|  |  |
| --- | --- |
| A picture of a winding road and trees  SAS  Project documentation | Abstract  This document shows a detailed documentation and project proposal in response to mini  By Brian Msane  Together with: Neliswa Maziya and Thandolwethu Nhlabatsi; 202203673, 202203763, 20202 |

Ail Academe Services Documentation

# Table of Content

1. Project Overview
   1. Introduction
   2. Problem Identification or Statement
   3. Project objective
   4. Contribution (significance of proposed solution)
   5. Stakeholders
2. Requirement Gathering and Feasibility Study
   1. Requirement gathering techniques
   2. Sample interview questions
   3. Sample questionnaire
   4. Functional and non-functional requirements
   5. Feasibility Study
      1. Technical feasibility
      2. Economic feasibility
      3. Schedule feasibility and scope
3. Solution Modeling (Use Case and DFD)
   1. Use case diagrams
   2. Context Diagram
   3. Level 1 data flow diagram
   4. Data dictionary
4. Database and Normalization
   1. Database used
   2. Schemas
   3. Progressive normalization (0NF, 1NF, 2NF, 3NF, BCNF)
   4. Query optimization
5. Testing and Validation
   1. Unit tests and CI/CD
   2. User acceptance testing (Automation)
   3. Validation
      1. Form validation
      2. Field validation
      3. Page validation
6. Development Tools
   1. Front-end tools
   2. Backend tools
   3. SCRUM framework
   4. Application programming interfaces (APIs)
7. Wireframe
   1. Tools
   2. System wireframe
8. Conclusion

# Project overview

* 1. Introduction

Often times when a student finishes writing their form five external examination they tend to aspire or even project themselves as university students. Unfortunately, between this fantasy and the reality lies a hurdle which is the application process. This process has so many problems which are: unawareness of application dates, misinformation about programs they want to pursue, lots of money spent, to name a few. This information gap can lead to uninformed decision, impacting their future careers and overall satisfaction during the course of their study. The need for a centralized system to assist student in making well-informed decision is apparent.

* 1. Problem identification

High school graduates encounter several critical issues during the university application process and those includes the following.

1. Limited understanding of available programs: Students often select programs based on advice from others, media influence, or long-held aspiration without fully understanding the program content, admission requirements, or career prospects. This can result in dissatisfaction and hinders career progression. In often times, students end up dropping out because they are required to do things they are not passionate about and their performance marks drop drastically.
2. Costly application procedures: The traditional application process requires students to physically visit multiple universities and colleges to gather information and submit applications. This methods is time-consuming, expensive, and inefficient, especially for those with limited financial resources. Also, the universities need more human labor to handle this process and that is ultimately costlier.
3. Unawareness of application timeline and costs: Many students are not informed about application opening and closing dates, acceptance periods, and costs associated with the entire process in different institutions. Many miss opportunities due to this.

These challenges not only necessitate a centralized system but also highlight a systematic problem affecting students’ successful transition into tertiary education. The lack of accessible, comprehensive information and streamlined processes necessitate a solution to support students during this critical phase of their lives.

* 1. Problem objective

The proposed system aim to alleviate the mentioned issues by building a very informative, user-friendly, and integrated system. The primary objective is to provide comprehensive program information to applicants. Detailed descriptions, prerequisites, admission criteria, curriculum details and potential career paths associated with each program will be provided. Further, we aim to simplify the application process by creating centralized, one-size-fits-all online platform which allows students to apply from the comfort of their homes to reduce physical visits and travelling costs. Lastly, to mention a few of our objectives, we aim to provide the awareness of application timelines and costs associated with applying.

* 1. Contribution
  2. Stakeholders

Project stakeholders are individuals or organizations who are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or successful project completion. The stakeholders involved in this project include:

* Tertiary Institutions: Universities, colleges and technical institutions who are responsible for managing admissions.
* Applicants: Students applying for admission to undergraduate programs.
* Government bodies: Agencies seeing education in Swatting, responsible for policy formulation and data collection.
* Sponsors and funders: Private organizations or individuals looking to sponsor students based on specific criteria.
* SAS development team: The team responsible for designing, implementing and maintaining the system.

# Requirement Analysis and gathering

* 1. Requirement gathering techniques
  2. Sample interview questions
  3. Sample questionnaire
  4. Functional and non-functional requirements
  5. Feasibility Study
     1. Technical feasibility
     2. Economic feasibility
     3. Schedule feasibility and scope

# Solution Modeling

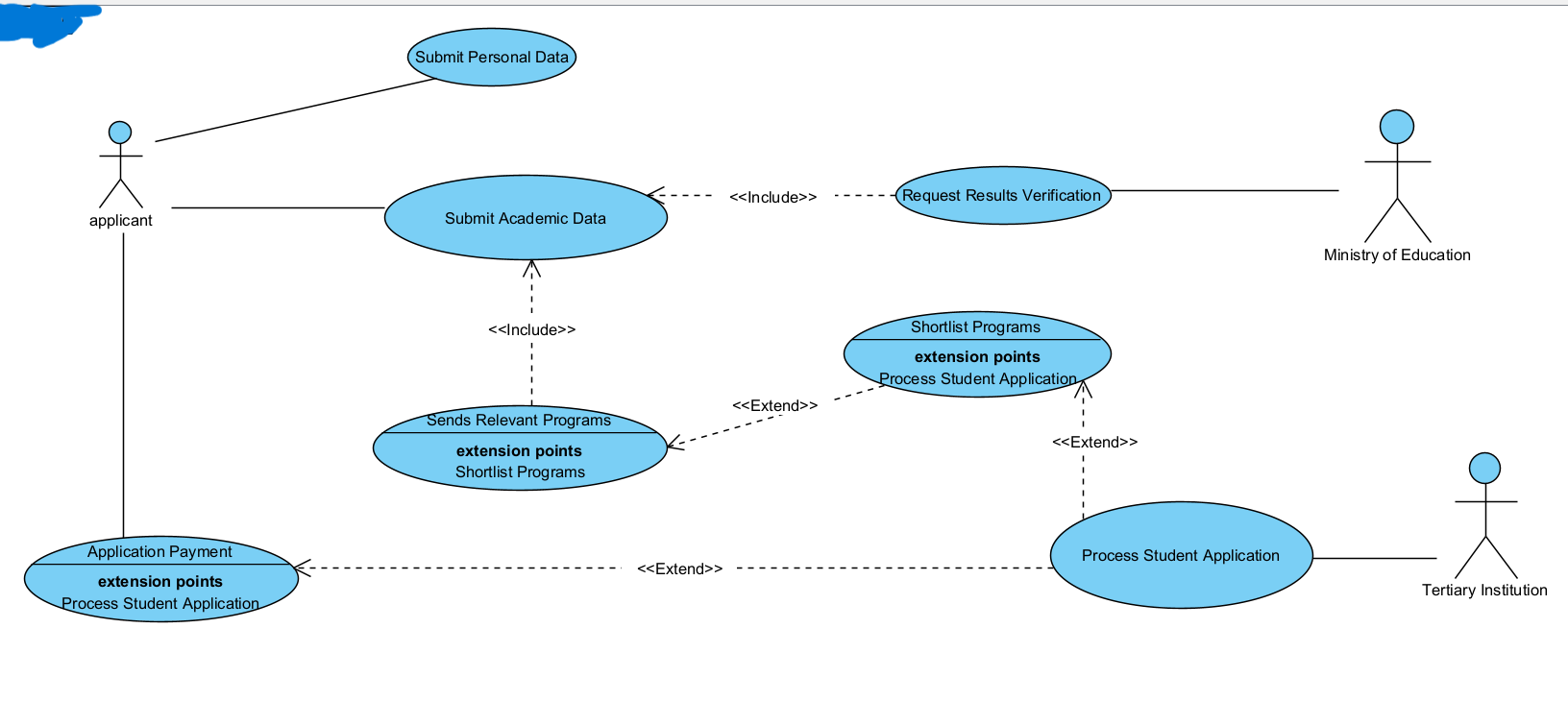
System modelling is one of the vital aspects in engaging with users because the diagrams let the users understand, contributes, and modify the system to suit their needs. Here we make use of use case diagrams and dataflow diagrams. Not only users but the developers also get a great deal of understanding and sorting or arranging modules and data inputs and outputs for the entire system.

* 1. Use case diagrams

From a high-level, the system can be depicted by a use case diagram which is a system modeling diagram which shows the system from the view of the user and is independent of technical or implementation details and jargon. It makes use of actors, use cases, and relationships. The actors are considered to be entities external to the system which play particular role to the system. Also, relationships shows how an entity relates to a use case, how entities relate to each other, or how two use cases relate to each other. The use cases can be perceived as a process within the system.

Below we have the SAS use case diagram which shows three entities which are: 1) applicant, 2) ministry of education, and 3) tertiary institution. We have six major use cases and the relationships are as thus.

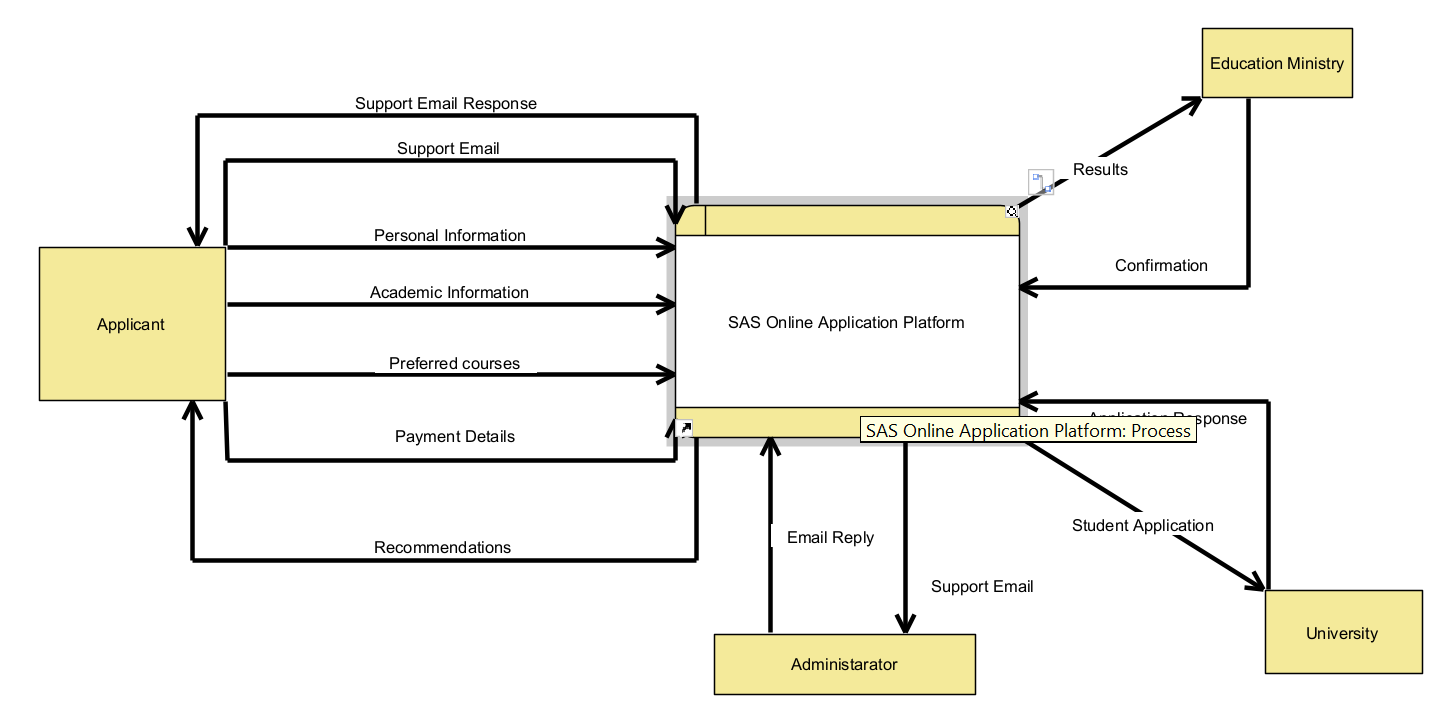
* The *applicant* communicates with **Submit Personal Data**
* The *applicant* communicates with **Submit Academic Data**
* Once the applicant has sent the academic data we need to verify them with the ministry of education we the use case **Submit Academic Data** has an extend relationship with **Request Results Verification**.
* Also, once we have the academic information of the applicant we need to give them our recommendations therefore there’s an include relationship between **Submit Academic Data** and **Send Relevant Programs**
* Once the high school graduate has given us a subset of the recommendation as his/her preferred programs we need to shortlist them to have the top three programs to use in the application process. Therefore, there is an extend relationship between Send Relevant Programs and Shortlist Programs
* There’s an extend relationship between Shortlist Programs and Process Student Application
* The graduate needs to pay for the application there he/she communicates with the Application Payment use case
* After paying we can then process the application therefore the Application Payment use case has an extend relationship with the Process Student Application use case
* Once processed, the application needs to be sent to the tertiary institution therefore the Process Student Application use case communicates with the student



* 1. Context Diagram

Another diagram which can be used to model the system is a dataflow diagram. This shows the movement of data within the systems and when drawing it we start from the special level called level 0 or the context diagram. This is a special in a way such that it depicts the entire system as just one process, labelled process 0, which can be exploded further in the other levels of dataflow diagramming. This ensures a top down approach to system modelling while making sure that even non-technical users can understand the system.

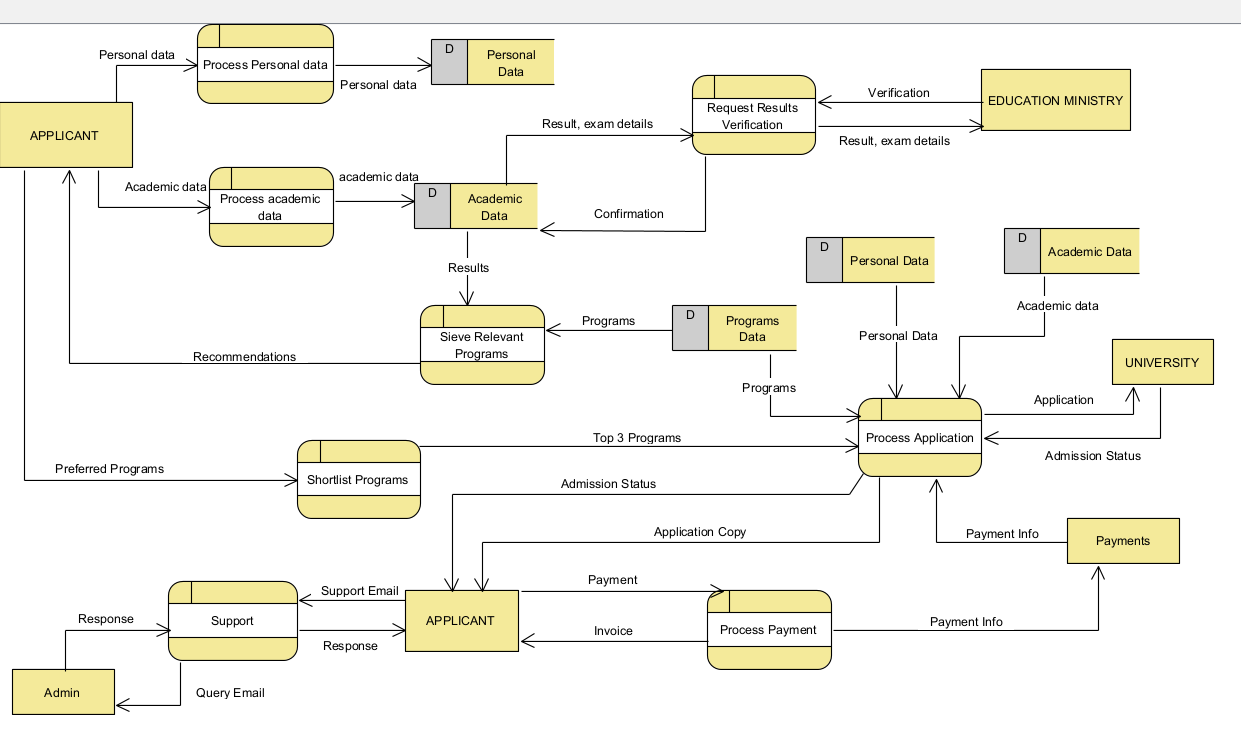
The context diagram is concerned about the major external entities and major data flows as compared to processes and data stores. Below we have the context diagram of our proposed system.



* 1. Level 1 data flow diagram

Exploding the process in the context diagram into sub-processed within the system gives birth to the level one diagram. We further broke down the system into processes which include, processing data, requesting results verification form the education ministry, sieving relevant programs, shortlisting those programs, adding customer support, processing payment, process application.

It is important to note that some, if not all, of the process can further be broken down or exploded so to create level two of the SAS data from diagram but we chose to go as further as this level. We needed to ensure that the diagram is free from technical jargons and it is a logical data flow diagram and going further within the level could result in a physical data flow diagram which show how the system will be implemented and almost often, the non-technical stakeholders do not understand that.



3.4 Data dictionary

# Database and Normalization

From definition of the databases used in this project, the schema definition, then normalization, and query optimization, this section details it all.

* 1. Database used

For this project, we initially had an idea of using

* 1. Schemas
  2. Progressive normalization (0NF, 1NF, 2NF, 3NF, BCNF)
  3. Query optimization

# Testing and Validation

This section speaks to the testing approaches we used for unit testing and user acceptance testing. Well not ignoring syntax testing and desk checking but that is done or is a subset of the programming side of things.

* 1. Unit tests and CI/CD

The code base could be so huge for us to write unit tests for every function, class, and methods instead we employed a strategy which includes Continuous Integration, Continuous Deployment, CI/CD and agility. This is a term widely adopted in DevOps, development operations which basically has two parts CI and CD. CI translates to a principle which requires users to frequently push changes they made into the codebase whether that is a new feature, bug fix, or anything else.

The CD part of it means that we need to deploy these changes into production. This calls for testing which is unit, integration, and UAT testing but mainly unit and integration testing. As mentioned we adopted this into our workflow.

The unit tests are triggered each time the developer makes changes and wants to deploy them. We have some parts of the code that we want to ensure that they are running properly before we let the users interact with the new version of the code. This is where we now write the critical unit tests. This strategy spares us the trouble of having to write unit tests for each function.

When the unit tests pass, we are then able to deploy to production because the build and release pipelines would have been function properly. If not, we are then prompted to fix the issues and continue this process until the unit tests are passing and everything integrates seamlessly.

* 1. User acceptance testing and automation

User acceptance testing ensures that the system behavior complies with the initial requirements. To perform this, we made use of two approaches, first, we gave the system to the applicants to make use of it and secondly we made an automation bot to test the behavior.

The first option revolves around making an MVP, minimal viable product, of the system and throwing in to the users to interact with and do prototyping to incrementally add features to it. We first develop the MVP, give it to the users and then proactively listen and attend to their feedback. Once we have the feedback we analyze how we add the changes to the system and code that in. We continue this loop until all the requirements have been addressed somehow.

The problem which this approach is scope creep. The users are most likely going to add new feature to the system that they initially did not point out. This increases the load on the development teams and goes against the time constraint and maybe the resources and budget constraint. Also, these iterations take time.

To address these issues we focused more into our second option which is developing and automated bot to interact with the system from the perspective of the user. The user interacts with the front-end therefore we made the bot to interact with the web pages, click buttons, navigate, make requests to APIs and doing a whole lot of processes.

The activities of this bot are derived from the requirements for instance, when the user clicks on a button request some data form the API, the response should be the intended data in the intended format and if not, we have an issue with meeting the user requirements.

We used selenium to help in the creation of this bot. This bot is fast, providing us with frequent and immediate feedback and will not result in scope creep so it is the best option for user acceptance testing.

* 1. Validation

Validation is a crucial process in software and web development that ensures the data input by used meets the required standards before being processed or stored in databases. It enhances security, improves data quality, and provides a better user experience by correcting errors early and immediately. For validation we mainly used two methods which are field and form validation and it is in our best desire to implement page validation as soon as possible.

* + 1. Field validation

Field validation focusses on individual form fields ensuring that each one contains acceptable data before the form is submitted. Common techniques include:

* Guiding the user to ensure they input the correct format through the use of placeholders in each field
* Real-time validation. Providing instant feedback as the user fills each field.
* Error messages when validation fails. They should be clear and detailed while guiding the user to input the correct data.
* Value ranges ensuring that the data is within the correct ranges. No one is expected to finish high school while less than 15 years old.
  + 1. Form validation

Form validation extends to check the entire HTML form before it is being submitted to the server. This was done on the client side using JavaScript and we also had the option of using PHP, Python, Node.js, or any other language to perform server-side validation but we chose to use python and that will be justified as we progress with the documentation. The use or client-side validation reduces server load, saves bandwidth, and enhances user experience. However, for security reasons, it should not only be relied on as users can be smart enough to bypass the JavaScript enforcement and so on. Hence, server-side scripting is necessitated. This maintain data integrity and security while ensuring that the data sent on the front-end is compatible with the data expected on the backend.

For validation could focus on:

* Ensuring that required fields are filled
* Ensuring expected data formats
* Verifying data types
* Enforce cross-field validation. This is where a combination of fields are validated, for instance username and password combination should match the one in the database, date of birth and age should be reasonably collating.
  + 1. Page validation

As mentioned, page validation is not yet enforced but it seems to be an exciting feature which involves verifying that all components of a web page meet certain standards and are functioning properly. For instance, ensuring that markup and styles comply with the web standards defined by the World Wide Web Consortium (W3C).

# Development Tools

This section highlights the tools and languages we used in developing this web application from the front-end, to the backend, APIs, and the software development methodology we adhered to during the course of this project.

* 1. Front-end tools

In developing the front-end of our website, we utilized a combination of foundational web development technologies which are: HyperText Markup Language, Cascading style sheets, and JavaScript. Each one of these plays a pivotal role in creating an interactive, intelligent, and responsive website.

HTML defines the structure and the content of our web pages. It uses a series of tags to organize elements such as text, images, videos, form, links, and so on. At this stage, the web pages is functional but lacks styling and interactivity, resembling what used to be the case in early stages of the web, particular Web 1.0.

To transform the basic HTML structure into a pleasing interface, we employ CSS. It allows us to style HTML elements by applying properties such as colors, fonts, layouts, spacing, and so on. It plays a crucial role in ensuring that the website adapts seamlessly to different screen sizes and devices.

JavaScript was used to add interactivity and dynamic behavior to the website. By manipulating the DOM – document object model, we created interactive elements such as drop-down menus, form validation, and sliders. Further, we used it to fetch data asynchronous using APIs, allowing dynamic content updates without reloading the page. Express, a JavaScript framework was used to simplify the process of handling HTTP requests, serving static files, and setting the middleware. This enabled efficient testing and development of our front-end.

* 1. Backend tools

The backend, core side, was mainly written in the Python programming language which is a fast, high-level language known for its readability and extensive libraries, which accelerate the development process and reduces complexity. If you think of something, it has already been implements and you can using the python repository to get is and use it in your application.

Mainly, we adopted object-oriented programming concepts in the development of the project.

We made use of a lot of open source libraries and frameworks provided by Python

* 1. Application programming interfaces (APIs)

The front-end is separated from the backend in as far as the client-server architecture is concerned so we need intermediaries and this is where an API comes into play. APIs allows for efficient communication between our backend and front-end and they define the rules and protocols for interaction.

We implemented a RESTful API using FastAPI, adhering to the REST (representational state transfer) principles which are statelessness, client-server separation, and uniform interface. Statelessness means that each API request contains all the information needed to process it, without relying on stored context on the server, basically, the server does not store any information about any particular request. Client-server separation means that the front-end and back-end evolve independently. The standard us of HTTP methods like GET, POST, PUT, DELETE, PATCH, and so on ensures the uniform interface.

The reason we chose Fast API, besides for the fact that it is an easy to use framework which has automatic documentation and has a SwaggerUI (OPENAPI) interface that allows you to have a user friendly want of requesting from your APIs, is that it implement server-side valuation. The uvicorn server is where the API and endpoints are running so it is our backend server. As each endpoint receives a request to serve, it checks the request body to see if the data match the defined schema and its data types and FastAPI uses type hints to ensure this. Also, when the response is being made, as response schema is also available to check whether the data being served adhere to the defined schema. This ensures data integrity and quality which making sure that the data being transmitted in or by the application will not break other parts of the system.

Alternatively, we explored using GraphQL as it allows the clients to request exactly the data they need, nothing more, nothing less. It serves as an efficient alternative to RESTful APIs in situations where we have interconnected resources. Its advantages are summarized below.

* Fetch specific data. Reduces over-fetching and under-fetching of data which is common in RESTful APIs.
* Single endpoint. Operates through a single endpoint simplifying the API architecture and reducing the need for multiple endpoints for different resources as it is the case in RESTful APIs.
* Strong typing. Uses a type system to define the schema of the data, providing clear expectation and reducing errors.
* Real-time data. Support subscription for real-time data updates in a much easier way as compared to when you try to implement in using RESTful APIs.
* Efficient network use. By minimizing the number of request and the amount of data transferred, it enhances performance, reduces server load, and bandwidth usage.

Despite these incredible advantages, we chose to stick with FastAPI reason being that we have prior experience with it and we did not have any technical reason. This is another thing we’d like to switch to in the near future.

* 1. SCURM framework

To manage our project efficiently, we adopted the SCRUM framework which is an agile methodology that emphasizes iterative development, collaboration, communication, and adaptability. The key concepts we exploited are:

* Product backlog. These are the features, enhancements, and bug fixes required for the project which we continuously updated with each sprint and feedback. Among them, we have their priority levels to help us in selecting the ones to be implemented first and with immediacy.
* Sprints. A week of iteration was good enough and we had a great deal of backlog done.
* Sprint planning. Before we delve into each sprint, we had a meeting to discuss the goal of the sprint, select backlog to work on and plan tasks to be done.
* Sprint review. After each we have to go back to the drawing board to see if we hit our targets, demonstrate implemented features and ensures goal alignment.
* Sprint retrospective. This is an infrequent meeting where we met to discuss what went wrong, and what went right and see how we can import. If not on its own, we often mix it with sprint review to get accelerated.
* Daily standups. On a daily basis, we had short meetups around campus to see what we did the previous day, what we’re to do and maybe fix bug and blockers if there are any within the team.

All these, combined with some minor elements of SDLC, made us realize some benefits which are enhanced collaboration, increased transparency, improved adaptability, continuous improvement, and better communication. In our case, we only had the SCRUM master and the development team and we also assumed the role of the product owner.