**Project Title:** Curriculum Development – Software Development Component

**TECHNOLOGY (MUST)**

**MBEYA UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FIELD PRACTICAL TRAINING**

**AT MBEYA UNIVERSITY OF SCIENCE AND**

.

**STUDENT NAME:** JOCTAN ELVIN MFUNGO

**REGISTRATION NUMBER:** 24100523140028

**COLLEGE:** COLLEGE OF INFORMATION COMMUNICATION AND TECHNOLOGY

**DEPARTMENT:** COMPUTER SCIENCE AND ENGINEERING

**LEVEL:** NTA 6

**ACADEMIC YEAR:** 2024/2025

**INDUSTRY/FIRM:** MBEYA UNIVERSITY OF SCIENCE AND TECHNOLOGY (MUST)

**INDUSTR/FIRM SUPERVISOR:** MR. EDWIN NCHIA

**UNIVERSITY SUPERVISOR:** MISS. PRISCA CHARLES MARO



**CONTENTS:**

1. **Acknowledgment** ........................................................ 3
2. **Summary** .................................................................. 4
3. **List of Abbreviations** ................................................ 5
4. **Chapter One: Introduction** .................................... 6
   * 1.0 Background of the Firm
   * 1.2 Vision, Mission and Core Values
   * 1.3 Strengths and Weaknesses of the Field Attachment Program
5. **Chapter Two: Activities……………11**
   * 2.0 Activities Performed
   * 2.1 Placement and Working Environment of the Organization
   * 2.2 Major Tasks Performed
   * 2.3 Overall Activities Performed
   * 2.4 System Overview
   * 2.5 New Things I Learned from During My Field Attachment
   * 2.6 Expectations Before the Training
   * 2.7 Things I Learned from Must
6. **Chapter Three: Recommendations………….38**
   * 3.0 Recommendations to the Firm
   * 3.1 Recommendations to The University
   * 3.2 Conclusion
   * 3.3 References
7. **Chapter Four:** **Appendices** .................................................. 41

**ACKNOWLEDGEMENT**

Firstly, I would like to thank God for giving me life, health, and strength during the entire period of my field practical training. Without His blessings, this journey would not have been possible.

I am very thankful to **Mbeya University of Science and Technology (MUST)** for giving me the chance to do my practical training within the institution. This opportunity gave me real exposure to the working environment, where I was able to connect what I had learned in class with actual practice. Through this training, I gained more confidence, skills, and experience in **software development**, especially in the area of **Curriculum Development**.

My special appreciation goes to my facilitator, **Mr. Edwin Nchia**, who guided me within the entire training. His advice, encouragement, and support helped me to carry out my tasks well and to learn new things that I could not have understood from theory alone. He was always available to assist me whenever I faced challenges, and his mentorship was very important in building my knowledge and professional growth.

I would also like to extend my gratitude to other staff members at MUST like lab technicians who welcomed me warmly and supported me whenever I needed help. Their cooperation and positive attitude made my training period smooth and enjoyable.

Lastly, I appreciate the support of my fellow students and colleagues, who worked with me, shared ideas, and encouraged me during this field training. Their teamwork and friendship made my experience even more meaningful.

**SUMMARY**

This report presents the activities, experiences, and knowledge which I gained during my Field Practical Training at Mbeya University of Science and Technology (MUST). The training was conducted under the supervision of **Mr. Edwin Nchia** and focused on the **Software Development component of the Curriculum Development Project**.

The report is organized into three main chapters.

**Chapter One** introduces the background of the institution, services offered, vision and mission, as well as the objectives of the field attachment.

**Chapter Two** describes the activities I performed during the training, which mainly involved software development tasks such as system design, coding, testing, debugging, and documentation. It also outlines the challenges I encountered and how solved them.

**Chapter Three:** presents recommendations and conclusions based on the knowledge and experience obtained during the training.

The overall experience was very beneficial, as it allowed me to put into practice the theories and concepts I learned in class, also developing problem solving skills, teamwork, and confidence in real software development tasks.

# **LIST OF ABBREVIATION:**

PC-Personal Computer.

WI-FI-Wireless Federation.

OS-Operating System.

MS-Microsoft.

RJ-Registered Jack.

CMOS-Complementary Metal Oxide Semiconductor.

ROM-Read Only Memory.

CMD-Command Prompt

USB-Universal Serial Bus

RAM-Random access memory

CPU-central processing unit

ICT-Information and communication technology.

CD-Compact Disk

**CHAPTER ONE**:- **INTRODUCTION**

1.0 BACKGROUND OF THE FIRM

Mbeya University of Science and Technology (MUST) has a rich history that began with its roots as a technical college. The institution's journey to becoming a full fledged university is a story of growth and expansion, driven by the need to provide specialized education in science and technology for Tanzania and the broader East African region.

**The Beginning: Mbeya Technical College**

The story of MUST began in **1986** when it was established as **Mbeya Technical College (MTC)**. This was a direct response to the government's push for more technical skills and expertise in the country. At its inception, MTC was a non-degree awarding institution. Its primary focus was to offer training in various technical fields, such as civil, electrical, and mechanical engineering. The college aimed to produce technicians and engineers who could contribute directly to the country's development projects.

The curriculum was designed to be practical and hands on, providing students with the skills needed for the job market. The initial student intake was small, but the college quickly gained a reputation for providing high-quality technical education. Over the years, it expanded its programs and infrastructure to accommodate more students and a wider range of courses.

**Transition to a University College**

The push for MTC to become a university began in the early 2000s. The government and the college's management saw the need to upgrade the institution to meet the growing demand for higher education in science and technology. This was a strategic move to help the country achieve its development goals, as outlined in national development plans.

In **2006**, Mbeya Technical College was officially upgraded to **Mbeya Institute of Science and Technology (MIST)**. This was a significant step, as it marked the beginning of its journey towards university status. Although it was now an institute, it still operated under the guidance of the University of Dar es Salaam (UDSM) as a constituent college. This meant that MIST could now offer degree programs, but they were accredited and awarded by UDSM. This period was crucial for developing the academic culture, research capacity, and quality assurance systems that a university needs.

During this time, the institution expanded its academic offerings to include more advanced programs in engineering, information technology, and applied sciences. The number of students increased significantly, and the campus saw new buildings and facilities being constructed to support the growth. The partnership with UDSM helped MIST to build its academic reputation and ensure its programs met national and international standards.

**Gaining Full University Status**

The final step in the institution's transformation happened in **2013**. After years of preparation, Mbeya Institute of Science and Technology was granted a charter by the government, officially making it a full-fledged university. It was renamed **Mbeya University of Science and Technology (MUST)**.

This new status gave MUST the autonomy to manage its own affairs, including developing its own curriculum, awarding its own degrees, and conducting independent research. The transition from an institute to a university was a major milestone, not just for the institution itself but for the entire southern highlands of Tanzania. It meant that students in the region had a world-class university specializing in science and technology right at their doorstep.

Since becoming a university, MUST has continued to grow. It has introduced new faculties and programs, focusing on areas like business, architecture, and environmental studies. The university's mission is to be a center of excellence in science and technology, producing graduates who are not only skilled but also innovative and entrepreneurial. It plays a vital role in local community development through outreach programs and research projects that address regional challenges. The university's history from a small technical college to a respected university shows a commitment to education and national development.

## **1.1 VISION, MISSION AND CORE VALUES**

Mbeya University of Science and Technology (MUST) has a clear vision, mission, and a set of core values that guide its operations and overall purpose. These elements define what the university aims to achieve, how it plans to do it, and the principles that govern its actions.

**Vision**

The vision of MUST is to be a **leading center of excellence for knowledge, skills, and applied education in science and technology**. This statement shows their ambition to not only be a top university but to also be a leader in practical, hands on education that applies scientific and technological knowledge to real world problems. They want to be the best at what they do.

**Mission**

The mission of the university is to **develop academically, technologically, and socially competent students, staff, and other stakeholders** who can respond to the broad needs and challenges of society. This mission expands on the vision by detailing how they will achieve their goals. It emphasizes the development of well-rounded individuals who are not only smart in their fields but also skilled in using technology and aware of their social responsibilities. It focuses on the importance of creating people who are a solution to societal problems.

**Core Values**

MUST's core values are the fundamental beliefs that shape its culture and guide its day-to-day decisions. They are:

1. **Leadership in Innovation and Technology:** The university is committed to being at the forefront of new ideas and technological advancements. They value high quality, modern programs and encourage continuous innovation.
2. **Culture of Excellence:** This value means that MUST strives for the highest quality in everything it does, from teaching and research to providing services. They are always working to improve and set challenging goals.
3. **Diversity and Equal Opportunity for All:** The university believes in creating an environment where everyone is respected and has an equal chance to succeed, regardless of their background. They promote a culture of fairness and respect.
4. **Partnerships:** MUST understands the importance of working with others. They actively collaborate with industries, government institutions, and other organizations both in Tanzania and internationally to achieve their goals.
5. **Integrity and Stewardship of Resources:** This value highlights the university's commitment to being honest, accountable, and responsible. They aim to manage their resources whether it's money, facilities, or human capital with the highest level of professional ethics.

## **1.2 OBJECTIVE OF THE FIELD ATTACHMENT**

The objective of the fieldwork is to put the theories into practical to gain the experience and clear knowledge of the different things concern a certain career. Also, field attachment is the field based practical training experience that prepare trainees for the task or activities they are expected to perform on the completion of the training. Therefore, the following are the specific objective of the field attachment program.

Field attachment aimed at measuring the level of understanding by applying the theory concept and knowledge into practical work. This is due to that during the field training or fieldwork student will be able to put into practical the different theories that he or she learn into class.

Field attachment program also aim in increasing confidence to a student during the work. This is due to that since a student are assigned to perform the different activities by the onsite supervisor during the practical training this normally help a student to build the strong confidence at work environment also it assists the future development once; he or she gets the job.

Also, field attachment aims to create the competence and develop a student to understand the different opportunity responsibilities and knowledge of the different employers also work ethics and on how the worker they behave in working area.

**1.3 STRENGTHS AND WEAKNESSES OF THE FIELD ATTACHMENT PROGRAM**

This part of my report is about the good and bad things I found during my field attachment at Mbeya University of Science and Technology (MUST). My work was on a software project for the Curriculum Development department, and my supervisor was Mr. Edwin Nchia.

**Strengths of the Program**

**I got to do real work.** Instead of just reading books, I actually got to build software. This helped me see how things are done in a real office, and it taught me about working with a team and finishing work on time.

**I used my class knowledge.** I was able to use the coding skills I learned at school. It was good to see that what I learned was useful in a real project.

**My supervisor was a great help.** Having Mr. Nchia to guide me was very useful. He is an expert, and he gave me advice and helped me when I had problems.

**I met important people.** I got to meet my supervisor and other people who work at the university. These connections can help me find a job in the future.

**Weaknesses of the Program**

**The tools were sometimes a problem.** Some of the computers or the internet were a bit slow. This made my work take longer than it should have.

**I only learned a few things.** Because I was working on one specific project, I didn't get to learn about other types of software or technologies.

**The work felt a bit simple sometimes.** The tasks I was given weren't always very challenging. I feel like I could have done more.

**CHAPTER TWO: ACTIVITIES**

**2.0 ACTIVITIES PERFORMED**

My main job during my field attachment was to work on a software project for the **Curriculum Development** section. The goal was to build a system that could manage all of the university's academic courses. This project was a team effort, and we followed a clear plan, starting with the front-end, then the back-end, and finally connecting the two.

**Front-End Development: Building What Users See**

We began by creating the part of the system that users would see and interact with. For this, we used **React** and **Vite**, along with **TypeScript**.

**Creating the User Interface (UI):** I was involved in designing and building the pages that university staff would use. This included creating forms where they could add new courses, view lists of existing curricula, and edit details like course descriptions or credit hours. We aimed to make these pages simple and easy to use.

**Using React for Components:** To make our work more efficient, we used **React** to break down the UI into smaller, reusable pieces called components. For example, a "course card" component could be used on multiple pages, saving us time and ensuring a consistent look.

**Using TypeScript for Stronger Code:** We chose to use **TypeScript** because it helped us write better, more reliable code. It acts like a safety net, catching many common coding mistakes before they become bigger problems. This made our development process smoother.

**Back-End Development: Storing and Managing Data**

After building the front-end, we worked on the "behind-the-scenes" part of the system. This is where the data is stored and all the business logic happens.

**Designing the Database:** I helped design the database using **PostgreSQL**. I created a structure (tables) to hold all the necessary information, such as course names, codes, module details, and their relationships. This step was like creating the blueprint for the system's "brain."

**Managing the Database with pgAdmin:** We used **pgAdmin** as a tool to easily manage and interact with our **PostgreSQL** database. This made it simple to create tables, run queries, and make sure our data was organized correctly.

**Coding the Back End Logic:** For the core of the back-end, we used **Java**. I wrote code that handles all the requests from the front-end. For instance, when a user fills out a form to add a new course, my code receives that information and tells the database to save it. This is what makes the system functional.

**Connecting the Systems: Making Everything Work Together**

The final and most important step was to connect the front-end and the back-end to make them work as a single, complete system.

**Building the API:** We made sure our **React** front-end could talk to the **Java** back-end using an **API** (Application Programming Interface). This API acted like a messenger, allowing the two parts to send and receive information from each other.

**Final Integration:** This step was about bringing everything together. When a user interacted with the beautiful front-end, the information would be sent through the API to our powerful back-end, which would then store it in our well-structured **PostgreSQL** database. This final step made our project a complete and working application.

**2.1 PLACEMENT AND WORKING ENVIRONMENT OF THE ORGANIZATION**

During my field attachment, I was placed at Mbeya University of Science and Technology (MUST) within the **Department of computer science and engineering**. My main task was to work on a software development project for the **Curriculum Development Project**. My facilitator for this attachment was **Mr. Edwin Nchia.**

The working environment was very good and supportive. After being given a task, my supervisor would give me a brief introduction on how to do it, and then I was encouraged to work on my own or as part of a team.

This level of independence was very helpful. It helped me build trust in my own skills and gave me the confidence to handle tasks on my own. I only asked for help from my supervisor when it was absolutely necessary.

There was also a great spirit of teamwork within the department. Everyone was friendly and worked together to make sure our duties were completed efficiently and on time. This cooperative environment made my time at MUST very productive and enjoyable.

**2.2. MAJOR TASKS PERFORMED**

My main job during my field attachment was the **Curriculum Development System Project**. My role was to help build a new software system from start to finish. This system was designed to help the university manage all of its academic courses and programs in a modern way.

My work on this project was done in three main parts:

**1. Front-End Development: Building What Users See**

This was the first part of the project. I worked with a team to build the part of the software that users would see and click on. We used a framework called **React** with **Vite** and **TypeScript** to build it. I helped to create user-friendly pages, forms, and buttons. For example, I built the page where staff can add a brand-new course, and the page where they can see all the details of an existing course. The goal was to make the system simple and easy for anyone to use.

**2. Back-End Development: The System's Engine**

After we built the front-end, we focused on the back-end. This is the part of the system that works behind the scenes to store and manage all the data. We used **Java** to write the main code. My job was to help create the code that handles all the business rules. For example, when a user saves a new course, my code made sure that the information was valid before it was sent to the database. This work was important to make sure the system worked correctly and securely.

**3. Database Design and Integration: Making Everything Work Together**

This was the final and most important part. I helped design the database using **PostgreSQL**. The database is like a big, organized file cabinet where all the information about courses, modules, and departments is stored. We used **pgAdmin** to manage it, which made it easy to create and organize our data tables. The final step was to connect our front-end and back-end so they could talk to each other. This allowed users' actions on the front-end to be saved correctly in the database through the back-end. This process made our project a complete and working application.

**2.3 OVERALL ACTIVITIES PERFORMED**

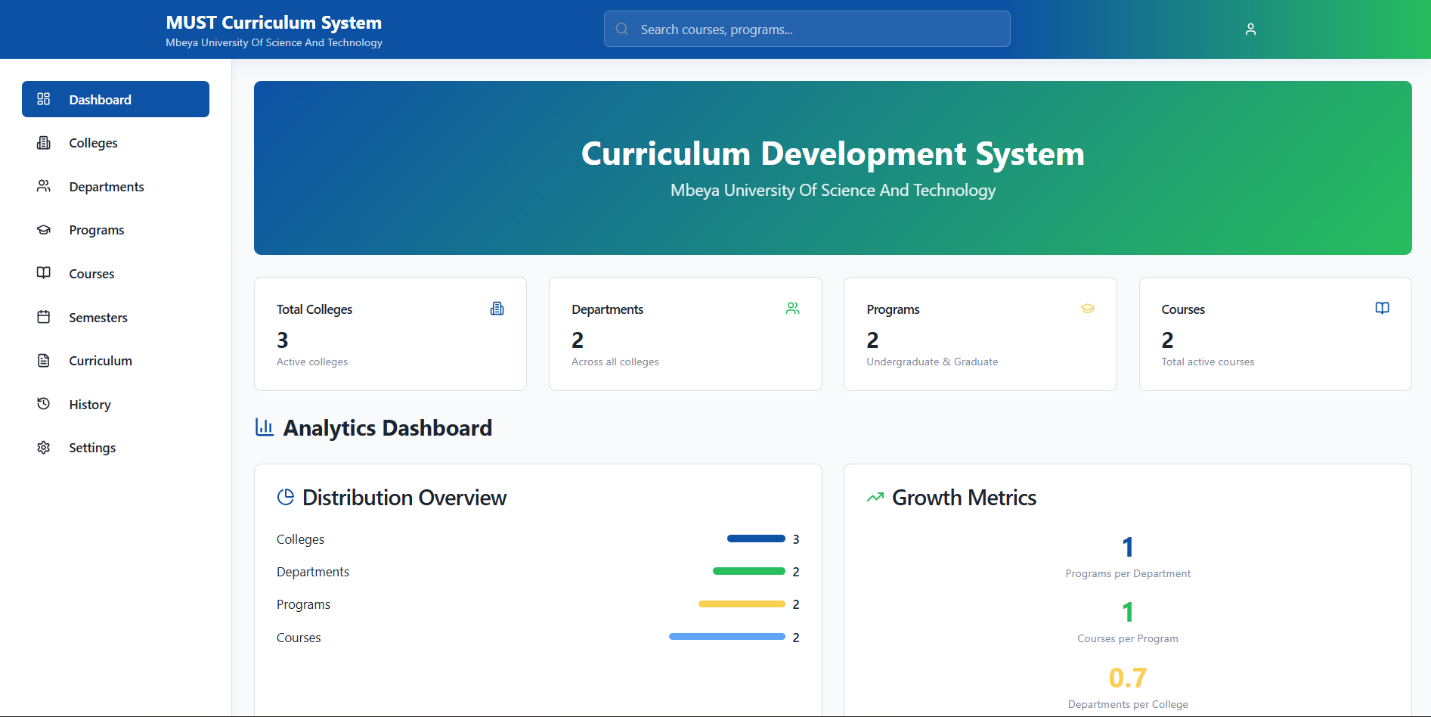
During my field attachment at Mbeya University of Science and Technology (MUST), I performed several key activities, both major and general.

**Major Activities: The Curriculum Development System Project**

My main task was to work on a new software system for managing the university's curriculum. This project was a complete cycle, and I was involved in all three main stages:

1. **Front-End Development:** I helped build the part of the system that users see. I used **React** and **TypeScript** to create the user-friendly pages, forms, and buttons.
2. **Back-End Development:** I worked on the part of the system that works behind the scenes. Using **Java**, I wrote the code that manages the system's logic and handles all the data.
3. **Database and Integration:** I helped design the database with **PostgreSQL** and connected the front-end and back-end so they could work together as a single system.

* 1. **FRONT-END DEVELOPMENT:**



**Front-End Development**

The first big part of our project was building the front-end, which is what the user sees and clicks on. Our goal was to create a modern and easy-to-use interface, just like the one you see in the picture. We followed a clear process from the very beginning until the end.

**1. Setup and Initial Installation**

Before writing any code, we had to prepare our development environment. This involved a few key steps:

First, we had to install all the software needed to build our project.

1. **VS Code:** We started by installing **Visual Studio Code (VS Code)**. This is the main program we used to write all our code. It's a very popular code editor that works well with all the technologies we used. Here are the steps for installing VS Code, written in past tense, as requested.

**How to Install VS Code**

Installing VS Code was a very easy process. I just followed a few simple steps to get it set up on my computer.

**Step 1: Downloading VS Code**

1. First, I needed to download the VS Code file from the official website.
2. I opened my internet browser, like Chrome, and went to the official VS Code website at [**https://code.visualstudio.com/**](https://code.visualstudio.com/).
3. The website automatically knew what kind of computer I had (Windows, macOS, or Linux) and showed a large download button. I clicked this button to start the download.

**Step 2: Installing VS Code**

Once the download was finished, I installed the program.

1. I went to my **"Downloads"** folder, where the file was saved.
2. I double-clicked the downloaded file. An installation window popped up. I accepted the agreement and clicked **"Next"**.
3. I followed the rest of the instructions and left most of the settings as they were, but I made sure to check the box that said **"Add to PATH"**. This made it easy to open VS Code from my Command Prompt or Terminal later on.
4. I then clicked **"Install"** and waited for the process to finish. When it was done, I clicked **"Finish"**.

*After that, VS Code was installed and ready to be used on my computer. I could open it from the Start Menu or by clicking the icon on my desktop.*

A screenshot of a computer

AI-generated content may be incorrect.

1. **Node.js and npm:** Next, we installed **Node.js**. This was like the main engine for our project and was required to run React applications. It came with a tool called **npm** (Node Package Manager), which we used to install all the other libraries and packages we needed

**How to install Node.js and npm**

Installing **Node.js** and **npm** was a straightforward process. I followed a few simple steps to get them set up on my computer.

**Step 1: Downloading Node.js**

I started by downloading the Node.js installer from the official website.

1. I went to the official Node.js website at [**https://nodejs.org/**](https://nodejs.org/).
2. The site had two versions for download: a "LTS" (Long-Term Support) version and a "Current" version. I chose the **LTS** version because it was more stable and recommended for most users. I clicked the download button that was right for my computer's operating system (Windows, macOS, or Linux).

**Step 2: Installing Node.js and npm**

After the installer file finished downloading, I ran it to install both Node.js and npm.

1. I found the downloaded file in my "Downloads" folder and double-clicked it to start the installation.
2. An installation window appeared. I clicked **"Next"** to begin.
3. I accepted the license agreement and clicked **"Next"** again.
4. I left all the default settings as they were, because they included installing npm along with Node.js. I clicked **"Next"** one more time.
5. Finally, I clicked **"Install"** and waited for the process to finish. When it was done, I clicked **"Finish"**.

*After this process, both Node.js and npm were successfully installed and ready to be used on my computer.*

1. **Git:** We also installed **Git**, which is a system for tracking changes in our code. This helped our team work together without problems and allowed us to save different versions of our project.

**How to install Git:**

Installing Git was a crucial step for our project, as it helped our team work together and manage changes to our code. The process was straightforward, and I followed these steps to get it set up.

**Step 1: Downloading Git**

I started by downloading the Git installer from the official website.

1. I went to the official Git website at [**https://git-scm.com/**](https://git-scm.com/).
2. The website automatically showed me a download button for my operating system. I clicked on it to start the download.
3. For example, if I was using Windows, I clicked the **"Download for Windows"** button.

**Step 2: Running the Installer**

After the download was complete, I ran the installer.

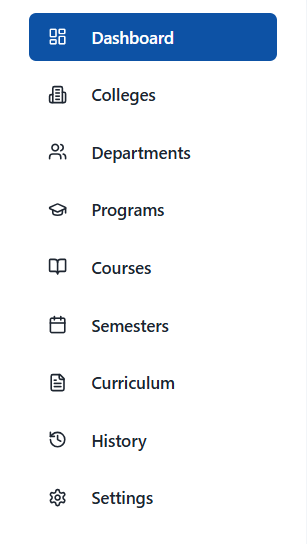
1. I found the downloaded file (which was a .exe file on Windows) in my "Downloads" folder and double-clicked it.
2. An installation window popped up. I clicked **"Next"** to continue.
3. I agreed to the license terms and clicked **"Next"** again.
4. I left most of the default settings as they were, since they were already set for a standard installation. I clicked **"Next"** through the various options.
5. Finally, I clicked **"Install"** and waited for the process to finish. When it was done, I clicked **"Finish"**.

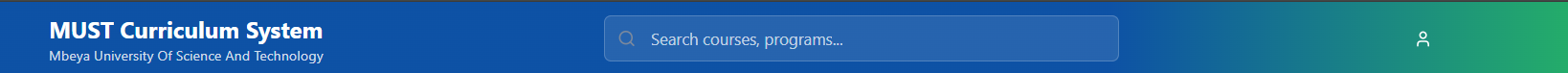
*After these steps, Git was successfully installed on my computer. I could then use it to manage my code from the Command Prompt or VS Code.*

**2. Drawing the Design;**

Before we wrote the actual code, we had to draw out how the front-end would look. This was like creating a map for the house we were about to build.

**Sidebar and Top Bar:** We planned to have a **sidebar** on the left with all the menu options (Dashboard, Colleges, etc.) and a **top bar** with the logo and a search button. This was the main layout for every page.





**Dashboard Layout:** We designed the main dashboard page. We decided to have small, colorful cards at the top to show important numbers like the total number of colleges. Below that, we planned to have the "Analytics Dashboard" section with charts and graphs to show more details about the data.



* 1. **Building the Pieces (Coding)**

After we finished our design and our tools were ready, we started to build the actual front-end. This was the part where we wrote the code for everything the user sees. We used **React** with **TypeScript** for this.

* + 1. **. Creating Reusable Components**

We didn't just write one long piece of code. Instead, we broke the user interface down into small, reusable parts called **components**. This made our work much more organized and efficient.

* + 1. **Header Component:**

We built a component for the very top of the page that held the university's logo, the title, and the search bar. We could then use this component on any page we wanted.

* + 1. **Sidebar Component:**

We created a component for the menu on the left side. It contained links for "Dashboard," "Colleges," "Departments," and so on.

* + 1. **Card Component:**

We also built a simple card component that we used over and over to show the total numbers of colleges, departments, and programs. This way, we only had to style it once.

**e) Building the Dashboard Page**

With our small components ready, we used them to build the main dashboard page.

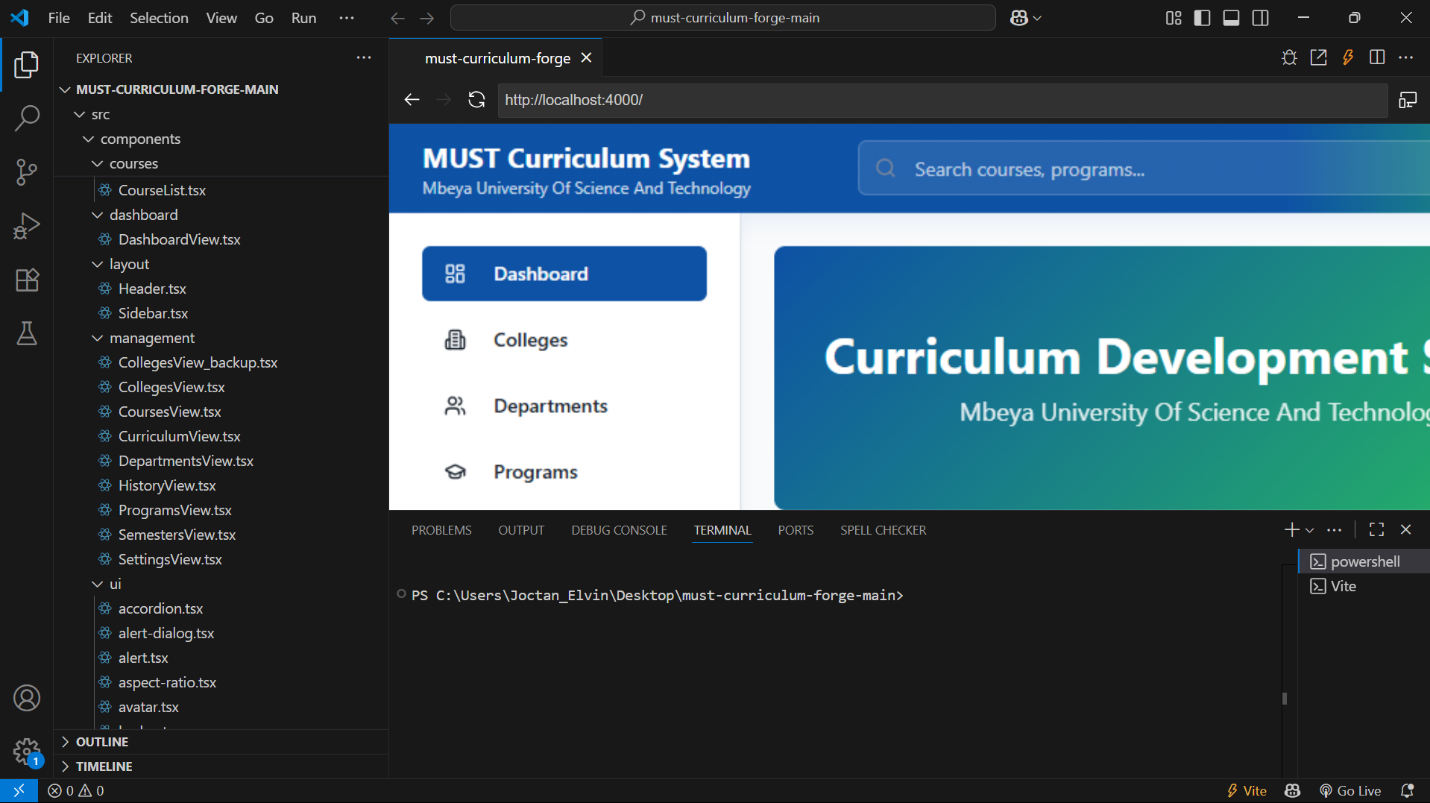
**f) Structuring the Page:**

We arranged our components to match our design. We put the Header component at the top, the Sidebar component on the left, and the main content area on the right.

**g) Coding the Analytics:**

We then wrote the code to show the data in the "Analytics Dashboard" section. We used our chart libraries to take the numbers and turn them into the bar charts and growth metrics that you saw in the picture. This helped us make the data easy to understand at a glance.

*By building in this way, we made sure our code was clean, organized, and easy to work with.*

**

* 1. **Making it Look Good (Styling)**

The final step was to add style and color to the front-end to make it look clean and professional.

1. **Using Tailwind CSS:** We used a special tool called **Tailwind CSS** to handle all the styling. Instead of writing long styling rules, we added small commands directly in our code to change things like color, spacing, and size. This is how we made the dashboard look modern and gave it the nice blue and green colors you see.
2. **Final Checks:** We went through everything one more time to make sure all the buttons worked, the colors looked right, and the pages loaded without any problems. The use of c **TypeScript** helped us avoid many simple errors from the beginning, so this step was smooth.
   1. **BACK-END DEVELOPMENT:**

A screenshot of a computer program

AI-generated content may be incorrect.

After we finished the front-end, our next major task was to build the back-end. This is the part of the system that users don't see; it's what stores, manages, and processes all the data. Our team used **Java**, **PostgreSQL**, and **pgAdmin** to build this part. The process involved three main steps, from setting up the environment to writing the code.

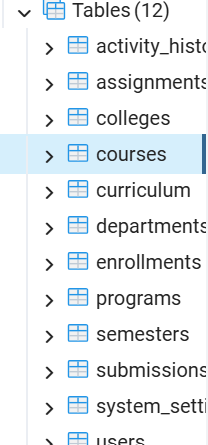
* + 1. **Setting Up the Back-End Environment**

Before we could write any back-end code, we had to install and set up our tools.

1. **Installing Java:** We started by installing the **Java Development Kit (JDK)**, which is needed to run and write Java applications.
2. **Installing PostgreSQL:** Next, we installed **PostgreSQL**, which is the database where all the curriculum information would be stored.
3. **Installing pgAdmin:** We also installed **pgAdmin**, a tool that made it easy to manage our PostgreSQL database. It gave us a visual way to create tables, add data, and run commands without having to write code every time.
4. **Setting Up the Project:** Finally, we used a tool to create our Java project. This set up all the necessary files and folders for our back-end.
   * 1. **Designing the Database (Schema)**

Once everything was installed, we designed the database to hold all the curriculum data.

1. We used **pgAdmin** to create the tables for our database. We created a table for Colleges, a table for Departments, a table for Programs, and a table for Courses.
2. We also set up relationships between these tables. For example, we made sure that each Department was linked to a specific College, and each Program was linked to a specific Department. This made our data organized and easy to work with.



* + 1. **Writing the Back-End Logic**

This was the main part of the back-end work. We wrote the Java code that made the system functional.

* We created **API Endpoints**. These are like special addresses that our front-end could send requests to. For example, when a user clicks "Save New Course" on the front-end, the front-end sends a request to a specific API endpoint on the back-end.
* We wrote code to handle these requests. For example, when the back-end received a request to save a new course, our Java code would process the information and then save it in the correct table in our **PostgreSQL** database.
* We also wrote code to get information from the database. When the front-end wanted to show a list of all colleges, our Java code would get that information from the database and send it back to the front-end to be displayed.

This back-end work prepared the system to handle, store, and manage all the curriculum data. The final step was to connect the front-end we had already built with this new back-end.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. **INTEGRATION OF FRONT-END AND BACK-END**

The final and most crucial step of our project was to connect the front-end we had built with the back-end we had developed. This process, known as integration, allowed the user interface to communicate with the database.

* + - 1. **Building the API**

The connection between the front end and back end was made possible through an **API** (Application Programming Interface). Think of the API as a messenger. It carried requests from our **React** front-end to our **Java** back-end.

1. We defined specific API endpoints. For example, we created an endpoint like /api/courses to handle all requests related to course data.
2. We made sure the API could handle different types of requests, such as getting data, sending new data, updating existing data, and deleting data.
   * + 1. **Connecting the Systems**

With the API ready, we connected our two separate parts.

1. On the front-end, we wrote code that would send requests to our API endpoints. For example, when a user clicked the "Save" button on the "Add New Course" form, the front-end would send a request to the back-end's /api/courses endpoint.
2. The back-end would then receive this request, process the data, and save it in our **PostgreSQL** database.
3. The back-end would also send a response back to the front-end to let it know if the action was successful or if there was an error. This made the system interactive for the user.
   * + 1. **Final Testing**

After connecting everything, we did a final round of testing to make sure the entire system worked as expected.

1. We tested all the functions, from adding a new college to editing a course, to make sure the front-end and back-end were communicating correctly.
2. We made sure that any changes made on the front-end were correctly reflected in the database and that the information shown on the screen was always accurate

**BACK END:**

A screenshot of a computer

AI-generated content may be incorrect.

**FRONT END:**

A screenshot of a computer

AI-generated content may be incorrect.

*By completing these steps, we successfully integrated our two main parts, creating a complete and fully functional* ***Curriculum Development System***

**2.4 SYSTEM OVERVIEW:**

**CURRICULUM MANAGEMENT SYSTEM:**

**SYSTEM OVERVIEW:**

This is a modern university curriculum management system that uses advanced web development technologies. The system is designed with a Frontend and Backend that are professionally integrated.

**SYSTEM ARCHITECTURE:**

**1. FRONTEND (USER SIDE)**

**Technologies Used:**

1. **React 18:** - A JavaScript library for building user interfaces.
2. **TypeScript:** - A programming language with type safety.
3. **Vite:** - A fast build tool for modern web development.
4. **Tailwind CSS:** - A utility-first CSS framework.

**How It Works:**

1. It runs on Port 5173 (http://localhost:5173).
2. It uses a Single Page Application (SPA) architecture.
3. **React Router** is used for navigation.
4. **Context API** is used for state management.
5. **Hooks** (useState, useEffect) are used for data management.

**2. BACKEND (SERVER SIDE)**

**Technologies Used:**

1. **Node.js:** - A JavaScript runtime environment.
2. **Express.js:** - A web framework for Node.js.
3. **TypeScript:** - Type-safe programming.
4. **PostgreSQL:** - A relational database.
5. **CORS:** - Cross-Origin Resource Sharing.
6. **Dotenv:** - Environment variables management.

**How It Works:**

1. It runs on Port 3001 (http://localhost:3001).
2. It provides **RESTful API endpoints**.
3. It connects to a **PostgreSQL database**.
4. It uses connection pooling for database efficiency.

**FRONTEND AND BACKEND INTEGRATION**

**1. API COMMUNICATION PATTERN**

**Frontend → Backend Communication:**

JavaScript

// Example of an API call from the frontend

const fetchColleges = async () => {

  try {

    const response = await fetch('http://localhost:3001/api/colleges');

    const data = await response.json();

    return data;

  } catch (error) {

    console.error('Error fetching colleges:', error);

  }

};

**Backend Response Pattern:**

JavaScript

// Example of an API endpoint on the backend

app.get('/api/colleges', async (req, res) => {

  try {

    const result = await pool.query('SELECT \* FROM colleges');

    res.json(result.rows);

  } catch (error) {

    res.status(500).json({ error: 'Database error' });

  }

});

**2. DATA FLOW**

[User Interface] → [Frontend React] → [HTTP Request] → [Backend Express] → [PostgreSQL Database                                                      ←

[UI Updates] ← [State Update] ← [JSON Response] ← [Database Query] ← [Data Storage]

**DATABASE SCHEM:**

**Main Tables:**

1. **colleges** - University colleges
2. **departments** - Departments for each college
3. **programs** - Study programs (degrees)
4. **courses** - Individual subjects
5. **semesters** - Academic periods
6. **curriculums** - Complete curricula

**Table Relationships:**

colleges (1) → (many) departments

departments (1) → (many) programs

programs (1) → (many) courses

semesters (1) → (many) courses

**SYSTEM WORKFLOW**

**1. SYSTEM STARTUP**

a) PostgreSQL Database starts

b) Backend Server starts (Port 3001)

c) Frontend Server starts (Port 5173)

d) Browser opens http://localhost:5173

**2. DATA LOADING PROCESS**

a) Frontend loads (React components load)

b) DataContext makes API calls

c) Backend receives requests

d) Database queries are executed

e) Data is returned in JSON format

f) Frontend updates UI with new data

**3. USER INTERACTION FLOW**

User clicks button → Frontend sends request → Backend processes → Database saves → Response returns → UI updates

**SYSTEM COMPONENTS:**

**Frontend Components:**

1. **Dashboard View:** - Main dashboard
2. **Colleges View:** - College management
3. **Departments View:** - Department management
4. **Programs View:** - Program management
5. **Courses View:** - Course management
6. **Curriculum View:** - Curriculum management

**Backend Routes:**

1. /api/colleges - CRUD operations for colleges
2. /api/departments - CRUD operations for departments
3. /api/programs - CRUD operations for programs
4. /api/courses - CRUD operations for courses
5. /api/curriculums - CRUD operations for curriculum

**DEVELOPMENT ENVIRONMENT**

**Environment Variables (.env)**

DB\_PORT=5432

DB\_NAME=curriculum\_db

DB\_USER=postgres

DB\_PASSWORD=your\_password

PORT=3001

**Package Management:**

1. **Frontend:** npm/yarn packages for the React ecosystem
2. **Backend:** npm packages for the Node.js ecosystem

**SECURITY AND PERFORMANCE**

**Security Measures:**

1. **CORS** configuration for API security
2. **Environment variables** for sensitive data
3. **Input validation** on the backend
4. **Error handling** for user experience

**Performance Optimization:**

1. **Connection pooling** for the database
2. Async/await for non-blocking operations
3. **Component optimization** in React
4. Efficient database queries

**BENEFITS OF THIS ARCHITECTURE**

1. **Scalability:** The system can grow easily.
2. **Maintainability:** The code is easy to maintain.
3. **Separation of Concerns:** Frontend and Backend are separated.
4. **Modern Technologies:** It uses modern technologies.
5. **Type Safety:** TypeScript ensures code quality.
6. **Responsive Design:** It works on all devices.

**DEVELOPMENT WORKFLOW**

**Development Process:**

1. **Database Design:** → Schema creation
2. **Backend Development:** → API endpoints
3. **Frontend Development:** → UI components
4. **Integration:** → Connect frontend and backend
5. **Testing:** → Ensure everything works
6. **Deployment:** → Production ready

**Best Practices:**

1. Code organization for maintainability
2. Error handling for user experience
3. Documentation for team collaboration
4. Version control for code management

**2.5 NEW THINGS I LEARNED DURING MY FIELD ATTACHMENT**

During my field attachment, I learned many new and valuable things that went beyond what I was taught in class. My work on the Curriculum Development System project gave me a lot of practical experience.

**Technical Skills**

1. **Modern Front-End Development:** I gained hands-on experience with modern front-end technologies like **React, Vite, and TypeScript**. I learned how to build reusable components, manage the application's state, and create a user-friendly interface using **Tailwind CSS**.
2. **Back-End Development with Java:** I learned how to use **Java** to build a back-end server from scratch. I now understand how to create a **RESTful API**, handle user requests, and write the logic that makes a system functional.
3. **Database Management:** I learned how to work with a relational database, **PostgreSQL**. Using **pgAdmin**, I learned how to design a database schema, create tables, and manage data.
4. **System Integration:** The most important technical skill I learned was how to connect a front-end and a back-end. I now understand how an **API** works as a bridge to allow the two parts of a system to communicate with each other.

**Professional and Soft Skills**

1. **Teamwork and Collaboration:** I learned how to work effectively with a team of developers. We shared ideas, solved problems together, and relied on each other to get the project done.
2. **Problem-Solving:** When I faced a difficult bug or a technical challenge, I learned to break down the problem into smaller parts. I also learned the importance of asking my supervisor, Mr. Nchia, for guidance when I was stuck.
3. **Time Management:** I learned how to manage my time and meet deadlines, especially when facing challenges like slow internet or old hardware. This experience taught me to plan my tasks and prioritize my work.
4. **Communication:** I improved my ability to communicate clearly, both with my team members and with non-technical people from the curriculum department. I learned to explain complex technical ideas in a way that was easy for them to understand.

**2.6 EXPECTATIONS BEFORE THE TRAINING**

Before I started my field attachment at Mbeya University of Science and Technology (MUST), I had a few key expectations for what I would learn and experience. My main goal was to put the knowledge I had gained from my university classes into practice in a real-world setting.

1. **To get hands-on experience:** I expected to work on a real software project from start to finish. I wanted to move beyond theory and actually write code that would be used by real people. I was particularly excited to work with modern technologies that are popular in the industry today.
2. **To improve my coding skills:** I expected to become a better programmer. I hoped that by working on a complex project, I would learn how to write cleaner, more efficient code and solve problems that I hadn't seen before. I also wanted to learn how to fix bugs in a large system.
3. **To learn about teamwork:** In school, we often work on projects alone. I expected to learn how to work effectively as part of a team. I wanted to understand how to share tasks, communicate with other developers, and collaborate to achieve a common goal.
4. **To learn from experts:** I hoped to have a good supervisor who could guide me and teach me new things. I expected to get mentorship from a professional in the field who could give me advice and help me grow my skills.
5. **To understand the professional environment:** I wanted to see what it's like to work in a real office environment. I expected to learn about professional conduct, time management, and how to interact with colleagues and managers.

**2.7 THINGS I LEARNED FROM MUST**

During my field attachment at Mbeya University of Science and Technology (MUST), I gained a lot of valuable experience. I not only applied what I learned in class but also acquired new skills that will be very important for my future career.

**Technical Skills**

I got hands-on experience with modern software development technologies that are widely used in the industry today. I learned how to:

* Build a user interface from scratch using **React** and **TypeScript**.
* Develop a back-end server using **Java** and manage data with a **PostgreSQL** database.
* Connect the front-end and back-end to make a full-stack application work, which taught me about **API integration**.

**Professional and Soft Skills**

Beyond coding, I also learned valuable professional skills that you can't get from a textbook. I learned how to:

* **Work in a team.** I collaborated with other students and my supervisor to solve problems and complete the project.
* **Manage my time effectively.** I learned to plan my tasks and meet deadlines, even when faced with technical challenges.
* **Communicate professionally.** I learned to ask for help when I needed it and to explain technical concepts to people who weren't in my field.

This experience gave me a much clearer picture of what it's like to work as a software developer and prepared me for a professional career.

**CHAPTER THREE: - RECOMMENDATIONS**

**3.0 RECOMMENDATIONS TO THE FIRM:**

Based on my experience during the field attachment at Mbeya University of Science and Technology (MUST), I have a few recommendations for the ICT Department to help improve future field programs and overall development efforts.

**1. Improve Equipment and Infrastructure**

I recommend that the department update some of its equipment, such as computers and networking tools. Faster computers and a more stable internet connection would allow students to work more efficiently and complete their tasks on time. This would also give students experience with the latest technologies used in the industry.

**2. Provide a Clear Project Plan**

Having a clear project plan from the start is very helpful. I recommend that the department create a detailed plan for each field project. This plan should include clear goals, a timeline, and the specific technologies that will be used. This would help students know exactly what is expected of them and would make the project run more smoothly.

**3.1 RECOMMENDATIONS TO THE UNIVERSITY**

The institute Mbeya University of Science and Technology (MUST) that they should search field for training to the student basic on difference course they undertake cause student they face and gate a hard time during searching for field in different company or organization

**3.2 CONCLUSION**

My field attachment at Mbeya University of Science and Technology (MUST) was an incredibly valuable and eye-opening experience. It successfully helped me move from just learning in a classroom to actually applying my knowledge in a real-world setting.

I got to work on a complete software project from beginning to end, which was a huge learning opportunity. My work on the **Curriculum Development System** gave me hands-on experience with modern technologies like **React** for the front-end, **Java** for the back-end, and **PostgreSQL** for the database. This showed me how all the different parts of a software system work together.

Beyond the technical skills, I also gained crucial professional skills. I learned how important teamwork is, how to communicate clearly with others, and how to solve problems when things don't go as planned. I now understand that being a good developer is not just about writing code; it's also about working with people and being a good problem-solver.

In conclusion, this field attachment was not just a requirement for my studies; it was a major step in my personal and professional growth. It gave me a clear picture of what a career in software development is like and has fully prepared me for the challenges and opportunities in the technology world.

**3.3 REFERENCES**

This field project was built upon the knowledge and technical documentation of various sources. The primary references are as follows:

1. **Academic Source:**
   * S.L. Pfleeger and J.M. Atlee, *Software Engineering: Theory and Practice*, 4th ed., Upper Saddle River, NJ: Prentice Hall, 2010.
2. **Technical Documentation:**
   * **Java:**
     + Oracle. (n.d.). *Java Platform, Standard Edition Documentation*. Retrieved from <https://docs.oracle.com/en/java/javase/>.
   * **PostgreSQL:**
     + PostgreSQL Global Development Group. (n.d.). *PostgreSQL Documentation*. Retrieved from <https://www.postgresql.org/docs/>.
   * **React:**
     + React. (n.d.). *React Documentation*. Retrieved from <https://react.dev/>.
   * **TypeScript:**
     + Microsoft. (n.d.). *TypeScript Documentation*. Retrieved from <https://www.typescriptlang.org/docs/>.
   * **Vite:**
     + Vite. (n.d.). *Vite Documentation*. Retrieved from <https://vitejs.dev/>.
3. **Online Resources:**
   * (Specific articles, tutorials, or blog posts you used)
   * (Specific GitHub repositories you referenced for code examples)

This list provides a template for how to format the references section. Be sure to fill in the exact sources you used during your project's research and development.

**APPENDICES:**

**FRONT END VIEW:**

1. DASHBOARD VIEW

A screenshot of a computer

AI-generated content may be incorrect.

1. COLLEGES VIEW

**A screenshot of a computer

AI-generated content may be incorrect.**

1. DEPARTMENTS VIEW

A screenshot of a computer

AI-generated content may be incorrect.

1. PROGRAMS VIEW

A screenshot of a computer

AI-generated content may be incorrect.

1. COURSES VIEW

A screenshot of a computer

AI-generated content may be incorrect.

1. SEMISTERS VIEW:

A screenshot of a computer

AI-generated content may be incorrect.

1. CURRICULUM VIEW

A screenshot of a computer

AI-generated content may be incorrect.

1. HISTORY VIEW

A screenshot of a computer

AI-generated content may be incorrect.

1. SETTING VIEW:

A screenshot of a computer

AI-generated content may be incorrect.

1. SOURCE OF CODE VIEW:

A computer screen shot of a computer screen

AI-generated content may be incorrect.