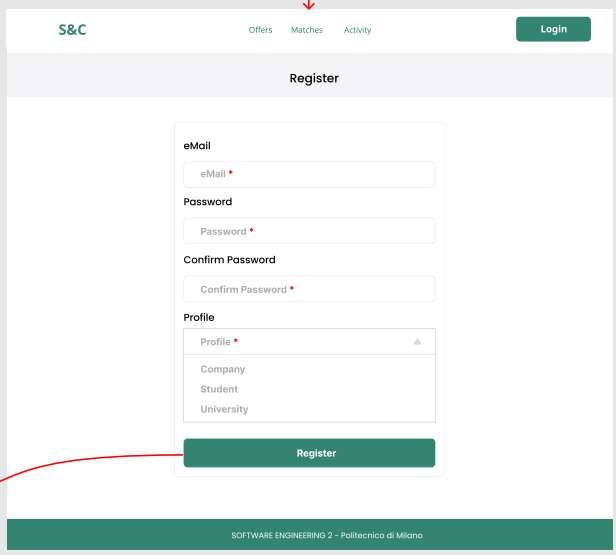
3.3.1. University Interaction Flow

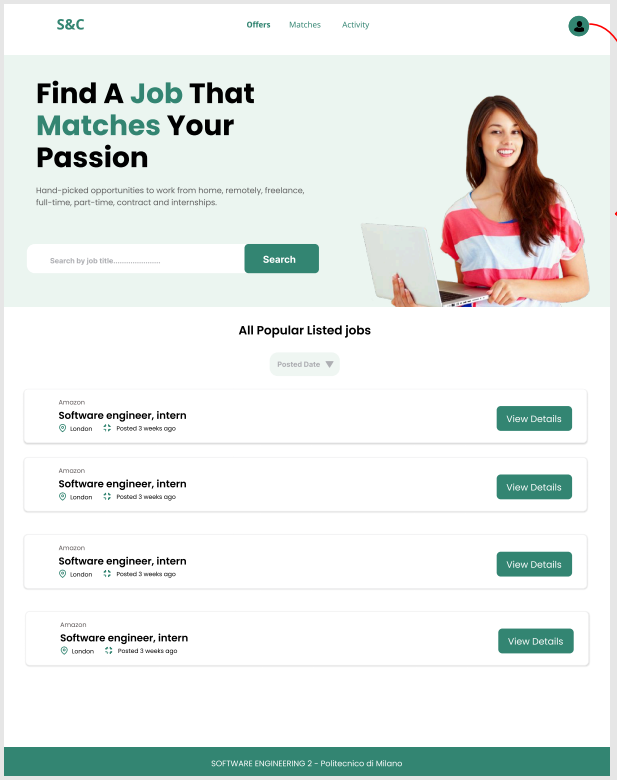
- Universities oversee the activities of their students, ensuring proper monitoring and progress tracking.
- Universities have access to detailed information about students, job roles, and their current statuses.
- Universities can send notifications to both students and companies to communicate important updates or deadlines.
- Universities can review feedback provided by both companies and students to ensure transparency and accountability.

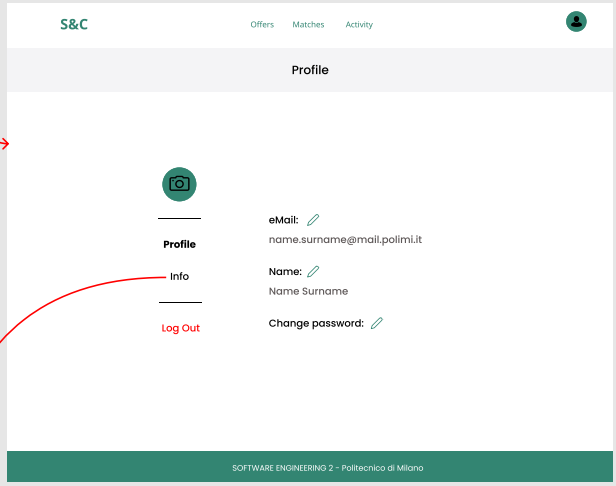
3.1.1


S&C
Offers
Matches
Activity
Login
Find A Job That Matches Your Passion
Hand-picked opportunities to work from home, remotely, freelance, full-time, part-time, contract and internships.
Search by job title.....
Search
All Popular Listed jobs
Posted Date
Amazon
Software engineer, intern
London
Posted 3 weeks ago
View Details
SOFTWARE ENGINEERING 2 - Politecnico di Milano


S&C
Offers
Matches
Activity
Login
Login
eMail
eMail *
Password
Password *
Login
Register
Forgot password?
SOFTWARE ENGINEERING 2 - Politecnico di Milano

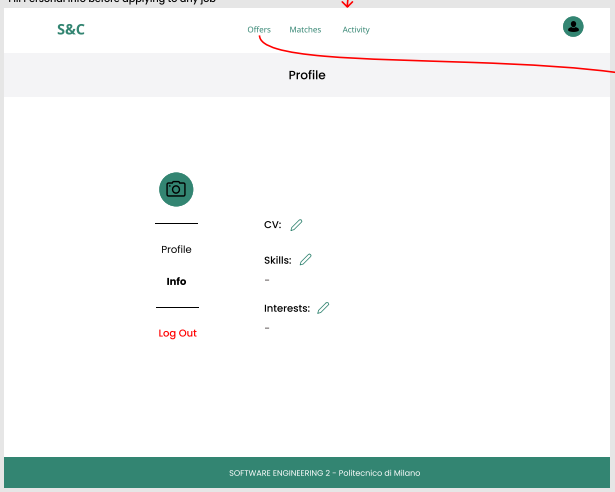

S&C
Offers
Matches
Activity
Login
Register
eMail
eMail *
Password
Password *
Confirm Password
Confirm Password *
Profile
Profile *
Company
Student
University
Register
SOFTWARE ENGINEERING 2 - Politecnico di Milano

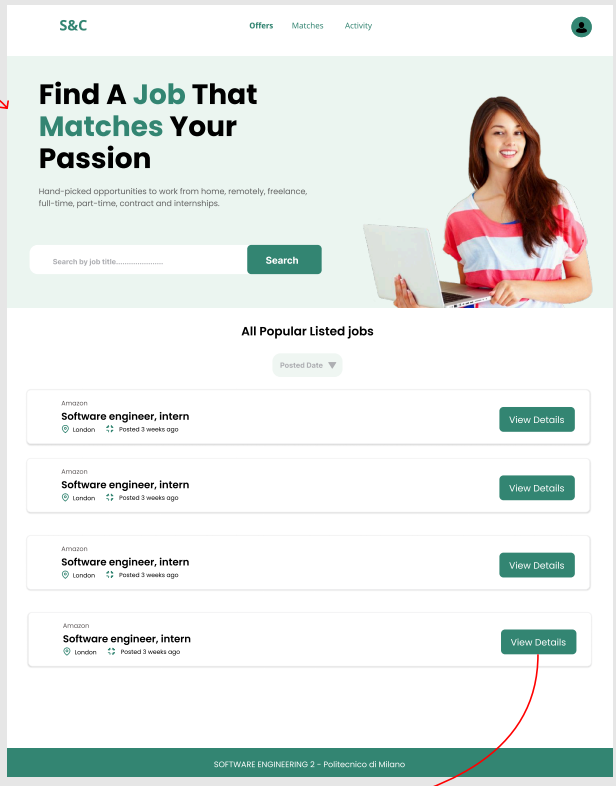

S&C
Offers
Matches
Activity
Find A Job That Matches Your Passion
Hand-picked opportunities to work from home, remotely, freelance, full-time, part-time, contract and internships.
Search by job title.....
Search
All Popular Listed jobs
Posted Date
Amazon
Software engineer, intern
London
Posted 3 weeks ago
View Details
SOFTWARE ENGINEERING 2 - Politecnico di Milano

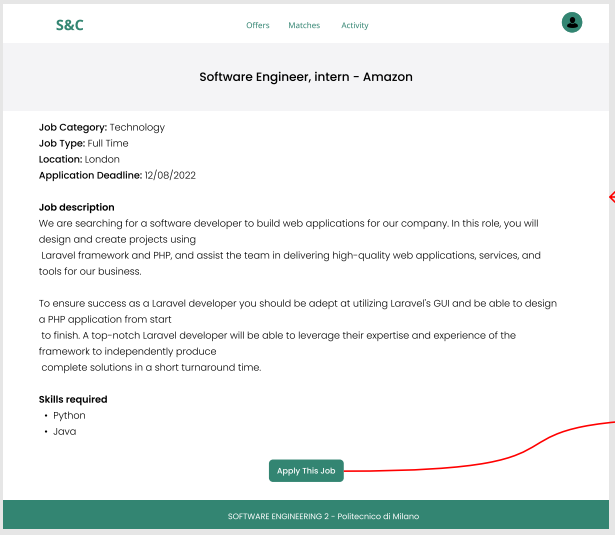

S&C
Offers
Matches
Activity
Profile
Profile
Info
Log Out
eMail:
name.surname@mail.polimi.it
Name:
Name Surname
Change password:
SOFTWARE ENGINEERING 2 - Politecnico di Milano

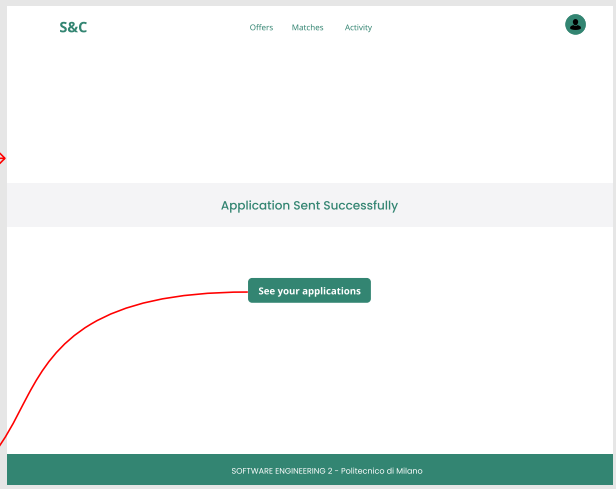
3.1.1

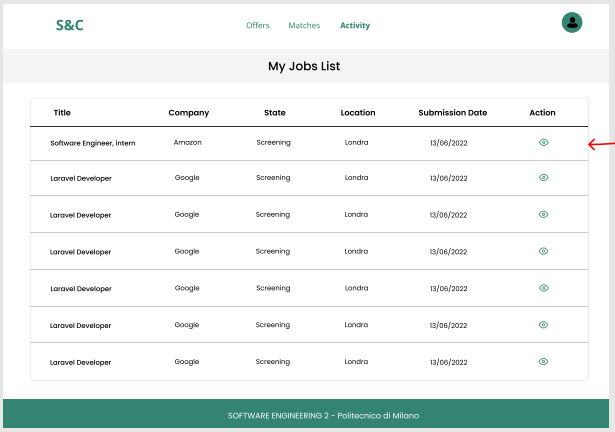
Fill Personal Info before applying to any job



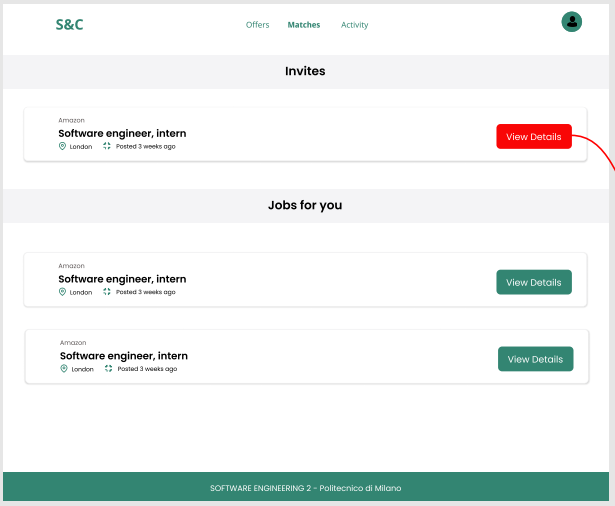


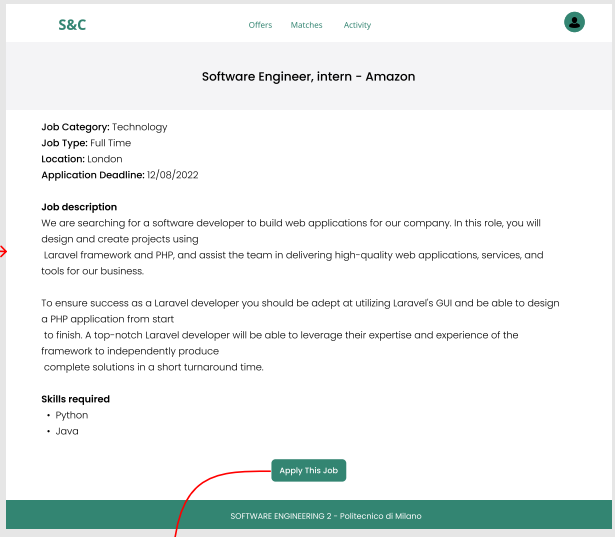


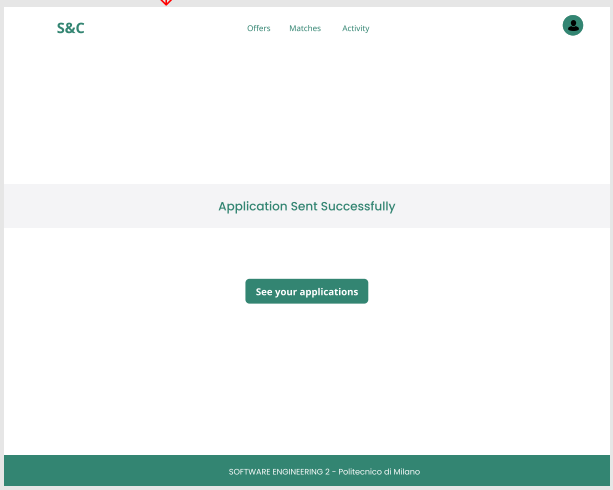




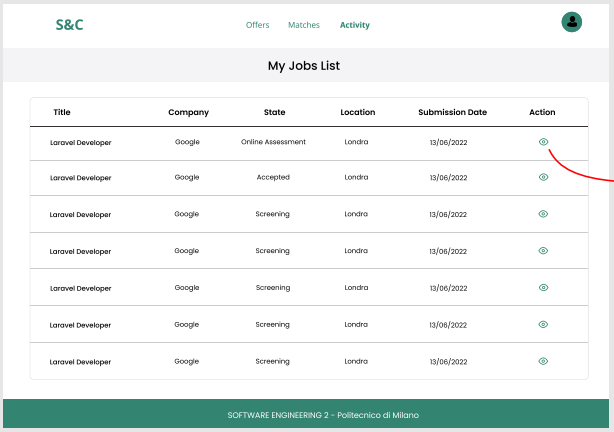
3.1.2

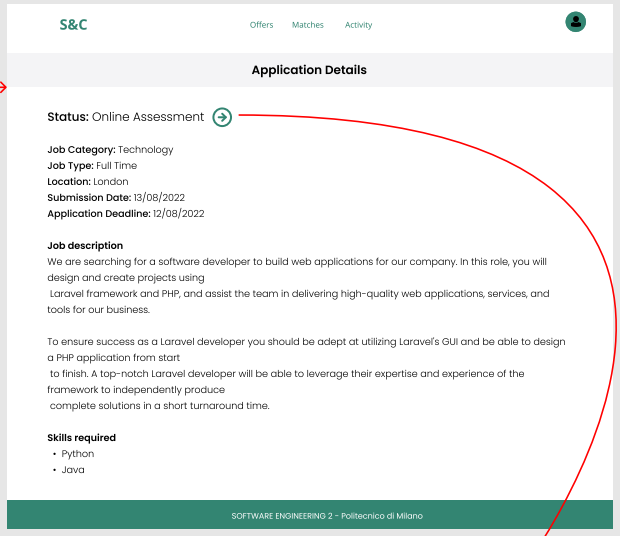

S&C
Offers
Matches
Activity
Invites
Amazon
Software engineer, intern
London
Posted 3 weeks ago
View Details
Jobs for you
Amazon
Software engineer, intern
London
Posted 3 weeks ago
View Details
Amazon
Software engineer, intern
London
Posted 3 weeks ago
View Details
SOFTWARE ENGINEERING 2 - Politecnico di Milano

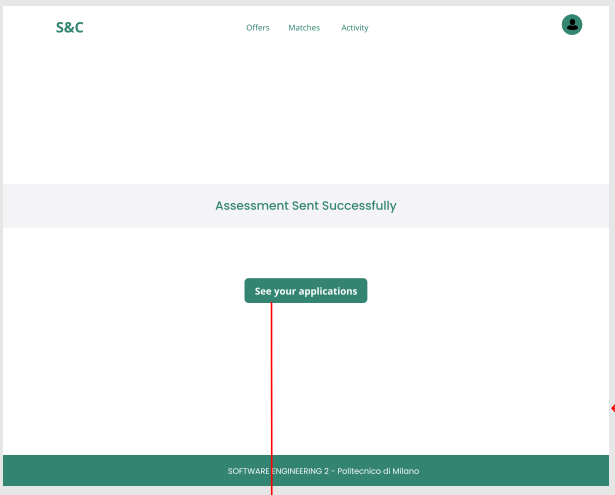

S&C
Offers
Matches
Activity
Software Engineer, intern - Amazon
Job Category: Technology
Job Type: Full Time
Location: London
Application Deadline: 12/08/2022
Job description
We are searching for a software developer to build web applications for our company. In this role, you will design and create projects using
Laravel framework and PHP, and assist the team in delivering high-quality web applications, services, and tools for our business.
To ensure success as a Laravel developer you should be adept at utilizing Laravel's GUI and be able to design a PHP application from start
to finish. A top-notch Laravel developer will be able to leverage their expertise and experience of the framework to independently produce
complete solutions in a short turnaround time.
Skills required
• Python
• Java
Apply This Job
SOFTWARE ENGINEERING 2 - Politecnico di Milano

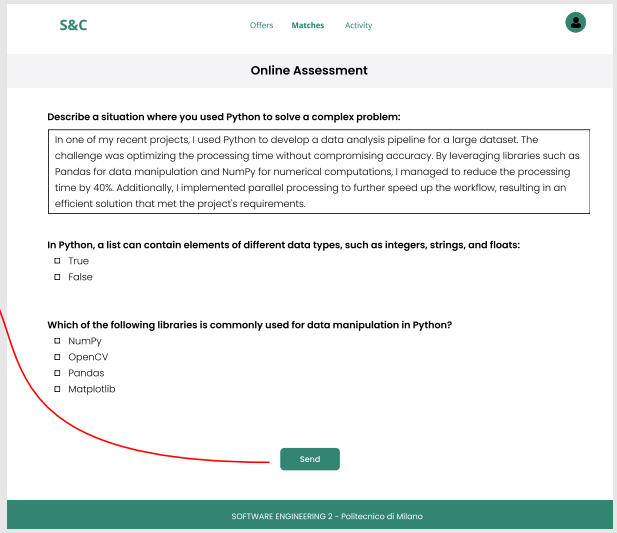

S&C
Offers
Matches
Activity
Application Sent Successfully
See your applications
SOFTWARE ENGINEERING 2 - Politecnico di Milano

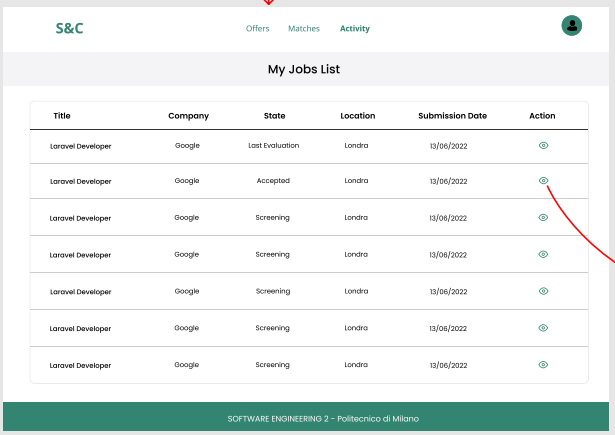
3.1.3

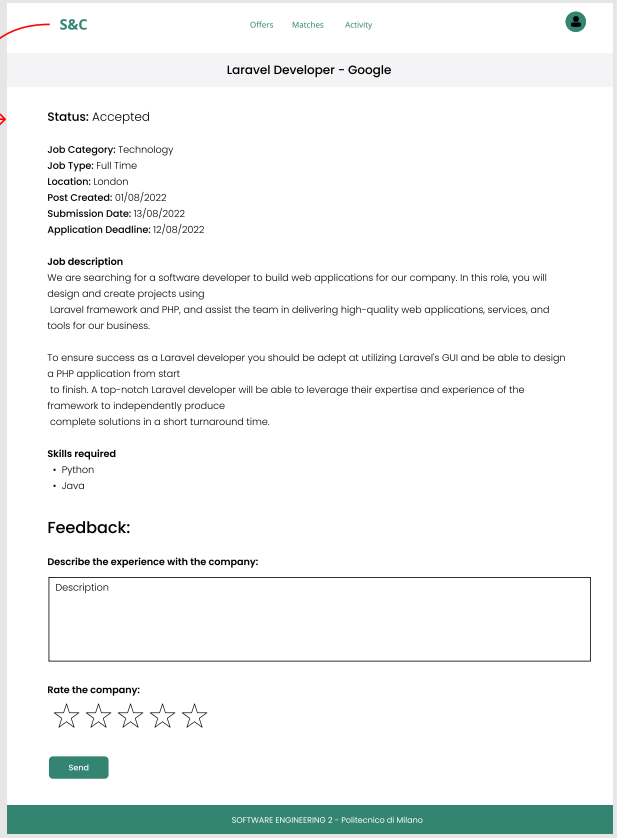

S&C
Offers
Matches
Activity
My Jobs List
Title
Company
State
Location
Submission Date
Action
Laravel Developer
Google
Online Assessment
Accepted
Screening
Londra
13/06/2022
SOFTWARE ENGINEERING 2 - Politecnico di Milano

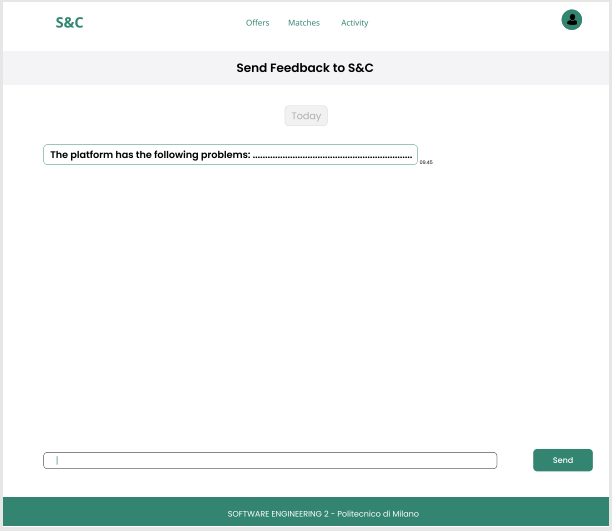

S&C
Offers
Matches
Activity
Application Details
Status: Online Assessment
Job Category: Technology
Job Type: Full Time
Location: London
Submission Date: 13/08/2022
Application Deadline: 12/08/2022
Job description
We are searching for a software developer to build web applications for our company. In this role, you will design and create projects using
Laravel framework and PHP, and assist the team in delivering high-quality web applications, services, and tools for our business.
To ensure success as a Laravel developer you should be adept at utilizing Laravel's GUI and be able to design a PHP application from start
to finish. A top-notch Laravel developer will be able to leverage their expertise and experience of the framework to independently produce
complete solutions in a short turnaround time.
Skills required
• Python
• Java
SOFTWARE ENGINEERING 2 - Politecnico di Milano


S&C
Offers
Matches
Activity
Assessment Sent Successfully
See your applications
SOFTWARE ENGINEERING 2 - Politecnico di Milano

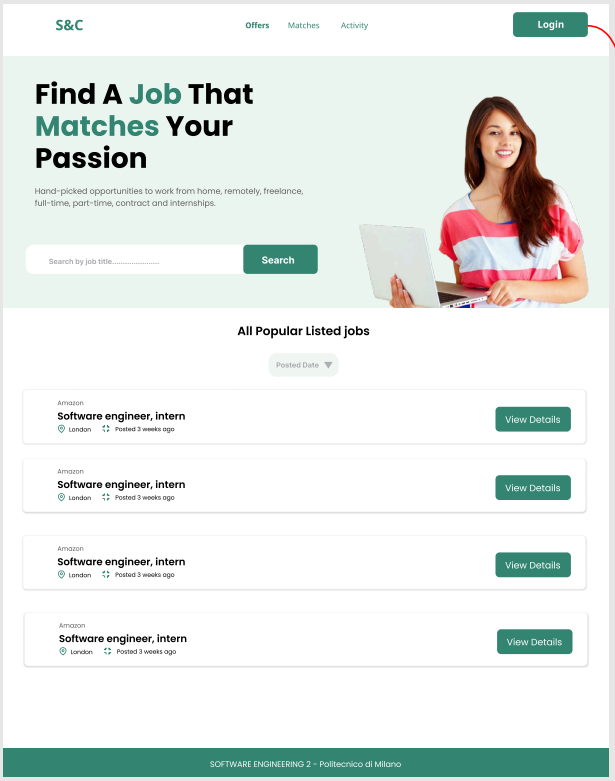

S&C
Offers
Matches
Activity
Online Assessment
Describe a situation where you used Python to solve a complex problem:
In one of my recent projects, I used Python to develop a data analysis pipeline for a large dataset. The challenge was optimizing the processing time without compromising accuracy. By leveraging libraries such as Pandas for data manipulation and NumPy for numerical computations, I managed to reduce the processing time by 40%. Additionally, I implemented parallel processing to further speed up the workflow, resulting in an efficient solution that met the project's requirements.
In Python, a list can contain elements of different data types, such as integers, strings, and floats:
True
False
Which of the following libraries is commonly used for data manipulation in Python?
NumPy
OpenCV
Pandas
Matplotlib
Send
SOFTWARE ENGINEERING 2 - Politecnico di Milano

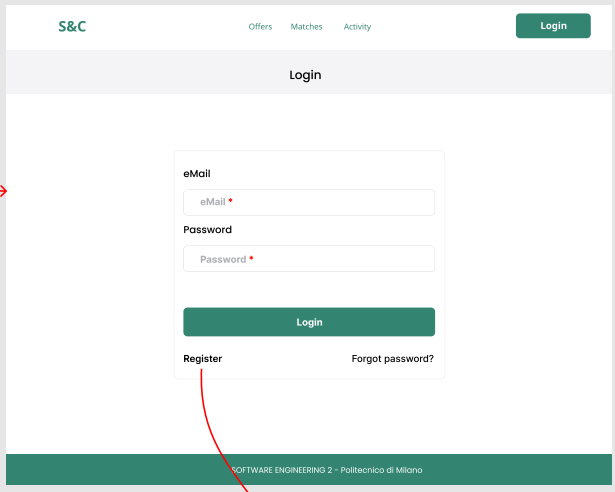

S&C
Offers
Matches
Activity
My Jobs List
Title
Company
State
Location
Submission Date
Action
Laravel Developer
Google
Last Evaluation
Accepted
Screening
Londra
13/06/2022
SOFTWARE ENGINEERING 2 - Politecnico di Milano

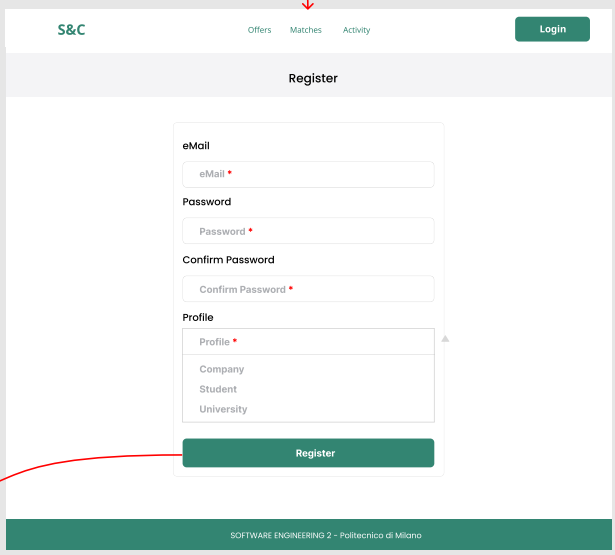

S&C
Offers
Matches
Activity
Laravel Developer - Google
Status: Accepted
Job Category: Technology
Job Type: Full Time
Location: London
Post Created: 01/08/2022
Submission Date: 13/08/2022
Application Deadline: 12/08/2022
Job description
We are searching for a software developer to build web applications for our company. In this role, you will design and create projects using
Laravel framework and PHP, and assist the team in delivering high-quality web applications, services, and tools for our business.
To ensure success as a Laravel developer you should be adept at utilizing Laravel's GUI and be able to design a PHP application from start
to finish. A top-notch Laravel developer will be able to leverage their expertise and experience of the framework to independently produce
complete solutions in a short turnaround time.
Skills required
• Python
• Java
Feedback:
Describe the experience with the company:
Description
Rate the company:
Send
SOFTWARE ENGINEERING 2 - Politecnico di Milano

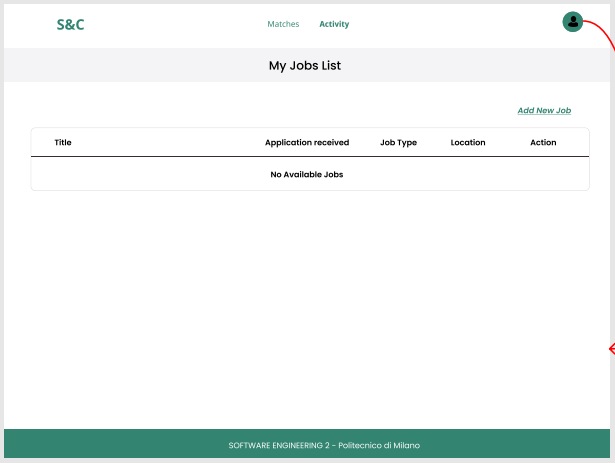

S&C
Offers
Matches
Activity
Send Feedback to S&C
Today
The platform has the following problems: .....
Send
SOFTWARE ENGINEERING 2 - Politecnico di Milano

3.2.1

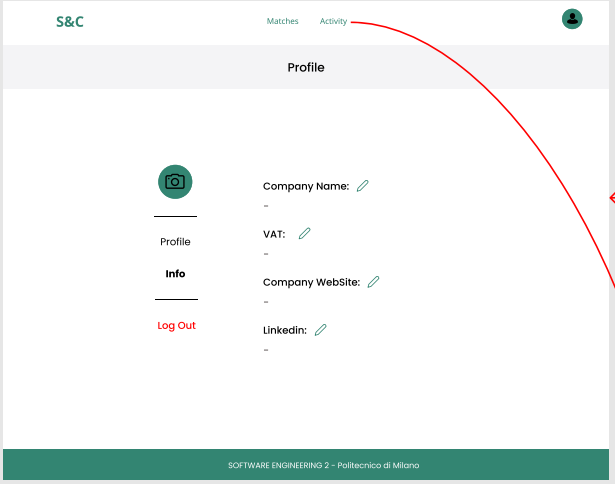


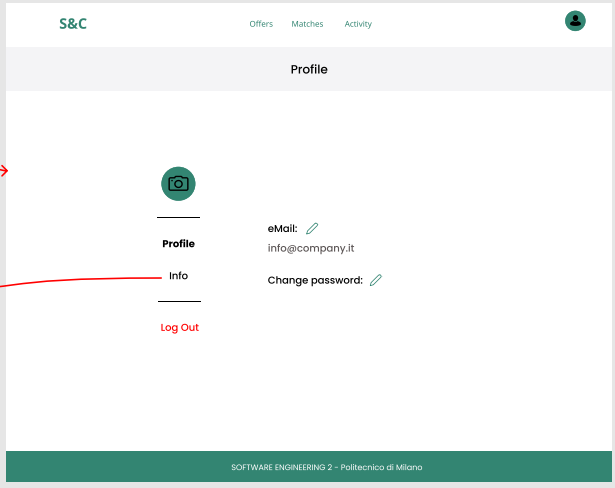




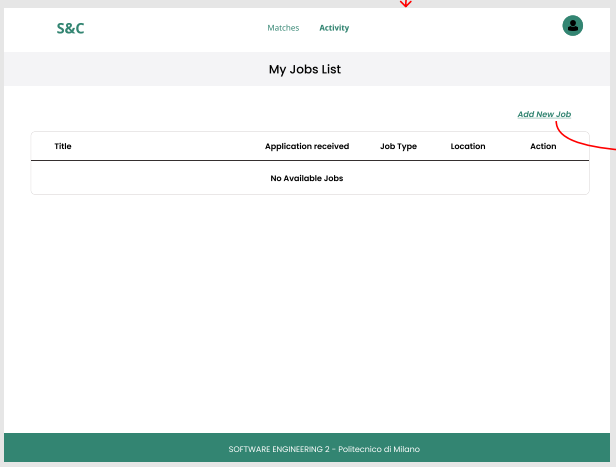


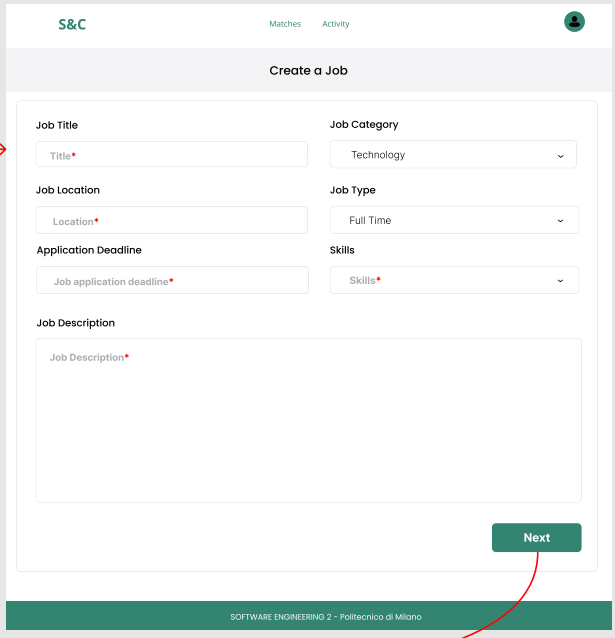
Fill Company Info before posting new jobs

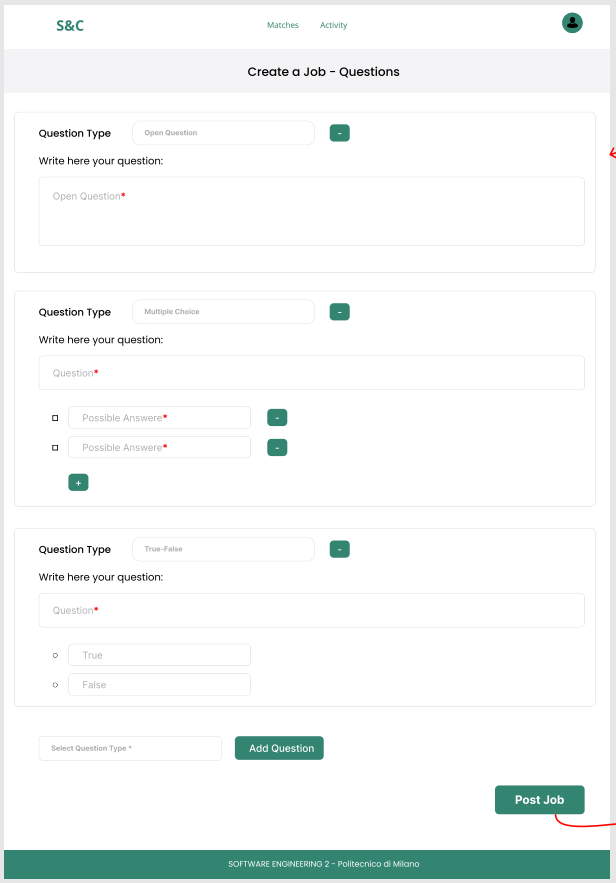


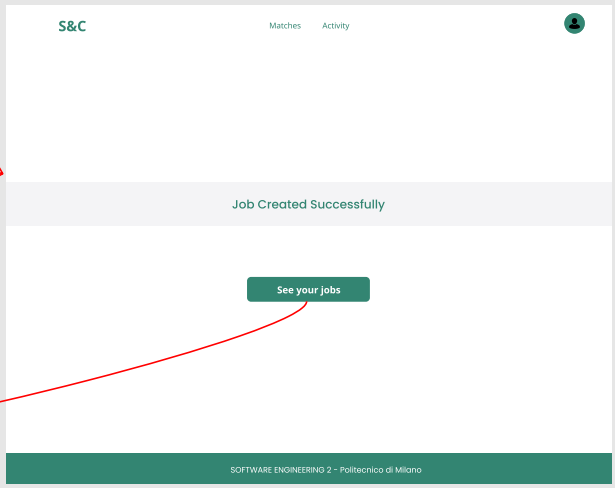


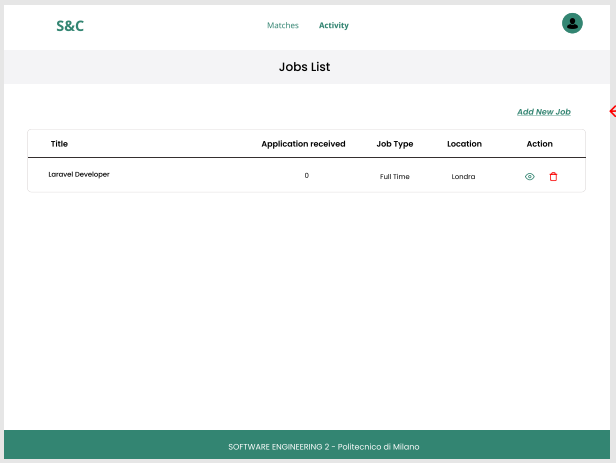
3.2.1


S&C
Matches
Activity
My Jobs List
Add New Job
Title
Application received
Job Type
Location
Action
No Available Jobs
SOFTWARE ENGINEERING 2 - Politecnico di Milano

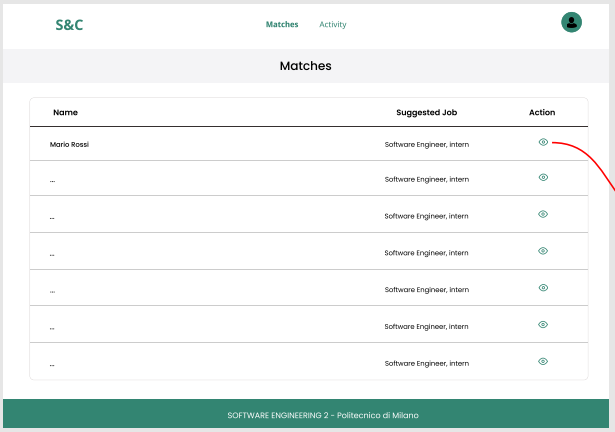

S&C
Matches
Activity
Create a Job
Job Title
Title*
Job Category
Technology
Job Location
Location*
Job Type
Full Time
Application Deadline
Job application deadline*
Skills
Skills*
Job Description
Job Description*
Next
SOFTWARE ENGINEERING 2 - Politecnico di Milano

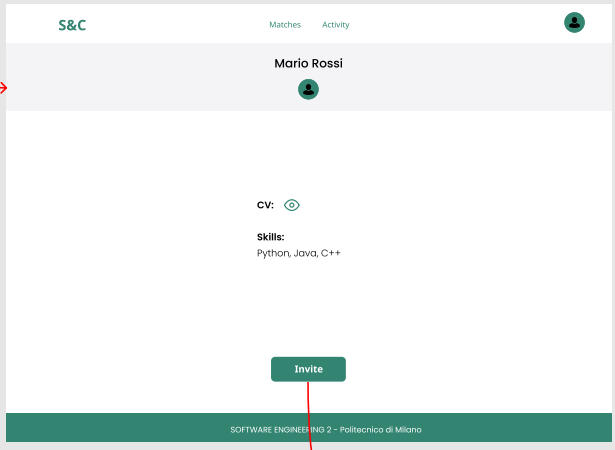

S&C
Matches
Activity
Create a Job - Questions
Question Type
Open Question
Write here your question:
Open Question*
Question Type
Multiple Choice
Write here your question:
Question*
Possible Answere*
Possible Answere*
Question Type
True-False
Write here your question:
Question*
True
False
Select Question Type *
Add Question
Post Job
SOFTWARE ENGINEERING 2 - Politecnico di Milano

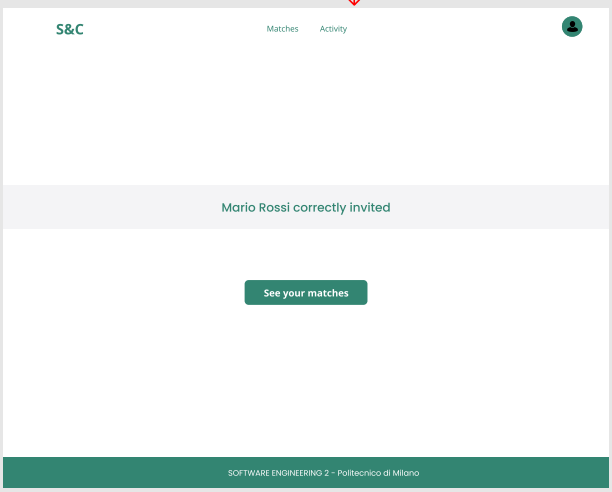

S&C
Matches
Activity
Job Created Successfully
See your jobs
SOFTWARE ENGINEERING 2 - Politecnico di Milano


S&C
Matches
Activity
Jobs List
Add New Job
Title
Application received
Job Type
Location
Action
Laravel Developer
0
Full Time
Londra
SOFTWARE ENGINEERING 2 - Politecnico di Milano

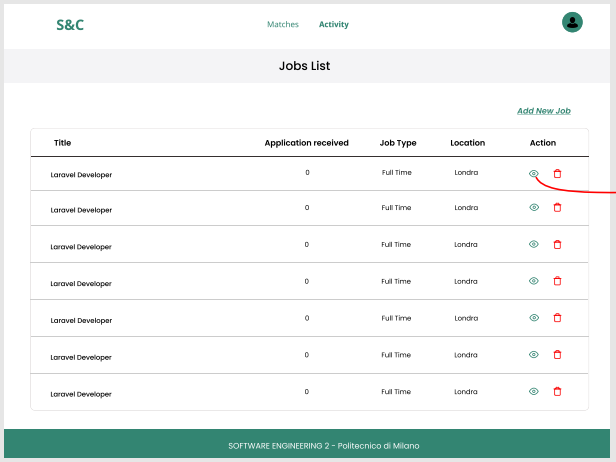
3.2.2

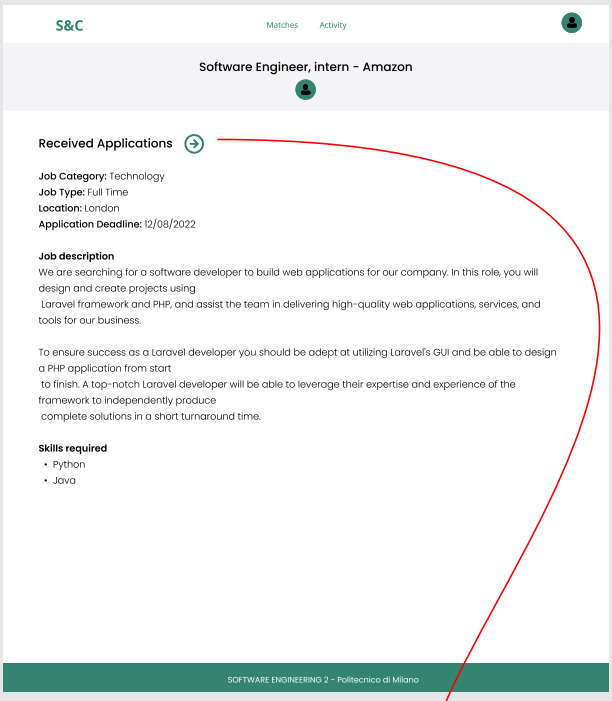

S&C
Matches
Activity
Matches
Name
Suggested Job
Action
Mario Rossi
Software Engineer, intern
...
Software Engineer, intern
..
Software Engineer, intern
..
Software Engineer, intern
...
Software Engineer, intern
..
Software Engineer, intern
..
Software Engineer, intern
SOFTWARE ENGINEERING 2 - Politecnico di Milano

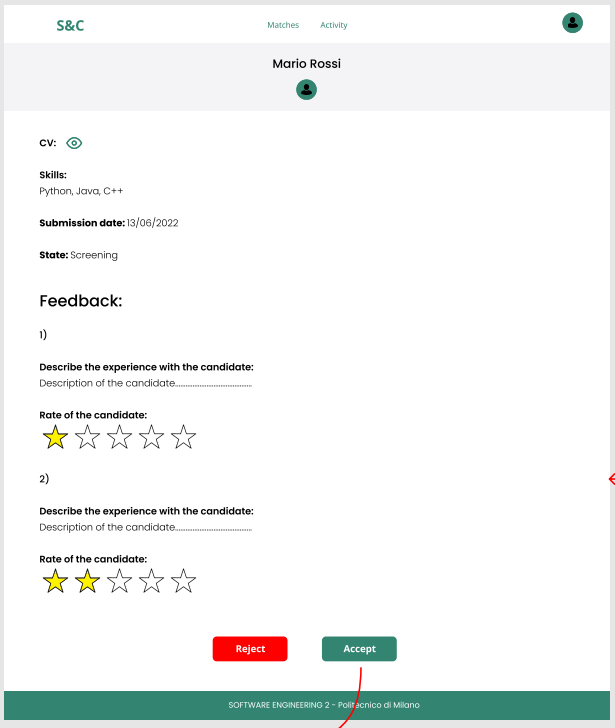

S&C
Matches
Activity
Mario Rossi
CV:
Skills:
Python, Java, C++
Invite
SOFTWARE ENGINEERING 2 - Politecnico di Milano

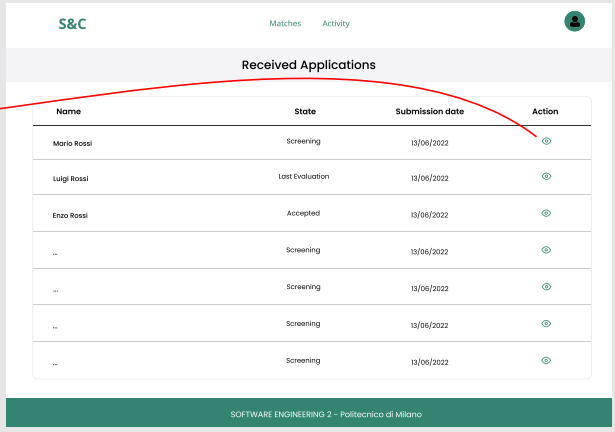

S&C
Matches
Activity
Mario Rossi correctly invited
See your matches
SOFTWARE ENGINEERING 2 - Politecnico di Milano

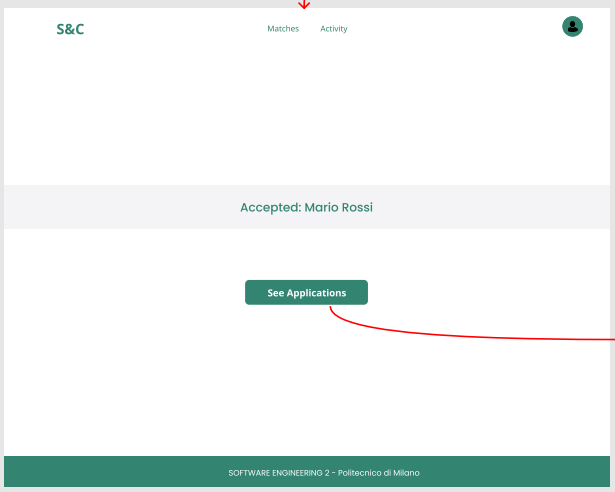
3.2.3

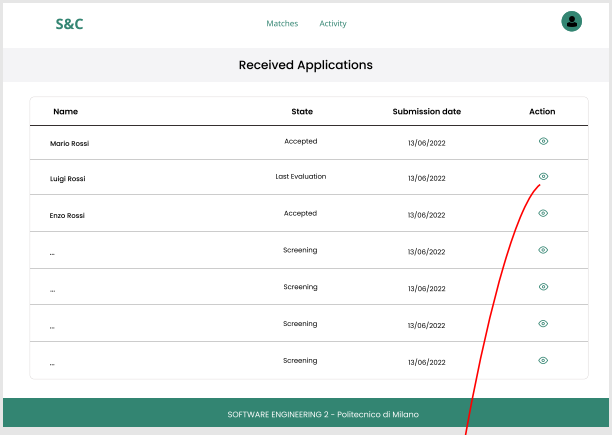

S&C
Matches
Activity
Jobs List
Add New Job
Title
Application received
Job Type
Location
Action
Laravel Developer
0
Full Time
Londra
SOFTWARE ENGINEERING 2 - Politecnico di Milano


S&C
Matches
Activity
Software Engineer, intern - Amazon
Received Applications
Job Category: Technology
Job Type: Full Time
Location: London
Application Deadline: 12/08/2022
Job description
We are searching for a software developer to build web applications for our company. In this role, you will design and create projects using
Laravel framework and PHP, and assist the team in delivering high-quality web applications, services, and tools for our business.
To ensure success as a Laravel developer you should be adept at utilizing Laravel's GUI and be able to design a PHP application from start
to finish. A top-notch Laravel developer will be able to leverage their expertise and experience of the framework to independently produce
complete solutions in a short turnaround time.
Skills required
Python
Java
SOFTWARE ENGINEERING 2 - Politecnico di Milano

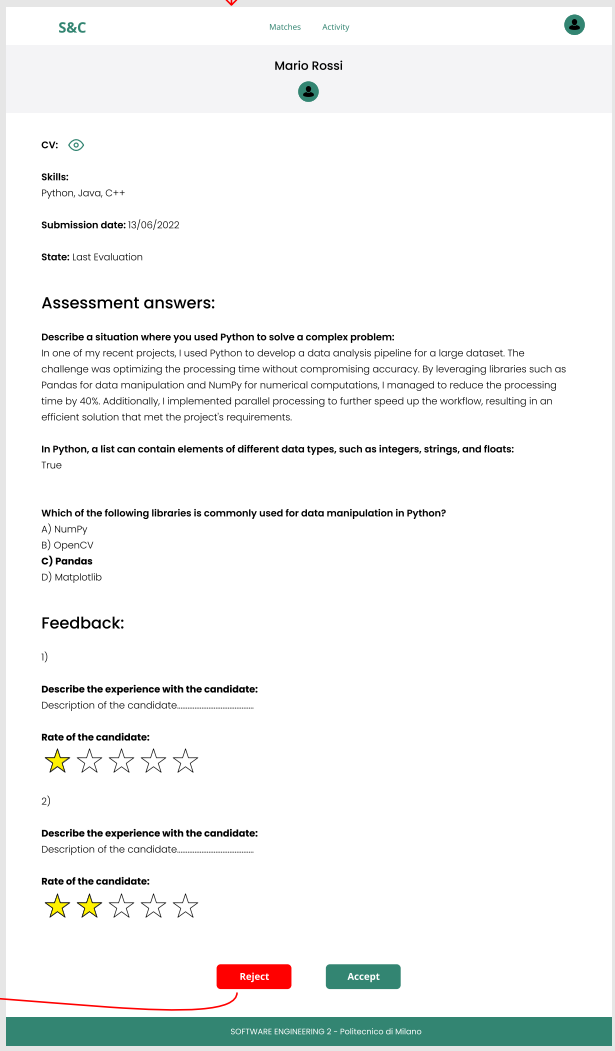

S&C
Matches
Activity
Mario Rossi
CV:
Skills:
Python, Java, C++
Submission date: 13/06/2022
State: Screening
Feedback:
1)
Describe the experience with the candidate:
Description of the candidate.....
Rate of the candidate:
2)
Describe the experience with the candidate:
Description of the candidate.....
Rate of the candidate:
Reject
Accept
SOFTWARE ENGINEERING 2 - Politecnico di Milano

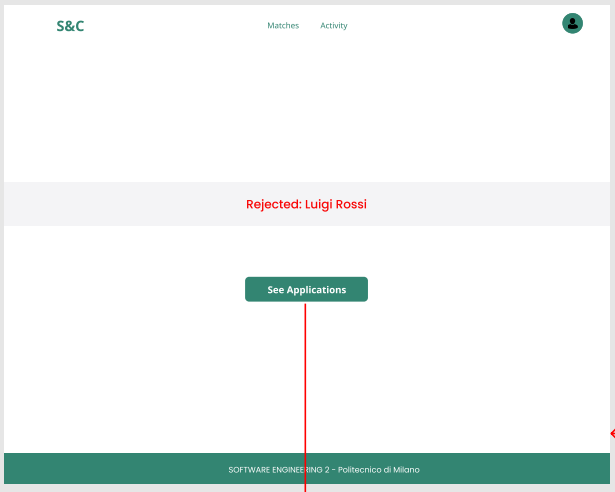

S&C
Matches
Activity
Received Applications
Name
State
Submission date
Action
Mario Rossi
Screening
13/06/2022
Luigi Rossi
Last Evaluation
13/06/2022
Enzo Rossi
Accepted
13/06/2022
SOFTWARE ENGINEERING 2 - Politecnico di Milano

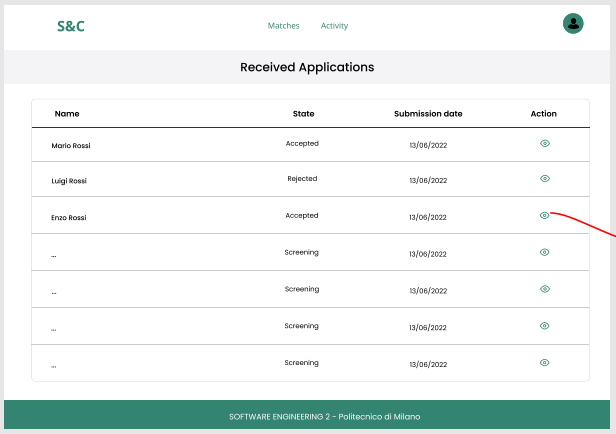

S&C
Matches
Activity
Accepted: Mario Rossi
See Applications
SOFTWARE ENGINEERING 2 - Politecnico di Milano

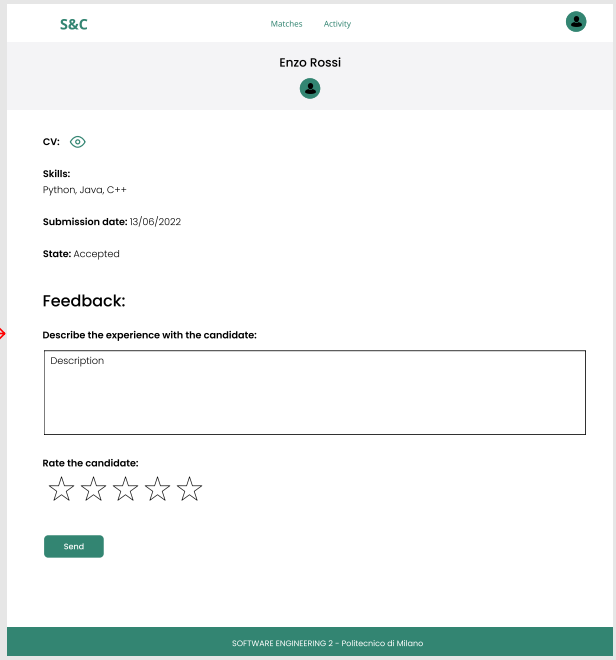

S&C
Matches
Activity
Received Applications
Name
State
Submission date
Action
Mario Rossi
Accepted
13/06/2022
Luigi Rossi
Last Evaluation
13/06/2022
Enzo Rossi
Accepted
13/06/2022
SOFTWARE ENGINEERING 2 - Politecnico di Milano

3.2.3

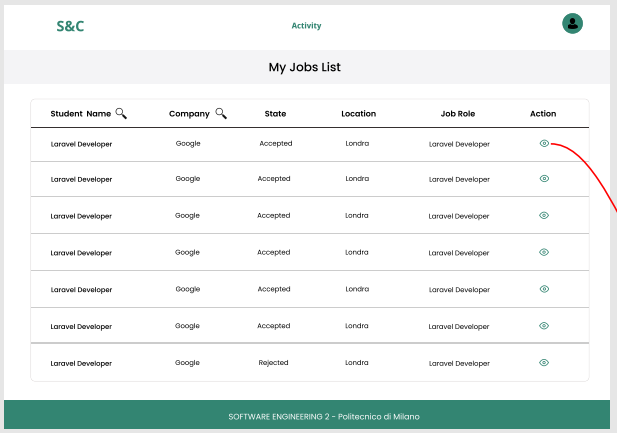

S&C
Matches
Activity
Mario Rossi
CV:
Skills:
Python, Java, C++
Submission date: 13/06/2022
State: Last Evaluation
Assessment answers:
Describe a situation where you used Python to solve a complex problem:
In one of my recent projects, I used Python to develop a data analysis pipeline for a large dataset. The challenge was optimizing the processing time without compromising accuracy. By leveraging libraries such as Pandas for data manipulation and NumPy for numerical computations, I managed to reduce the processing time by 40%. Additionally, I implemented parallel processing to further speed up the workflow, resulting in an efficient solution that met the project's requirements.
In Python, a list can contain elements of different data types, such as integers, strings, and floats:
True
Which of the following libraries is commonly used for data manipulation in Python?
A) NumPy
B) OpenCV
C) Pandas
D) Matplotlib
Feedback:
1)
Describe the experience with the candidate:
Description of the candidate.....
Rate of the candidate:
2)
Describe the experience with the candidate:
Description of the candidate.....
Rate of the candidate:
Reject
Accept
SOFTWARE ENGINEERING 2 - Politecnico di Milano

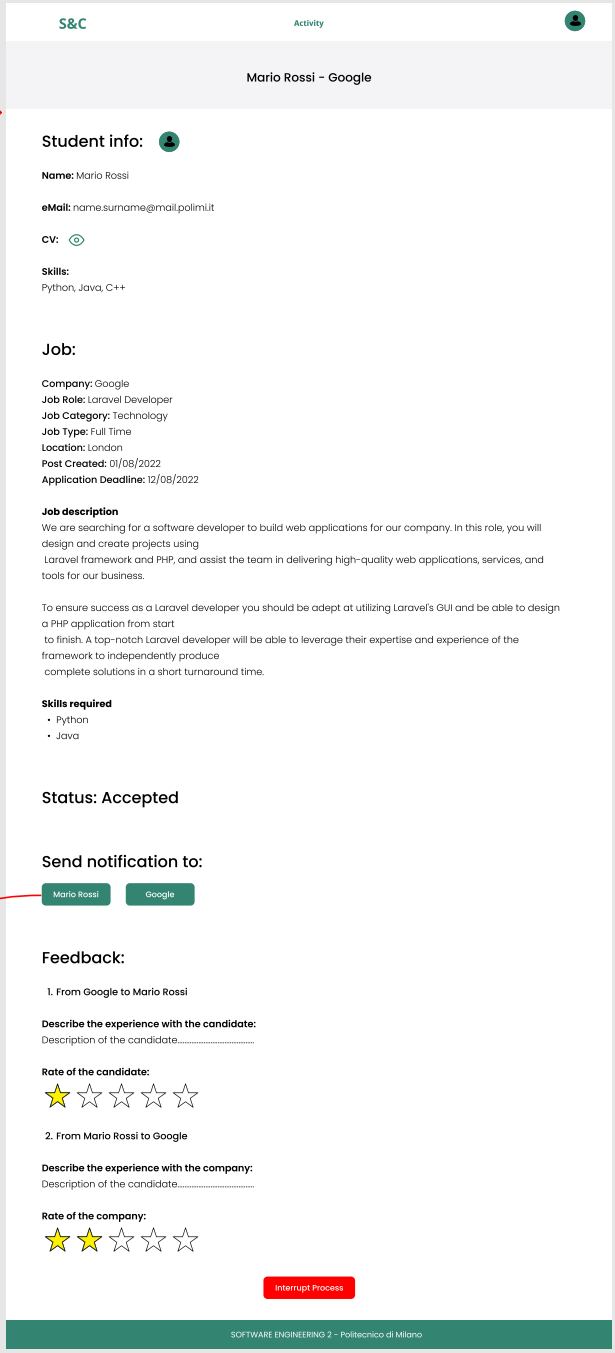

S&C
Matches
Activity
Rejected: Luigi Rossi
See Applications
SOFTWARE ENGINEERING 2 - Politecnico di Milano

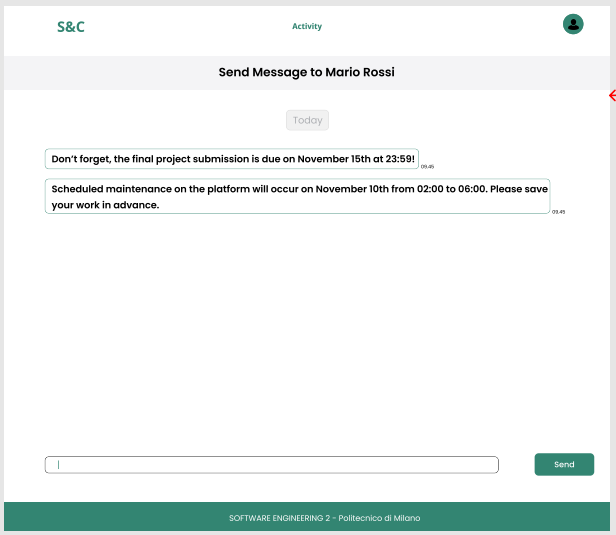

S&C
Matches
Activity
Received Applications
Name
State
Submission date
Action
Mario Rossi
Accepted
13/06/2022
Luigi Rossi
Rejected
13/06/2022
Enzo Rossi
Accepted
13/06/2022
...
Screening
13/06/2022
SOFTWARE ENGINEERING 2 - Politecnico di Milano


S&C
Matches
Activity
Enzo Rossi
CV:
Skills:
Python, Java, C++
Submission date: 13/06/2022
State: Accepted
Feedback:
Describe the experience with the candidate:
Description
Rate the candidate:
Send
SOFTWARE ENGINEERING 2 - Politecnico di Milano

3.3.1


S&C
Activity
My Jobs List
Student Name
Company
State
Location
Job Role
Action
Laravel Developer
Google
Accepted
Londra
Laravel Developer
Rejected
SOFTWARE ENGINEERING 2 - Politecnico di Milano


S&C
Activity
Mario Rossi - Google
Student info:
Name: Mario Rossi
eMail: name.surname@mail.polimi.it
CV:
Skills:
Python, Java, C++
Job:
Company: Google
Job Role: Laravel Developer
Job Category: Technology
Job Type: Full Time
Location: London
Post Created: 01/08/2022
Application Deadline: 12/08/2022
Job description
We are searching for a software developer to build web applications for our company. In this role, you will design and create projects using
Laravel framework and PHP, and assist the team in delivering high-quality web applications, services, and tools for our business.
To ensure success as a Laravel developer you should be adept at utilizing Laravel's GUI and be able to design a PHP application from start
to finish. A top-notch Laravel developer will be able to leverage their expertise and experience of the framework to independently produce
complete solutions in a short turnaround time.
Skills required
• Python
• Java
Status: Accepted
Send notification to:
Mario Rossi
Google
Feedback:
1. From Google to Mario Rossi
Describe the experience with the candidate:
Description of the candidate.....
Rate of the candidate:
2. From Mario Rossi to Google
Describe the experience with the company:
Description of the candidate.....
Rate of the company:
Interrupt Process
SOFTWARE ENGINEERING 2 - Politecnico di Milano


S&C
Activity
Send Message to Mario Rossi
Today
Don't forget, the final project submission is due on November 15th at 23:59!
Scheduled maintenance on the platform will occur on November 10th from 02:00 to 06:00. Please save your work in advance.
Send
SOFTWARE ENGINEERING 2 - Politecnico di Milano

4 | Requirements traceability

Requirements	Description	Components
R9, R10	Allow students to view internship recommendations and trends.	Student Controller
R5, R6	Allow students to create/edit profiles, upload CVs, and receive validation.	Student Controller
R8, R19	Provide a search tool with filtering options for internships.	Student Controller, Internship Controller
R12, R18	Guide students through the application process and track application statuses.	Student Controller, Internship Controller
R15, R16, R28, R29	Companies can create, edit, and manage internship postings with deadlines, and various criteria.	Company Controller, Internship Controller
R17, R20	Enable companies to access ad hoc pages for reviewing applications and analyzing applicants.	Company Controller
R19, R9	Provide a recommendation feed for companies to find candidates and view suitable profiles.	Company Controller, Internship Controller, Match Controller
R24, R25	Universities can monitor internship progress, manage complaints, and track student engagement.	Internship Controller, Feedback Controller
R26, R27	Integrate securely with university databases and external platforms for data verification and feedback collection.	Internship Controller, Feedback Controller

R10, R9	Improve recommendation engine and notify students of suitable internship.	Internship Controller, Student Controller
R16, R21	Allow companies to manage interview schedules and outcomes, including feedback to the platform.	Company Controller, Feedback Controller
R22, R23	Collect and display post-internship feedback and endorsements.	Feedback Controller
R1, R2, R3	Implement security measures to the login and registration of the account.	Authentication Controller
R23	Display top-rated internships based on feedback.	Feedback Controller, Internship Controller
R7	Assessment tool for answering the application answers.	Student Controller
R10, R19	Rank and score internship recommendations for students.	Internship Controller, Match Controller

5 | Implementation, integration and test plan

In this section, we outline the implementation strategy, the integration methodology, and the comprehensive testing plan for the job boarding web application, developed using React for the frontend and ASP.NET Core as a monolithic server for the backend. The architecture chosen ensures high decoupling of code, primarily facilitated by dependency injection.

5.0.1. Development Plan

The development process will follow a bottom-up approach, focusing on the incremental building of backend components and simultaneous development of the frontend. The backend is structured into multiple sections, each serving a distinct purpose and responsible for various features. The key features to be developed are as follows:

- **Authentication Management**: Manages user registration, login, and session handling. It is foundational, as most other features depend on authenticated access.
- **Student Information Management**: Handles CRUD operations for student profiles, including their personal details, resumes, and application history.
- **Company Information Management**: Supports company profiles
- **Internship Management**: Manages internship postings, applications, and status updates.
- **Matching Management**: Manages matching processes and recommendation improving system.
- **Feedback Management**: Manages the feedback system that guarantees students and companies to provide feedbacks on the platform and on companies(for students) and students(for companies) .

Each feature will be developed in sequence, respecting dependencies (e.g., the Authentication Module must be implemented before Student and Company Information Management). Development will be conducted alongside unit testing for each component to ensure correctness and stability. Additionally, testing of the backend functionalities is performed using Postman to validate API endpoints, ensure expected behavior, and facilitate the later integration with the frontend components.

5.0.2. Component Integration and Testing

Once individual components are developed and unit tested, the integration phase begins. Integration will be performed in stages:

- **Phase 1: Backend Integration**: Integrate and communicate between the several backend sections developed.
- **Phase 2: Backend-Frontend Integration**: Connect the frontend React application to the backend API endpoints. This step involves verifying that data requests and responses between the client and server are functioning as intended.

During this phase, integration tests will be conducted to verify that the combined components function as expected when communicating with each other.

5.0.3. System and End-to-End (E2E) Testing

Once all components have been successfully integrated, system testing will be conducted to ensure that the complete solution functions cohesively. This phase focuses on verifying that all integrated components interact correctly and perform as expected in a unified environment. System testing will comprehend the following type of tests:

- **Load Testing**: Assess the system's ability to handle varying levels of user traffic and operational demands, ensuring stability under stress.
- **Performance Testing**: Measure the system's response times and performance metrics under both typical and peak usage conditions to ensure reliability and scalability.

End-to-End (E2E) testing will simulate real user interactions with the application through the frontend interface, validating the application's functionality, workflow, and overall behavior from start to finish.

5.0.4. Acceptance Testing

The final phase of testing is acceptance testing, which involves stakeholders and representative users. This testing stage is crucial to ensure that the system meets the business requirements and user expectations. Once a beta version of the software is available, the acceptance testing process will proceed as follows:

- **User-Based Scenarios**: Real users, including stakeholders, will perform typical tasks on the system (e.g., creating an account, browsing job postings, and applying for internships).
- **Feedback Collection**: User feedback will be gathered to identify usability issues, missing features, or areas for improvement.
- **Validation**: Confirm that the system meets the acceptance criteria defined in the project requirements.

Acceptance testing will conclude with a review meeting, where results and feedback are

discussed, leading to necessary adjustments before the final release.

6 | Effort spent

Member of Group	Effort Spent	Hours
Giovanni Vaccarino	Introduction	0h
	Architectural Design	10h
	User Interface Design	1h
	Requirements Traceability	1h
	Implementation, Integration and Test Plan	5h
Vittorio Palladino	Introduction	3h
	Architectural Design	8h
	User Interface Design	0h
	Requirements Traceability	2h
	Implementation, Integration and Test Plan	1h
Nicolò Vacis	Introduction	0h
	Architectural Design	8h
	User Interface Design	6h
	Requirements Traceability	1h
	Implementation, Integration and Test Plan	1h

Table 6.1: Effort spent by each member of the group

7 References

7.1. References

- **2024-2025 Software Engineering 2** - RASD Document
- **Aws RDS** - https://aws.amazon.com/it/rds/
- **Aws SES** - https://aws.amazon.com/it/ses/
- **Aws S3** - https://aws.amazon.com/it/s3/
- **Fat and thin client** -https://www.geeksforgeeks.org/difference-between-thin-clients-and-thick-clients/
- **Monolithical vs microservices** - https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith
- **CDN** - https://www.ibm.com/it-it/topics/content-delivery-networks
- **NGINX** - https://nginx.org/en/
- **Kuberneetes** - https://kubernetes.io/docs/home/
- **UX principles** - https://www.uxdesigninstitute.com/blog/ux-design-principles/