Students & Companies

Project General Info

R&DD + IT

Repository -> https://github.com/Slaitroc/RicciPaoliGrisoni

Group Members:

- Lorenzo Ricci -> https://github.com/Slaitroc
- Matteo Giovanni Paoli -> https://github.com/Krotox
- Samuele Grisoni -> https://github.com/dedepivot

Documents:

- RASDv1.1
- DDv1.1
- ITDv1.1
- ATDv1

RASD

DD

Software Design Documentation

Product Name	Students & Companies
Date Updated	07/01/2025
Written By	RicciPaoliGrisoni

Architectural Styles and Paradigms

The purpose of this document is to provide a comprehensive overview of the software design for the XYZ application. This includes the system overview, design considerations, specifications, detailed design, implementation plan, testing plan, and maintenance plan.

Interfaces

The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

System Components

The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

Unit and Integration Test

ITD

UML

Sequence Diagrams

Goals

Requirements

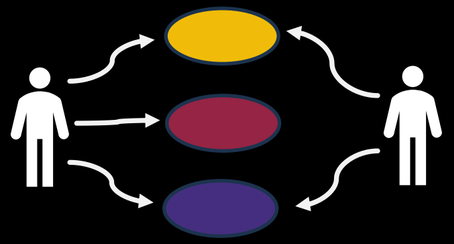
User Interface

User Scenarios

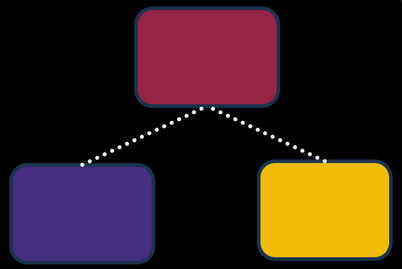
RASD

Within our Requirements and Analysis Specification Document, we provide a detailed description of the domain along with the necessary requirements that our platform must meet to achieve its intended goals.

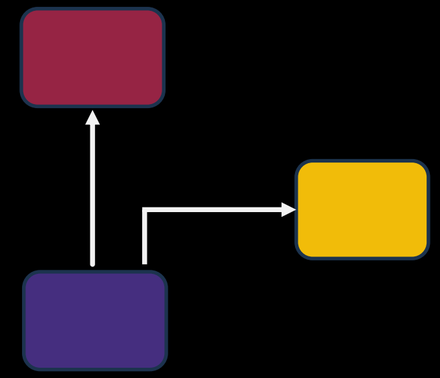
Use Case Diagrams



Alloy

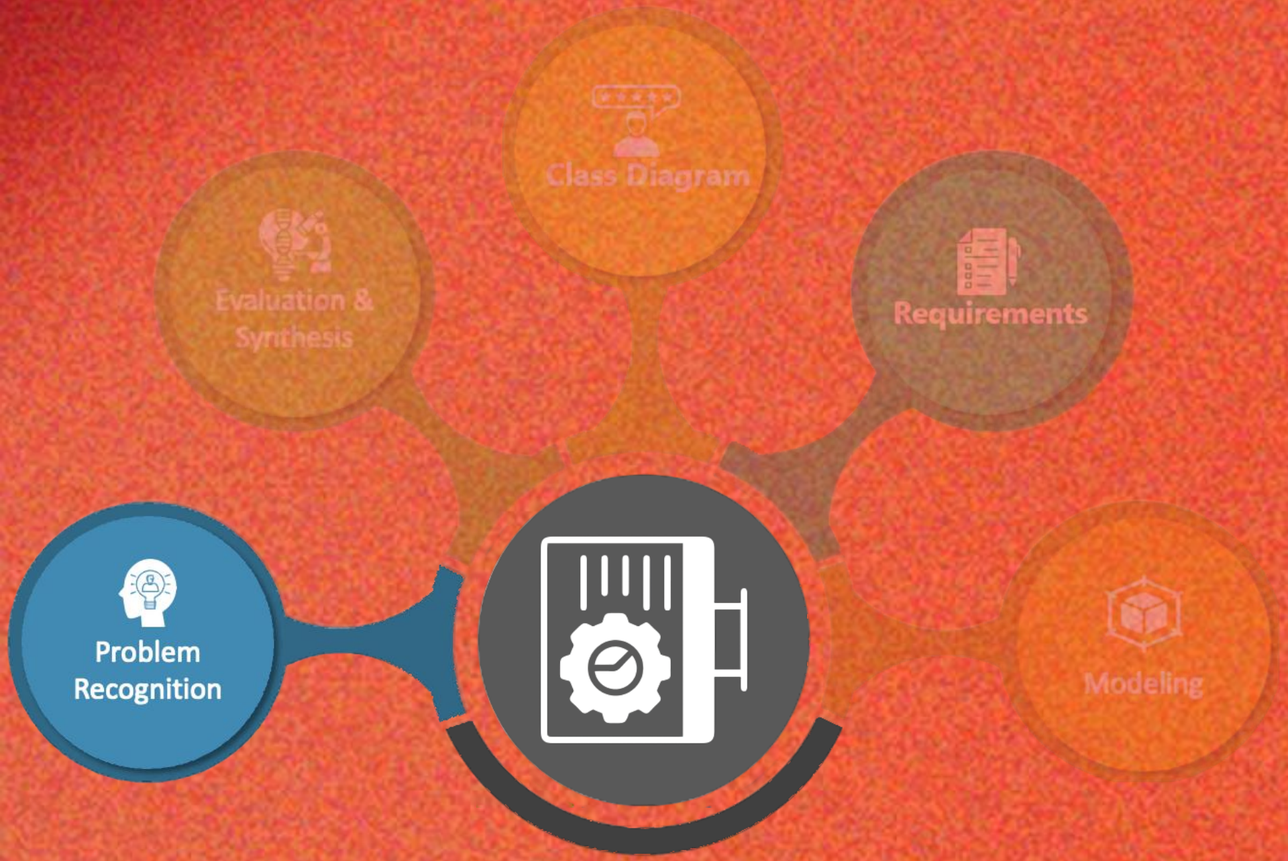


SW & HW Dependencies



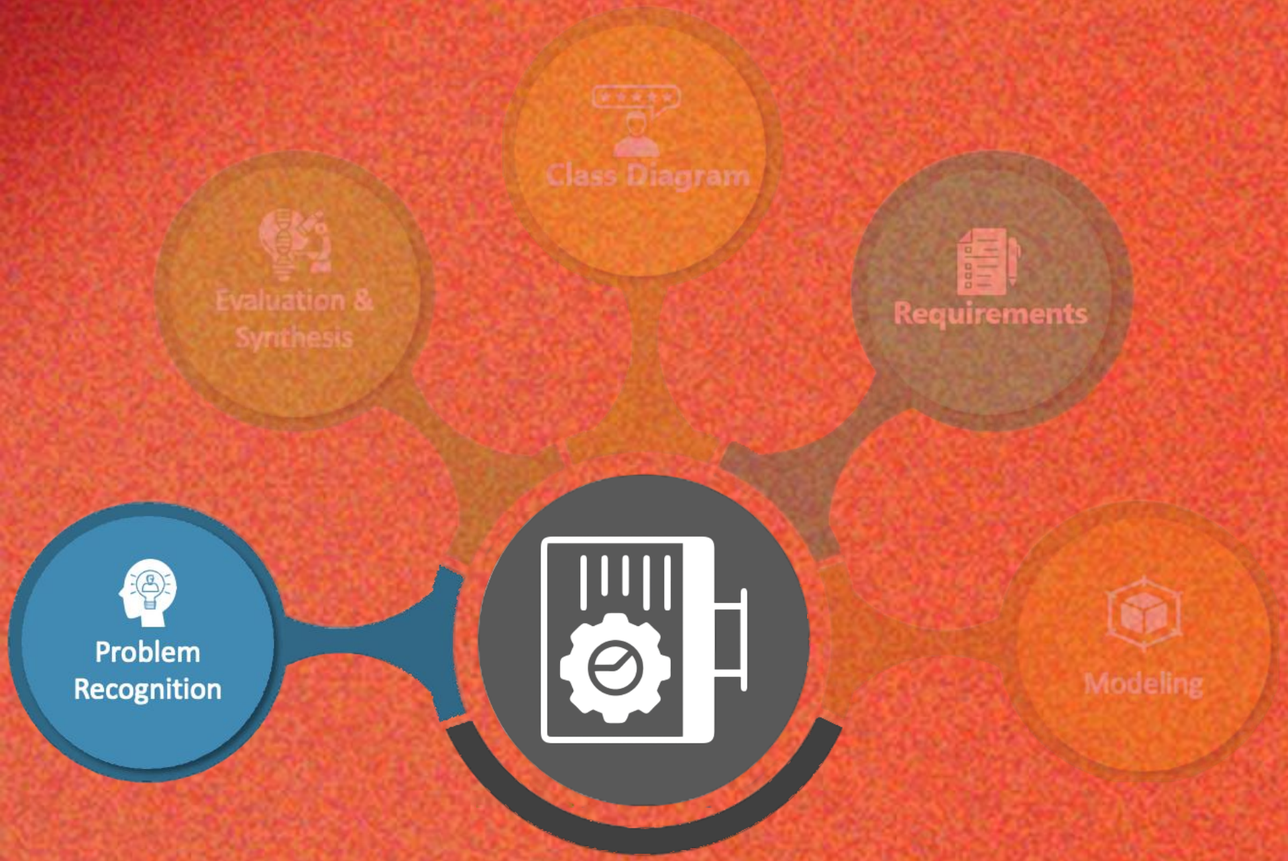
Class Diagram

Domain Class Diagram



GOALs

1. Companies would like to advertise the internships they offer.
2. Students would like to autonomously candidate for available internships.
3. Students would like to be matched with internships they might be interested in.
4. Companies would like to perform interviews with suitable students.
5. Students and companies would like to complain, communicate problems, and provide information about an ongoing internship.



GOALs

6. Students and companies would like to be provided with suggestions about how to improve their submission

7. Universities would like to handle complaints about ongoing internships.

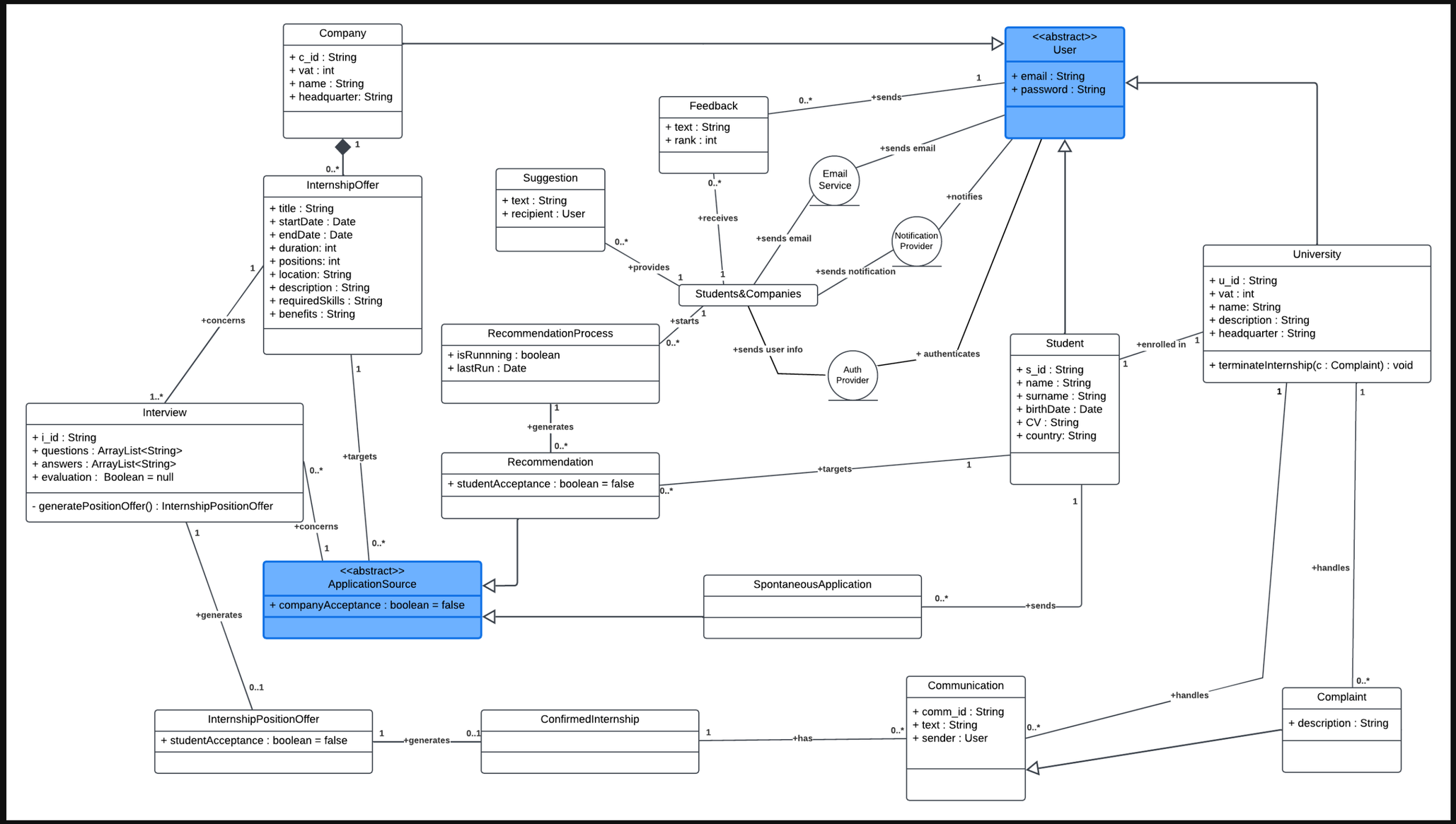
8. Students would like to choose which internship to attend among those for which they passed the interview.

9. Companies would like to select students for the internship position among those who passed the interview.

CLASS DIAGRAM

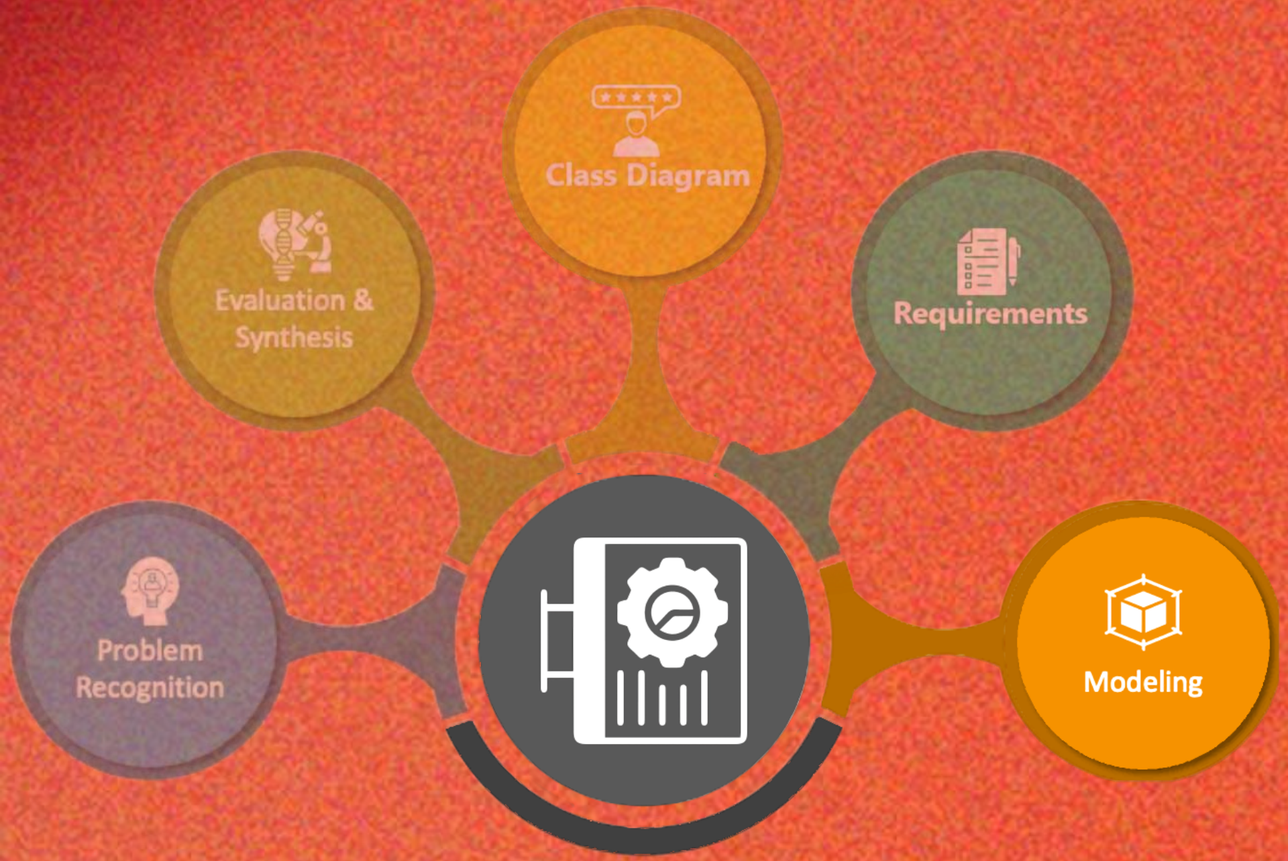
- Represents key entities: Student, Company, University, Internship, Interview, Complaint.
- Displays key relationships between entities
- Highlights core attributes such as personal information, internship details, and interview outcomes
- Shows associations between users and platform actions like applications, offers, and complaints.

CLASS DIAGRAM



Recommendation Process Use Cases

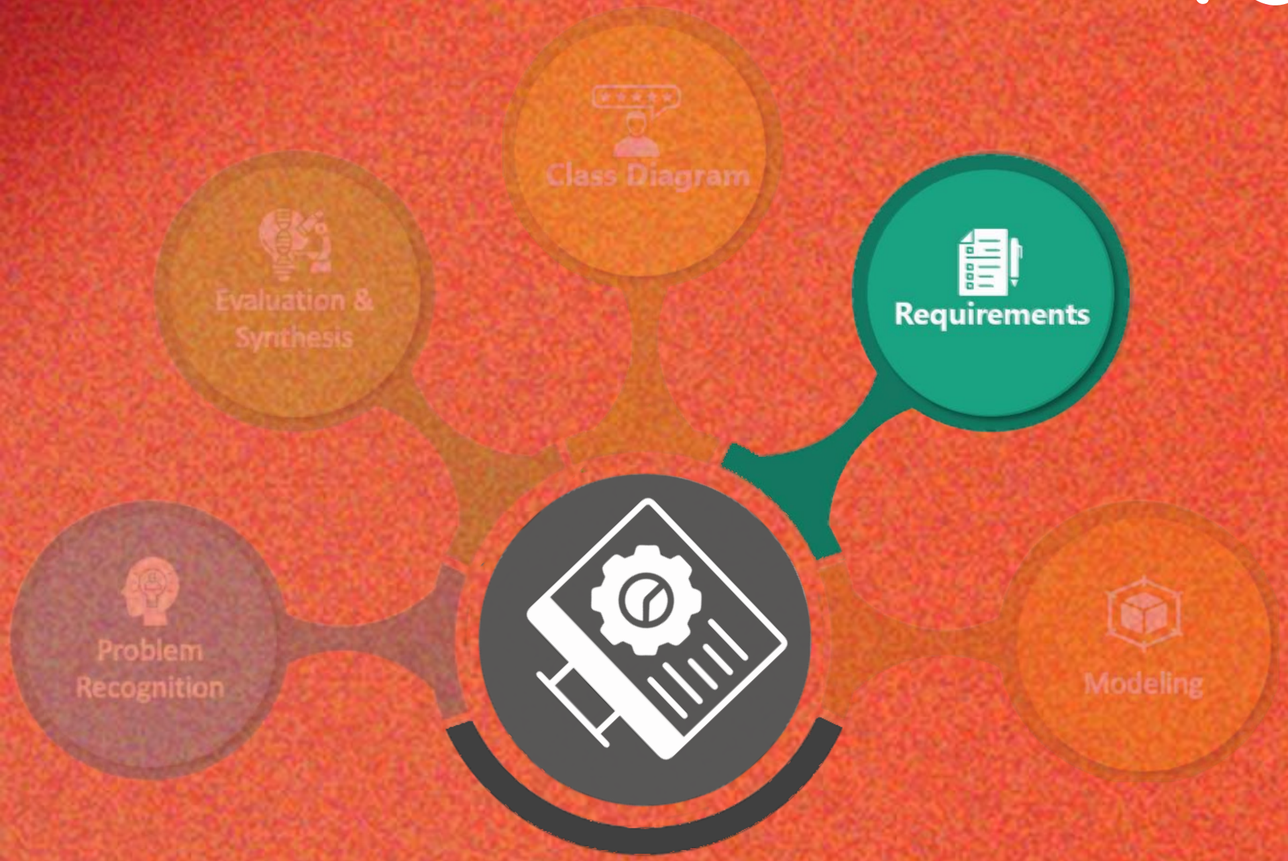
- Matching algorithm based on student CVs and internship criteria
- Triggered when a new submission is published on the platform
- Moves to a "ToBeAccepted" state for approval.
- Terminated if either party rejects.
- Dual acceptance initiates the interview process.



ALLOY

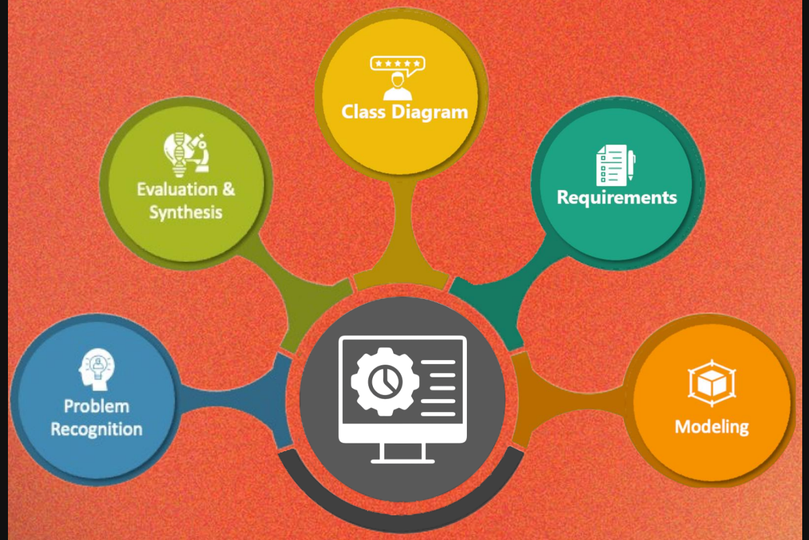
- Represents core domain entities such as Students, Companies, Universities, and Internship Offers.
- Defines relationships that mirror the internship matching and application processes.
- Encodes business rules and constraints for valid system interactions.
- Captures invariants that ensure consistency in the internship lifecycle.
- Visualizes the domain structure for clear analysis and verification.

REQUIREMENTS



- [R1]: The platform shall allow any unregistered students to register by providing personal information and selecting their University.
- [R6]: The platform shall allow Companies to create and publish Internship offers specifying details.
- [R8]: The platform shall provide Students with Matches automatically obtained by the Recommendation Process
- [R21]: The platform shall allow Companies to create Interviews.
- [R34]: The platform shall allow registered Universities to handle Complaints and to interrupt an Internship at their own discretion..

RASD



DD

Software Design Documentation

Product Name	Students & Companies
Date Updated	07/01/2025
Written By	RicciPaoliGrisoni

Architectural Styles and Paradigms

The purpose of this document is to provide a comprehensive overview of the software design for the XYZ application. This includes the system overview, design considerations, specifications, detailed design, implementation plan, testing plan, and maintenance plan.

Interfaces

The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

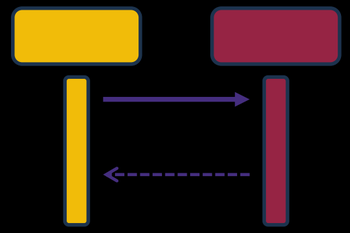
System Components

The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

Unit and Integration Test

ITD

UML



Sequence Diagrams

Runtime View

Goals

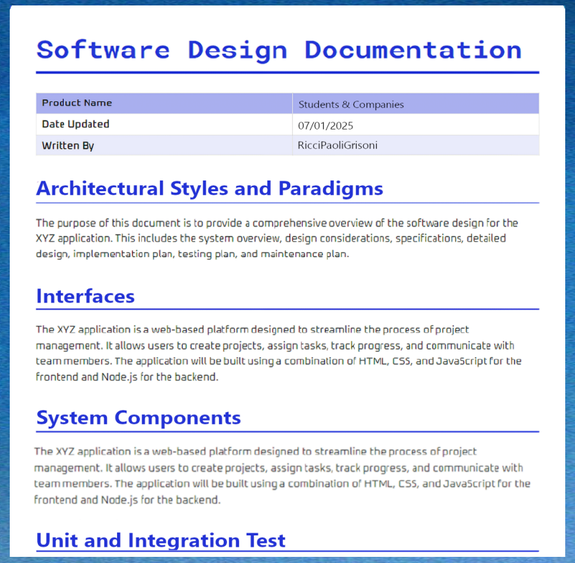
Requirements

User Interface

Test Plan

DD

Within our Requirements and Analysis Specification Document, we provide a detailed description of the domain along with the necessary requirements that our platform must meet to achieve its intended goals.



Software Design Documentation

Product Name	Students & Companies
Date Updated	07/01/2025
Written By	RicciPaoliGrisoni

Architectural Styles and Paradigms

The purpose of this document is to provide a comprehensive overview of the software design for the XYZ application. This includes the system overview, design considerations, specifications, detailed design, implementation plan, testing plan, and maintenance plan.

Interfaces

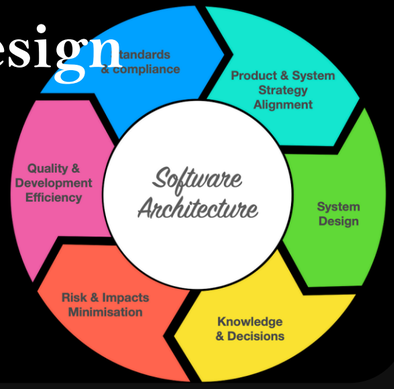
The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

System Components

The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

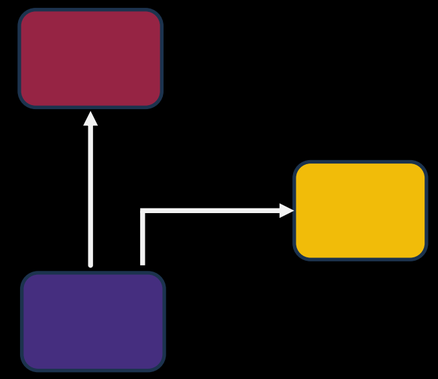
Unit and Integration Test

Architectural Design



Deployment

Technologies



Interface & Component Diagrams

Component View & Interfaces

Design Document

Software Design Documentation

Product Name	Students & Companies
Date Updated	07/01/2025
Written By	RicciPaoliGrisoni

Architectural Styles and Paradigms

The purpose of this document is to provide a comprehensive overview of the software design for the XYZ application. This includes the system overview, design considerations, specifications, detailed design, implementation plan, testing plan, and maintenance plan.

Interfaces

The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

System Components

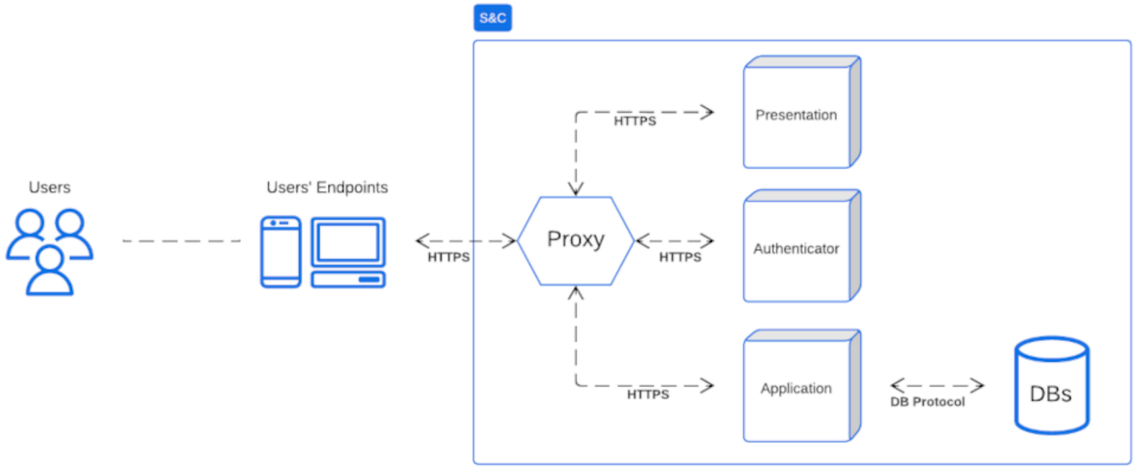
1. **Architectural Styles and Paradigms:** a high-level overview of the system composition
2. **Interfaces: a**llow the components to talk with the external world
3. **Components:** independently deployable services that perform a specific business functions.
4. **Test:** plan and execution of unit and integration test with the aim to discover and fix any problem of the platform

Architectural Style and Paradigms

Software Design Documentation

Product Name	Students & Companies
Date Updated	07/01/2025
Written By	RicciPaoliGrisoni

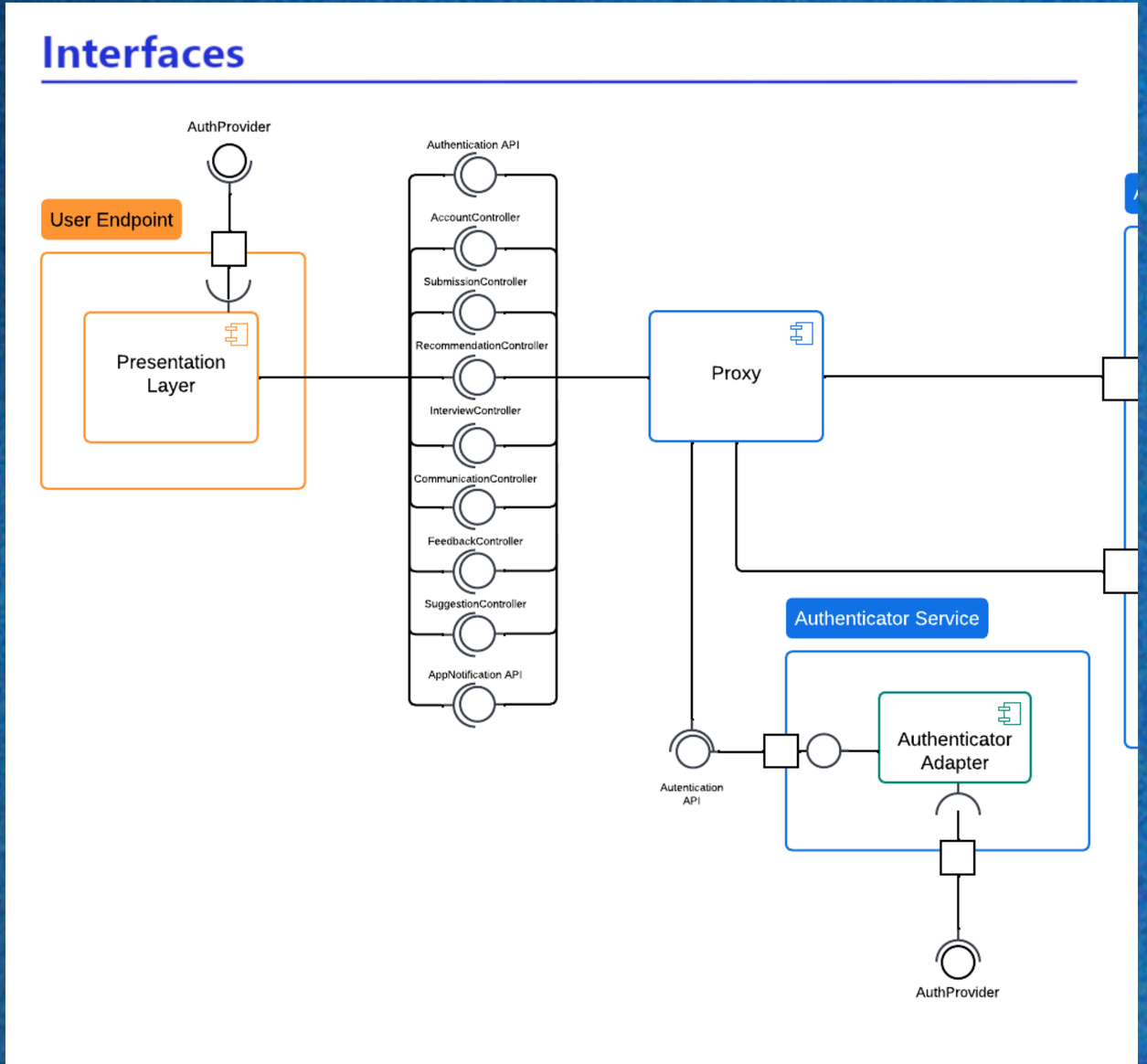
Architectural Styles and Paradigms



Interfaces

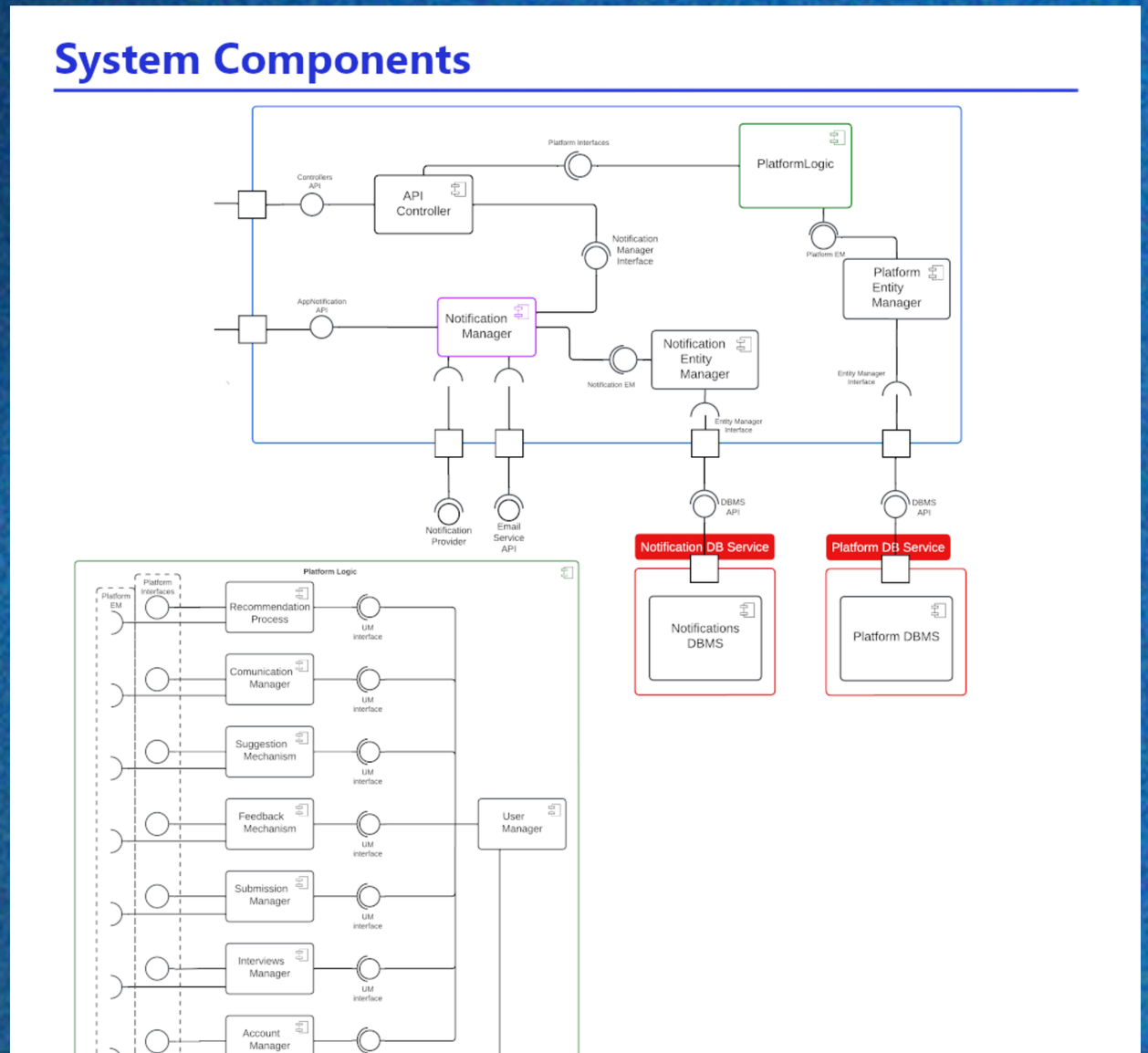
1. A **Proxy** re-directs incoming requests to the appropriate service.
2. The **authenticator service** handles the authentication before forwarding the request
3. The **application** service is the **heart of the Platform**, computing information and performing CRUD operations on the Databases

Interfaces



1. **User Endpoints** interacts with the Platform through different **REST API** calls
2. **Exposed Controllers endpoints:** represent all the back-end operations that can be called by users.
3. **Authenticator API:** interacts with the external auth provider and returns data in a back-end-friendly format.
4. **Front-end AuthProvider Interface:** handle sign-up, login and token refreshen operations

System Components



1. The **backend has two main controllers**, each with its own Entity Manager and database. This forms the foundation for a more scalable architecture

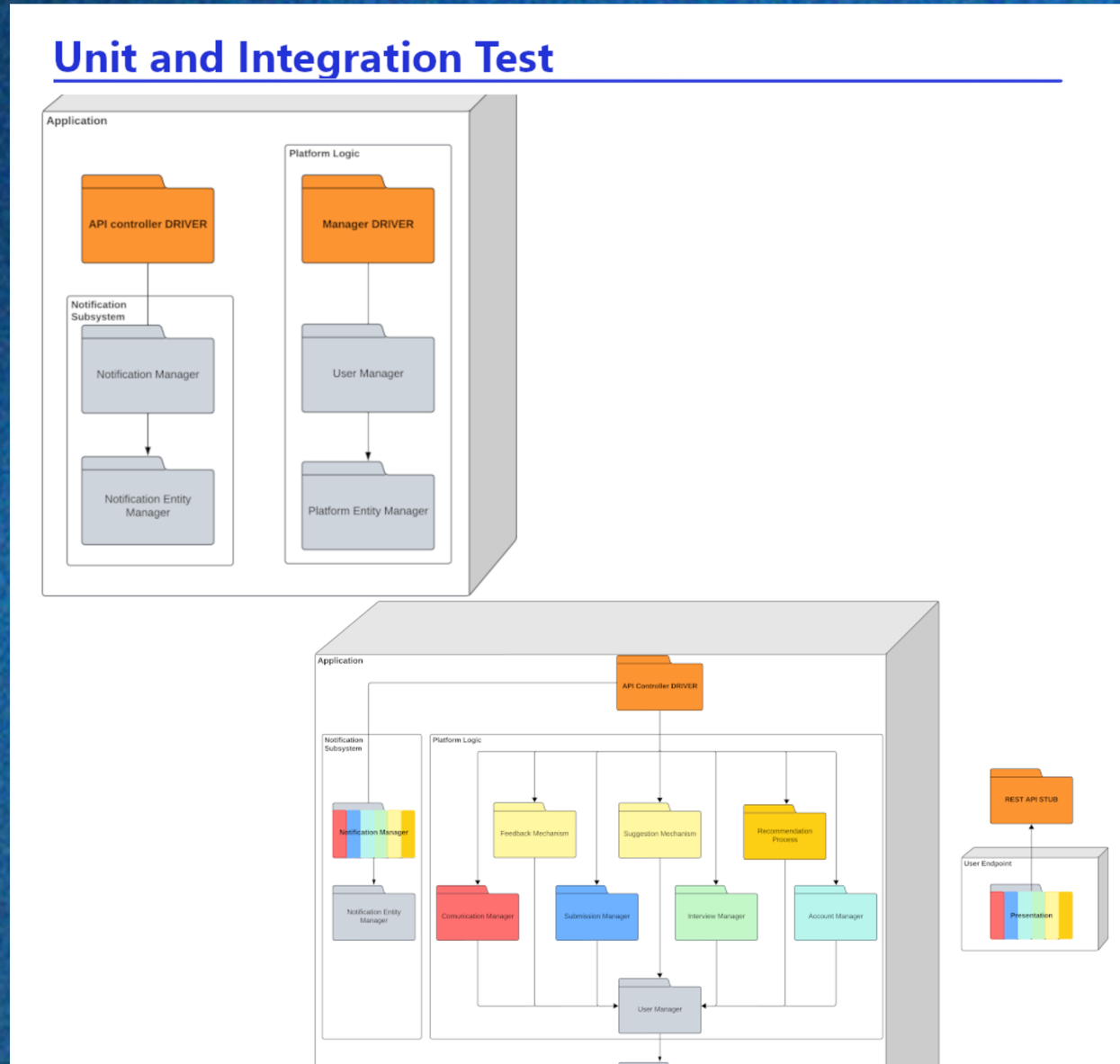
2. **API Controller:** contains controllers that execute Platform Logic functions based on user request

3. **Notification Manager:** Manages notifications, integrating and adapting external services and supporting future containerization.

4. **Platform Logic**: Encapsulates business logic and CRUD operations. It consists of multiple components, each independently managing a specific part of the system's logic and its own data via the Entity Manager.

5. **Possible Service Split-up:** Due to the presence of different interfaces and databases, we believe it is possible to split the application service into at least two subservices.

Unit and Integration Test



1. **Hybrid Development Approach:** combines thread-based and bottom-up strategies for parallel development and testing.

2. **Feature-Based Implementation:** develops one Platform Logic feature at a time, implementing both front-end and back-end components

3. **Early Integration Testing:** allows component integration to be tested early, reducing risks of major integration issues later.

4. **Testing Strategy:** uses **API Controller Driver** and REST **API Stubs to simulate interactions**, ensuring thorough backend and frontend testing.

RASD

DD

Software Design Documentation

Product Name	Students & Companies
Date Updated	07/01/2025
Written By	RicciPaoliGrisoni

Architectural Styles and Paradigms

The purpose of this document is to provide a comprehensive overview of the software design for the XYZ application. This includes the system overview, design considerations, specifications, detailed design, implementation plan, testing plan, and maintenance plan.

Interfaces

The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

System Components

The XYZ application is a web-based platform designed to streamline the process of project management. It allows users to create projects, assign tasks, track progress, and communicate with team members. The application will be built using a combination of HTML, CSS, and JavaScript for the frontend and Node.js for the backend.

Unit and Integration Test

ITD

ITD

Within our Requirements and Analysis Specification Document, we provide a detailed description of the domain along with the necessary requirements that our platform must meet to achieve its intended goals.

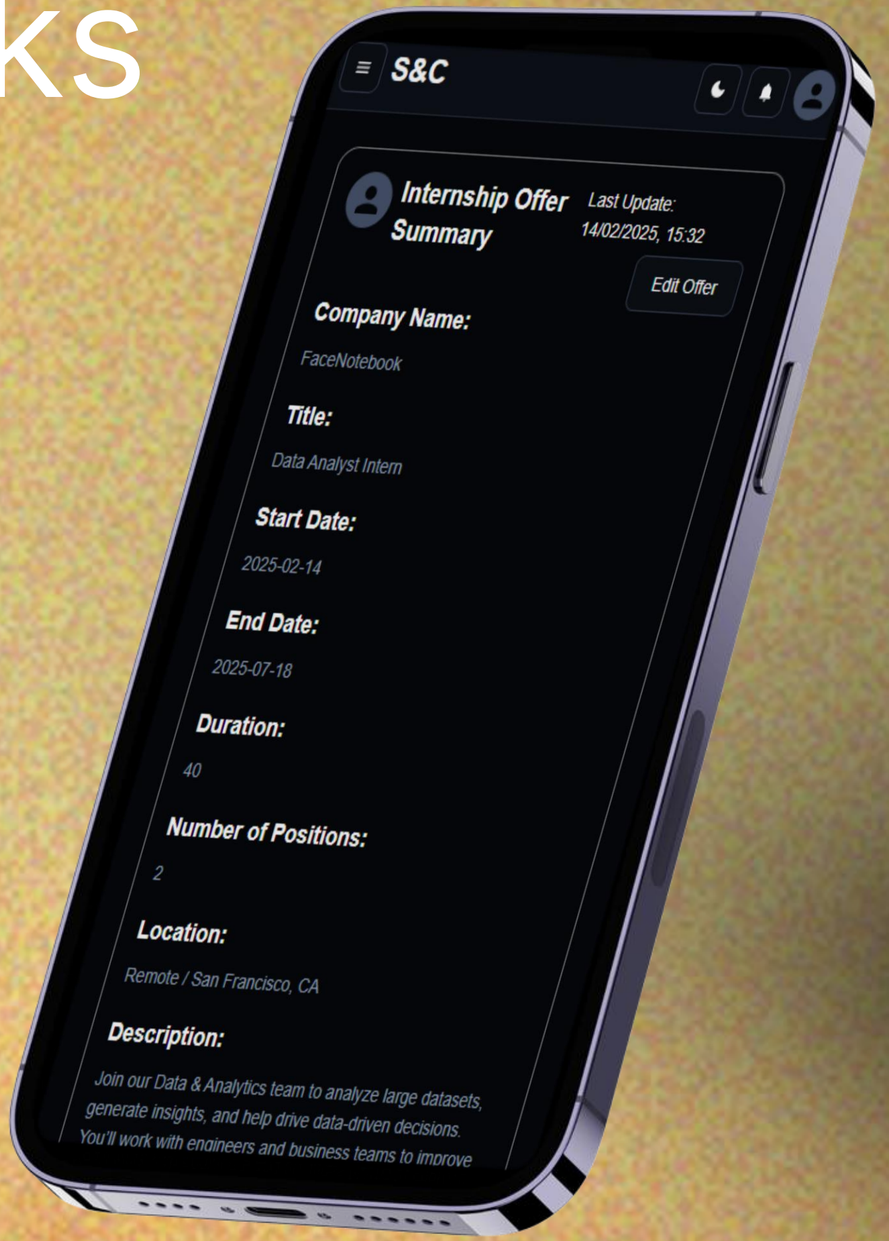
Adopted Frameworks

Frontend

- **React:** Chosen for its simplicity, flexibility, and performance. Libraries like **Material-UI** enhance animations and responsive design.

Backend

- **Spring Boot:** Used to create RESTful APIs, manage business logic, and integrate **Spring Data JPA** for database operations.
- **Lucene:** high-performance text search engine used for the **matching algorithm**
- **Firebase:** A backend platform handling **authentication** and **push notifications** via **Firebase Cloud Messaging**.



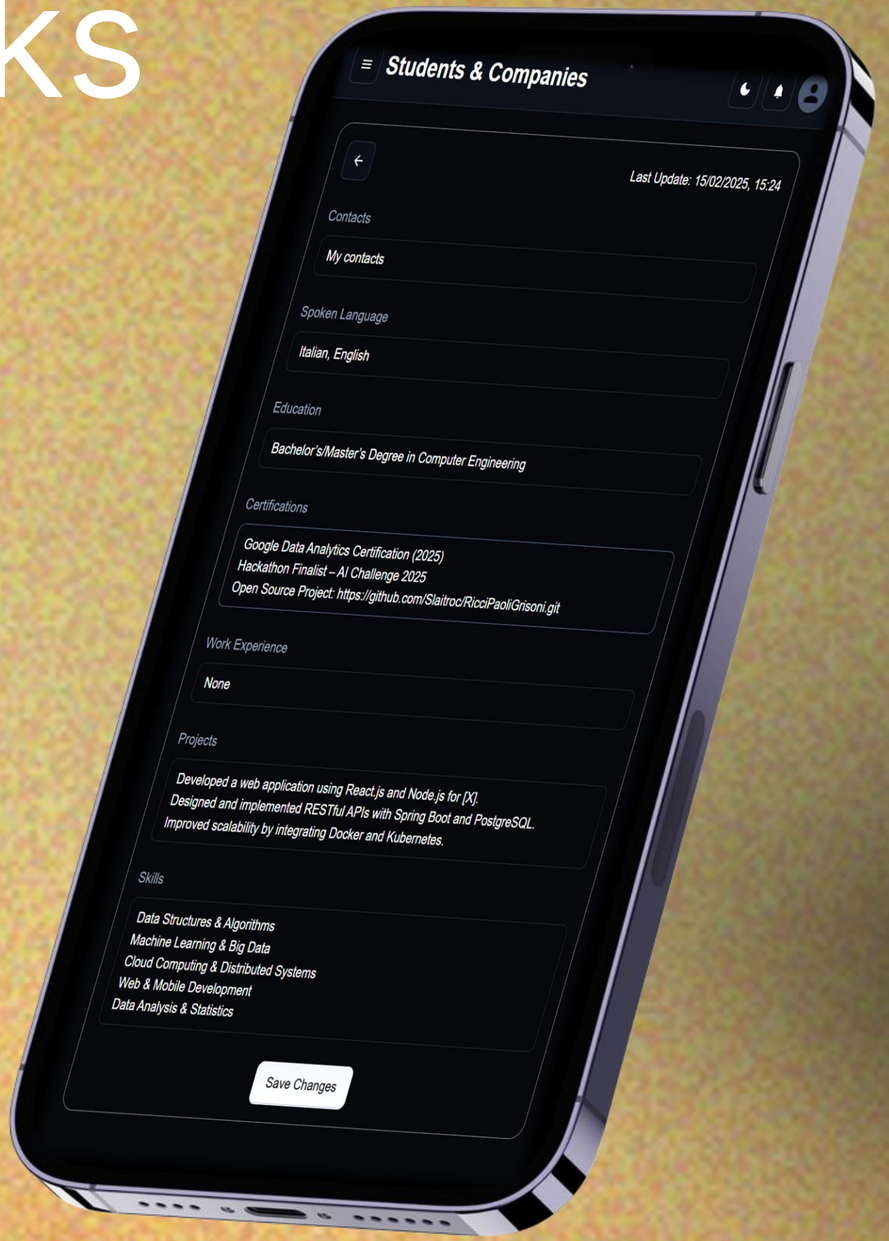
Adopted Frameworks

Data Layer:

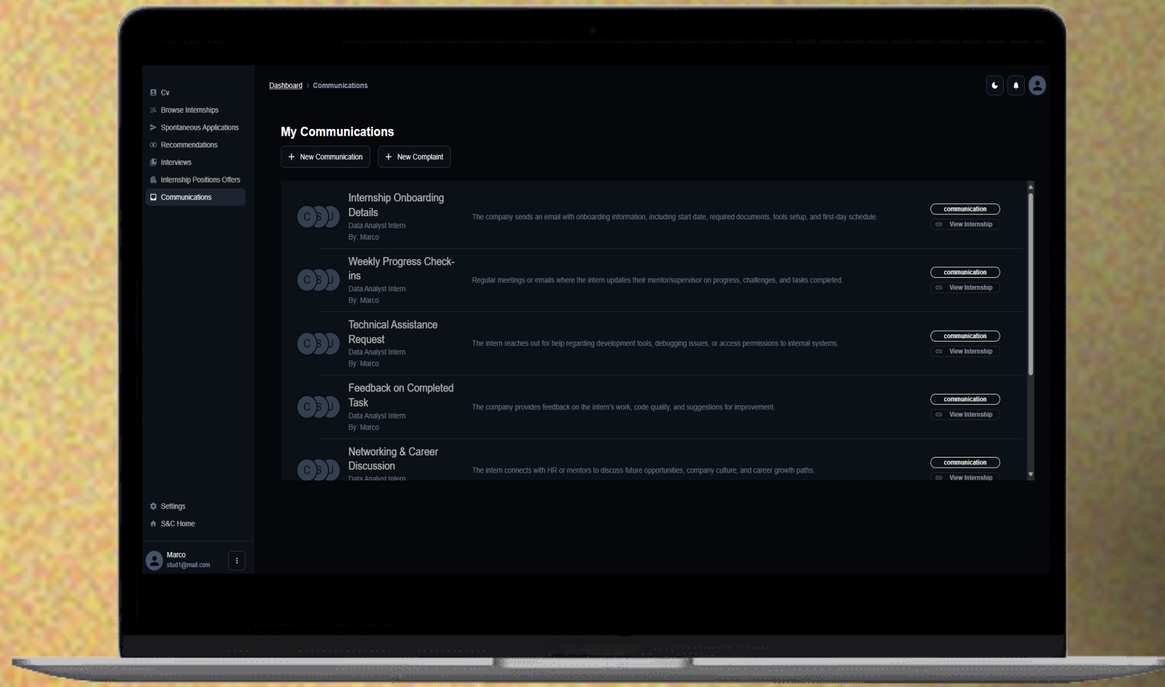
- **JPA:** provides an ORM abstraction for mapping Java objects to database tables
- **Hibernate:** enhances performance with caching, transaction management, and query optimization.

Tools:

- **MariaDB:** An open-source, MySQL-compatible relational database chosen for its integration with JPA/Hibernate, scalability, and cost-effectiveness.
- **Traefik:** A **reverse proxy & load balancer** for routing traffic in the microservices architecture.



Code Structure



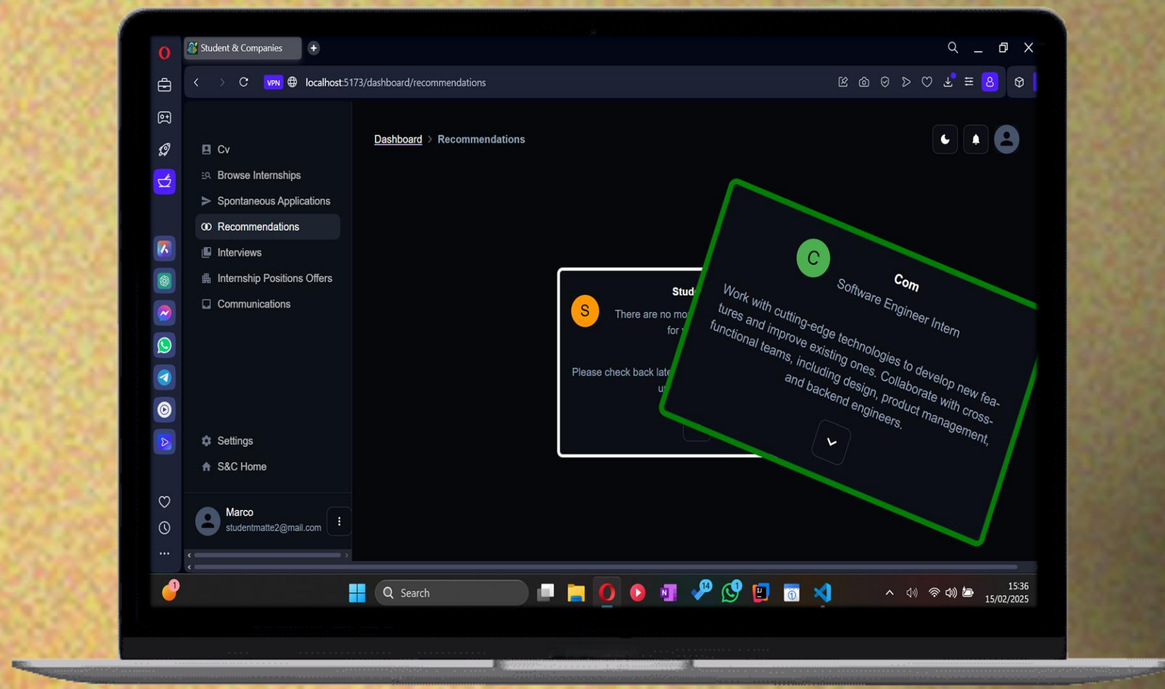
Recommendation Process

Recommendation Process

- **Lucene:** a search engine used by popular sites like Wikipedia and LinkedIn.
- **Stop Words:** words without semantic meaning, such as articles or conjunctions.
- **Words Vector:** represents a set of words as a vector.
- **Score**: represents the distance between two vectors the **closer the vectors, the higher the score**.

Feedback System:

- **Dynamic Threshold:** The minimal score that a pair CV-IntershipOffer must have to be considered a Match
- **EMA:** a type of moving average (MA) that places a greater weight and significance on the most recent data points



Thank you for your attention!

R&DD + IT

Repository -> https://github.com/Slaitroc/RicciPaoliGrisoni

Group Members:

- Lorenzo Ricci -> https://github.com/Slaitroc
- Matteo Giovanni Paoli -> https://github.com/Krotox
- Samuele Grisoni -> https://github.com/dedepivot

Documents:

- RASDv1.1
- DDv1.1
- ITDv1.1
- ATDv1

Let's Test it Out!