POLITECNICO
MILANO 1863

Students&Companies

Design Document (DD)

SOFTWARE ENGINEERING II

Computer Science and Engineering
Academic Year: 2024-2025

Authors:

ALMANDOZ FRANCO Rodrigo
Personal Code: 10752609

BRANDI Mattia
Personal Code: 10771744

Version 2 - 25 January 2025

Table of Contents

1. Introduction

1.1 Purpose

The main purpose of the present document is to support the development team in the realization of the S&C platform. It provides an overall description of the adopted system architecture and a breakdown of the various components, describing their interaction. Moreover, the DD describes the implementation, integration and testing plans which are defined taking into account priority, required effort and impact of the single components.

1.2 Scope

Students&Companies platform is a software system designed to ease the connection between students searching for an internship and companies in an efficient and user-friendly way. The platform allows students to browse, apply for, and manage applications, while providing companies tools to post internships, review applications and schedule interviews. Moreover, thanks to an efficient recommendation system powered by machine learning algorithms, both students and companies can have tailored recommendations based on their needs and preferences. In addition, S&C supports the entire internship process, from application to feedback. The detailed requirements and functionality are outlined in the Requirement Analysis and Specification Document (RASD).

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Internship:** the position of a student (intern) who works temporarily in an organization, sometimes without pay, in order to gain work experience.
- **CV:** It stands for Curriculum Vitae, and it is a short written description of your education, qualifications and previous jobs
- **API:** It is a set of functions and procedures allowing the creation of applications that access the features or data of an application
- **SSO:** Single Sign-on (SSO) is an identification method that enables users to log in to multiple applications with one set of credentials
- **Demilitarized Zone:** It is a segment of a network that acts as a zone between a private internal network and untrusted external networks (e.g., the Internet)

1.3.2 Acronyms

- **S&C** Students and Companies
- **RASD**: Requirement Analysis and Specification Document.
- **UI** : User interface
- **UML**: Unified Modelling Language
- **DD:** Design Document
- **DMZ:** Demilitarized Zone

1.3.3 Abbreviations

- [Gn] - the n-th goal of the system
- [WPn] - the n-th world phenomena
- [SPn] - the n-th shared phenomena
- [UCn] - the n-th use case
- [Rn] - the n-th functional requirement
- DB - Database
- API - Application Programming Interface

1.4 Revision History

- Version 1
- Version 2
    - We changed the Component Diagram image due to an interaction mismatch between the CompliantManager/FeedbackManager and InternshipManager. Additionally, we have specified the correct internship interface for each component that uses it to ensure an additional layer of safety.

1.5 Reference Documents

This document is based on:

- The specification of the RASD and DD assignment of the Software Engineering 2 course, held by professors Matteo Rossi, Elisabetta di Nitto and Matteo Camilli at the Politecnico di Milano, A.Y 2024/2025
- Slides of course on WeBeep

1.6 Document Structure

The current document is divided into six chapters, which are the following.

1. **Introduction**: it aims to explain the purpose of the document and recall the concepts introduced in the RASD.

2. **Architectural Design**: it includes a detailed description of the architecture of the system, in particular a high level view of the elements, software components and a runtime description of the system's functionalities.
3. **User Interface Design**: it contains mock-ups of the graphical user interface.
4. **Requirements Traceability**: it describes the connection between the requirements defined in the RASD document and the components described in the second chapter. In particular, it highlights how the design decisions have been taken with respect to the requirements and, as a consequence, how those decisions fulfill the goals described in RASD.
5. **Implementation, Integration and Test Plan**: it describes the process of implementation, integration and testing that the developers must respect in order to deliver a high standard system.
6. **Effort Spent**: it contains the time spent to realize the following document per person.
7. **References:** it contains the references to any document and to softwares used in this document.

2. Architectural Design

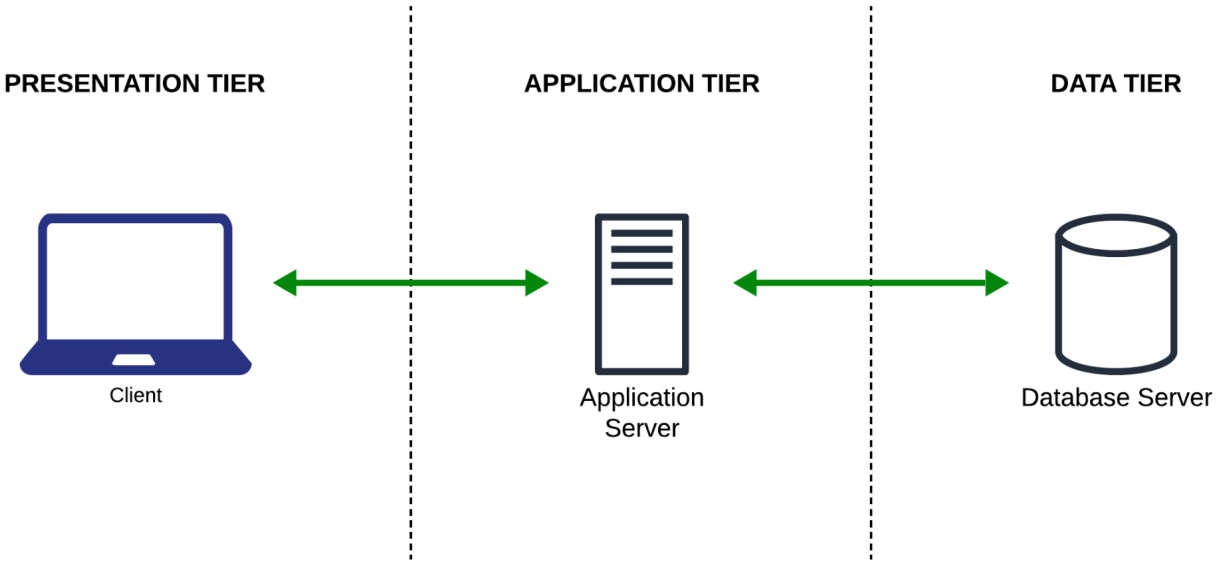
2.1 Overview

In the following section, it is described a high-level overview of the architectural elements that compose the system and their primary interaction.

S&C platform employs a three-tier architecture, which allows the system to be divided into three distinct computing levels. The first level is the presentation layer, which manages the presentation's logic and the platform's user interface (front end of the platform); the second level is the application layer, which is responsible for the functions provided to the user. In this layer the data is processed; the third is the data layer, which stores the data and it manages the information and access to the database. The last two tiers represent the back end of the platform and they are accessed through methods exposed by a REST API.

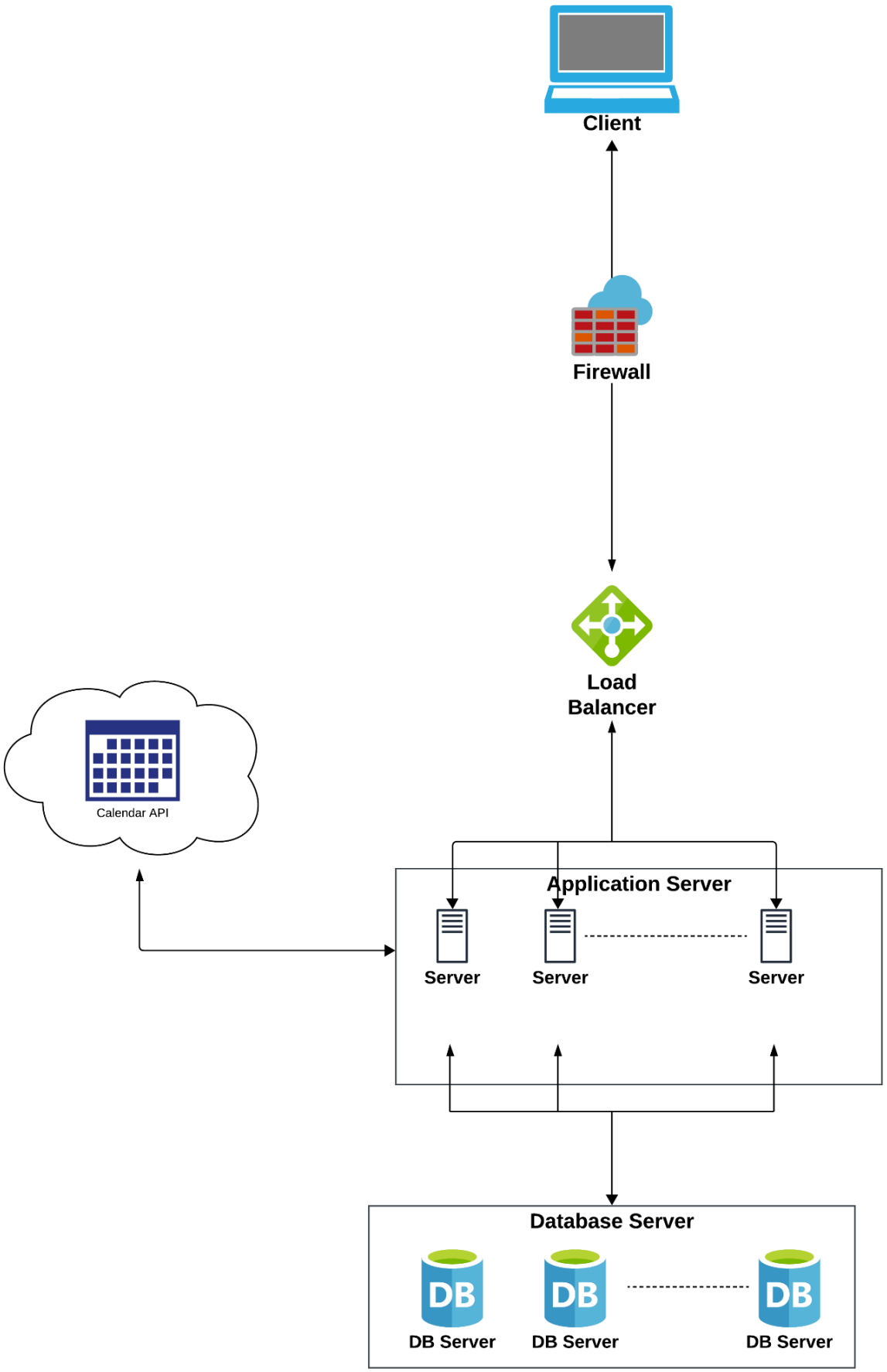
Each tier can be developed and maintained independently without affecting the other tiers. Each layer can communicate only with the adjacent one.

S&C platform adopts a thin client communication, which means that the client node manages only the presentation logic, such as user interactions, and it does not require significant computational power



2.1.1 Distributed View

The S&C platform has the following distributed view.


Client
Firewall
Load
Balancer
Calendar API
Application Server
Server
Server
Server
Database Server
DB
DB
DB
DB Server
DB Server
DB Server

2.2 Component View

The following components belong to the back-end of the S&C platform:

WebApp Server: the main purpose of the component is to expose endpoints to the Web application, enabling client-server communication. It is the REST API of the S&C platform. It processes incoming client requests and manages them. It allows the exchange of data between the front-end and the back-end. It contains all graphical user interface units. Moreover, it allows requests to be correctly guided to the appropriate microservice.

SSO Manager: this component manages user authentication across multiple systems or platforms within the S&C ecosystem. It enables users to access S&C platform with their organization credentials, making it easy for them to access the platform.

Registration Manager: this component provides methods to manage the organization registration workflow. It allows authorized employees to create the respective organization account by supplying the necessary personal and organization details. It validates the provided information.

Company Manager: this component is responsible for implementing functionalities that enable a Company's employees to access their dashboard and interact with them. It allows employees to visualize the Company's details, to manage the provided Internships and to access other relevant resources.

Statistics Manager: this component is specialized to process and represent data analytics once retrieved by the user. It aggregates data such as internship performance, user engagement rates and other key indicators.

Feedback Manager: this component allows users to create, manage and publish feedback that are relevant in terms of fostering user engagement and enhancing the internship's value.

University Manager: this component allows Universities to manage their profile and access their private dashboard. It allows employees to monitor their students by creating insights based on the data received by Model.

Notification Manager: this component implements all the functionalities required by S&C platform to send notifications. It also provides users the tools to manage their notifications ensuring timely and relevant alerts regarding recommendations and scheduled interviews.

DBMS: this component allows the creation and manipulation of the database which is necessary for the correct behaviour of the S&C platform.

Model: this component serves as an abstraction layer between the application logic and the DBMS. It provides all the necessary methods to access data and the logic to manipulate them ensuring consistency. Whenever data is required by a component, this last component

interacts with the model, which is responsible to retrieve that data by interacting with the DBMS.

Internship Manager: this component allows Company's employees to interact with the internships provided by their respective Companies. With those functionalities, he can publish internships, check and review the CVs of the not recommended students, who have applied on the Internship personal page. The employee can also manage the list of recommended applicants. Moreover, both complaints and feedback can be published by all Users by accessing Internship Manager.

Interview Manager: this component allows the Company's employee to organize and manage the interviews for a specific Internship offered by the Company itself. It communicates with the Calendar API to efficiently schedule interviews based on the availability of both candidates and recruiters.

ComplaintManager: this component allows User to create, manage and publish complaints. It manages the communication between the parts that could end with a positive resolution of the Complaint, otherwise the Internship will be prematurely terminated for the specific student.

RecommendationManager: this component is designed to manage recommendations by both students and Company's employees. It uses data-driven algorithms to identify patterns in successful matches. Both anonymized complaints and feedback are used to weigh the model in order to provide better recommendations in the future.

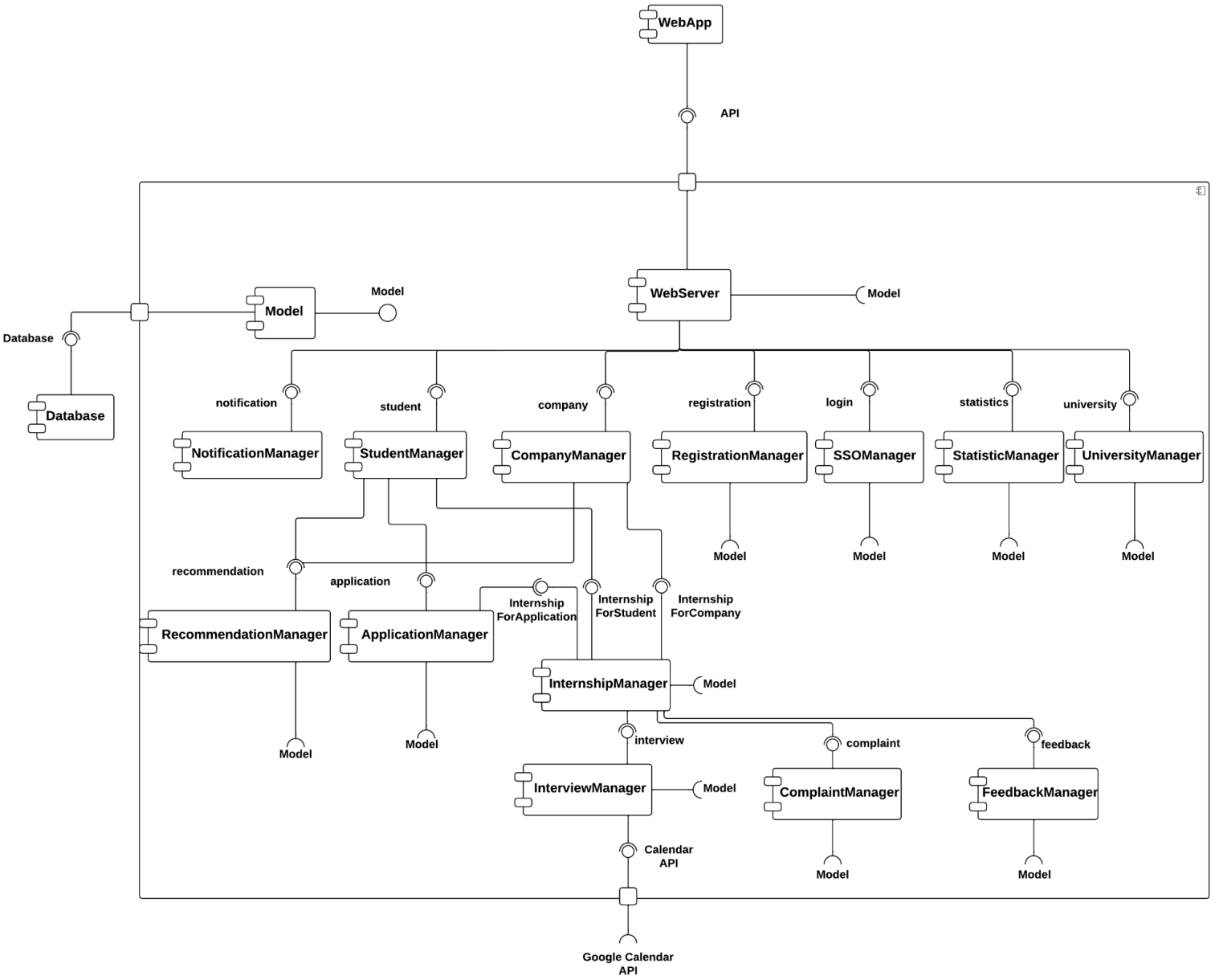
StudentManager: this component allows Students to interact with their private dashboards containing all the personal data and a summary about all the activity related to that specific Student. It has all the methods for uploading, editing and maintaining CVs.

ApplicationManager: this component supports functionalities for searching and applying for internships. It allows students to apply filters in order to have an overview of the available options. In addition it offers functionalities to apply for a specific Internship and to track the selection process.

The following components belong to the front-end of the S&C platform:

WebApp: this component is the client-side application that provides the user interface for interacting with the S&C platform. The WebApp communicates with the back-end components through API calls.

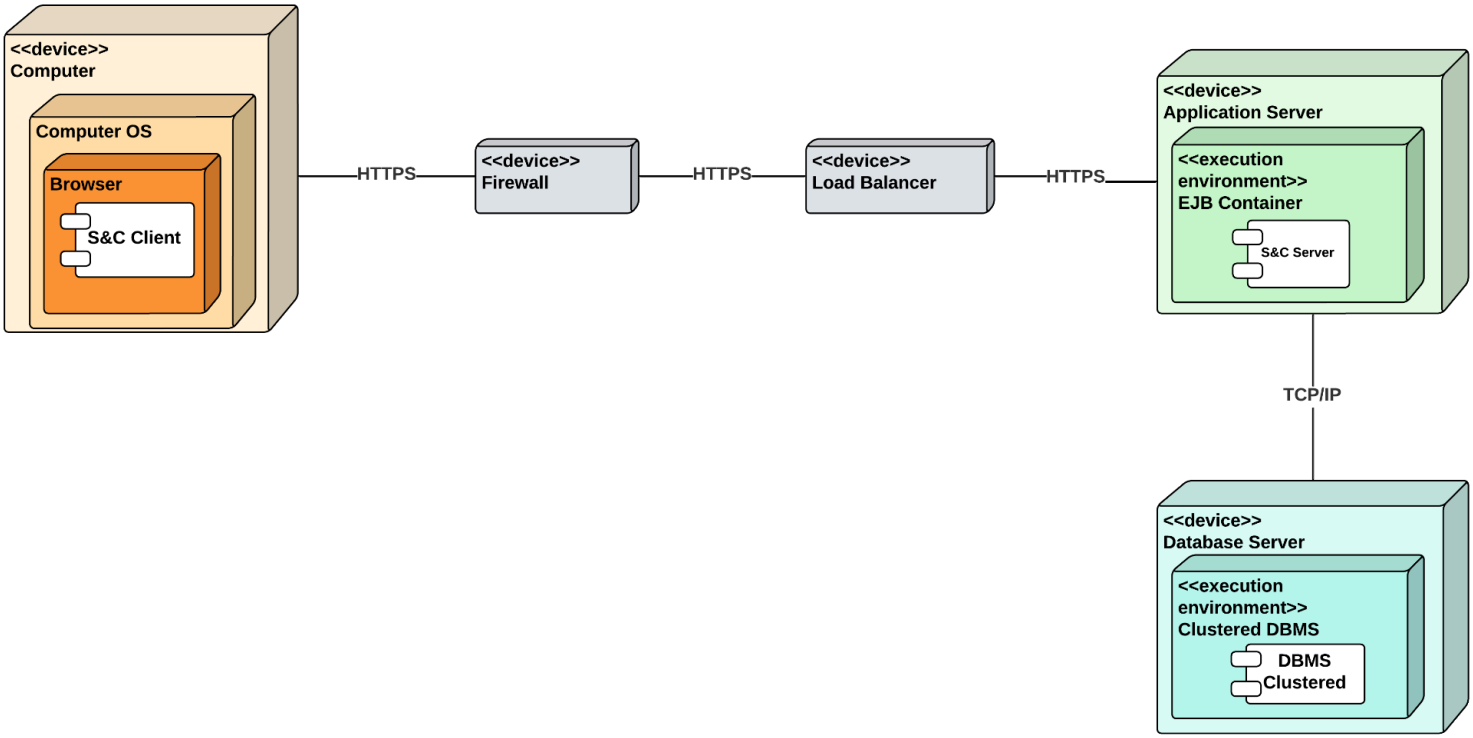
The S&C platform has the following component view.



2.3 Deployment View

In this chapter, it is described the deployment view for S&C. It provides an overview of the system's execution environment, detailing the distribution of the hardware components that host and run the software.

The S&C platform has the following deployment view.



The deployment view illustrates the interaction and distribution of the system's components. It consists of the following key elements:

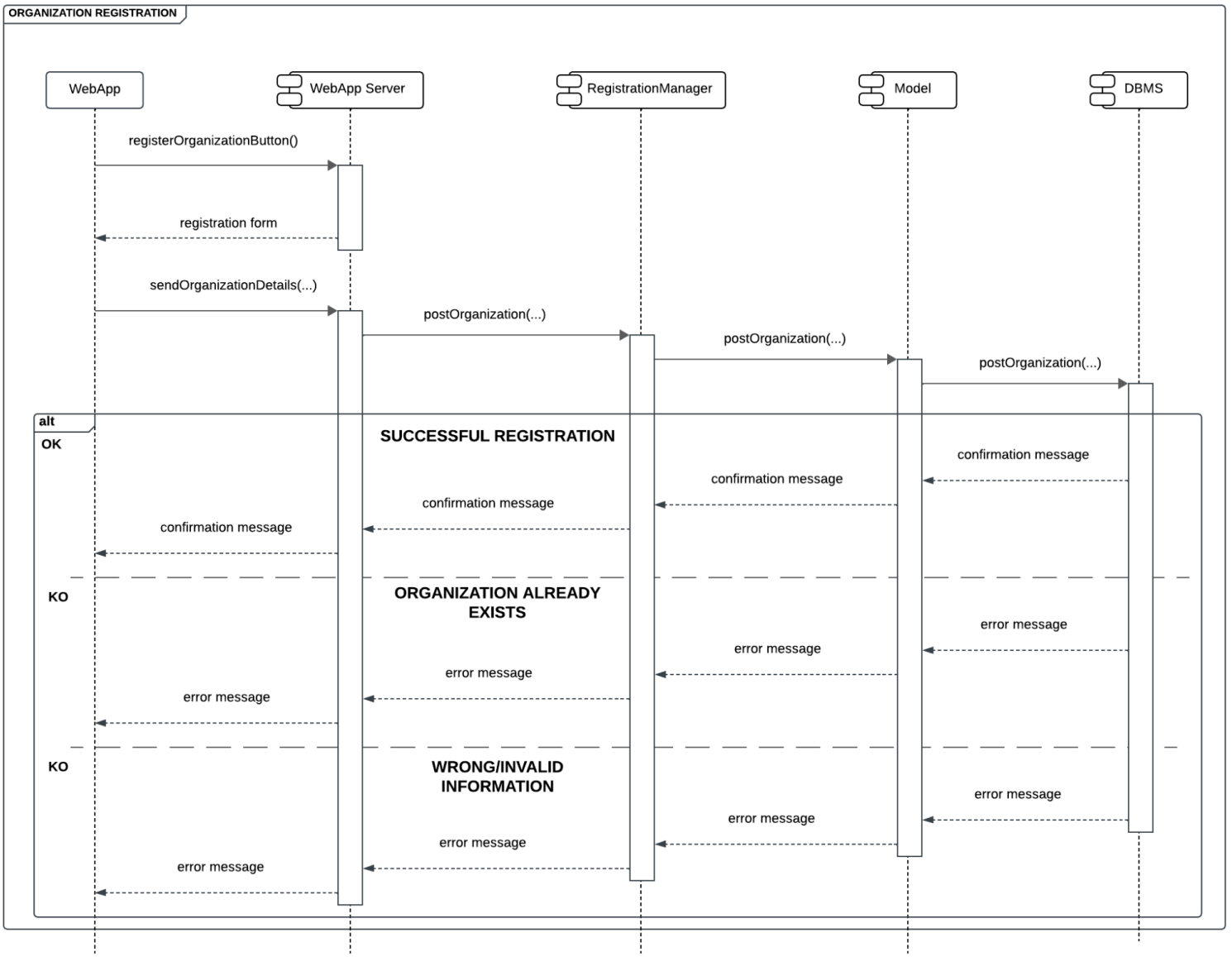
- **Computer**: it represents a client device used to access the system. It runs a web browser through which the client-side application is initiated. The computer establishes an HTTPS connection to the system via a firewall.
- **Firewall**: it acts as a security layer, inspecting incoming requests to ensure no malicious traffic reaches the system. It prevents harmful packets from progressing further and it provides a secure gateway for data transmission.
- **Load Balancer**: it is responsible for distributing incoming HTTPS requests among the available resources, such as the application server. This mechanism enhances system scalability, optimizes performance, and prevents server overload.
- **Application Server**: the application server is a critical component of the architecture, running the server-side application within an EJB container. It processes client requests, manages the S&C platform's logic, and communicates with the database server. This interaction occurs through a reliable TCP/IP connection, ensuring reliable data updates and retrievals.
- **Database Server**: it hosts the Clustered DBMS, which ensures data consistency and availability. It provides synchronous replication across the cluster, enabling high

availability and minimizing the risk of errors due to data inconsistencies. This guarantees up-to-date and accurate information for all system operations.

2.4 Runtime View

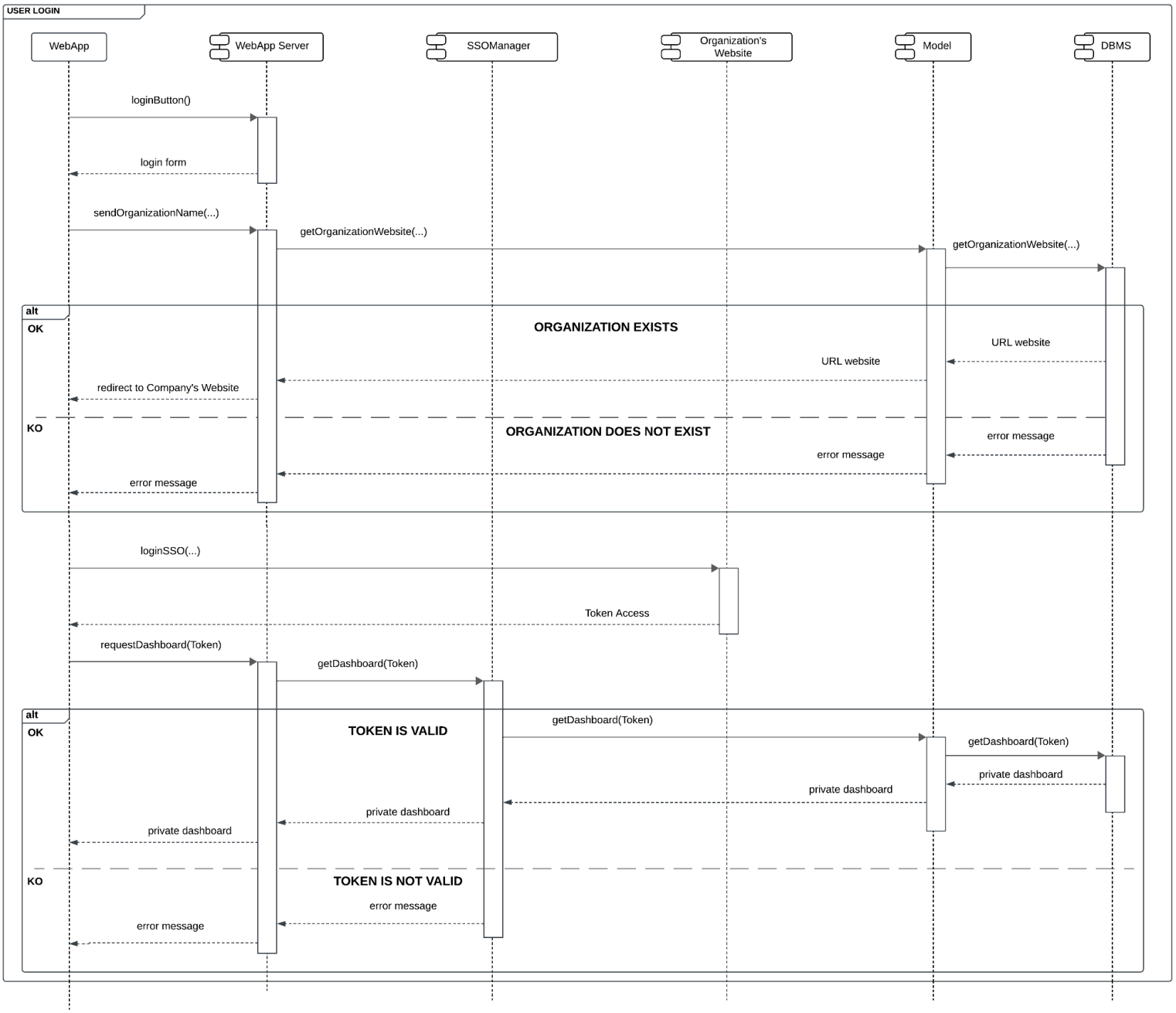
[UC1] - Organization Registration

The following runtime view shows the organization registration. Apart from the main components, WebAppServer, Model and DBMS, it involves the Registration Manager. The DBMS is responsible for determining the outcome of the operation, which can be successful or unsuccessful due to wrong or invalid information or if the organization already exists inside S&C.



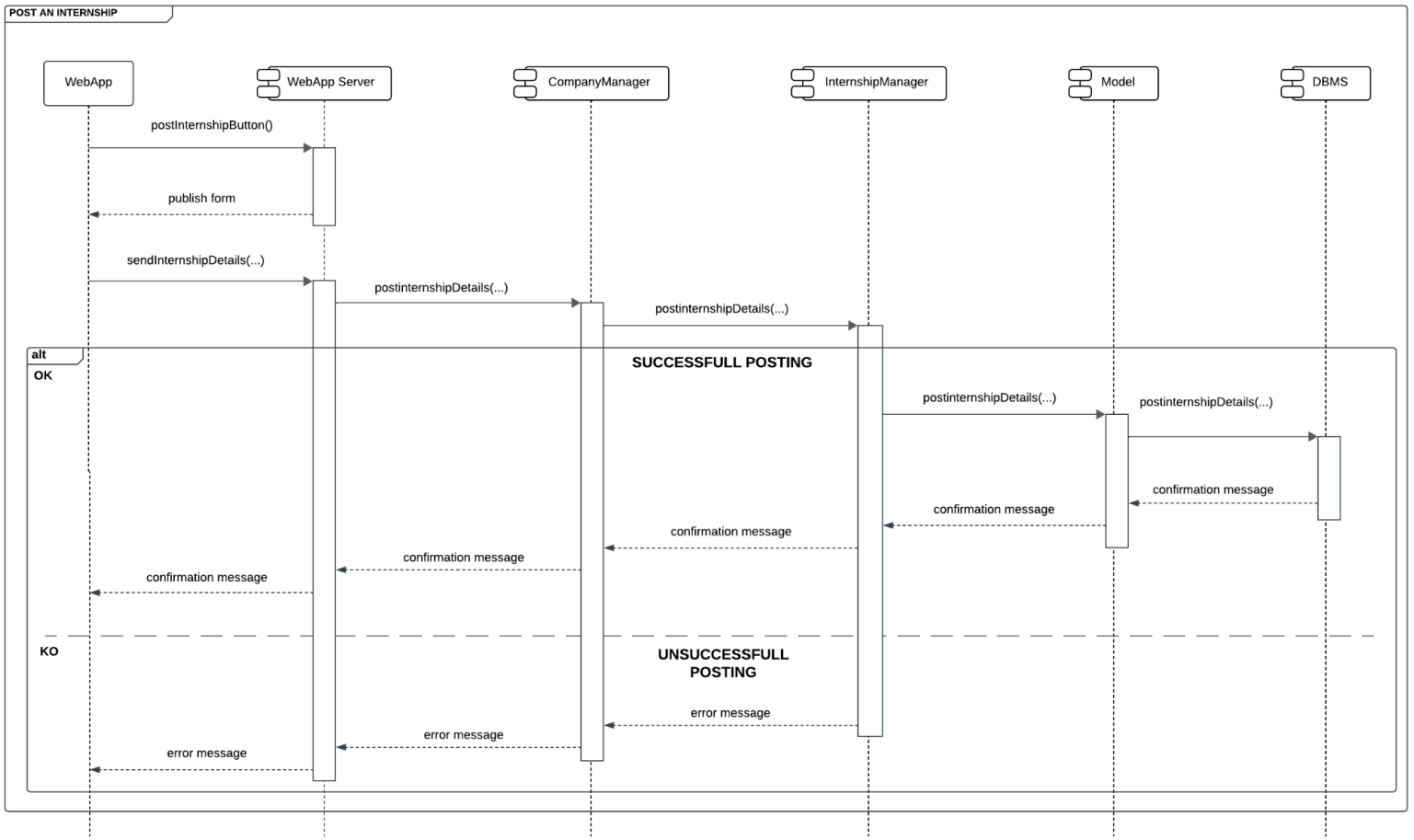
[UC2] - Login User

The following runtime view shows the user login procedure. Apart from the main components, WebAppServer, Model and DBMS, it involves the SSO Manager and the outside organization's website. The organization's website returns to the WebApp a Token and, consequently, the SSO Manager determines the validity of that Token.



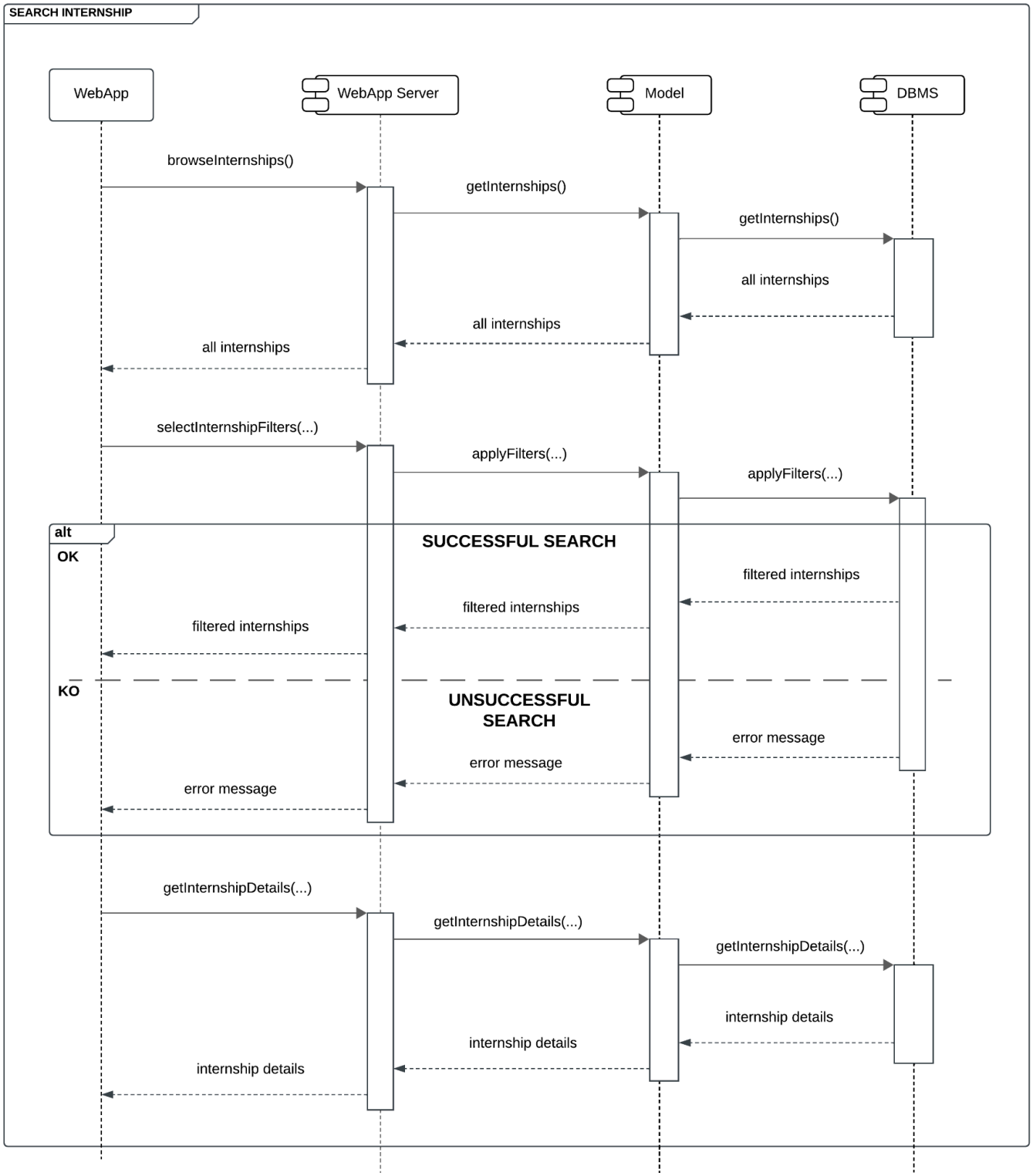
[UC3] - Post an Internship

The following runtime view shows the internship posting procedure. Apart from the main components, WebAppServer, Model and DBMS, it involves the Company Manager and the Internship Manager. The internship manager determines if the posting is successful or if it is unsuccessful.



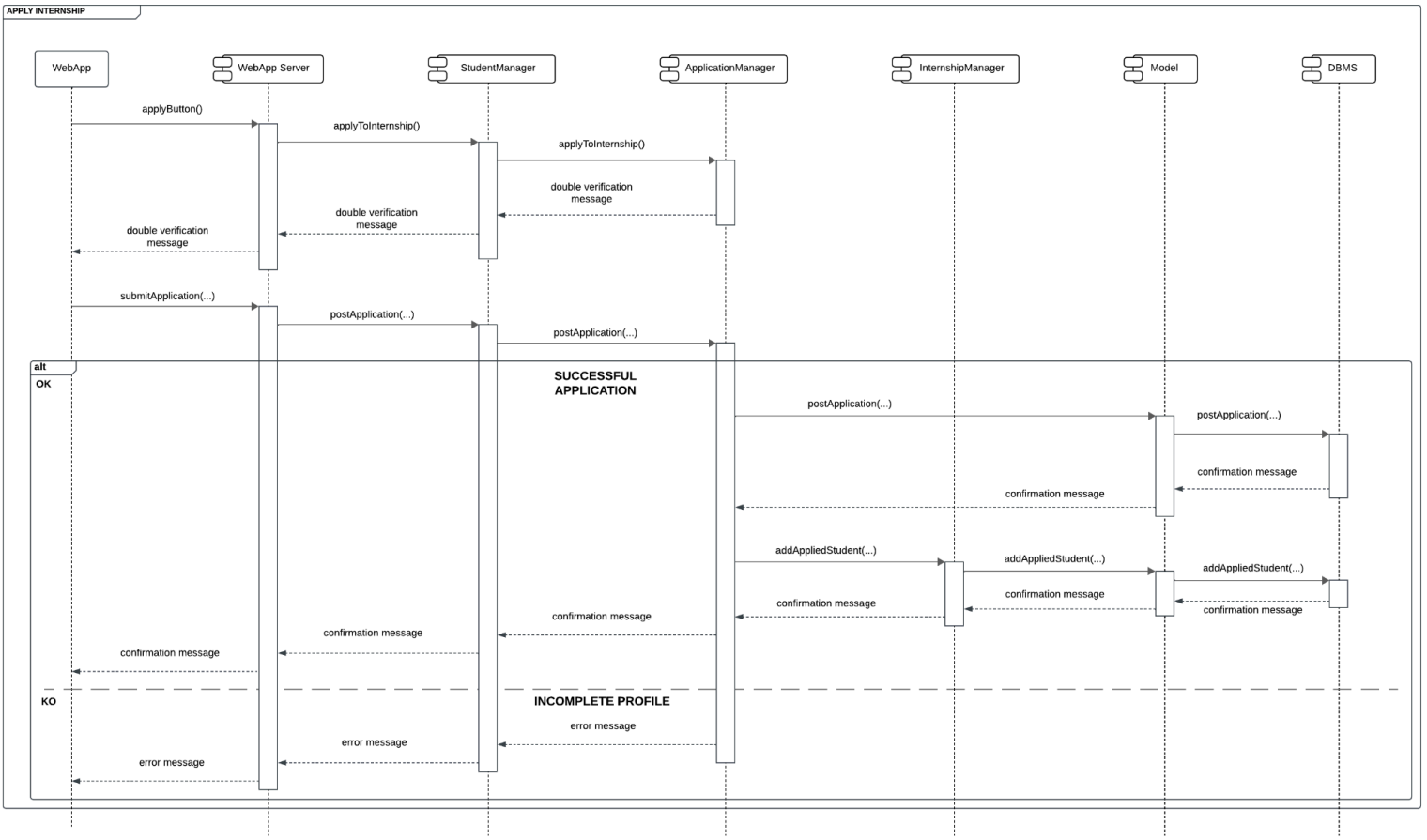
[UC4] - Search for Internship

The following runtime view shows the internship searching procedure. It involves only the main S&C components because it does not require any computational effort by components. In fact, in order to get the list of all internships, the selection of filters and details of an internship, the system requires only the main information provided by the client.



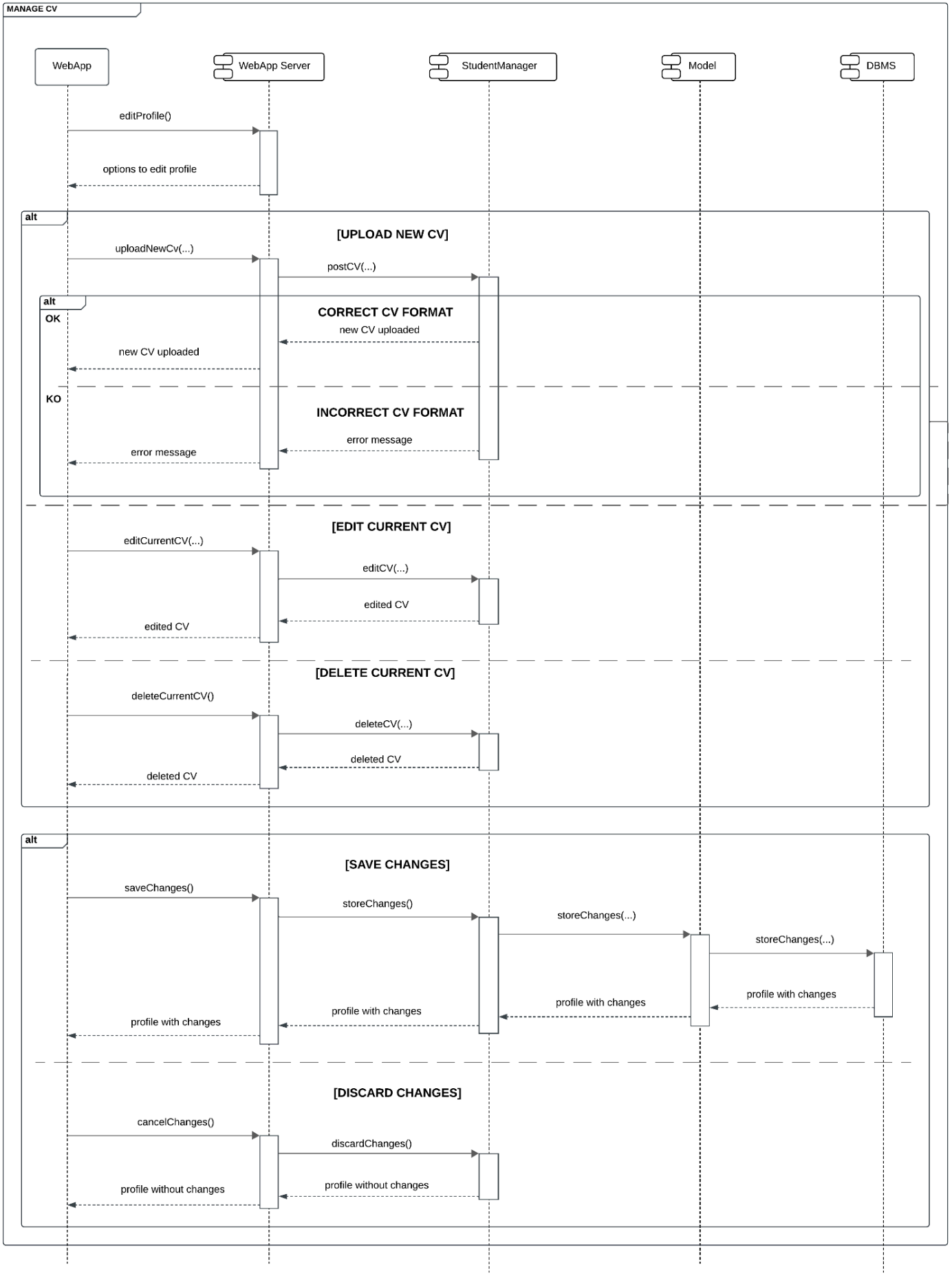
[UC5] - Apply for Internship

The following runtime view shows the procedure for applying to an internship. Apart from the main components, WebAppServer, Model and DBMS, it involves the Student Manager, Application Manager and the Internship Manager. The Application Manager determines if the application is successful by checking the completeness of the student profile. Moreover, the Application Manager is responsible for adding the student into the selected Internship by calling the method *AddAppliedStudent* offered by the Internship Manager.



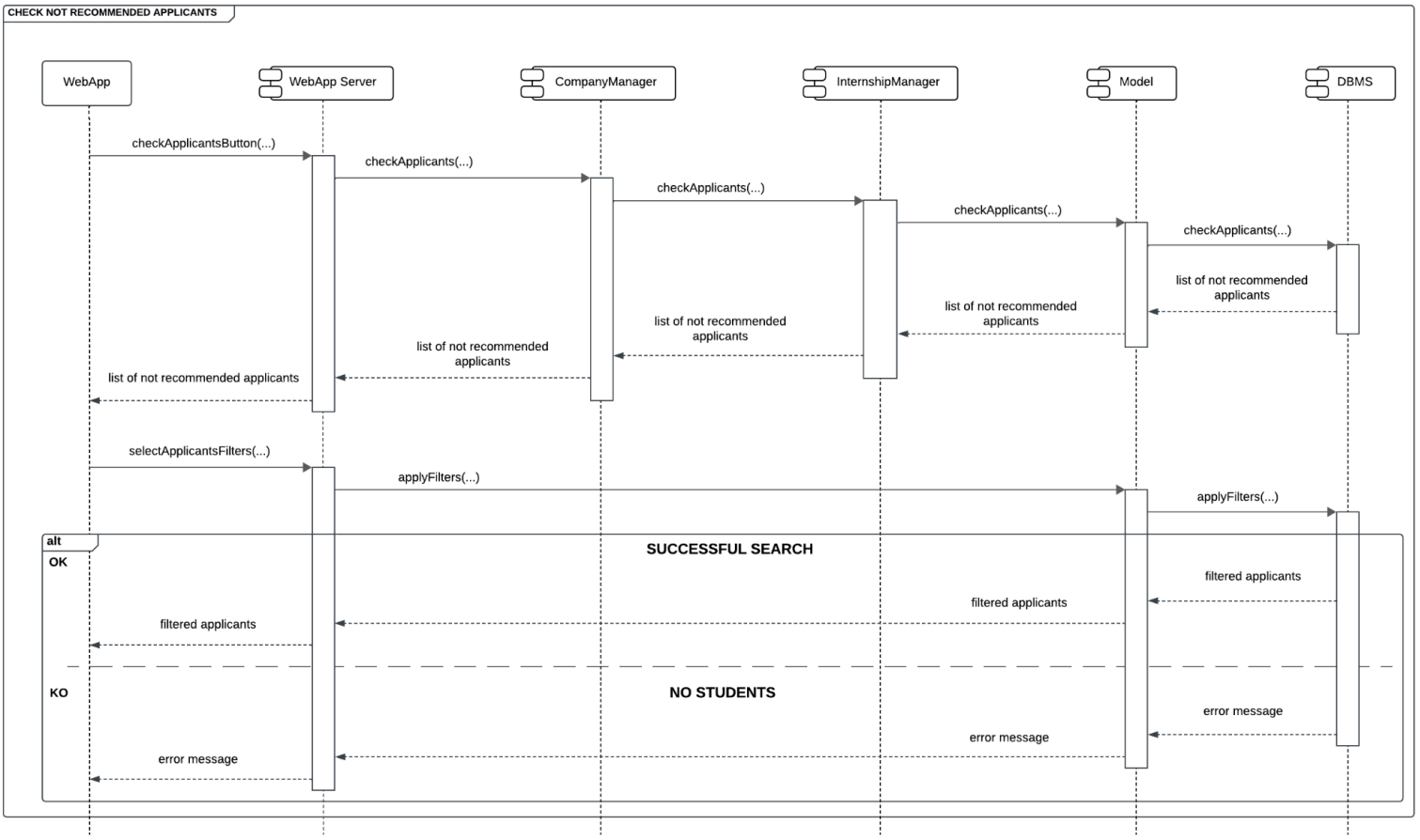
[UC6] - Manage CV

The following runtime view shows the procedure for managing the CV. Apart from the main components, WebAppServer, Model and DBMS, it involves the Student Manager. The Student Manager can reject the modification made only when uploading a new CV in case of an incorrect file format. Thanks to the Student Manager, the tudent can edit or delete the current CV. Moreover, the client may save or discard the changes made to the CV section. It has a pessimistic approach by saving into the DBMS only if the student wants to commit the modifications.


MANAGE CV
WebApp
WebApp Server
StudentManager
Model
DBMS
editProfile()
options to edit profile
alt
[UPLOAD NEW CV]
uploadNewCv(...)
postCV(...)
alt
OK
CORRECT CV FORMAT
new CV uploaded
new CV uploaded
KO
INCORRECT CV FORMAT
error message
error message
[EDIT CURRENT CV]
editCurrentCV(...)
editCV(...)
edited CV
edited CV
[DELETE CURRENT CV]
deleteCurrentCV()
deleteCV(...)
deleted CV
deleted CV
alt
[SAVE CHANGES]
saveChanges()
storeChanges()
storeChanges(...)
storeChanges(...)
profile with changes
profile with changes
profile with changes
profile with changes
[DISCARD CHANGES]
cancelChanges()
discardChanges()
profile without changes
profile without changes

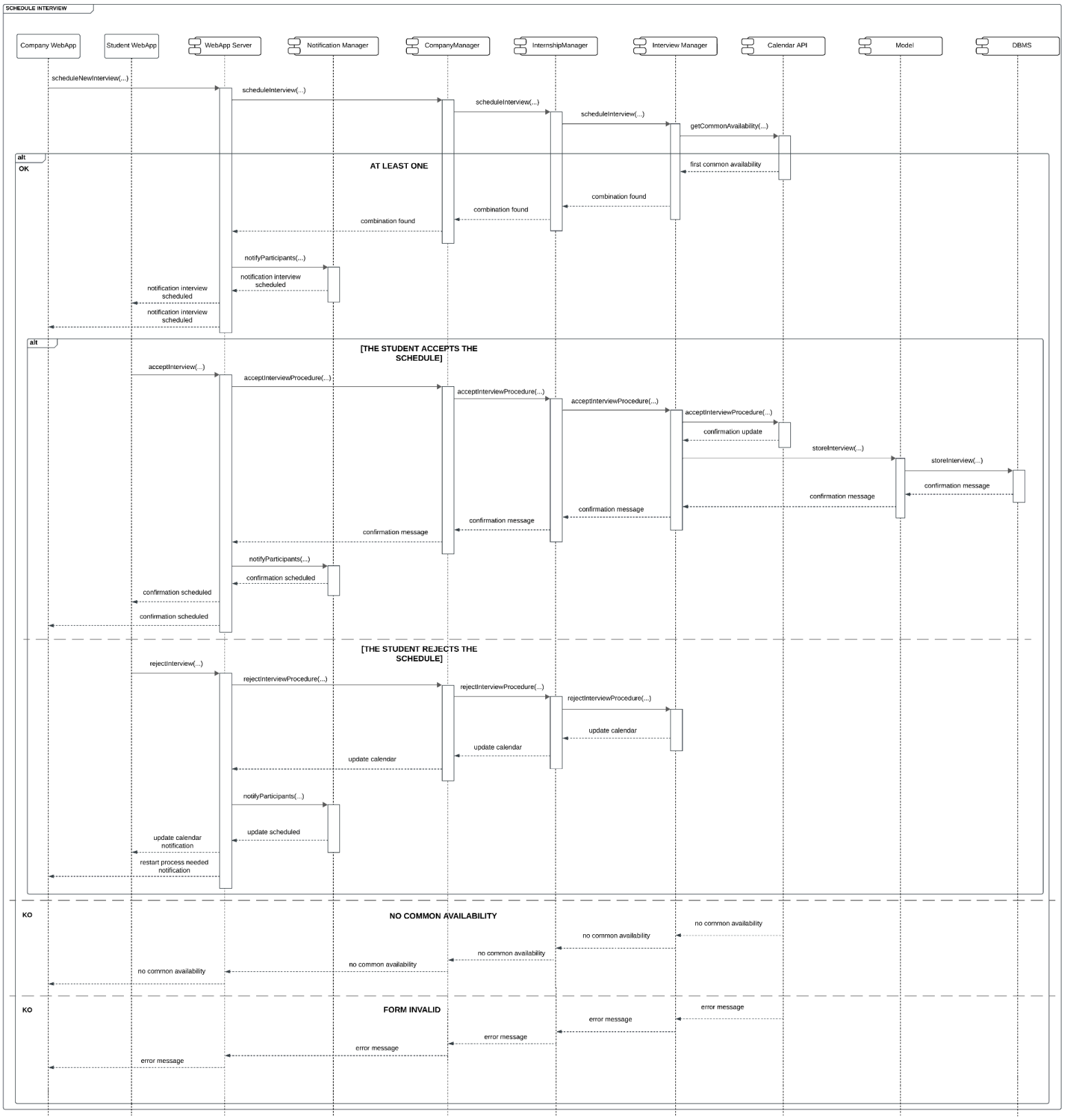
[UC7] - Check Not Recommended Applicants

The following runtime view shows the procedure for checking the not recommended applicants. Apart from the main components, WebAppServer, Model and DBMS, it involves the Company Manager and the Internship Manager. The Internship Manager is responsible for getting the list of not recommended applicants.



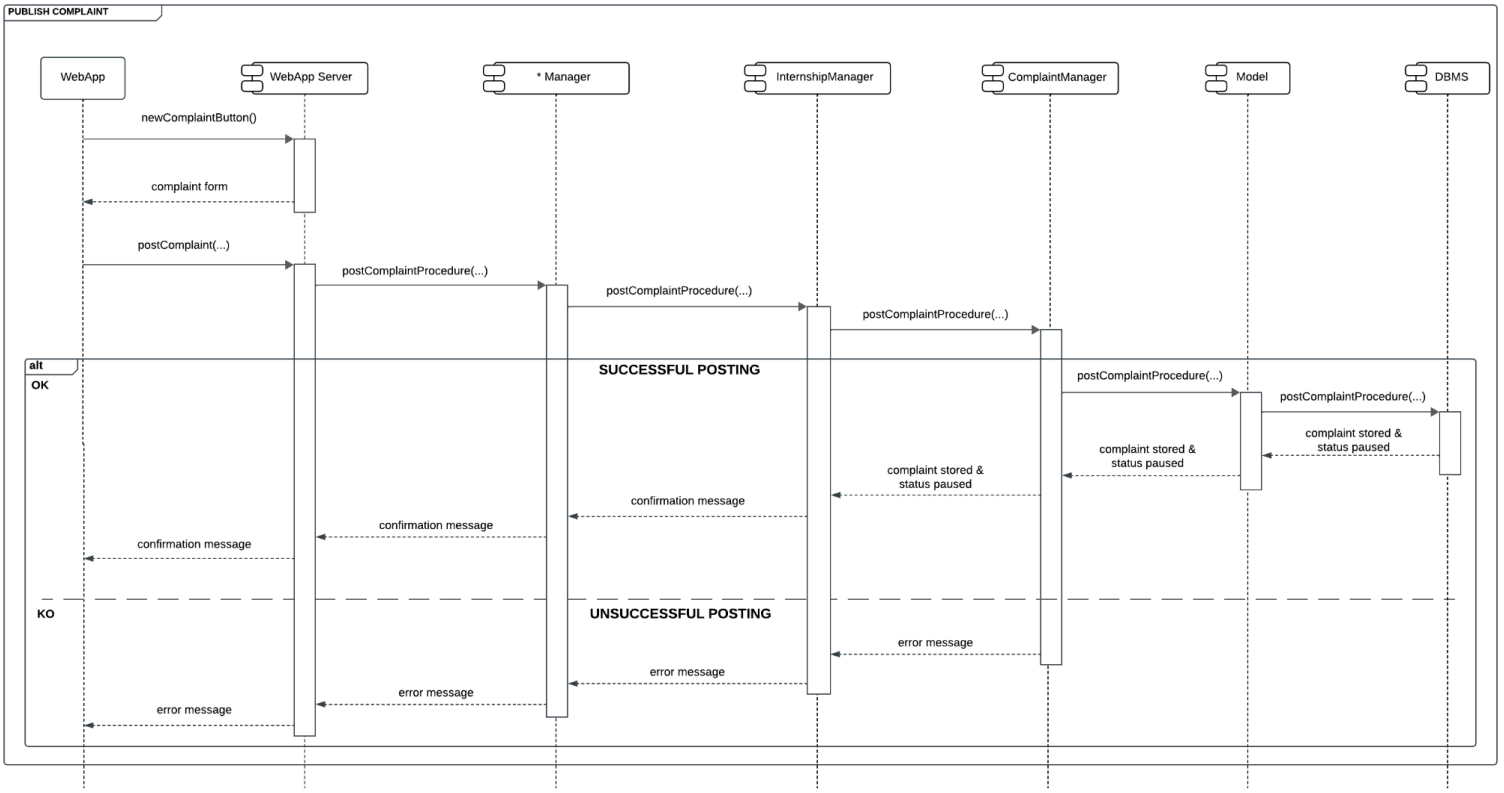
[UC8] - Schedule Interview

The following runtime view shows the procedure for scheduling interviews. Apart from the main components, WebAppServer, Model and DBMS, it involves the Notification Manager, Company Manager, Internship Manager, Interview Manager and the outside component Calendar API. The Notification Manager, called by the WebApp Server, is responsible for the notifications to participants (schedule interview, accept interview and reject interview). The Interview Manager is responsible for communicating with the Calendar API and, if thestudent accepts the interview, it is responsible for updating the DBMS by calling the appropriate method on Model. If there are no common availabilities, the Interview Manager, after the first interaction with Calendar API, returns an error message.


SCHEDULE INTERVIEW
Company WebApp
Student WebApp
WebApp Server
Notification Manager
CompanyManager
InternshipManager
Interview Manager
Calendar API
Model
DBMS
scheduleNewInterview(...)
scheduleInterview(...)
scheduleInterview(...)
scheduleInterview(...)
getCommonAvailability(...)
alt
OK
AT LEAST ONE
first common availability
combination found
combination found
combination found
notifyParticipants(...)
notification interview scheduled
notification interview scheduled
notification interview scheduled
alt
[THE STUDENT ACCEPTS THE SCHEDULE]
acceptInterview(...)
acceptInterviewProcedure(...)
acceptInterviewProcedure(...)
acceptInterviewProcedure(...)
acceptInterviewProcedure(...)
confirmation update
storeInterview(...)
storeInterview(...)
confirmation message
confirmation message
confirmation message
confirmation message
confirmation message
notifyParticipants(...)
confirmation scheduled
confirmation scheduled
confirmation scheduled
[THE STUDENT REJECTS THE SCHEDULE]
rejectInterview(...)
rejectInterviewProcedure(...)
rejectInterviewProcedure(...)
rejectInterviewProcedure(...)
update calendar
update calendar
update calendar
notifyParticipants(...)
update scheduled
update calendar notification
restart process needed notification
KO
NO COMMON AVAILABILITY
no common availability
no common availability
no common availability
no common availability
no common availability
KO
FORM INVALID
error message
error message
error message
error message
error message

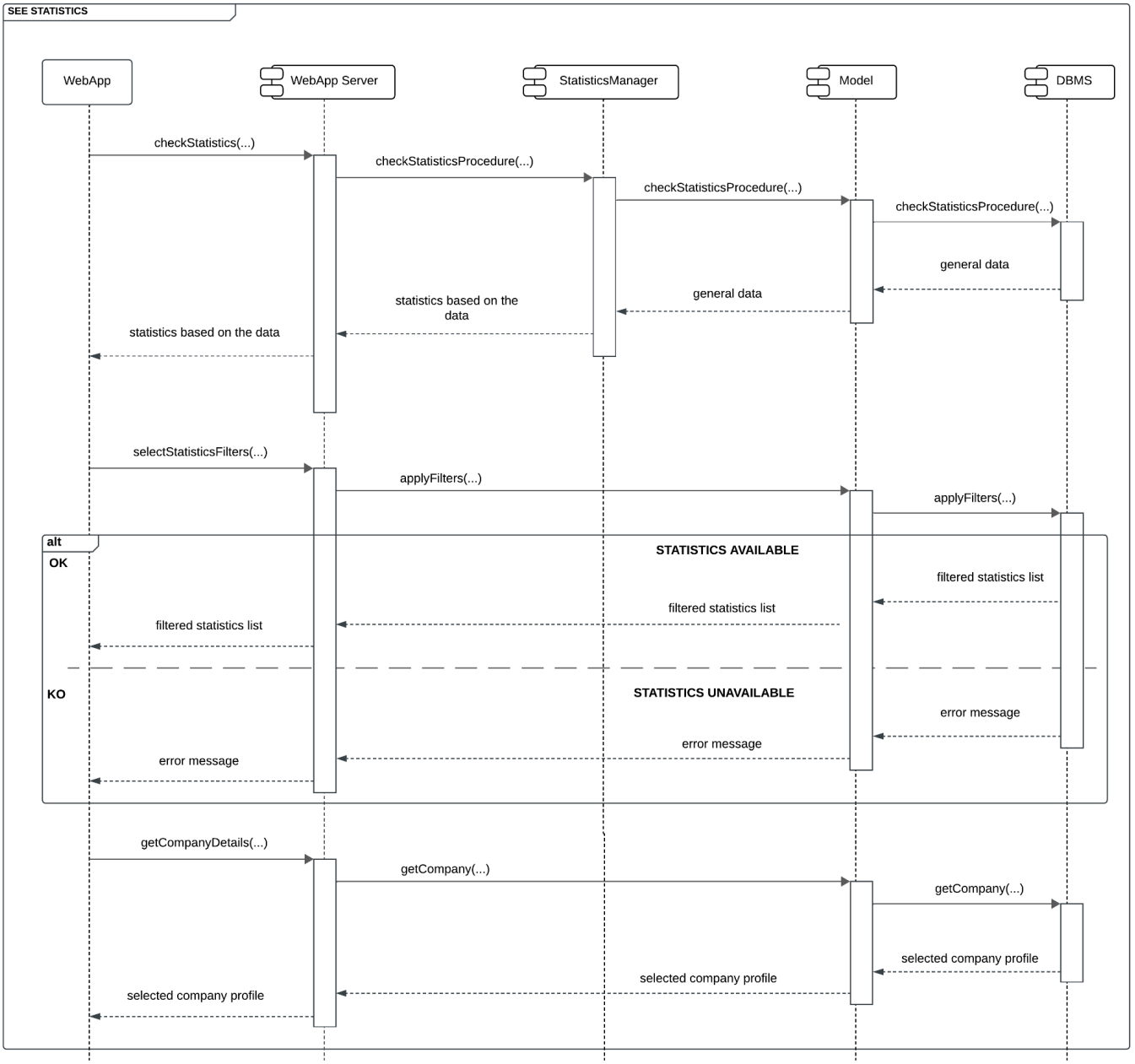
[UC9] - Publish complaint

The following runtime view shows the procedure for publishing complaints. Apart from the main components, WebAppServer, Model and DBMS, it involves the Student or Company Manager as it depends on the situation, the Internship Manager and the Complaint Manager. The Complaint manager is responsible for calling the appropriate Method on the model in order to successfully publish a complaint. Moreover, if the Complaint Manager encounters an error on the complaint procedure, it can stop the posting.



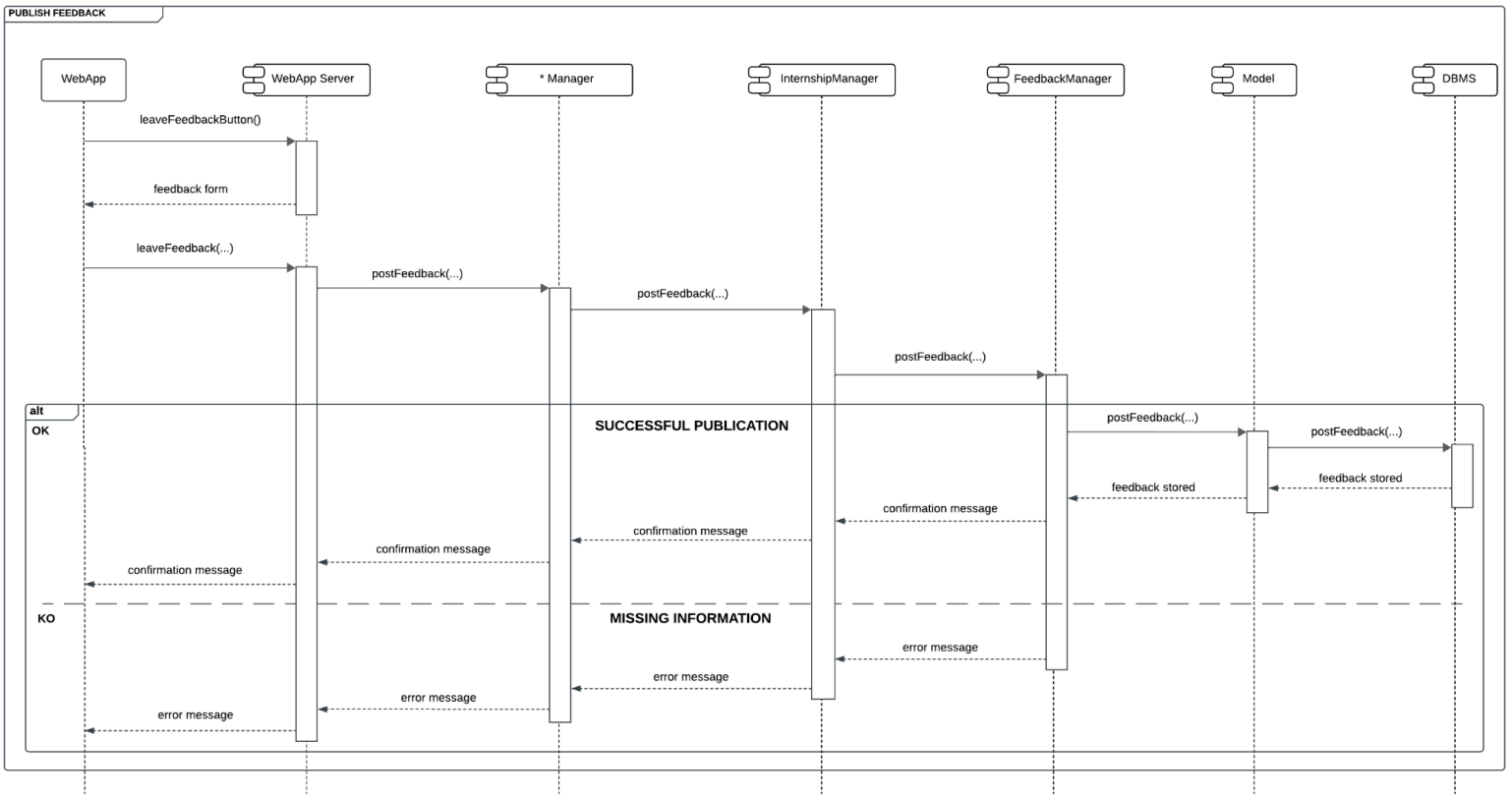
[UC10] - See statistics

The following runtime view shows the procedure for seeing statistics. Apart from the main components, WebAppServer, Model and DBMS, it involves only the Statistics Manager. The Statistics Manager is responsible for retrieving the appropriate data on the DBMS and it generates interesting statistics based on that data.



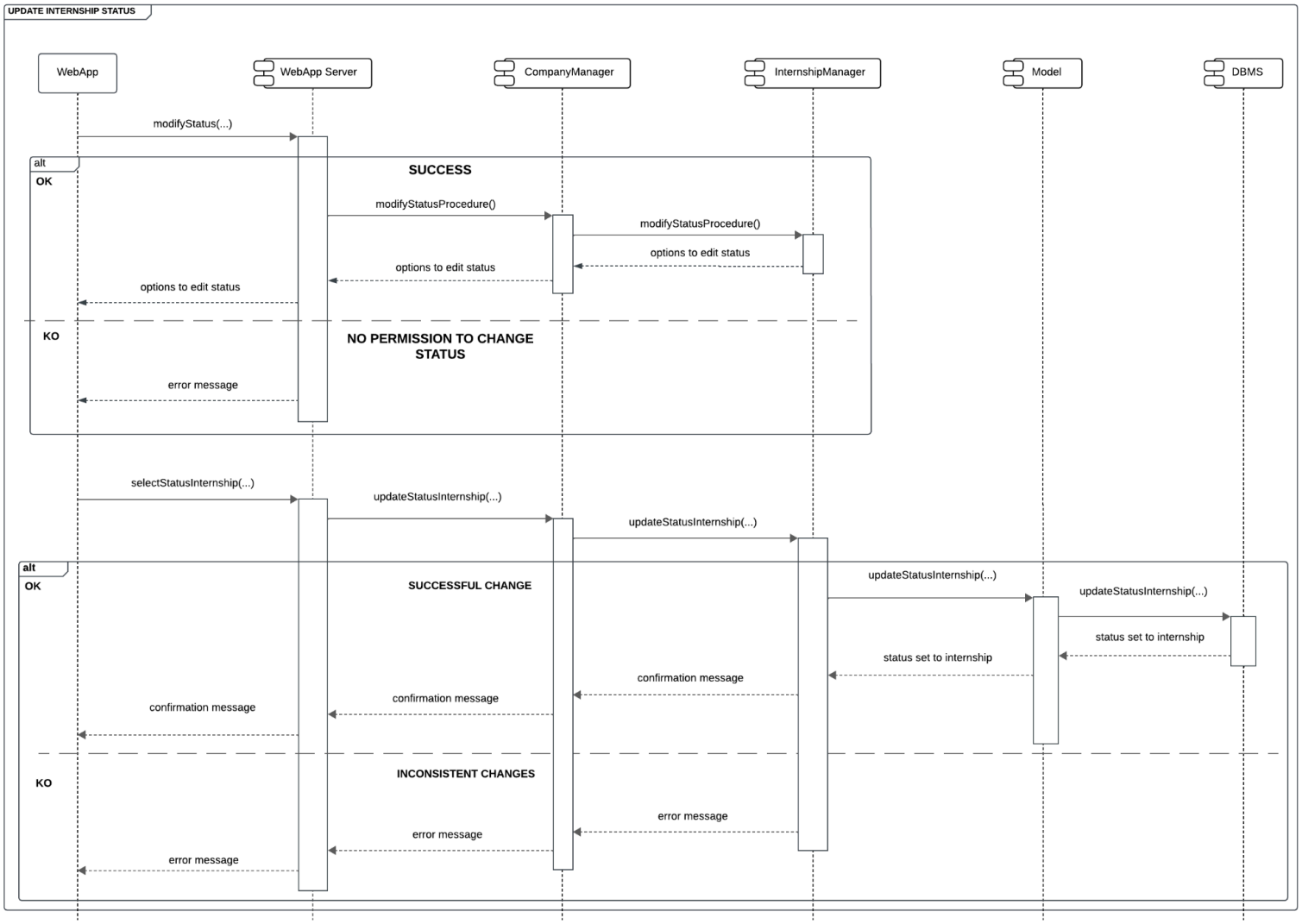
[UC11] - Publish feedback

The following runtime view shows the procedure for publishing feedback. Apart from the main components, WebAppServer, Model and DBMS, it involves the Student or Company Manager as it depends on the situation, the Internship Manager and the Feedback Manager. The Feedback Manager is responsible for calling the appropriate method on the model in order to successfully publish a feedback. Moreover, there is an error on the feedback posting, the Feedback Manager detects it and stops the procedure.



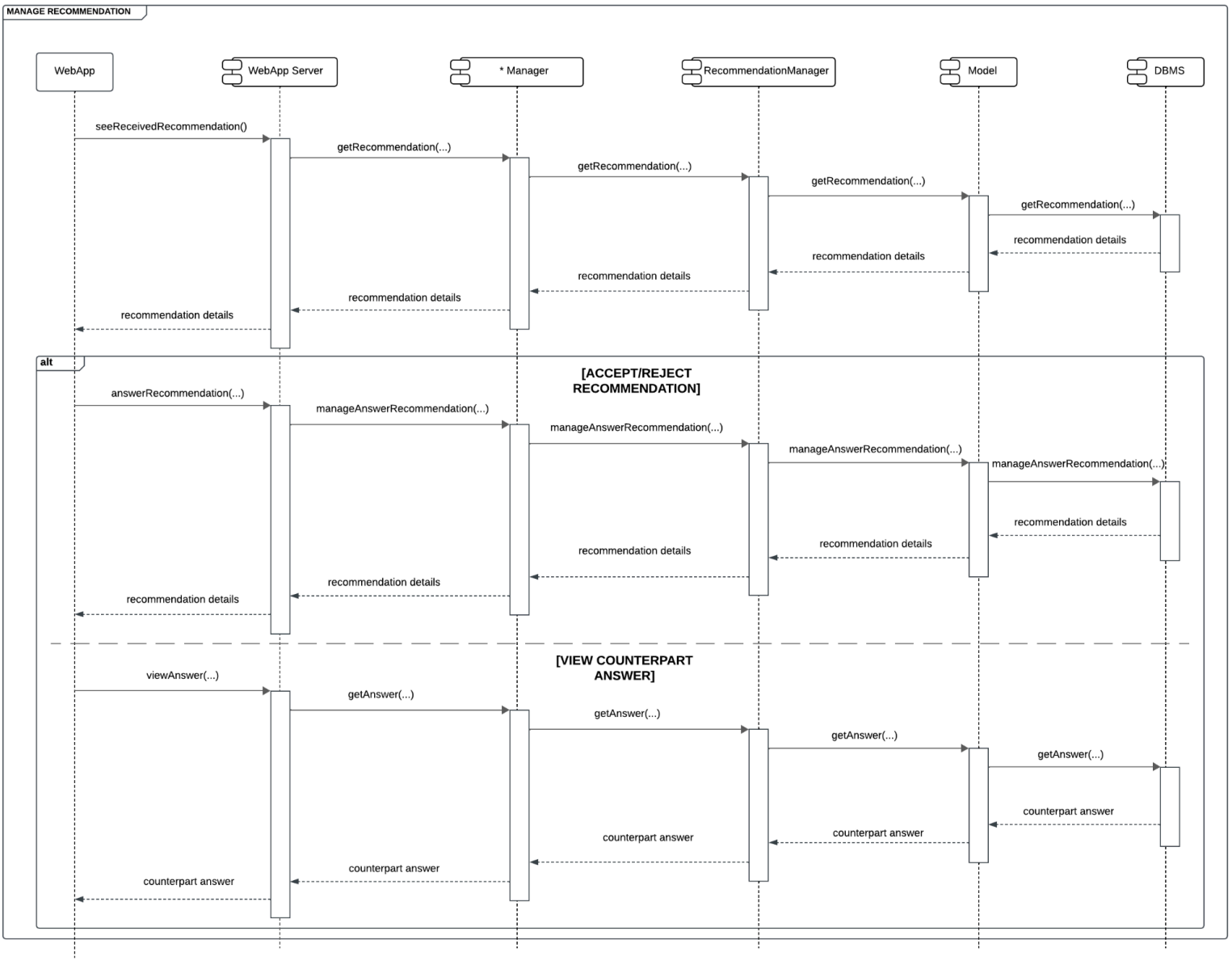
[UC12] - Update status of the internship

The following runtime view shows the procedure for updating the status of an internship. Apart from the main components, WebAppServer, Model and DBMS, it involves the Company Manager and the Internship Manager. The WebApp Server checks if the employee has the permission to change the status of an internship. Afterwards, the Internship Manager is responsible for calling the appropriate method on the model in order to modify the DBMS. The Internship Manager may stop the procedure if the new change is inconsistent with the status already present in the system.



[UC13] - Manage Recommendation

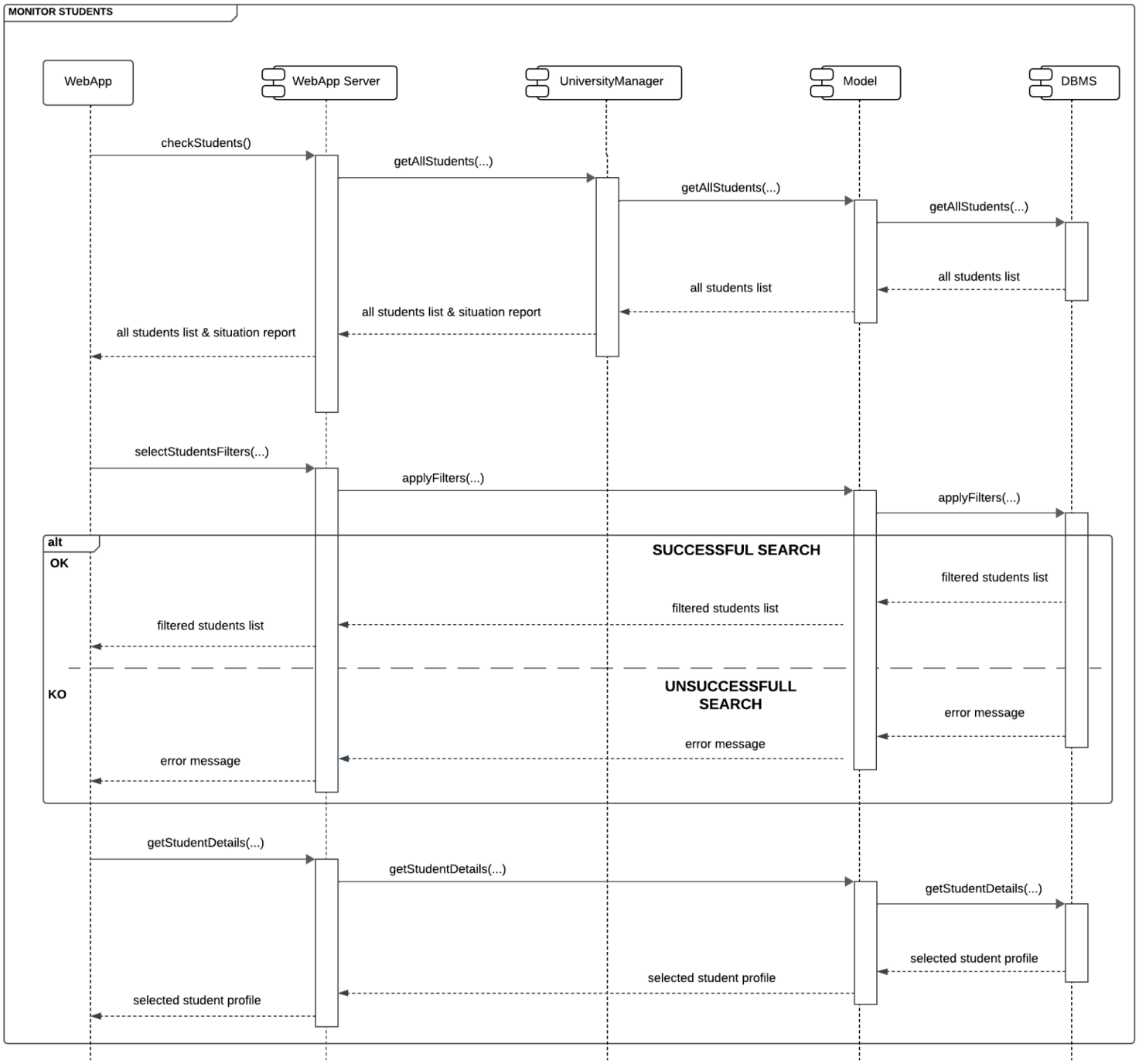
The following runtime view shows the procedure for managing recommendations. Apart from the main components, WebAppServer, Model and DBMS, it involves the Student or Company Manager as it depends on the situation and the Recommendation Manager. The Recommendation Manager is responsible for retrieving recommendations to the user, for updating the user recommendation, whether it has been accepted or rejected, and for visualizing the counterpart answer.



[UC14] - University monitors its students

The following runtime view shows the procedure for monitoring students. Apart from the main components, WebAppServer, Model and DBMS, it involves the University Manager. The University Manager is responsible for retrieving the student list of the university and for

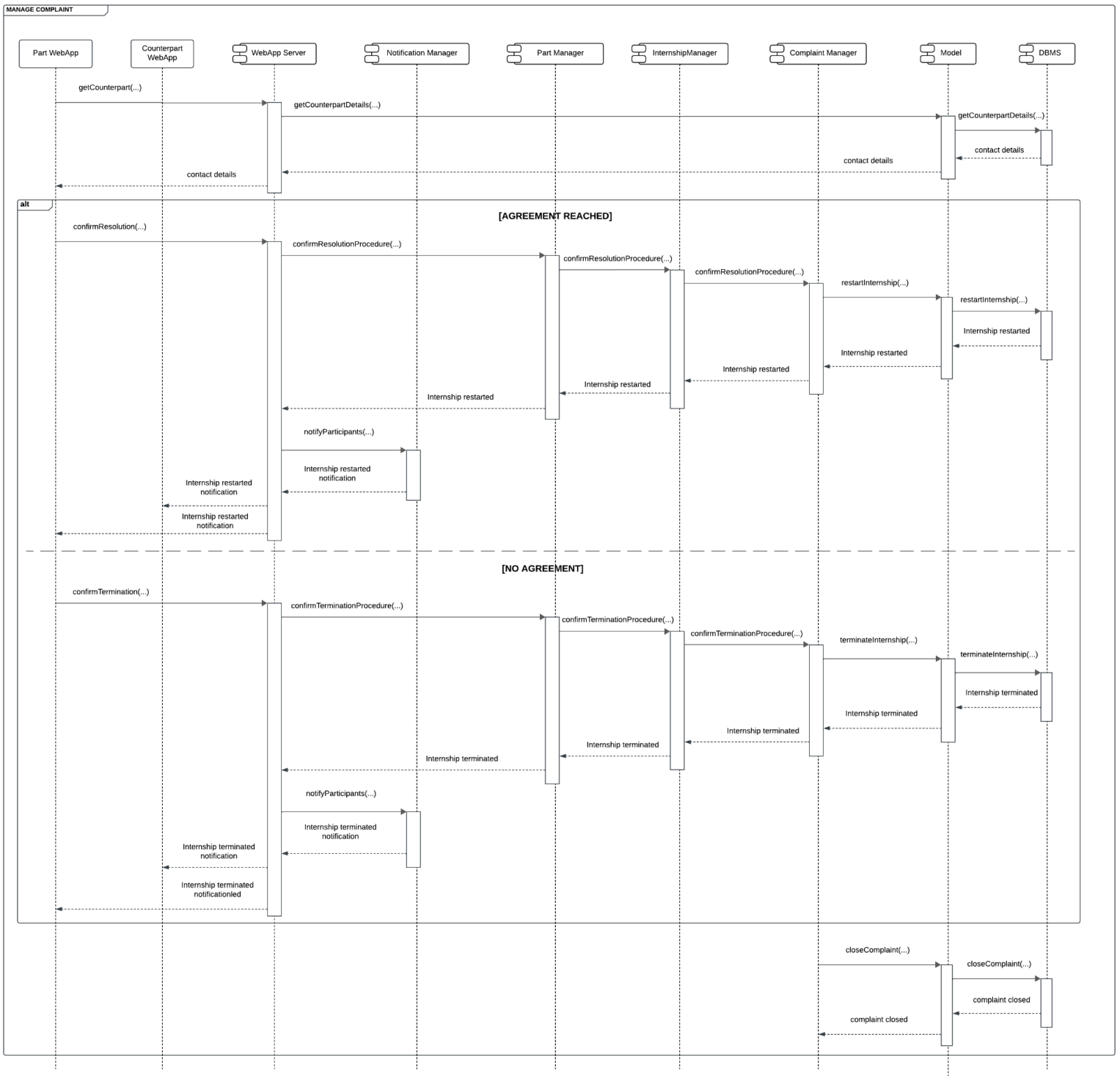
creating a report in order to show the employee what is the situation of their students in an accurate format.



[UC15] - Manage Complaint

The following runtime view shows the procedure for managing complaints. Apart from the main components, WebAppServer, Model and DBMS, it involves the Notification Manager, the Company Manager and the Complaint Manager. This scenario shows the complaint

made by the Company. The reversed scenario is equivalent. The notification manager, called by the WebApp Server, is responsible for the notifications to participants (internship restarted or internship terminated). The complaint manager is responsible for calling the methods on Model in order to restart the internship, if a positive agreement has been reached, or in order to terminate the internship otherwise. In any case, at the end of the scenario, the complaint manager calls on Model the appropriate methods in order to close the complaint.



2.5 Component Interfaces

WebApp Server

- registerOrganizationButton()
- sendOrganizationDetails(name, email, details)
- loginButton()
- sendOrganizationName(name)
- requestDashboard(Token)
- postInternshipButton()
- sendInternshipDetails(name, status, deadlineApplication, project, terms)
- browseInternships()
- selectInternshipFilters(internships, filters)
- getInternshipDetails(internshipName)
- applyButton()
- submitApplication(internshipName)
- editProfile()
- uploadNewCv(cvFile)
- editCurrentCV(changedText)
- deleteCurrentCV()
- saveChanges()
- cancelChanges()
- checkApplicantsButton(internshipName)
- selectApplicantsFilters(applicants, filters)
- scheduleNewInterview(studentName, internshipName)
- acceptInterview(companyName, interviewID, internshipName)
- rejectInterview(companyName, interviewID, internshipName)
- newComplaintButton()
- postComplaint(internshipName, complaintText, studentName, companyName)
- checkStatistics(category)
- selectStatisticsFilters(category, filters)
- getCompanyDetails(companyName)
- leaveFeedbackButton()
- leaveFeedback(internshipName, feedbackText, studentName, companyName)
- modifyStatus(hasPermission)
- selectStatusInternship(internshipName, newStatus)
- seeReceivedRecommendation()
- answerRecommendation(recommendationID, answer)

- viewAnswer(recommendationID)
- checkStudents()
- selectStudentsFilters(students, filters)
- getStudentDetails(studentName)
- getCounterpart(counterpartName)
- confirmResolution(internshipName, complaintID, counterpartName)
- confirmTermination(internshipName, complaintID, counterpartName)

SSO Manager

- getDashboard(Token)

Registration Manager

- postOrganization(name, email, details)

Company Manager

- postinternshipDetails(name, status, deadlineApplication, project, terms)
- checkApplicants(companyName, internshipName)
- scheduleInterview(companyName, studentName, internshipName)
- acceptInterviewProcedure(companyName, studentID, interviewID, internshipName)
- rejectInterviewProcedure(companyName, studentID, interviewID, internshipName)
- postComplaintProcedure(internshipName, complaintText, studentName, companyName)
- postFeedback(internshipName, feedbackText, studentName, companyName)
- modifyStatusProcedure()
- updateStatusInternship(companyName, internshipName, newStatus)
- getRecommendation(companyName)
- manageAnswerRecommendation(userID, recommendationID, answer)
- getAnswer(userID, recommendationID)
- confirmResolutionProcedure(userID, internshipName, complaintID)
- confirmTerminationProcedure(userID, internshipName, complaintID)

Statistics Manager

- checkStatisticsProcedure(category)

Feedback Manager

- postFeedback(internshipName, feedbackText, studentName, companyName)

University Manager

- getAllStudents(universityID)

Notification Manager

- notifyParticipants(studentName, companyName)

DBMS

- postOrganization(name, email, details)

- getOrganizationWebsite(name)
- getDashboard(Token)
- postinternshipDetails(name, status, deadlineApplication, project, terms)
- getInternships()
- applyFilters(listOfItems, filters)
- getInternshipDetails(internshipName)
- postApplication(studentID, internshipName)
- addAppliedStudent(studentID, internshipID)
- storeChanges(logOfChanges)
- checkApplicants(companyName, internshipName)
- storeInterview(companyName, internshipName, interviewID, studentID)
- postComplaintProcedure(internshipName, complaintText, studentName, companyName)
- checkStatisticsProcedure(category)
- getCompany(companyName)
- postFeedback(internshipName, feedbackText, studentName, companyName)
- updateStatusInternship(company, internship, newStatus)
- getRecommendation(userID)
- manageAnswerRecommendation(userID, recommendationID, answer)
- getAnswer(userID, recommendationID)
- getAllStudents(universityID)
- getStudentDetails(universityID, studentID)
- getCounterpartDetails(counterpartID)
- restartInternship(internshipName)
- terminateInternship(internshipName)
- closeComplaint(internshipName, complaintID)

Model

- postOrganization(name, email, details)
- getOrganizationWebsite(name)
- getDashboard(Token)
- postinternshipDetails(name, status, deadlineApplication, project, terms)
- getInternships()
- applyFilters(listOfItems, filters)
- getInternshipDetails(internshipName)
- postApplication(studentID, internshipName)
- addAppliedStudent(studentID, internshipID)
- storeChanges(logOfChanges)

- checkApplicants(companyName, internshipName)
- storeInterview(companyName, internshipName, interviewID, studentID)
- postComplaintProcedure(internshipName, complaintText, studentName, companyName)
- checkStatisticsProcedure(category)
- getCompany(companyName)
- postFeedback(internshipName, feedbackText, studentName, companyName)
- updateStatusInternship(company, internship, newStatus)
- getRecommendation(userID)
- manageAnswerRecommendation(userID, recommendationID, answer)
- getAnswer(userID, recommendationID)
- getAllStudents(universityID)
- getStudentDetails(universityID, studentID)
- getCounterpartDetails(counterpartID)
- restartInternship(internshipName)
- terminateInternship(internshipName)
- closeComplaint(internshipName, complaintID)

Internship Manager

- **InternshipForCompany**
    - postinternshipDetails(name, status, deadlineApplication, project, terms)
    - checkApplicants(companyName, internshipName)
    - scheduleInterview(company, studentName, internshipName)
    - acceptInterviewProcedure(company, studentID, interviewID, internshipName)
    - rejectInterviewProcedure(company, studentID, interviewID, internshipName)
    - postComplaintProcedure(internshipName, complaintText, studentName, companyName)
    - postFeedback(internshipName, feedbackText, studentName, companyName)
    - modifyStatusProcedure()
    - updateStatusInternship(company, internshipName, newStatus)
    - confirmResolutionProcedure(user, internshipName, complaintID)
    - confirmTerminationProcedure(user, internshipName, complaintID)
- **InternshipForStudent**
    - postComplaintProcedure(internshipName, complaintText, studentName, companyName)
    - postFeedback(internshipName, feedbackText, studentName, companyName)
    - confirmResolutionProcedure(user, internshipName, complaintID)
    - confirmTerminationProcedure(user, internshipName, complaintID)

- **InternshipForApplication**
    - addAppliedStudent(student, internshipName)

Interview Manager

- scheduleInterview(company, studentName)
- acceptInterviewProcedure(company, studentID, internship, interviewID)
- rejectInterviewProcedure(company, studentID, internship, interviewID)

Complaint Manager

- postComplaintProcedure(internshipName, complaintText, studentName, companyName)
- confirmResolutionProcedure(internship, complaintID)
- confirmTerminationProcedure(internship, complaintID)

Recommendation Manager

- getRecommendation(user)
- manageAnswerRecommendation(user, recommendationID, answer)
- getAnswer(user, recommendationID)

Student Manager

- applyToInternship()
- postApplication(studentName, internshipName)
- postCV(cvFile)
- editCV(changedText)
- deleteCV(cvFile)
- storeChanges()
- discardChanges()
- postComplaintProcedure(internshipName, complaintText, studentName, companyName)
- postFeedback(internshipName, feedbackText, studentName, companyName)
- getRecommendation(studentID)
- manageAnswerRecommendation(userID, recommendationID, answer)
- getAnswer(userID, recommendationID)
- confirmResolutionProcedure(userID, internshipName, complaintID)
- confirmTerminationProcedure(userID, internshipName, complaintID)

Application Manager

- applyToInternship()
- postApplication(student, internshipName)

In Model and DBMS, the method applyFilters has the parameter List<T> listOfItems that is a generic list.

2.6 Selected architectural styles and patterns

2.6.1 Three-Tier Architecture

The S&C platform is built over a three-tier architecture, which provides numerous advantages thanks to the modularization of the system into independent tiers:

1. **Presentation Tier:** it includes the user interface for students, university employees and company employees. The primary function is to organize and display the data received for the application tier in an easy and user-friendly interface.
2. **Business Logic Tier:** it includes the business and core logic of the S&C platform, containing the main algorithms. It processes data exchanges between the presentation and data tiers.
3. **Data Tier:** it includes the database system, which stores all critical data allowing the platform to function. It exposes an API to the business logic tier for an efficient data exchange.

Thanks to the adoption of this architecture, the platform has a high level of flexibility, because the tiers can be updated and maintained independently and it is a scalable solution, enabling the system to handle increased user traffic efficiently. Moreover, the middle tier between the presentation and data tiers ensures a restricted access to the database, improving the protection level to the data tier.

2.6.2 Model View Controller Pattern

The S&C platform follows the Model-View-Controller (MVC) pattern. This architectural pattern divides the system into three interconnected components:

- **Model:** it represents the core logic and data of the application. In particular, it includes the algorithms which make possible the use of the platform. Moreover, it provides methods for retrieving, updating and processing the data.
- **View:** it represents the presentation layer of the application. It includes all possible visual representations of the data, efficiently translating the data from Model to a user-friendly interface.
- **Controller:** it represents the intermediary between the model and the view. It handles user inputs and events, triggering the corresponding actions on the Model. Consequently, the modifications of the model may trigger updates in the View to reflect the changes.

The main advantage of using MVC is the distribution of tasks between each component, making it easier to maintain and scale the system. Moreover, the modular design ensures

reusability of individual components. By adopting the MVC pattern, the S&C platform results in a robust, scalable and maintainable system.

2.6.3 Facade Pattern

The facade pattern is used in the implementation of the Web App Server component within the S&C platform to abstract the complexities of the application server from the client applications. The pattern ensures an easy interaction between the client side and server side. Moreover, the facade pattern provides a strong decoupling of the various components of the S&C platform, in particular between the client application and the application servers.

2.7 Other Design Decision

2.7.1 Data Storage

The S&C platform employs a robust and scalable database in order to manage large volumes of structured and unstructured data. For efficient data retrieval, relational databases handle the core application data, such as user profiles and internship posting, while object storage is used to store documents, such as CVs. S&C platform regularly performs backups and employs encryption mechanisms to ensure data security.

2.7.2 Availability

The S&C platform must be highly available to ensure uninterrupted access for users. Therefore, it is necessary to deploy redundancy mechanisms and load balancers to minimize downtime and increase the availability and reliability of the system. Moreover, the use of distributed systems principles, such as fault tolerance, ensures a highly available system.

2.7.3 Security

The S&C platform ensures security of user data, placing integrity and security as a priority. The platform employs multiple layers of security mechanisms to protect itself. In fact, the platform supports Single Sign-On (SSO) for user authentication, ensuring that authorized users can access only their information and perform the available actions. Moreover, the data stored in the database is encrypted following industry standard protocols. Security is a priority for S&C platform, as well as the compliance with data protection regulations.

3. User Interface Design

As stated in the RASD file, the following images briefly describe the UI for the S&C platform. In particular, the first image shows the Welcome page of the platform, where users can login using SSO or register their specific organization. The other two images describe the dashboard of users. The first dashboard is a company's dashboard, where the employee can surf on the platform using the menu on the left side. On the right side of the screen, he will be able to get details about what he is looking for (in this case, the employee is looking for a qualified student). The second dashboard is a student's dashboard, where the student can surf on the platform using the menu on the left side. On the right side of the screen, he will be able to get details about what he is looking for (in this case, the student is looking for an internship).

Company's Profile

Publish Internship

Internships in Progress

Applications in Progress

Recommendations

Settings

Contact Us

Log Out

Are you looking for a qualified student, TechNova?

You are in the right place!

Search on Students&Companies.com

Rodrigo Almandoz Franco
- University: Politecnico di Milano
- Skills: Foreign languages, Data Structures....

Details

Mattia Brandi
- University: Politecnico di Milano
- Skills: Computer Science, Data Structures....

Details

Paola Rossi
- University: Università di Milano
- Skills: Marketing, Corporate Communications....

Details

Giancarlo Certo
- University: Università di Pisa
- Skills: Mathematics, Physics, Chemistry...

Details

University
- ☐ Politecnico di Milano
- ☐ Politecnico di Torino
- ☐ Università di Trento

>>> see others

Required Skills
- ☐ Verbal communication
- ☐ Work in group
- ☐ Problem Solving

>>> see others

Location
- ☐ Milan, Italy
- ☐ Rome, Italy

My Profile

Search Internships

Applications in Progress

Recommendations

Settings

Contact Us

Are you looking for an internship, Rodrigo?

You are in the right place!

Search on Students&Companies.com

Data Science Intern, **TechNova Analytics**
- Location: Berlin, Germany
- Paid Internship

Details

Web Development Intern, **WebSites Solutions**
- Location: Madrid, Spain
- Unpaid Internship

Details

Human Resources Intern, **TalentSeeker**
- Location: Paris, France
- Unpaid Internship

Details

Marketing and Communications Intern, **GlobalReach Marketing**
- Location: London
- Paid Internship

Details

Company
- ☐ TechNova Analytics
- ☐ Deloitte
- ☐ Apple

>>> see others

Required Skills
- ☐ Verbal communication
- ☐ Work in group
- ☐ Problem Solving

>>> see others

Duration
- ☐ 6 months

4. Requirements Traceability

Requirements	[R1] The system allows users to access their private area through SSO.
Components	• WebApp • WebApp Server • SSO Manager • Model • DBMS

Requirements	[R2] The system allows Universities and Companies to sign an agreement for registering in the S&C platform.
Components	• WebApp • WebApp Server • Registration Manager • Model • DBMS

Requirements	[R3] The system allows Companies to create and publish the internships (projects and terms) they offer. [R6] The system allows Company to see Student CVs [R13] The system allows Companies to review the applications. [R18] The system allows Companies to finalize the selection of the best candidates. [R19] The system allows Companies to offer an internship through proper tools.
Components	• WebApp • WebApp Server • Company Manager

	• Internship Manager • Model • DBMS

Requirements	[R4] The system allows Students to manage their CV (skills, experiences, attitudes).
Components	• WebApp • WebApp Server • Student Manager • Model • DBMS

Requirements	[R5] The system allows Students to search for internships in the search bar. [R7] The system allows Students to filter and order the internships based on preferences.
Components	• WebApp • WebApp Server • Model • DBMS

Requirements	[R9] The system allows Students to review their specific applications. [R20] The system allows Students to manage the offer by the Company.
Components	• WebApp • WebApp Server • Student Manager • Application Manager

	• Model • DBMS

Requirements	[R8] The system allows Students to apply at most once to a specific internship
Components	• WebApp • WebApp Server • Student Manager • Application Manager • Internship Manager • Model • DBMS

Requirements	[R10] The system allows to send recommendations to interested parties. [R11] The system allows interested parties to manage received recommendations.
Components	• WebApp • WebApp Server • Student Manager/Company Manager • Recommendation Manager • Model • DBMS

Requirements	[R12] The system allows to send notifications to interested parties.
Components	• WebApp • WebApp Server • Notification Manager

Requirements	[R15] The system allows Companies to schedule an interview. [R16] The system allows Companies to use structured questionnaires. [R17] The system allows interested parties to see the details of an interview.
Components	● WebApp ● WebApp Server ● Notification Manager ● Company Manager ● Internship Manager ● Interview Manager ● Model ● DBMS

Requirements	[R14] The system allows to establish contact between interested parties. [R21] The system allows Companies to pause the internship. [R22] The system allows Companies to terminate the internship prematurely. [R23] The system allows interested parties to publish complaints about problems during the internship.
Components	● WebApp ● WebApp Server ● Notification Manager ● Company Manager ● Internship Manager ● Complaint Manager ● Model ● DBMS

Requirements	[R24] The system allows interested parties to publish feedback about the internship or the counterpart after the internship.
Components	• WebApp • WebApp Server • Student Manager or Company Manager • Internship Manager • Feedback Manager • Model • DBMS

Requirements	[R25] The system allows Universities to track and oversee the progress and status of internships undertaken by their students.
Components	• WebApp • WebApp Server • University Manager • Model • DBMS

Requirements	[R26] The system allows users to see and analyze up to date statistics.
Components	• WebApp • WebApp Server • Statistics Manager • Model • DBMS

5. Implementation, Integration and Test Plan

5.1 Overview

The following chapter focuses on the implementation of the S&C platform, detailing the approach for integrating and validating the various components. The aim of the tests implemented is to identify and solve the application bugs before each release. Integration and implementation are strictly correlated and therefore, the integration sequence depends on the implementation order. As a consequence, this chapter also considers the integration test plan to ensure a unique process. Finally, during the implementation, the developers must document the codebase to ensure clarity and maintainability.

5.2 Implementation Plan

The implementation of the system will follow a combination of bottom-up and thread strategies to leverage the advantages of both approaches. On one hand, the thread strategy approach involves the identification of the primary features of the platform, determining the components and sub-components necessary to deliver them. By implementing the features as threads, it is possible to deliver intermediate results for the evaluation by stakeholders. On the other hand, the bottom-up strategy facilitates the integration of sub-components in order to deal with interdependencies. This strategy allows the testing of intermediate results, simplifying the identification of bugs and ensuring that integrated modules work as required. The combination of these strategies enables an efficient parallelization of the development process. In fact, features can be assigned to independent teams working concurrently. Anyway, it is critical to identify common components before starting to work on specific features in order to promote productivity and reduce redundancy.

5.2.1 Features Identification

The features below described are taken from the S&C platform requirements. It is relevant to highlight that some features require the implementation of entire components while others may require small modifications to existing ones.

This modular implementation strategy allows for the identification of independent tasks, which can be implemented and tested in parallel by independent development teams.

The following features are the most important of the platform.

[F1] Organization registration and Login

These two basic features can be implemented and tested in parallel by different groups of developers. The first one enables the registration of the Companies and the Universities in the database. On the other hand, the second feature provides the necessary elements to obtain and redirect the user to the required webpage where he can execute the login procedure.

[F2] Company profile and internship management

This is a core feature which contains all the functionalities necessary to create and manage a Company profile. It enables Company's employees to publish, edit them and view detailed information about internships.

[F3] University profile and student monitoring

With this feature, the system allows University's employees to create a University profile with an intuitive dashboard, necessary to see the list of their students in order to monitor them efficiently.

[F4] Student profile and application management

This is another core feature that contains all the functionalities necessary to create and manage a Student profile. The student can use the provided functionalities to manage its own CV, with whom he can apply for internships and visualize the list of submitted applications.

[F5] Interview management

This feature represents the support that the platform offers to the interview process managed by the Company's employees. It contains the set of functionalities for scheduling interviews with selected candidates. Moreover, employees can easily interact with the student in order to agree on suitable dates whenever it is necessary. Additionally, the system provides the employee with all the tools to interact with the questionnaires useful to conduct the interview.

[F6] Notification mechanism and feedback management

These two features are independent from each other and they can be implemented in parallel. The first one manages the notification mechanism which handles the system's delivery of notifications from the server to client devices. The second one is necessary to grant a smooth experience to the users which have all the tools to leave positive/negative feedback on specific internships.

[F7] Recommendation and Complaint management

These two features are independent from each other and they all depend on the functionalities provided by the *Notification mechanism* feature previously implemented.

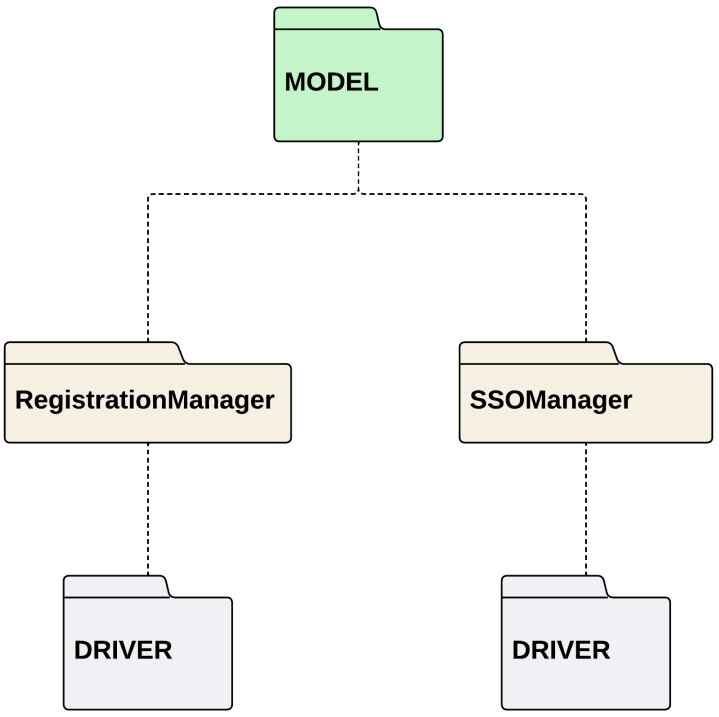
The first one is a core feature of the system that implements the recommendation system which enables S&C to promote suitable students to Companies and viceversa. Meanwhile, the second one provides all the tools and a dedicated space to report issues during internships, as well as to resolve complaints once a satisfactory solution, either positive or negative, is found.

5.3 Component Integration and Testing

In the following section, it is detailed which component is implemented in every stage of the development process and how different components are implemented and tested. The components under testing are shown in light brown while the components already tested are in green.

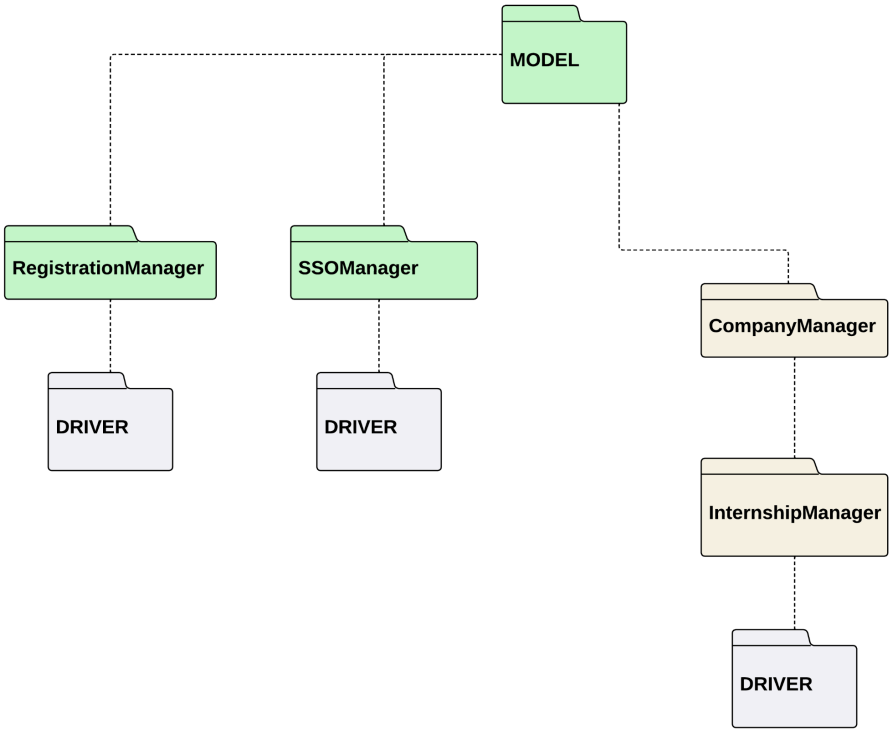
[F1] Organization registration and Login

The following image shows the new components developed and tested for the organization registration and login.



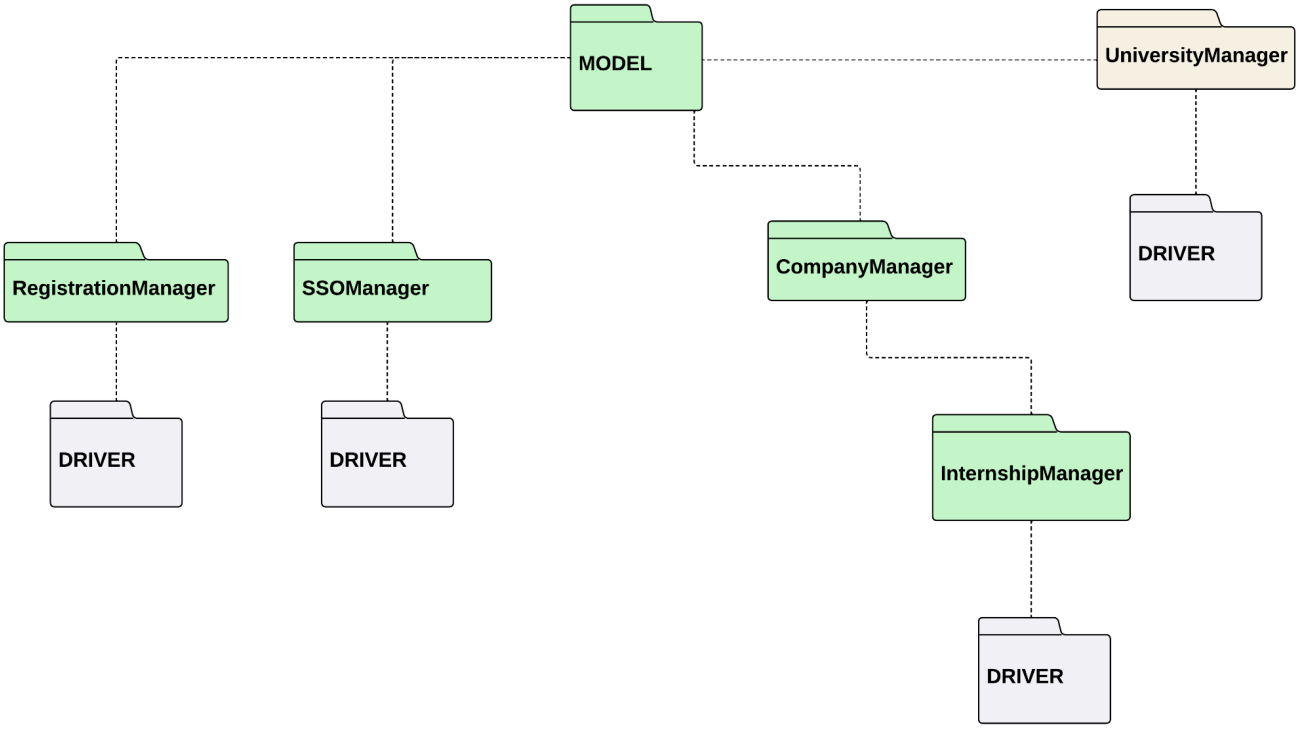
[F2] Company profile and internship management

The following image shows the new components developed and tested for the company profile and internship management.



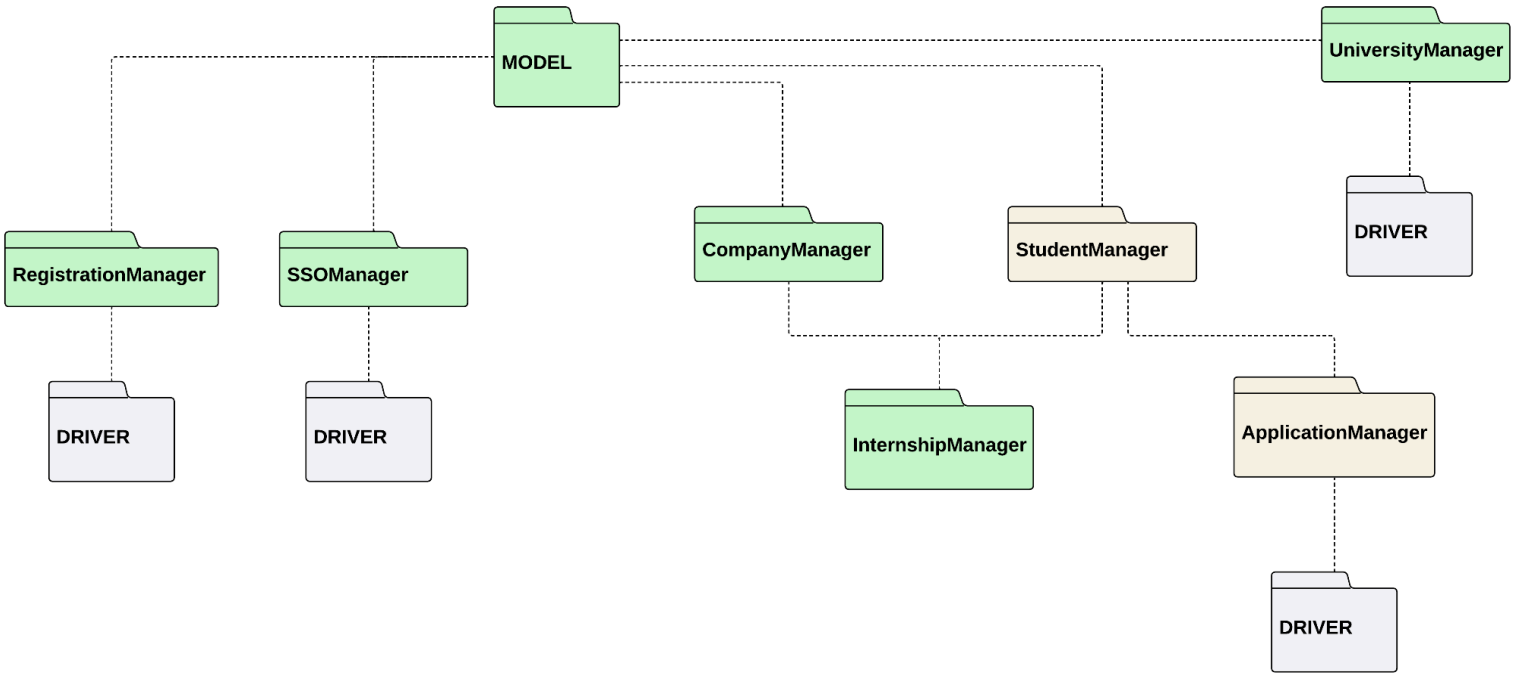
[F3] University profile and student monitoring

The following image shows the new components developed and tested for the university profile and student monitoring.



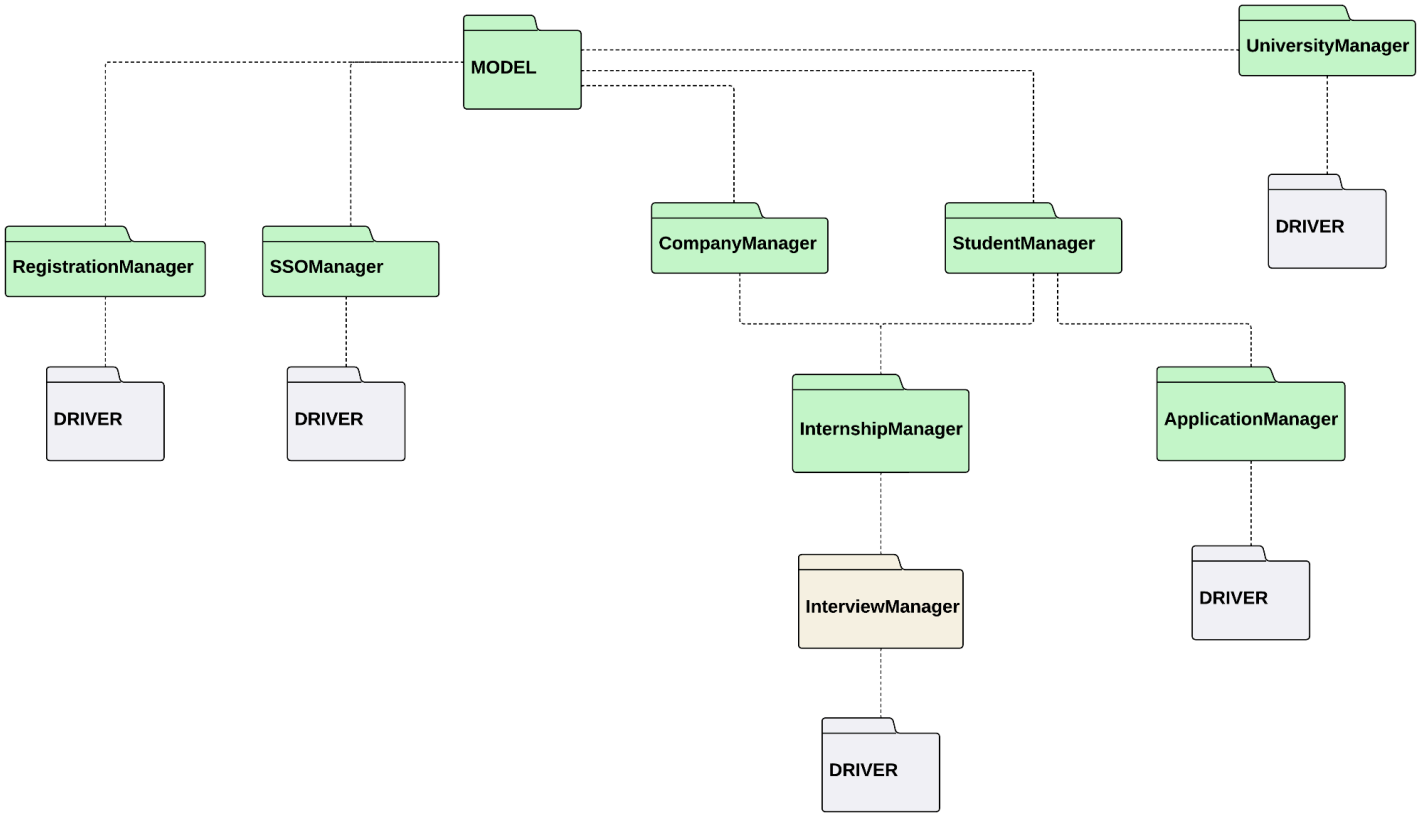
[F4] Student profile and application management

The following image shows the new components developed and tested for the student profile and application management.



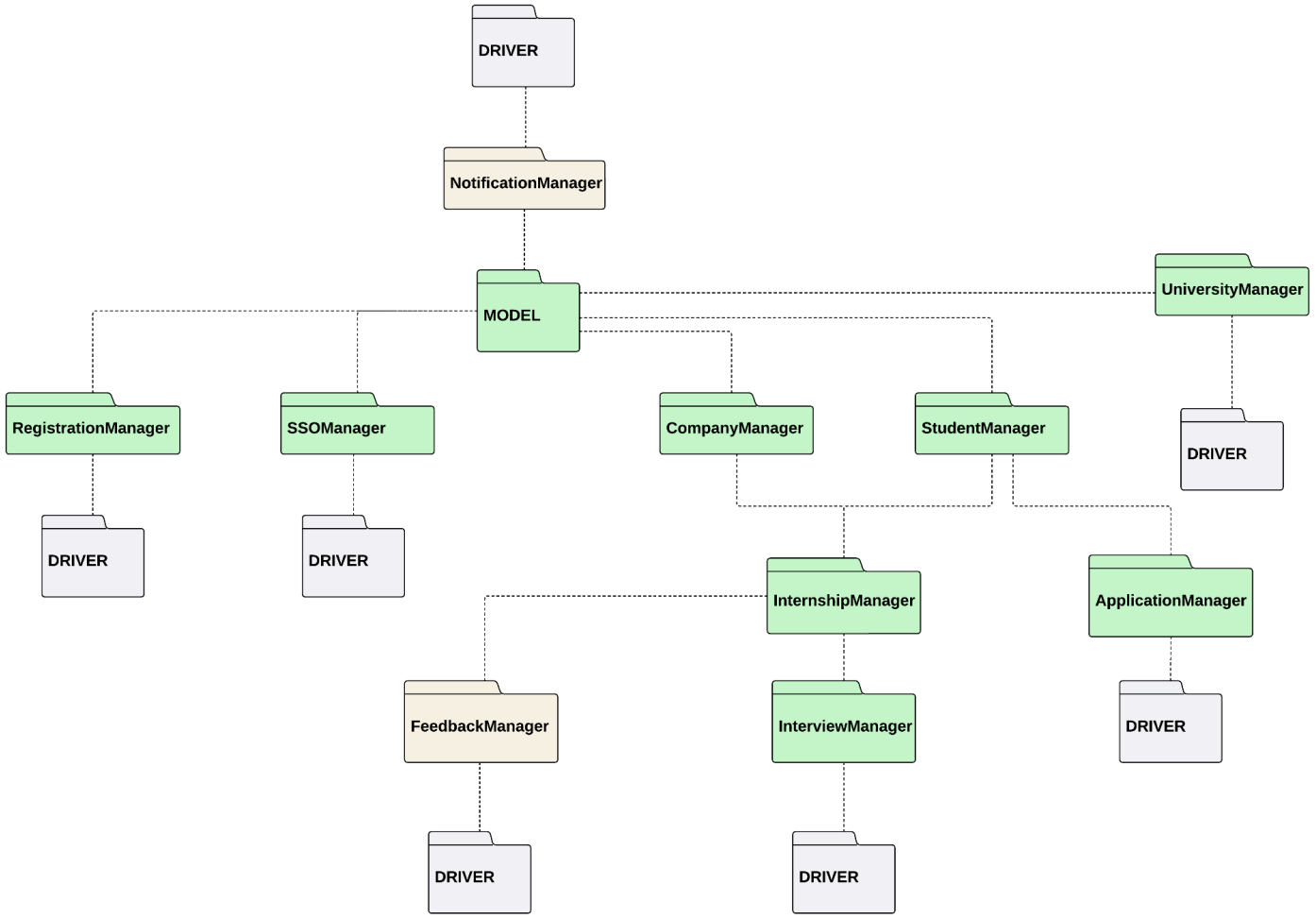
[F5] Interview management

The following image shows the new components developed and tested for the interview management.



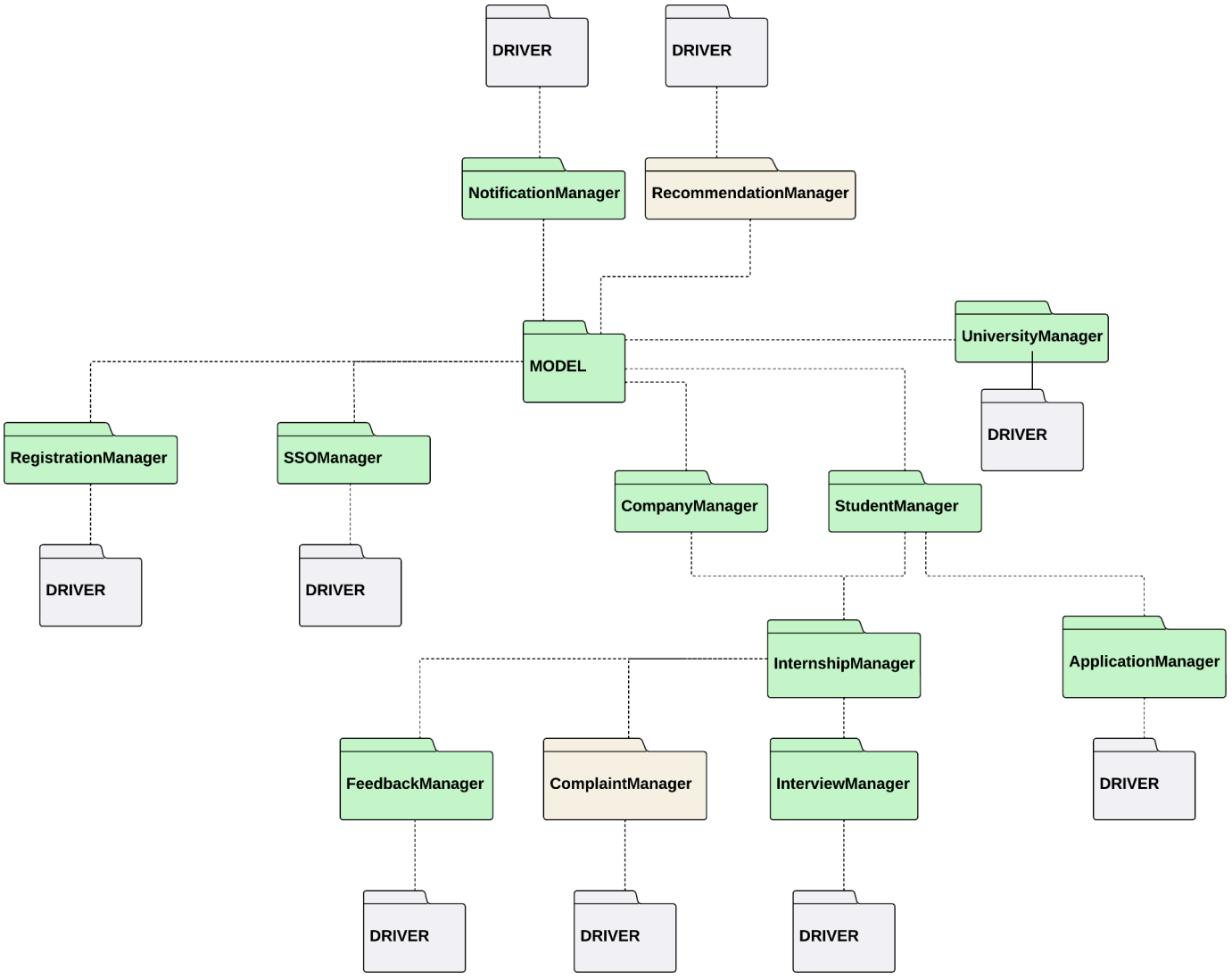
[F6] Notification mechanism and feedback management

The following image shows the new components developed and tested for the notification and feedback management.



[F7] Recommendation and Complaint management

The following image shows the new components developed and tested for the recommendation and complaint management.



5.4 System Testing

The S&C system undergoes a series of testing activities to ensure that all components of the system meet the desidered functional and non-functional requirements. Each S&C module is tested individually to verify its expected performance. Since there may be some component dependency, *Driver* components may be implemented to simulate interactions to ensure the system works as expected. Once individual components have been validated, these are integrated and the integration between them is tested as well. S&C employs a combination of bottom-up and thread-based strategies for integration testing to ensure that connected modules are well integrated and to verify the cooperation between them. After the S&C system has been implemented, unit and integration tested, it must pass the system testing phase. At this point, the system is considered to be as close to the final product as possible. The system testing includes contributions from stakeholders and uses black-box techniques to verify the correctness of the system's features and that they comply with the functional and non-functional requirements defined in the RASD. In particular, the system testing phase includes the following subtests.

- **Functional Testing:** it verifies that the platform satisfies all the requirements specified in the relative Requirement Analysis and Specification Document (RASD). During this phase, it is possible to identify new features to improve user experience.
- **Performance Testing:** it identifies potential bottlenecks or inefficiencies that could degrade the platform's response time. It requires to define an expected workload and acceptable performance target
- **Usability Testing:** it ensures that the platform is intuitive and easy to use for users. GIven the importance of user friendliness, this testing evaluates the effectiveness of web interfaces
- **Load Testing:** it determines the system's ability to handle the maximum workload over extended periods. It helps identify issues like memory leaks, buffer overflows, and mismanagement of memory under high load conditions.
- **Stress Testing:** It examines the system's resilience under extreme conditions and recovery capabilities after failure. By overloading the system, this phase ensures that S&C recovers from failures.

5.5 Additional Specifications on Testing

The system must be rigorously tested whenever new functions are added to ensure that no bugs are being introduced. In addition, during the implementation, the testing phases described above must be periodically executed to verify the correctness of the system and to

identify promptly any issue that may arise. In this way, S&C is ensuring reliability and robustness of the system. Moreover, it is important, during the development of the system, to receive feedback and recommendation from stakeholders and end users representatives. This action should be done periodically, whenever a feature is implemented. Stakeholders should receive a description of the functionalities implemented and therefore, they can verify whether the system aligns with their expectations. Furthermore, alpha versions of the platform should be provided to users, allowing them to report malfunctions and share their satisfaction levels. The continuous feedback mechanism aims to validate the system during its creation. This allows developers to identify possible misalignments between the system and stakeholder expectations.

6. Effort Spent

ALMANDOZ FRANCO Rodrigo

Chapter	Effort (in hours)
1	2
2	18
3	0
4	8
5	7

BRANDI Mattia

Chapter	Effort (in hours)
1	2
2	18
3	1
4	8
5	5

7. References

- Diagrams made with: https://lucid.co/
- UserInterface made with: https://www.canva.com/