Design Report: BMS Incoming Exchange Student Administration

Braam, Harry (s2562952)
Huh, Jeongyeon (s2425181)
Kaewborisut, Damon (s2532409)
Mutruc, Vlad (s2277816)
Van het Reve, Stijn (s2395894)
Qureshi, Faizan Mazhar (s2314118)

Supervised by Ferreira Pires, Luís

November 10, 2023

Contents

1 Introduction

The Faculty of Behavioural, Management, and Social Sciences (BMS) is a dynamic group of self-described tech-savvy social scientists committed to improving society. They are passionate about both the human aspect and the potential of technology. Through innovative and responsible integration of people and technology, they strive to enhance both aspects. The main approach encompasses academic education, research, and addressing societal challenges. The faculty of BMS offers Bachelor's and Master's degrees, Professional Learning and Development programs, and cross-disciplinary research programs.

Every semester, the faculty welcomes a diverse cohort of students from all around the world who are embarking on their exchange period at the University of Twente. This international influx of students brings with it a multitude of administrative challenges requiring organization and coordination.

To date, the primary tool for managing the information of incoming exchange students has been an Excel file. While it has served its purpose in the past, time has taken its toll on this resource. Buttons that once navigated the system now don't work, links have become unreliable, certain functions have become obsolete, and the software cannot generate critical reports and the outputs required. In essence, the current system is not productive and needs renewal. The issues must be addressed in time, ideally before the beginning of the spring semester in 2024.

This report outlines the proposed and implemented solution of a group of Technical Computer Science Bachelor students from the University of Twente regarding an upgrade to the system, which will not only streamline the organization of student information but also enhance the efficiency and effectiveness of the BMS student exchange program management.

This report is further structured as follows: section 2 presents how the administration of incoming exchange students is currently done. It highlights the issues and challenges faced by the current system. Section 3 describes how the requirements were formulated, prioritized, and categorized into functional and non-functional requirements. Next, section 4 offers an overview of possible solutions identified and an explanation of the chosen solution. Section 5 discusses the technologies used and the reasons for the selection. Section 6 explains how the system was implemented in terms of database and UI design to provide insights into the rationale behind the design choices. Section 7 analyzes the risk which is the crucial aspect of project management. It is dedicated to a comprehensive exploration of the various risks that our team has identified, drawing from real-life experiences encountered during the setup of the project. Section 8 describes the testing strategies employed, such as unit testing, integration testing, and user testing, along with the results obtained. Section 9 gives future work with details on the potential improvements that were not met due to any limitations encountered and explores additional features that could enhance the system. Section 10 delves into the fulfillment of requirements and reflects any challenges faced. It provides an overall evaluation of the product created. Section 11 summarizes the key findings and conclusion. Lastly, section 12 evaluates the overall progress of the design project, including planning, responsibility divisions, and teamwork.

2 Current Situation

2.1 Client and Stakeholders

The client and main stakeholders of this project are the employees of the BMS International Office, which is the team using the system to manage the data on the incoming exchange students.

Other stakeholders are the teachers and study coordinators of the BMS faculty, as they depend on the system to get an overview of the current exchange students enrolled in the modules and courses offered by the faculty. The exchange students themselves are evidently dependent on the system in order for them to be successfully enrolled. However, it is important to note that the exchange students do not directly interact with the system. Instead, they entrust the administrative responsibilities to the faculty and its coordinators to manage the enrollment process on their behalf.

2.2 Current System

In the current system, the faculty of BMS relies on a manual process for all the data transfer and management. This process involves the utilization of a web application called Mobility Online to access and gather information from exchange students. This application contains all the data students have submitted during the application process. However, transferring the necessary data from Mobility Online to an Excel file is entirely manual and carried out by employees at the BMS International Office. The Excel file comprises various worksheets, including an overview of students, review process records, course details for different student blocks, and many more. This process creates two lists: one listing students with their respective courses/modules and another listing student information for a specific module/course.

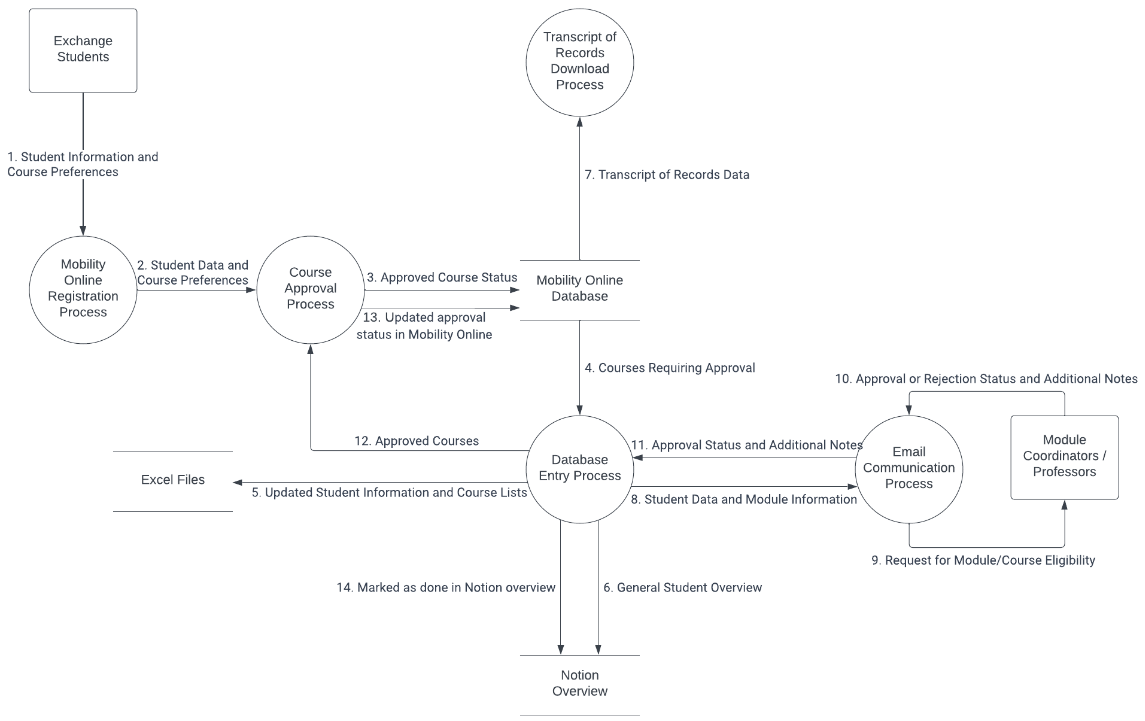

Exchange Students
1. Student Information and Course Preferences
Mobility Online Registration Process
2. Student Data and Course Preferences
Course Approval Process
3. Approved Course Status
13. Updated approval status in Mobility Online
Mobility Online Database
Transcript of Records Download Process
7. Transcript of Records Data
4. Courses Requiring Approval
10. Approval or Rejection Status and Additional Notes
12. Approved Courses
11. Approval Status and Additional Notes
Email Communication Process
Module Coordinators / Professors
Excel Files
5. Updated Student Information and Course Lists
Database Entry Process
8. Student Data and Module Information
9. Request for Module/Course Eligibility
14. Marked as done in Notion overview
6. General Student Overview
Notion Overview

Figure 1: Data Flow Diagram

*Notion - A third-party tool that BMS employees decided to use to keep track of the students they manage

The following is a summary of the current workflow as depicted in the data flow diagram above. Each connection is numbered to signify the order of data flow.

1. Exchange students provide their information and course preferences through Mobility Online during the application process.
2. Mobility Online processes student data and initiates the course approval process based on the provided information.
3. The Course Approval Process updates the approval status of students back to Mobility Online.
4. Mobility Online provides data on courses that require approval or modification. They are then manually introduced into the Excel database.
5. The Database Entry Process updates the personal and main Excel files with student data and course information.
6. The Excel files update the Notion overview with student information.
7. Mobility Online provides the option to download the Transcript of Records.
8. The Database Entry Process involves sending emails to module coordinators for course approvals.
9. Email Communication Process sends emails to module coordinators requesting course approvals.
10. Module coordinators respond to emails sent by the Email Communication Process.
11. The Email Communication Process receives responses and updates the personal and main Excel files.
12. The Database Entry Process adds approved courses to the course approval process.
13. The Course Approval Process updates Mobility Online with the completed course approval.
14. Database Entry Process marks students as done in the Notion overview.

2.3 Problems

The existing Excel-based solution for managing incoming exchange students discussed above presents several critical issues. Firstly, its reliance on manual data entry makes the process extremely labor-intensive and prone to human errors, leading to many inefficiencies and inaccuracies. It is also a very time-consuming process, making the task a lot slower than it has to be. Secondly, it lacks alphabetical sorting and filtering functionalities, further hampering the efficient organization and retrieval of student information and creating an administrative bottleneck. Furthermore, the output sheets only categorize students into semesters as 'A' and 'B, It fails to accommodate the distinction between quartiles, such as '1A,' '1B,' '2A,' and '2B'.

This misalignment further complicates the accuracy of the data representation. Additionally, the assignment of student numbers, often issued and updated later, lacks a structured system and introduces even more inconsistencies in data with no sequential number assignment. In addition, it does not offer any properly structured way to enroll the students into the courses/modules they will be following, and this has to be done manually by the staff. Lastly, the review process for student applications within the Excel file lacks efficiency and organization, making it challenging to monitor progress and update the status of their application on the mobility portal. While these issues represent most of the system's problems, the extensive list of challenges and the time-intensive nature of the current approach requires a new and updated administration system.

3 Requirements

3.1 Requirements Definition

The requirements were formulated after discussing them with the clients. In a meeting, current issues and the expectations of the project were communicated and made clear. With the information gathered, user stories were defined first. Then, going over each of the user stories, requirements were written such that all the user stories are covered.

3.2 Requirements Prioritization

The requirements are prioritized based on the MoSCoW method. This method prioritizes the requirements by considering how important and crucial they are or if there is any drawback in case a requirement is not met by the end of the project. Each of the requirements was classified into either Must, Should, Could, or Won't. Must requirements are those related to security risks and the most basic functionalities as required by the clients. Should requirements are preferred to be implemented, but the project is still successful without them. Could requirements are nice to have, but acceptable if not implemented due to lack of time. Lastly, Won't requirements are those that will not be implemented but can be done in the future.

3.3 System Requirements

The requirements were categorized into functional and non-functional requirements. The non-functional requirements represent specific qualities or characteristics that a system must possess.

Functional Requirements

3.3.1 Must Have

- R1: The user should be able to export Data in Excel formats such as .xls and .xlsx
- R2: The user should be able to add Student data manually
- R3: The user should be able to use the web interface to add, view, and edit exchange students
- R4: The user should be able to add courses/modules manually
- R5: The user should be able to assign students to courses manually
- R6: Courses should have distinct categorizations by quartiles '1A', '1B', '2A', '2B

3.3.2 Should Have

- R7: The user should be logged in to view or edit data.
- R8: The user should have the option to upload data in Excel format and link students to courses/modules
- R9: The user should be able to assign student numbers to students using uploaded data after the student has been created

3.3.3 Could Have

- R10: The user should be able to export and import student admission progress using the given data format
- R11: The admin user should be able to create multiple users

3.3.4 Won't Have

- R12: The system won't automatically enter data through an API connected to Mobility Online or Osiris.

Non-functional Requirements

- Performance
  - The system should process Excel files containing personal and course data efficiently, with a response time of less than 2 seconds for files of up to 300 records.
  - Reports and overviews should be generated quickly, with a maximum generation time of 2 seconds, even for large datasets.
- Scalability
  - The system should be designed to scale horizontally to accommodate an increasing number of exchange students, more than 100, and data records without compromising performance.
- Usability
  - The interface should be intuitive and user-friendly, requiring minimal training for (existing) users to perform their tasks productively.
- Security
  - Personal data should be securely stored and processed, following data protection regulations (e.g., GDPR).
  - Access to the data and functionalities should be restricted to authorized users with proper authentication and authorization mechanisms in place.
- Error Handling / Data Integrity
  - The system should display meaningful error messages and handle exceptions to prevent data corruption and minimize downtime.
  - Error logs should be maintained for troubleshooting and debugging purposes.
- Backup and Recovery
  - Regular automated data backups should be performed to ensure data recovery in case of system failures or data loss incidents.
- Maintainability
  - The system should be designed with modular and well-documented functionalities to facilitate maintenance and future updates.
- Testing
  - The system should be thoroughly tested before release to ensure the system's reliability and accuracy with unit, integration, and user acceptance testing methods.

4 Design Choices

In this section, we will talk about the possible solution identified for the problem, and an explanation of the chosen design choices that have been made for the project will be discussed.

4.1 Overview of Possible Solutions

The following Solution were proposed to the BMS Faculty :

4.1.1 Updated Excel Sheet

The familiarity of an updated Excel sheet is a significant advantage, easing the transition for staff as they are already well-acquainted with Excel. However, concerns arise regarding its potential for breaking or facing compatibility issues with future software updates. This can disrupt workflow and would be harder to maintain. Scalability is also an important issue that can arise as data volume and complexity increase, and Excel lacks robust security features, making it susceptible to data exposure. Additionally, its limitations in handling advanced or customized processes could hinder adaptation to evolving faculty needs

4.1.2 Local Application

This would allow for the implementation of more complex functions and faster processing times. However, with a local application, we would not have access to modern UI libraries, encounter compatibility issues, and encounter problems with collaboration. Next to this, we personally have very little experience working with such an application.

4.1.3 Cloud-Hosted Web App

Hosting a web app on a cloud provider or UT servers could address previous solution drawbacks. However, it introduces the potential challenge of storing student personal data on a third-party server, likely incurring additional costs. Knowledge gaps about UT's data storage policies and reliance on UT's ICT support department, LISA, for solutions, are considerations for this approach

4.1.4 Local Web App

A web app running on a local computer offers flexibility beyond Excel's built-in features, with the potential for a user-friendly interface accessible through a web browser. However, it would still have some drawbacks for the staff to import and export data from their portals but it would provide an easier way to handle and process the data.

4.2 Chosen Solutions

After carefully considering the current issues and the identified solutions, we have decided to move forward with developing a web application that can be run locally and, at a later stage, be deployed to a server. This choice is based on several factors. Firstly, a web app provides a modern and flexible platform that can overcome the limitations of an ageing Excel-based system. It offers a user-friendly interface accessible via web browsers, improving accessibility and ease of use for faculty members. Secondly, a web app can address scalability concerns and enhance data security compared to Excel. Thirdly, if it were to deploy to a cloud provider or UT servers, we can ensure a more robust backup and security infrastructure, mitigating potential risks associated with manual backups and local storage. The advantages of a web application in terms of functionality, scalability, and security make it a compelling choice for streamlining the management of incoming exchange students.

4.3 Architectural Design

In the figure below, you will see a general overview of the architectural design, including the chosen technologies to support the client's requirements

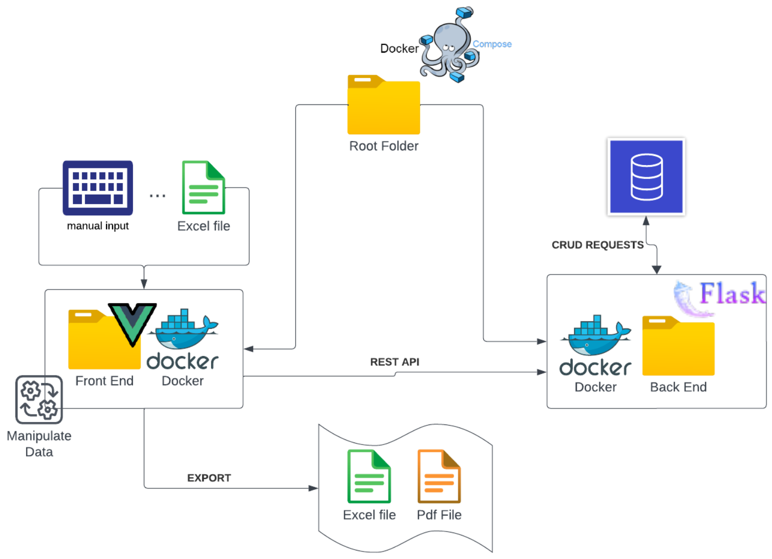


Figure 2: Architectural design with Technologies

4.4 Overview of Data Flow

The figure below will provide an overview of how data flow works in the system.

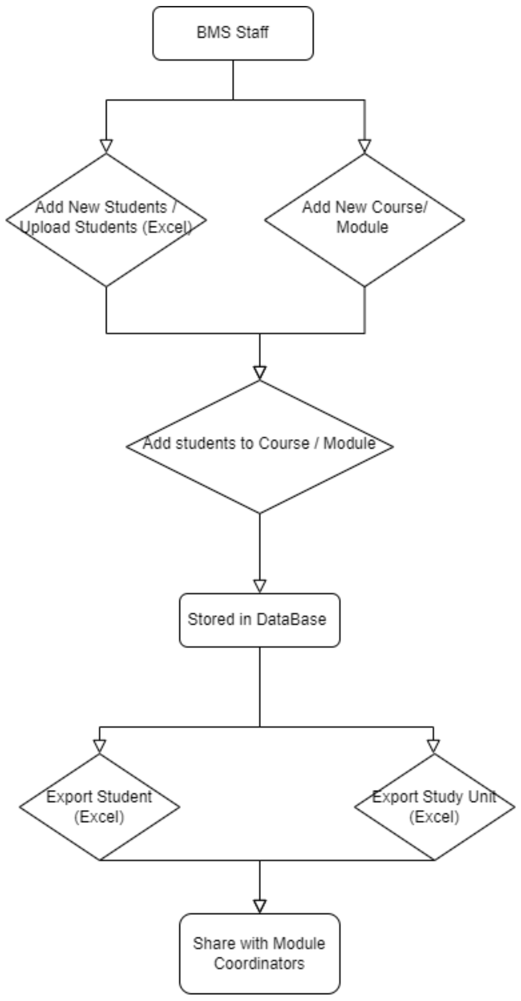

BMS Staff
Add New Students / Upload Students (Excel)
Add New Course/ Module
Add students to Course / Module
Stored in DataBase
Export Student (Excel)
Export Study Unit (Excel)
Share with Module Coordinators

Figure 3: Data Flow

5 Technologies

The following software tools have been selected for use in the project.

5.1 Python

Python is a versatile and powerful programming language that we selected for the back-end development of the project. It's known for its simplicity and readability, and it offers a variety of libraries like SQLAlchemy and Pandas, which proved helpful for the project, making it our preferred choice for rapid development. The fact that our team was already experienced with Python further solidified our decision.

5.2 Vue.js

Vue.js is a progressive JavaScript framework utilized for building user interfaces. It's approachable design and seamless integration. It was used to develop all the web pages of the project. It was also chosen because some of the team members had experience working with it.

5.3 Flask

Flask is a lightweight and flexible web framework for Python, selected for its simplicity and extensibility. It offers essential tools and features that enable developers to efficiently create robust and scalable back-end systems. Its user-friendly features and ease of use were key factors in our decision to choose Flask.

5.4 MySQL

MySQL was chosen for several reasons. It is a relational database that closely aligns with the key requirements of our project. MySQL excels in critical areas like data management, security, scalability, and data storage for web applications. Additionally, MySQL has a strong track record and a positive reputation in the industry, making it a reliable and suitable choice for our project

5.5 Docker / Docker-Compose

Docker-Compose will simplify application deployment by encapsulating components in isolated containers, ensuring consistent operation across environments. It will streamline managing service configurations, allowing easy addition, modification, or scaling without affecting other parts. Docker's reliance on "Dockerfiles" will help specify project structure and dockerized images. Our backend Dockerfile, based on a standardized container with Python, will set up a Flask project for development and deployment. This approach will also ease the client's use and integration in the future

5.6 RESTful

RESTful is known for simplicity and scalability, it uses a stateless, client-server model with standard HTTP methods. Its widespread usage for APIs and ease of integration made it our project's choice.

6 Implementation Details

6.1 Implementation Guidelines

We aimed to minimize data storage in the database and create a flexible structure to meet future requirements. However, we were also limited by Mobility Online's provided data, such as the name field, where four distinct names are provided. The diverse range of international names and Mobility Online's constraints led us to include all these fields. Additionally, a debate ensued about whether to strictly follow UT policy or provide the administrators with more flexibility. Especially surrounding whether to allow Bachelor students to follow Master's courses and vice versa. We decided to follow UT policy to avoid potential confusion and errors for administrators, as course codes and names don't need to be unique between bachelor and master courses, causing unnecessary potential confusion.

In the User Interface (UI), we prioritized functionality over aesthetics, given that this tool is designed for employee use. Emphasizing practicality, some pages display significant data to enable quicker data location and manipulation. This larger amount of data on screen might cause new users to be overwhelmed potentially. Recognizing that this tool is exclusively for Utwente employees, we anticipate they will become proficient with it over time. Moreover, to minimize the time needed to navigate the tool, we have tried to minimize page reload to ensure a fast and responsive interface.

6.2 Database

The following class diagram of the database in Figure 4 will give you an overview of the database. The database has been kept small and simple, as mentioned above. There is a student table with some basic information about the exchange student, such as whether they require a visa, study year, level, etc. Next, a bachelor student can follow modules and master student courses. This connection is made in the Selected Courses or Module table. Here is also some basic information about when the students will follow the course and the status of the course by the administrator, whether it has been approved, canceled, review start, etc. The data about the courses and modules is stored in their own respective tables. Lastly, a table containing the account information with a boolean for admin. If this were to be true, a user could create new users. Otherwise, they would merely be able to use the system.

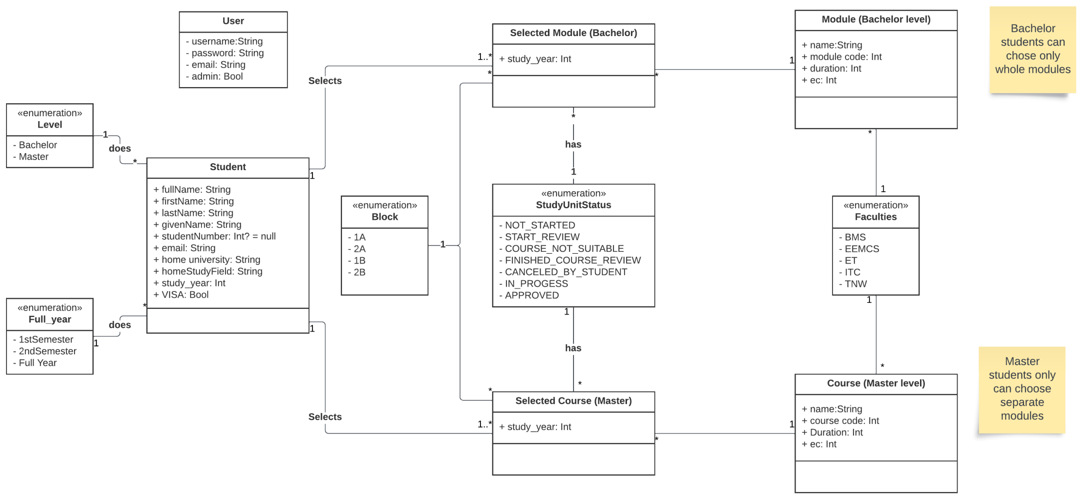
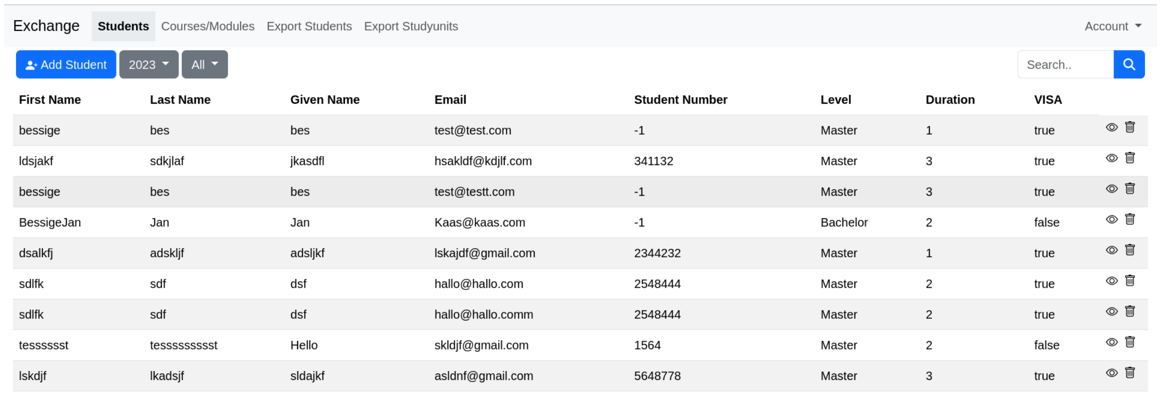


Figure 4: Class Diagram of the database design

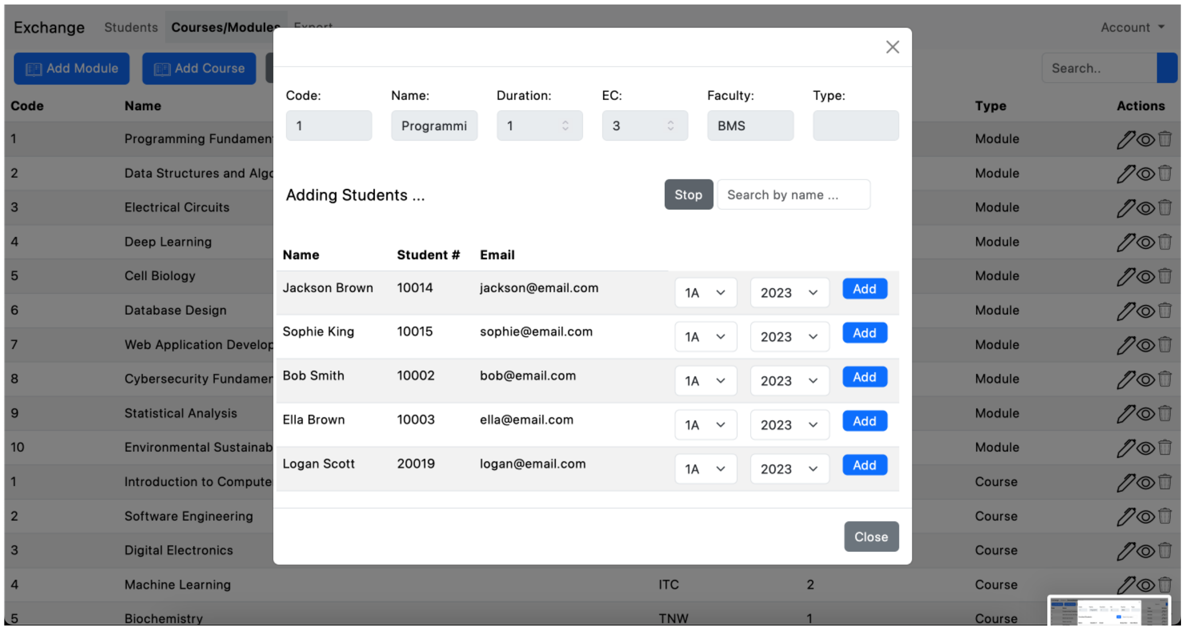
6.3 User Interface

One of the main pages of the UI is the student overview, as seen in Figure 5. This page gives an overview of all the students in the system. In the top left corner is the functionality to add a student through a model by uploading a file or by hand. Next are two dropdown menus where users can filter on an academic year and full year, first or second-semester students. This was requested by the stakeholders as this allows them not to be overwhelmed by the amount of students in the system. On the right side is search functionality so that specific students can be easily found by searching on the name or student number. Below is a table with all the students with alternating white and grey rows to help distinguish between different rows of students. It is possible to sort the students by clicking on the table's headers. This will sort on the table depending on the attribute clicked. By clicking on the icons at the end of the row, it is possible to view all data, such as attributes not shown in the table, or delete a student. We chose to show all the students on one page and not have a maximum and overflow to different pages. This allowed for easier navigation of all students without clicks. The number of students per year is around 60 so we don't expect any performance difficulties from this decision.



First Name	Last Name	Given Name	Email	Student Number	Level	Duration	VISA
bessige	bes	bes	test@test.com	-1	Master	1	true
ldsjakf	sdkjlaf	jkasdfl	hsakldf@kdjlf.com	341132	Master	3	true
bessige	bes	bes	test@testt.com	-1	Master	3	true
BessigeJan	Jan	Jan	Kaas@kaas.com	-1	Bachelor	2	false
dsalkfj	adskljf	adsljkf	lskajdf@gmail.com	2344232	Master	1	true
sdlfk	sdf	dsf	hallo@hallo.com	2548444	Master	2	true
sdlfk	sdf	dsf	hallo@hallo.comm	2548444	Master	2	true
tessssst	tesssssssst	Hello	skldjf@gmail.com	1564	Master	2	false
lskdjf	lkadsjf	sldajkf	asldnf@gmail.com	5648778	Master	3	true

Figure 5: Student Overview page



Code	Name			Type	Actions
1	Programming Fundamen			Module	
2	Data Structures and Algo			Module	
3	Electrical Circuits			Module	
4	Deep Learning			Module	
5	Cell Biology			Module	
6	Database Design			Module	
7	Web Application Develop			Module	
8	Cybersecurity Fundamen			Module	
9	Statistical Analysis			Module	
10	Environmental Sustainab			Module	
1	Introduction to Compute			Course	
2	Software Engineering			Course	
3	Digital Electronics			Course	
4	Machine Learning	ITC	2	Course	
5	Biochemistry	TNW	1	Course	

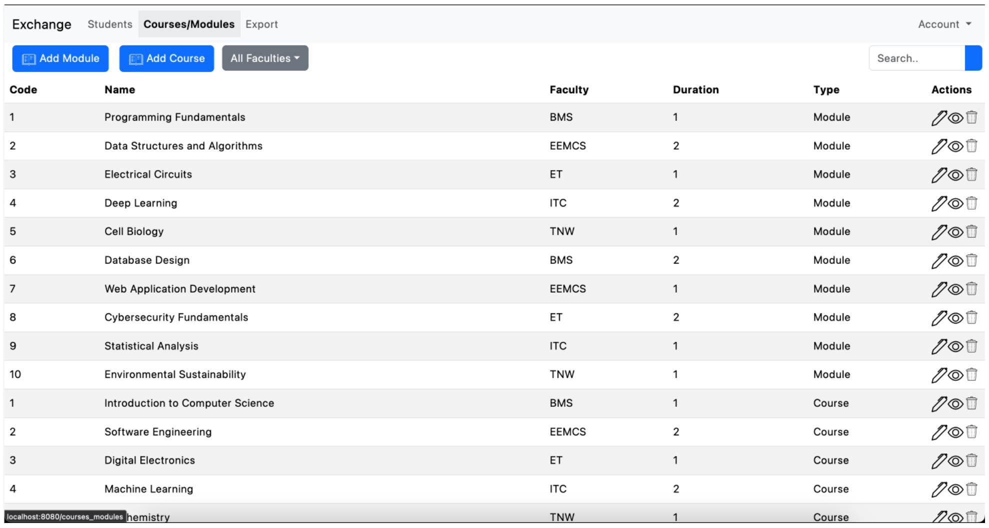
Code:	Name:	Duration:	EC:	Faculty:	Type:
1	Programmi	1	3	BMS	

Adding Students ...

Name	Student #	Email			
Jackson Brown	10014	jackson@email.com	1A	2023	Add
Sophie King	10015	sophie@email.com	1A	2023	Add
Bob Smith	10002	bob@email.com	1A	2023	Add
Ella Brown	10003	ella@email.com	1A	2023	Add
Logan Scott	20019	logan@email.com	1A	2023	Add

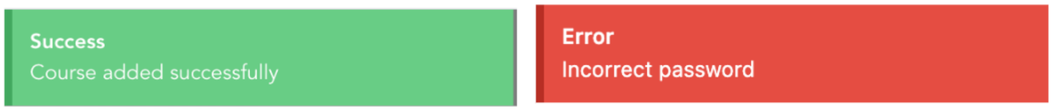
Figure 6: Add students from the course modal

The other important page in the prototype is the course overview seen in Figure 7. This page gives an overview of the courses and functionality to assign students to courses. The courses page shares many of the same design features as the student overview page. The table is the same with search and sort functionality, and courses can be added via a model triggered in the left corner. It is possible to view a course and enrolled students as seen in Figure 6. This is contained in a model triggered by the eye icon in the table. It is possible to search for students in an enrolled course. Lastly, it is possible to add a student to a course as seen in figure 6. All functionality has been achieved without page reloads through modals and the notification system.



Code	Name	Faculty	Duration	Type	Actions
1	Programming Fundamentals	BMS	1	Module	✎ 👁 🗑
2	Data Structures and Algorithms	EEMCS	2	Module	✎ 👁 🗑
3	Electrical Circuits	ET	1	Module	✎ 👁 🗑
4	Deep Learning	ITC	2	Module	✎ 👁 🗑
5	Cell Biology	TNW	1	Module	✎ 👁 🗑
6	Database Design	BMS	2	Module	✎ 👁 🗑
7	Web Application Development	EEMCS	1	Module	✎ 👁 🗑
8	Cybersecurity Fundamentals	ET	2	Module	✎ 👁 🗑
9	Statistical Analysis	ITC	1	Module	✎ 👁 🗑
10	Environmental Sustainability	TNW	1	Module	✎ 👁 🗑
1	Introduction to Computer Science	BMS	1	Course	✎ 👁 🗑
2	Software Engineering	EEMCS	2	Course	✎ 👁 🗑
3	Digital Electronics	ET	1	Course	✎ 👁 🗑
4	Machine Learning	ITC	2	Course	✎ 👁 🗑
localhost:8080/courses_modules	hemistry	TNW	1	Course	✎ 👁 🗑

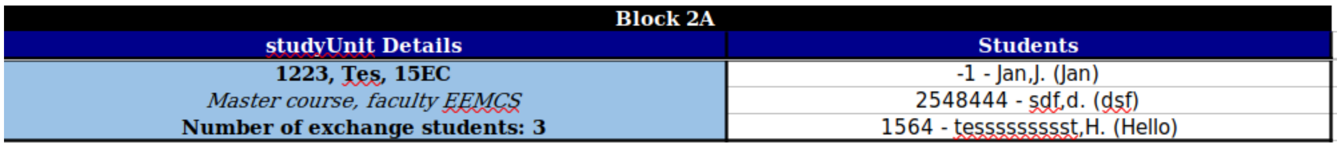
Figure 7: Courses and Modules page


Success
Course added successfully

Error
Incorrect password

Figure 8: Example Notification System

To give feedback to the user if their action was successful, a notification system is used example of a notification can be seen in figure 8. These would pop up on the top right corner of the screen.



Block 2A	
studyUnit Details	**Students**
1223, Tes, 15EC	-1 - Jan,J. (Jan)
Master course, faculty EEMCS	2548444 - sdf,d. (dsf)
Number of exchange students: 3	1564 - tesssssssst,H. (Hello)

Figure 9: Example Export File

Figure 9 shows a small section of an example export file. This file can be shared with module coordinators to show them how many and which exchange students will be following their courses. These files can be downloaded by clicking the export buttons in the header as seen in Figure 5.

7 Risks analysis

Several risks may arise during the project setup for deployment or development, which some of us have encountered. We want to discuss these risks in this section to prevent further time wastage in troubleshooting.

7.1 Docker related problems

Depending on the operating system and additional less known factors the process of building and/or running the docker containers might result in some problems.

- **Yarn not installed** - This problem usually occurs if the operating system used to run docker is *Windows* and is simply solved by manually installing the yarn package inside the docker container.

  ```
  #!/bin/bash
  docker-compose exec backend bash
  sudo apt-get install yarn
  ```

- **Package xyz not found** - This error message pops up on the back end in one of the Python files during development. There might be cases when not all the packages are included in the requirements.txt or the installation process of the packages from the requirements.txt did not start often enough. To fix this problem you need to manually install the packages in the back-end container.

  ```

  docker-compose exec backend bash
  pip install <xyz> --save # also saves the installed packages in the requirements file
  ```

- **Error "CANCELED [internal] load build context" while (re)building the containers** - This problem could occur when changes were made in the container itself, such as in the dependencies, libraries or even in the code itself. To fix this error, you need to delete the container and rebuild the associated images by deleting the images and build the containers from scratch. In the Docker Desktop Application, you can easily delete the containers and images within the application and build the container by using the ***docker compose up*** in the project directory. It is also possible to do this procedure within the Command Prompt by doing the following:

  ```
  docker stop $(docker ps -a -q) # first stop all running containers
  docker system prune # delete all non-running containers
  docker images # returns a list of images and find all image ID's to delete
  # for every image that must be deleted with their ID:
  docker rmi ID ID ID ID ...  # for multiple images, place ID with space
  docker compose up # build the container from scratch
  ```

8 System Testing

8.1 Approach

System testing is important in developing our application for organizing foreign exchange student information. This testing section evaluates the website as a whole, ensuring that all individual components, modules, and integration work together to deliver the intended functionality and meet the project's requirements. We will be using Unit testing, Integration testing and User testing to observe specific components and their interactions. These tests can assess the entire website's performance, security, usability, and functionality.

8.2 Unit testing

Unit testing was conducted to test the individual components or units of code, with the goal of delivering a dependable software product with enhanced robustness and code reliability. The scenario-focused unit testing approach was adopted in this project to evaluate the code's functionality rigorously. This testing is vital to identify potential issues and ensures reliable operation under diverse conditions that could arise from real usages. Additionally, testing of edge cases validates the resilience and ability of the system to handle unexpected or extreme situations. This assists in identifying vulnerabilities and shortcomings in the code, increasing the overall quality of the project. Systematically testing different scenarios and edge cases strengthens the project's capacity to fulfill the client's requirements. This helps the system to be well-prepared for real-world usage and ensures it provides the clients with a trustworthy experience.

The tested functionalities include file type detection using the 'is_file_excel' function, random filename generation together with some specific characteristics, serialization of data for UUID and student data, creation of structured API responses including both success and error responses, enforcement of administrative access using 'admin_required' decorator, retrieving and formatting information and categorizing data.

The code coverage measures how well the unit tests cover the project. The results were 66% file coverage in the entire development back-end directory and 91% line coverage within the covered files. This indicates that a significant portion of the code is tested, which helps to ensure the reliability and correctness of the functionality components of the project. However, the file coverage does not cover all the files. This is because the routes.py file was tested using integration testing.

8.3 Integration testing

With integration testing, we can test if the components in the front-end and back-end work well together as a coherent and functional whole. For this project, the project members are sufficient as testers to ensure the validation of the test cases. The test cases will consist of:

- database interactions (CRUD operations): By using the POST or DELETE method, an object from the database could be altered, which must be shown correctly in the database.
- API endpoints and data retrieval: the HTTP request must return expected responses such that the data is correctly sent and received between the front-end and back-end.
- Authentication and authorization flows: the application must ensure that only authorized users can access specific features and data (home page, admin page).
- Data consistency across components: Check that data remains consistent and synchronized between different components in the front-end, back-end and database.
- Data validation and error handling between components: Ensure that invalid or malicious data does not cause system failures or security vulnerabilities, test error-handling mechanisms to verify that the system gracefully handles unexpected errors without crashing or exposing sensitive information.

By manually testing all the steps and functionalities within the application, we can evaluate if the test cases are valid. The database will consist of dummy data and users to test the application.

While testing, there were no major problems with the functionalities of each test case. But some error handling is not properly handled because the notification still gives a non-user-friendly error message ("mysql.DATABASERROR"). So, we must properly catch the error and send a readable error message to the front end. One thing to note is that everyone can register and gain access to the system and view all the students. Although they are not admins, they cannot view the course/module page to assign students to courses/modules. Depending on the client's use case, it is better to let admins gain the ability to create new accounts.

8.4 User testing

The purpose of user testing is to obtain valuable feedback from the user by testing features or the whole system. Since this is the first user testing procedure, we are more interested in letting the users orient themselves around the web application, testing different functionalities along the way. We will be listening to all feedback (or lack thereof). The key is to focus on collecting valuable insights. We organized a meeting with the client to test some of the features such as Usability, Aesthetics and Visual Appeal, Learnability, or even Feedback and Error Handling. The following table shows all the functionalities that need to be tested:

feature	Functionality to be tested
Authorization	Log in Log out Register an account
Pages	View the students page View the courses/modules page View the export page View the account page
Students	Overview a student Add a student Edit a student Add students via Excel
Modules/Courses	View details about a module/course Add module/course Edit module/course Delete module/course Add students to module/course
Miscellaneous	Use search functionality for students and courses/modules Export modules/courses and students Sort modules/courses based on X Sort students based on X

Table 1: The functionalities that should be tested

While testing, the client was pleased and enthusiastic about the project prototype. This was expected because our prototype looks and works more efficiently than their previous solution with an Excel file. They mentioned that the prototype is simple but straightforward and easy to use, which is crucial for the clients and upcoming clients who could use it. The client also mentioned that filling in all student information and their courses/modules is time-consuming, but that is unavoidable because Mobility Online does not give any options to automate this process, such as an API. Lastly, they also talked about some improvements to the prototype. Although the sorting features were not obvious, they mentioned it would be nice if the sorting had an up-and-down arrow icon. Also, the error handling with notifications sometimes returns an error message, which is not user-friendly.

9 Future Work

9.1 improvements

In a short amount of time, 10 weeks, we have created a basic application for all the data of the foreign exchange students. This could export necessary exports for the study coordinators, connect students with their desired modules/courses, and provide a global view of all the foreign exchange students' information. But with additional time and effort, some functionalities could be improved to make it more time-efficient and user-friendly with the requirements they gave to us. Some examples are:

- Error handling: Although we have user-friendly error handling for log in and filling in information, it needs to improve to ensure that all users can understand and interpret the application and its errors, which includes technical and non-technical users.
- Database table: Create distinct courses and modules for when they are given (PRIMARY KEY of their table in the database are the study code, study year and study block), such that you can easily find all students following that study unit for a certain study year and study block.
- Exported excels: Indicate students with different colors in the exported excels for distinguishing Visa students, full-year students or even master students. This would be helpful for the study coordinators and even for the stakeholders (BMS).
- Filtering by year: By having all the previous years' data, the stakeholders can see through all the students if needed. With multiple years, it is essential to have filtering by year for the whole system, such that it only shows and interacts with students, modules, and courses in a certain year. This will also help for exported Excel only to show data for a year.
- Edit function for chosen courses/modules: One of the stakeholders' requirements is to have a status for each chosen course/module of each student. Depending on the usage of this feature, a PUT method for chosen courses/modules must be created to update this status.

9.2 additional features

Besides all the improvements, additional features could be implemented to enhance the accessibility of the system and/or improve the efficacy of the system for time-consuming tasks. Some discussed features are:

- Deployment on a server: Instead of running the Docker container locally, the Docker container could be deployed and run on a server, such that the system could be accessed remotely. This deployment will give implications such as the GDPR and potential data leaks.
- Automation of filling information: Depending on the future usability of mobility online or a similar website for reading all external information of the foreign exchange students, it is possible to automate filling all the information of the students and their modules/courses. One option is to go through the exported Excel files of Mobility online, which contains the student personal information and their modules/courses. If available, the system could also use an API of Mobility Online to automatically and directly access all the data instead of the user importing the Excel files into the system.
- Export/import whole database: By implementing the exporting/importing the whole database or part of the database (year) feature, the system could use it for backups and recovery, archiving, and data retention or database migration.
- Multi-user database: If the system is deployed on a server, then it is possible that multiple users could request the same API request, which could cause concurrency problems in the database regarding incorrect data writes and data reads. Therefore, the database must be Multi-user friendly by locking or other concurrency solutions.
- security measurements: Lastly, if we deploy on a server, security measurements must be taken to ensure that malicious data does not cause system failures or security vulnerabilities. Even if the system is only locally run, it is still essential to create the measurements because the users could accidentally use invalid data and cause security vulnerabilities.

10 Discussion

10.1 Requirements

We have successfully addressed most requirements, except for R6, R7, and R8, which relied on Excel sheets exported from Mobility Online. However, during the development phase, we encountered a setback as we discovered that the corresponding functionality in Mobility Online was broken. Currently, it only allows for the export of student data, which can be imported into the current version of the system. If this functionality is restored in the future, integrating it would be straightforward, as the code for deconstructing Excel sheets and updating the database is already in place. Unfortunately, despite our efforts to reach out, repairing the system in time was impossible.

10.2 UT Systems

While working on this system, we had to wade our way through many UT systems. Many of these systems had broken export functionality, lacked an API, and generally were not user-friendly. This heavily fragmented ecosystem of closed systems made executing our assignment more difficult and created a worse product. We could not solve the issue's core, mainly these many different systems where data streams between them are impossible. Unfortunately, because of our limited influence within UT, our only solution was adding another system. Therefore, we also urge the UT to reflect on its IT strategies seriously, bring back the number of deployed systems, and make sure new systems have open APIs.

10.3 End Product

As described above, most requirements have been met. The product comes with a manual which can be found in the Appendix, with a more elaborate version on GitLab. This manual will allow the stakeholders to further develop the prototype. We wished we could have created a better system, but our limitations made that impossible. However, we are still proud of what we managed to achieve within the set limitations.

11 Conclusion

The project has resulted in a web application that streamlines the organization of exchange student information, this upgrade to the old Excel-based system improves the efficiency of the BMS student exchange program management. The team implemented a functioning front-end and back-end using RESTful API to facilitate communication between the two, while utilizing Docker for effortless deployment.

This design project not only tested the team's understanding of all phases of the development process but also gave insight on how to work around existing systems and the different challenges that come along with it. The project resulted in a working system meeting most requirements and expectations of the client. The team has great optimism that the web app will be taken into consideration by BOZ BMS.

12 Evaluation

in this section, we will evaluate the whole project regarding the planning, the responsibilities of each team member, and overall team evaluations throughout the project. Then, we reflect on and discuss some flaws within the UT systems. And lastly, the conclusion of our project.

12.1 Planning

The planning for the project was very flexible because our client (BMS) did not demand us project deadlines or other assignments except for the project itself. But that does not stop us from organizing meetings with the client. So, we planned to have a meeting with the client once per 1 or 2 weeks. To have discussions regarding the requirements, current problems, feedback on project ,Q&A sessions, and lastly, testing the prototype. This ensures that we and the client are on the same page regarding the project. We also had a meeting with our supervisor per week to show and gain feedback on our project. This meeting would also gather all the team members and discuss potential questions within the project. Lastly, each week, we meet with all the team members to discuss potential to-do's for the upcoming week and put all the tasks into a SCRUM-board using Trello such that the to-do's are written and known by everyone with their progress.

As mentioned in the previous section, the client did not demand any assignments. However, there were deadlines and assignments from the actual Module itself. First, in weeks 3,5,7, and 9, we needed to showcase our progress within our project with a presentation. Second, at the end of the module, we needed a final presentation with a demo for the supervisor of our whole project. Lastly, in the final week, the poster and final report of the project must be delivered to complete the module. The latter is the most time-consuming, thus, we started working with our final report in week 5 and received feedback from our supervisor throughout the documentation of the final report.

12.2 Responsibilities

From the start of the project, all team members had different expertise and interests within the project. So, we tried to give each member their preferred responsibilities within the project. Fortunately, there were no conflicts regarding giving each team member their responsibilities. Even if some responsibilities are tied to some team members, that does not restrict us from working together on those responsibilities. By reviewing and reflecting together on the assignments, we are able to make the best out of our project. The assigned responsibilities of our team members are as follows:

- **Anna:** Presentation Slides, Interface Designer, Requirement Analysis, and Head of Testing.
- **Damon:** Database Designer & Developer, Back-end Developer, and Communication Manager with the Client.
- **Faizan:** Presentation Slides, Interface Developer, User Support and Community Manager.
- **Harry:** Front-end Developer, Interface Designer, and Head of Mailing Services and Preferences.
- **Stijn:** Presentation Slides, Interface Designer, Requirements Specification, and Documentation Manager.
- **Vlad:** front-end Designer & Developer, Back-end Designer & Developer, and Poster Design.

12.3 Team Evaluations

Overall, there were no conflicts among the team members, although everyone had their own schedule and sometimes could not attend a meeting. Therefore, we choose to have a weekly meeting with each other or at least most of us, to reflect on each other work, ask questions to each other and make another planning for the next week. Then, someone will send all the necessary and useful information to our Whatsapp group to inform everyone about the planning and to-dos. This worked really well, especially with the Whatsapp group, because we could ask questions outside the meetings.

Appendix A Manual

This section is meant to assist the system's maintainers, developers or users in setting it up.

A.1 Requirements

Before proceeding with the installation, ensure the following dependencies are met:

- Docker
- npm
- yarn (recommended)
- git

A.2 Installation process

Clone the repository

```
#!/bin/bash
git clone git@gitlab.utwente.nl:s2277816/complex-excel.git
```

Then you can navigate to the project directory, make sure you are on the **master** branch and then build the docker services so all the dependencies and environments will be installed by the docker framework.

```
#!/bin/bash
docker-compose build
docker-compose up
```

After the last command, you should end up with a working docker setup that contains both front-end and back-end services. Back-end will be exposed at port 8000 and the front end will be available at port 8080.

A.3 Development guidelines and tips

The project was developed using the two main branches and multiple feature branches that interacted with those main branches during their development.
Branches

- *master* - Branch for production commits, all code here should be bug-free and fully working
- *acceptance* - Testing branch for implemented features and solved issues. Can have code problems and bugs.
- *feature-xyz* - A dynamically named branch named after the issue it solves or functionality it implements. It is used for development to be then merged into **acceptance** and eventually **master**

The main methodology we followed was as follows:

1. Create an issue for a specific functionality introduction or bug fix
2. Create a new branch corresponding to the issue from **master** to do all the issue-related changes
3. When the changes are complete merge to **acceptance** where all the team tests it and approves or sends it back to development.
4. If approved by team members create a merge request to **master** from the feature branch and merge it to create a release. (Multiple feature branches could also be combined into one release)

More detailed information about the git guidelines used in the development of this project is described on the GitLab page of the repository.