**BackEnd**

We will start the process of developing the database structure and site foundations after the stage of identifying goals and gathering requirements for the following stage, “Sprint 4”, but we must be understand the following information before we start the design process:

**What is the purpose of backend web architecture?**

The purpose of backend web architecture is to develop programs that generate functional experiences for users while separating them from the internal logic of the website. By creating architecture on the backend of a website, you can develop streamlined websites where users can easily navigate through pages and use different features without having to look at complicated code or use external devices. It also allows users to access websites without having to exclusively use their computer's processing power, which makes websites more accessible.

**What are the components of backend web architecture?**

Here is a list of common components of backend web architecture:

* **Servers**

A server is a computer that gathers details about the interactions users have with a website, interprets them and sends them across a network. You can program servers to run specific instances of code depending on the inputs they receive. For instance, if a user enters their credit card information on a website, your server may automatically process the payment through a banking application.

* **Logic**

Logic is the sequence of operations that programmers code into the backend to accomplish specific tasks. Website logic contains algorithms and functions that allow for websites to perform different actions and output information depending on how users behave and interact with website features. Programmers in backend web architecture create logic that runs almost exclusively on servers, interpreting inputs and producing outputs

* **Frameworks**

A software framework is a foundation where developers can make mobile and web apps faster and more standardized. Using a backend framework makes the development and clean pragmatic design, and the example below, available at **StackOverFlow**, is pretty helpful in understanding the concept of a framework.

Backend frameworks are server-side frameworks designed to make tasks easier for developers. They provide tools, libraries, and other components that help developers create the framework for a website or application. Backend frameworks can automate some aspects of web development, making it faster and simpler.

* **Databases**

Databases contain the information that servers access to direct website functions. In website backend architecture, databases include information such as integers, characters and arrays, which are sets of variables that have something in common. Databases have many functions that you can use to organize information for users to access. For example, a server may use a database to retrieve the options for a pull-down menu on a website.

* **APIs**

APIs, or application programming interfaces, allow software programs to communicate with other servers and databases to exchange information. For example, an API may allow a travel website to retrieve flight price information from various other websites in order to present the best price to users. APIs exist in the backend to simplify the presentation of information from multiple databases in a single place.

For the backend development of the application for the internal bus in IMSIU, there are several programming languages that can be considered. Some popular choices for backend development include:

1. **Python**: Python is a versatile language with a wide range of frameworks like Django and Flask, which are well-suited for web development. It has a large community and offers numerous libraries and tools for backend development.
2. **Node.js**: Node.js is a JavaScript runtime that allows you to build scalable and high-performance applications. It has a non-blocking, event-driven architecture, making it suitable for real-time applications.
3. **Java**: Java is a robust and widely adopted programming language. It has a vast ecosystem of frameworks, such as Spring, which can simplify backend development.

As for the database schema, considering the requirements of the application, a relational database management system (RDBMS) would be a suitable choice. Some popular options for RDBMS include:

1. **PostgreSQL**: PostgreSQL is a powerful open-source RDBMS known for its reliability, scalability, and extensive feature set. It supports complex queries and has good support for geospatial data, which could be useful for bus route management.
2. **MySQL**: MySQL is another popular open-source RDBMS that is widely used. It is known for its performance, ease of use, and compatibility with various platforms.