
Software Test Documentation: Test Plan

for

<A Software for Internship>

Prepared by Mert Damburaci, Ali Firat Ozdemir,

Efekan Uysal, Umutcan Celik

Middle East Technical University Northern Cyprus Campus

Computer Engineering

Supervised by İdil Candan

18.04.2025

84/100

Contents

1	Introduction.....	2
2	Test items.....	2
3	Features to be tested	3
4	Features not to be tested	3
5	Approach.....	5
5.1	Unit Testing	5
5.2	Integration Testing	5
5.3	System Testing	5
5.4	Performance Testing	5
5.5	User Testing	6
6	Item pass/fail criteria	7
7	Schedule.....	10
7.1	Milestones and Tasks	10
7.2	Gantt Chart.....	14
8	References.....	16

1 Introduction

The software product being tested is a web-based application titled Internship Management System, developed to facilitate and streamline the end to end management of university internship processes. The system is designed exclusively for browser use.

It provides role based access and tailored functionalities for various stakeholders, including students, coordinators, instructors, companies, and student affairs. Each stakeholder interacts with the system according to their responsibilities within the internship lifecycle.

The high level functionalities of the Internship Management System include:

- Role-based user access and control for different stakeholder types.
- Online internship application submission and approval workflows.
- Internship report submission by students and evaluation by instructors.
- Document upload and validation.
- Email notifications triggered by key events such as application submission, evaluation results, and approvals.
- Real time data updates and consistent data handling between system modules.
- Browser-based responsive interface with dark and light mode support.
- Performance handling under high load periods, such as peak application deadlines.

2 Test items

• **User Interface Compatibility Testing** : This test involves verifying the internship portal's user interface across multiple browsers. The aim is to ensure consistency in layout, functionality, and responsiveness, enhancing user experience for both coordinators , students , companies and instructors.

• **Data Management and Integrity Testing**:Test aims to validate the system's data management processes, including accurate storage, retrieval, and updating of internship details, student records, and coordinator operations and instructor operations also companies operations. Ensuring data integrity minimizes inconsistencies and errors.

• **Performance and Load Testing** :This involves assessing the system's responsiveness under various loads, particularly during peak periods such as application deadlines or document submission dates. The test ensures that the system maintains stability and performance under high user demand.

• **Email Notification Testing** :This involves validating the automatic email notification functionality triggered by key events, such as new announcements, application status updates, and document submission confirmations, report feedback via instructor. Tests ensure timely and accurate notifications to relevant users.

• **User Interaction Testing**:This test focuses on verifying the functionalities available to different user roles (students, coordinators, and supervisors). It includes actions such as submitting internship

applications, updating internship details, and providing feedback or evaluations, ensuring accuracy in capturing and reflecting user inputs.

- **Recommendation System Testing:** This involves evaluating the accuracy and relevance of internship recommendations provided to students based on their uploaded resumes and past internship experiences. The test ensures the recommendation algorithms effectively match student profiles with suitable internship opportunities.

3 Features to be tested

The features and functionalities below will be tested to ensure the system meets its functional and non-functional requirements:

- ***User Interface Compatibility***

- Consistency of UI across major browsers (Chrome, Firefox, Safari).
- Responsiveness across different screen sizes (desktop, tablet).
- Proper rendering of Tailwind CSS dark/light mode styling across all pages.

- ***Internship Workflow***

- Internship application submission and editing by students.
- Approval and assignment processes by coordinators.
- Report uploading by students and evaluation by instructors.



13 • ***User Role Validation***

- Accurate role-based access for each user type (student, coordinator, instructor, company).
- Authorization enforcement for restricted operations (e.g., only instructors can evaluate reports).

- ***Email Notification System***

- Triggering of notifications for announcement submission, approval, and report feedback.
- Delivery to correct user based on action.

- ***Data Integrity and Management***

- Accurate creation, update, and retrieval of user and internship records.
- Consistent data handling across frontend and backend modules.

- ***Performance and Load***

- System response time during simultaneous submissions and usage.
- Stress testing during application deadline periods.

● **Usability Testing**

- User satisfaction with report upload, internship tracking, and feedback features.
- Clarity and simplicity of user flows for first time users.

● **Data Protection & Integrity**

- Encryption of sensitive user and internship data in transit and at rest.
- Input validations prevent SQL injection and cross-site scripting (XSS) attacks via strict input sanitization.
- Consistent data management with accurate creation, updates, and retrieval of records across frontend and backend modules.

● **Partial Recommendation System (Tentative)**

- Resume-based internship suggestions for students.
- Matching relevance based on past experiences (may be tested in future milestones).

4 Features not to be tested

Email Delivery Mechanism & Third-Party Integration:

While users receive notifications and reminders via email regarding approvals, deadlines, or announcements, the actual email delivery mechanism and third party email integration will not be tested. We assume email services function as expected.

External Authentication Systems:

Even if the system will use the METU authentication system later on, those authentication mechanisms will not be tested.

External Storage Systems:

The Google SQL Cloud Database service where the database is being stored will not be tested.

Coordinator Dashboard Statistics:

Any summary statistics (e.g. number of internships completed, average evaluation scores, report pass percentage) shown to coordinators, which our supervisor Idil Candan said they never use, will not be tested.

Deployment on a Live Domain:

The system is not yet deployed under a live domain. Therefore, testing related to live hosting, domain configuration, and deployment environments is not included in the scope.

High-Traffic Load Testings:

The system is designed exclusively for the Computer Engineering Department, with an expected user number of fewer than 100 active users, even during peak periods such as the summer internship season. Given the limited scale of operations, extensive stress testing under high-traffic conditions is not required. However, basic load testing will still be conducted under normal usage scenarios to ensure system stability and functionality.



5 Approach

5.1 Unit Testing



In this section, we focus on testing the most important backend functions of our system using unit tests. The purpose of unit testing is to ensure that each function works correctly on its own, without depending on other parts of the system.

We will write test cases for critical features such as:

- functions that trigger automatic email notifications based on user actions or system events,
- internship application handling,
- trainee form approval,
- report evaluation.



We will use the JUnit testing framework, which is commonly used for testing Java-based applications developed with Spring Boot. Each function will be tested using valid and invalid inputs, and the output will be checked to see if the function behaves as expected.

We plan to check the following in our tests:

- Does the function process valid data correctly?
- Does it return the correct error message when the input is wrong or missing?
- Does the function check the user role correctly before performing sensitive actions?

Before testing, we will review all service functions and prioritize the ones that directly affect users. Basic checks will be applied to the parts already covered by automated tests, while detailed testing will be handled in the next assignment. This approach helps us detect logic errors early and increase system reliability before integration testing.

5.2 Integration Testing

Ensures that all modules and components of the internship system harmoniously when integrated. Specific attention is given to the interaction between the email notification module, database management, authentication services, and user interface components. Tests will be conducted using Postman and automated integration test scripts to simulate realistic user scenarios, such as applying for internships, approving internship, giving feedback for students via instructor, verifying the correct delivery and format of emails, and ensuring accurate and timely synchronization of data between front-end and back-end components. Additionally, test cases will validate that the system maintains data consistency and integrity during module interactions and database operations. Results will be documented and reviewed regularly to identify and resolve integration issues promptly.



5.3 System Testing

The goal of system testing is to ensure that the entire internship system meets all specified requirements, functions correctly, and performs effectively in its intended environment. The system testing approach involves collaborative efforts among team members who will be responsible for

creating comprehensive test scenarios and executing them. Scenario creation begins with identifying relevant scenarios based on defined functional requirements and system specifications, followed by detailed documentation including inputs, expected outcomes, and step by step execution processes. Emphasis will be placed on critical functionalities such as internship application workflows, internship management, email notifications, and access control for different user roles (students, coordinators, supervisors, instructors). Test scenarios undergo tidy review and validation to confirm accuracy and completeness. During the execution phase, testers will carry out tests in environments closely resembling the actual production setting. All identified defects will be promptly reported, diligently tracked, and thoroughly resolved before marking the test scenarios as completed.

5.4 Performance Testing

The goal of performance testing is to verify that the internship system meets defined speed, responsiveness, and stability criteria under expected load conditions. Particular emphasis will be placed on ensuring efficient system behavior during peak usage periods, such as deadlines for internship application, fetching data from database to getting that current available internships. Initially, benchmarks will be established by simulating typical user loads without any optimization mechanisms in place. Following this initial testing, system optimizations, including database indexing, query optimization, and caching strategies, will be implemented. Subsequent performance tests will then measure improvements by comparing critical metrics such as page load times, API responsiveness, and system resource utilization. Test results will be systematically analyzed to inform ongoing performance enhancements and ensure that the internship system consistently meets performance expectations.

24

5.5 Usability Testing

In this part, we performed Usability Testing to see if our system is easy and understandable for real users. We asked a few users to try important features of the system and give feedback using a short form.

The users were asked to test these tasks:

- Filling the trainee information form
- Understanding the form approval process
- Applying to a company for internship
- Uploading a report and viewing its evaluation

After completing these tasks, users answered simple questions like:

- Was it easy to fill out the internship form?
- Did the system clearly explain the approval steps?
- Did you feel guided when applying to a company?
- Was the report evaluation process understandable?

Our goal was to understand whether users could complete tasks comfortably and without confusion. The feedback helped us see which parts of the system were user friendly and which needed improvement.

6 Item pass/fail criteria

6.1 Unit Testing – Pass/Fail Criteria

Test Objective:

To verify that each backend function works correctly and independently by checking valid and invalid inputs.

Pass Criteria:

- All selected unit tests must pass with 100% success for core functions such as internship application processing, report evaluation, and trainee form approval. [1]
- The function should return the correct output for valid input.
- The function should return an appropriate error message for invalid input.
- Role-based restrictions must be enforced (e.g., only instructors can evaluate reports).

Fail Criteria:


- If any of the core functions return incorrect results.
- If invalid input causes unexpected system behavior or crash.
- If any restricted function can be accessed by unauthorized users.

Justification:

These functions are essential for the system's basic operation, and even one failure could prevent users from completing key tasks.

6.2 Integration Testing – Pass/Fail Criteria

Test Objective:

To check if different parts of our system work well together, like when the frontend and backend  are data, or when one module sends something to another.

Pass Criteria:

- Users should be able to complete full tasks, such as filling and submitting a trainee form, without any system error.
- All approvals and evaluations should be saved in the database and shown correctly on the user interface.
- Data must stay correct when moving between modules. Nothing should be missing or wrong.

Fail Criteria:

- If one part of the system cannot send data to another part correctly.
- If email notifications are not sent during report or form submission.
- If the interface shows wrong information (for example, a form looks approved but it is not saved like that in the system).

- If one module crashes while trying to use another module.

Justification:

Our system depends on multiple modules working together, so a failure in integration would affect the overall process.

6.3 System Testing – Pass/Fail Criteria

Test Objective:

To check if the entire system works correctly as a whole by testing common user flows from start to finish.

Pass Criteria:

- All main user scenarios (form submission, report upload, approval, evaluation) must complete without errors.
- Role-based access control must work properly; users should only see and do what their role allows.
- Navigation between pages must be smooth, and all features should behave as expected.
- No missing links, incorrect page redirects, or broken functionalities should appear.

Fail Criteria:

- If any critical user scenario fails to complete.
- If role restrictions are not enforced.
- If the system does not respond correctly across pages or shows incorrect information.
- If essential pages or buttons do not work as intended.

Justification:

These criteria ensure that the system works from the user's perspective and that all parts function together in a realistic usage environment.

6.4 Performance Testing – Pass/Fail Criteria

Test Objective:

To check how fast and responsive our system is when users perform important actions like opening the dashboard, submitting forms, and uploading reports.

Pass Criteria:

- The system should respond to all key actions within 3 seconds. [2]
- The system should support at least 50 users working at the same time without crashes or major slowdowns.



- Page loading and data-saving operations should complete smoothly during regular usage.

Fail Criteria:

- If important actions take more than 3 seconds regularly.
- If the system becomes unresponsive or crashes when more than 50 users use it at the same time.
- If pages fail to load or freeze during normal use.

Justification:

These limits are based on usability guidelines. If the system responds in under 3 seconds, the user experience is not interrupted and is considered acceptable. The 50 user concurrency target reflects the expected use of the system in a real university setting.

10

6.5 Usability Testing – Pass/Fail Criteria

Test Objective:

To evaluate how easily users can complete important tasks in our system, such as logging in, filling out forms, applying to companies, and uploading reports.

Pass Criteria:

- 20 users tested the system and filled out a feedback form.
- If 75% or more of the feedback answers are positive (e.g., "yes" to questions like "Was it easy to fill the form?"), the test is considered passed. [3]
- Users must be able to complete tasks without external help.

Fail Criteria:

- If more than 25% of the answers are negative.
- If users ask for help, are confused, or cannot complete tasks.

Justification:

Usability testing helps to understand how real users interact with the system. A 75% satisfaction rate is accepted as a standard usability threshold based on common usability guidelines.

7 Schedule

7.1 Milestones and Tasks

Week 1 (February 17- February 23):

	Tasks	Responsible	Duration	Dependencies
Task 1	Migration to Github	Umutcan	1 week	All source files are compiled in one system.
Task 2	Database redeployment	Efekan, Umutcan, Mert	2 weeks	No backend development is taking place until the database is redeployed.

Week 2 (February 24 - March 2):

	Tasks	Responsible	Duration	Dependencies
Task 1	Term planning of the project & identification of missing functions	Efekan, Mert, Umutcan, Ali	1 week	To attend the Feedback Session for CNG 491 Final Products on February 24th and receive feedback.

✓ **Milestone:** GitHub migration started, Database migration kicked off.

Week 3 (March 3 - March 9):

	Tasks	Responsible	Duration	Dependencies
Task 1	Student side Trainee Form Page Fixes	Efekan	1 week	Database must be redeployed.
Task 2	Designing a new login page	Ali	1 week	Database must be redeployed.
Task 3	Backend development of the company side	Mert, Umutcan	1 week	Database must be redeployed.
Task 4	Configuration Management Plan writing	Efekan, Umutcan, Mert, Ali	2 weeks	None.

Week 4 (March 10 - March 16):

	Tasks	Responsible	Duration	Dependencies
Task 1	Coordinator side Trainee Forms Page & Announcements Page Fixes	Efekan	1 week	General dependency on DB
Task 2	Backend development of Internship application	Umutcan	1 week	General dependency on DB
Task 3	Improving page loading speeds	Mert, Umutcan	2 weeks	General dependency on DB
Task 4	Backend development of internship recommending system	Mert	2 weeks	General dependency on DB

Week 5 (March 17 - March 23):

	Tasks	Responsible	Duration	Dependencies
Task 1	Backend development of deadline functions	Umutcan	1 week	General dependency on DB
Task 2	Development of student side of the internship reports pages.	Efekan	1 week	General dependency on DB
Task 3	Development of mail notifications for internship applications and deadlines.	Mert, Umutcan	2 weeks	General dependency on DB
Task 4	Development of the Instructor side of trainee forms page	Ali	1 week	General dependency on DB
Task 5	Backend development of instructor assigning & report grading functions	Umutcan	1 week	General dependency on DB

Week 6 (March 24 - March 30):

	Tasks	Responsible	Duration	Dependencies
Task 1	Development of the Student Affairs frontend and backend.	Umutcan & Ali	1 week	General dependency on DB
Task 2	Integration of deadlines into the system.	Efekan	1 week	General dependency on DB, Backend codes of the deadline functions.
Task 3	Development of mail notifications for internship applications and deadlines.	Mert, Umutcan	2 weeks	General dependency on DB
Task 4	Development of the Instructor side of trainee forms page	Ali	1 week	General dependency on DB

✓ **Milestone:** Frontend and Backend Core Functionalities Ready

Week 7 (March 31 – April 6):

Bayram Holiday

	Tasks	Responsible	Duration	Dependencies
Task 1	Development of the export to excel function	Umutcan	1 week	General dependency on DB
Task 2	Company side trainee forms page & internship applications page	Ali	2 weeks	General dependency on DB, Backend codes of the company side functions.

Week 8 (April 7– April 13):

	Tasks	Responsible	Duration	Dependencies
Task 1	Integration of report grading to the frontend	Efekan	1 week	General dependency on DB, Backend codes of the report grading functions.
Task 2	Student affairs page fixes	Ali	1 week	General dependency on DB, Backend codes of the student affairs side functions.
Task 3	Instructor assigning page	Ali, Efekan	1 week	General dependency on DB, Backend codes of the instructor assigning functions.
Task 4	Test Plan writing	Efekan, Umutcan, Mert, Ali	2 weeks	None.

Week 9 (April 14– April 20):

	Tasks	Responsible	Duration	Dependencies
Task 1	Development of student affairs mail notification	Mert	1 week	General dependency on DB, Backend codes of the student affairs side functions.
Task 2	Development of company internship postings create page	Efekan	1 week	General dependency on DB, Backend codes of the company side functions.
Task 3	Resume uploading and sending companies through the system	Ali, Mert	2 weeks	General dependency on DB

Week 10 (April 21– April 27):

	Tasks	Responsible	Duration	Dependencies
Task 1	Frontend development of internship recommending system	Efekan	1 week	General dependency on DB, Backend codes of internship recommending.
Task 2	Frontend fixes	Ali	1 week	General dependency on DB
Task 3	Backend fixes	Umutcan, Mert	1 week	General dependency on DB

✓ **Milestone:** All Features Implemented.

Week 11(April 28– May 4):

	Tasks	Responsible	Duration	Dependencies
Task 1	Backend Unit Testings	Umutcan, Mert	3 weeks	General dependency on DB, The test plan, Implementation of the last functions
Task 2	Database Testing	Umutcan	2 weeks	General dependency on DB, The test plan
Task 3	Frontend Unit Testings	Efekan, Ali	3 weeks	General dependency on DB, The test plan
Task 4	Integration Testing	Efekan, Mert	2 weeks	General dependency on DB The test plan

Week 12 (May 5– May 11):

	Tasks	Responsible	Duration	Dependencies
Task 1	System Testing	Efekan, Umutcan, Mert, Ali	2 weeks	General dependency on DB, The test plan
Task 2	Performance Testing	Umutcan, Mert	2 weeks	General dependency on DB, The test plan
Task 3	Usability Testing	Efekan	2 weeks	General dependency on DB, The test plan

Week 13 (May 12– May 18):

	Tasks	Responsible	Duration	Dependencies
Task 1	Test Results Report Writing	Efekan, Umutcan, Mert, Ali	1 week	All test needs to be concluded in order to write the report.
Task 2	Test Results Report Feedback	Efekan, Umutcan, Mert, Ali	1 week	Test Results Report needs to be written

✓ **Milestone:** Testing Phase Completed.

Week 14 (May 19– May 25):

	Tasks	Responsible	Duration	Dependencies
Task 1	System Improvements	Efekan, Umutcan, Mert, Ali	2 weeks	Test Results Report Feedback needs to be taken, General dependency on DB
Task 2	Comprehensive backend fixes	Umutcan, Mert	2 weeks	Test Results Report Feedback needs to be taken, General dependency on DB
Task 3	Comprehensive frontend fixes	Efekan, Ali	2 weeks	Test Results Report Feedback needs to be taken, General dependency on DB
Task 4	Final Report writing	Ali, Umutcan, Efekan , Mert	2 weeks	Test Results Report Feedback needs to be taken, General dependency on DB

Week 15 (May 26– June 1):

	Tasks	Responsible	Duration	Dependencies
Task 1	Project finalization	Efekan, Umutcan, Mert, Ali	1 week	All fixes and improvements need to finish.

✓ **Milestone:** Project Finalized & Project Final Report Completed.

Week 16-18 (June 2– June 22):

	Tasks	Responsible	Duration	Dependencies
Task 1	Final Presentation Preparation	Efekan, Umutcan, Mert, Ali	3 weeks	Project needs to be finalized.
Task 2	Final Demo Video Preparation	Umutcan, Efekan	3 weeks	Project needs to be finalized.

10

✓ **Milestone:** Final Presentation Preparations

7.2 Gantt Chart



You can examine the Gantt chart from Figure 1 . Since it is consisting 18 weeks schedule and it is large, if you want to examine it in detail, you can see it divided into two in Figures 2 and 3 . The meanings of the tasks written in the Gantt chart are explained in 7.1.

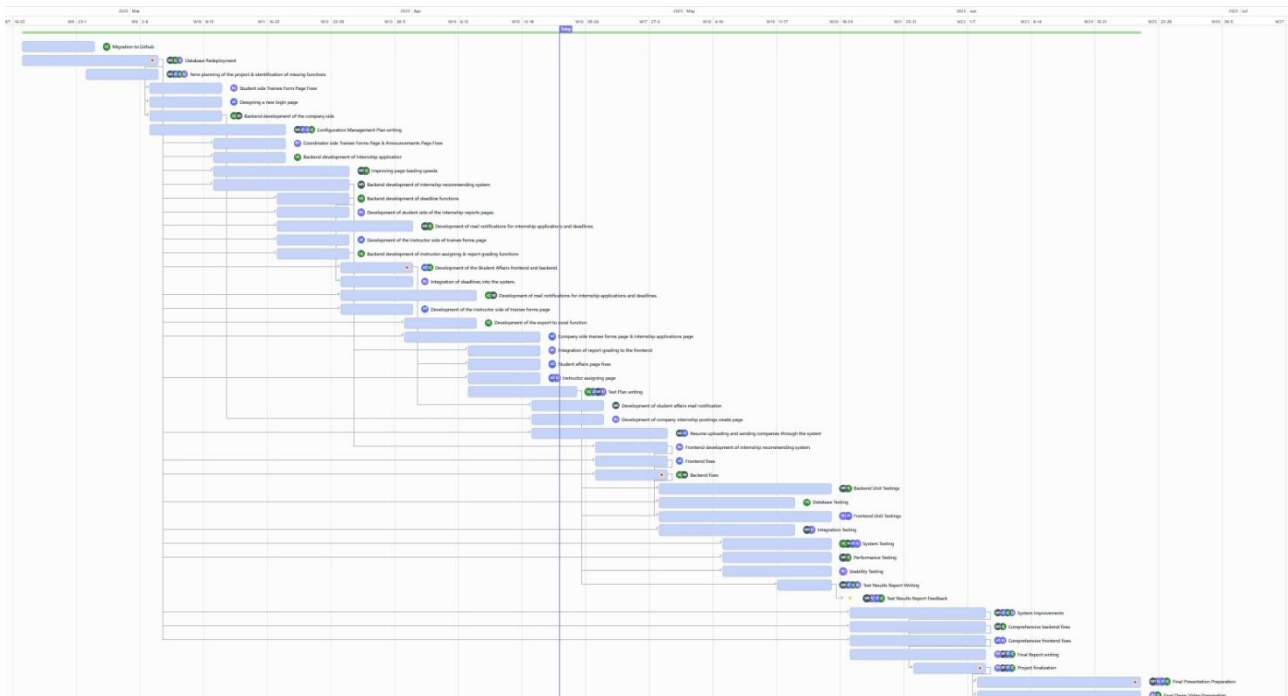
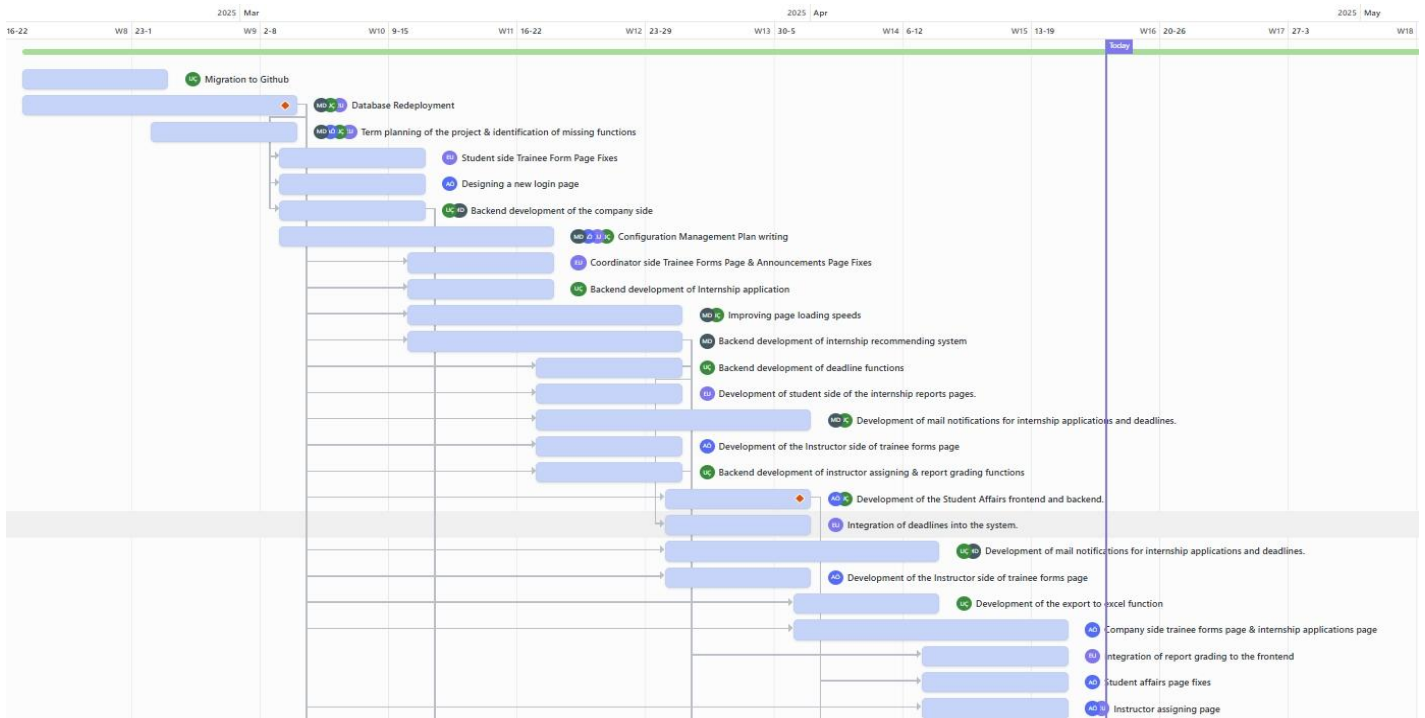
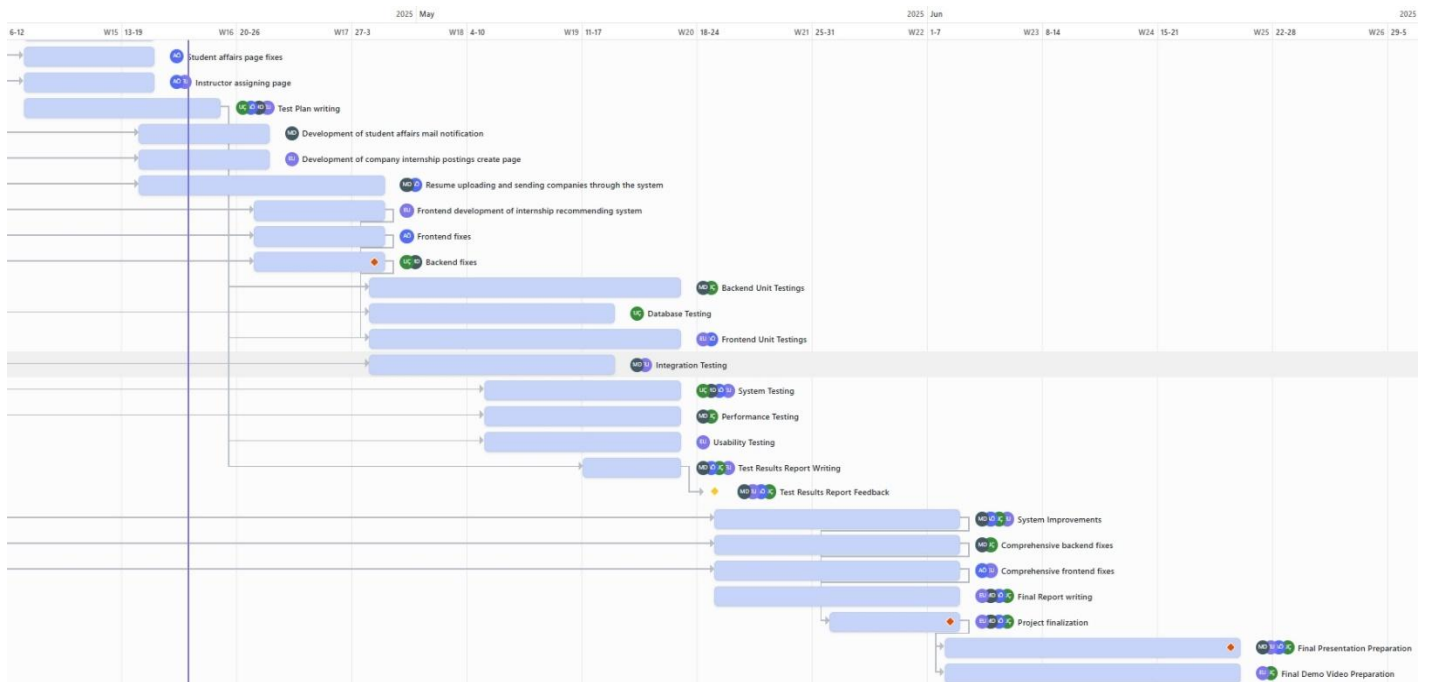


Figure 1: Gantt Chart



Gantt Chart (Week 1 - Week 8)

Figure 2: Gantt Chart



Gantt Chart (Week 9 - Week 18)

Figure 3: Gantt Chart

8 References

- 1- I Fowler, M. (n.d.). bliki: Test Coverage. martinfowler.com.
<https://martinfowler.com/bliki/TestCoverage.html>
- 2- Nielsen, J. (2020, May 16). Website response times. Nielsen Norman Group.
<https://www.nngroup.com/articles/website-response-times/>
- 3- Nielsen, J. (2024, January 31). How many test users in a usability study? Nielsen Norman Group.
<https://www.nngroup.com/articles/how-many-test-users/>