**THE UNIVERSITY OF DODOMA**

**THE COLLEGE OF INFORMATICS AND VIRTUAL EDUCATION**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (CSE)**

INDUSTRIAL TRAINING REPORT

**ORGANISATION NAME: TELESOFT ENTERPRISE LTD**

**DISTRICT: DODOMA**

**REGION: DODOMA**

**BY**

**Student name:** MELKIZEDEKI YUSUPH MAGOTA

**Student regno:** T21-03-08644

**Course code:** CS 331

**Degree program:** BSc - SE

**Submitted to:** MR. FERUZI HASSAN

**Academic year:** 2023/2024.

TABLE OF CONTENTS

[LIST OF FIGURES vi](#_Toc182814732)

[ACKNOWLEDGEMENT vii](#_Toc182814733)

[SUMMARY viii](#_Toc182814734)

[CHAPTER ONE 1](#_Toc182814735)

[INTRODUCTION 1](#_Toc182814736)

[1. React Development: 1](#_Toc182814737)

[2. Tailwind CSS: 1](#_Toc182814738)

[3. GraphQL 1](#_Toc182814739)

[4. Figma: 1](#_Toc182814740)

[5. Computer Maintenance: 1](#_Toc182814741)

[CHAPTER TWO 2](#_Toc182814742)

[ABOUT THE COMPANY 2](#_Toc182814743)

[Background 2](#_Toc182814744)

[Goals: 2](#_Toc182814745)

[Core Values 2](#_Toc182814746)

[Activities Performed by the Organization and Experience 2](#_Toc182814747)

[Organization Structure 3](#_Toc182814748)

[CHAPTER THREE 4](#_Toc182814749)

[REACT 4](#_Toc182814750)

[JavaScript Concepts and React Fundamentals: 4](#_Toc182814751)

[Rendering Components and React Structure: 4](#_Toc182814752)

[State Management and React Lifecycle: 5](#_Toc182814753)

[Hands-On Projects: 5](#_Toc182814754)

[CHAPTER FOUR 6](#_Toc182814755)

[TAILWIND CSS: STYLING REACT APPLICATONS 6](#_Toc182814756)

[Introduction to Tailwind CSS: 6](#_Toc182814757)

[Responsive Design: 6](#_Toc182814758)

[Styling Components with Tailwind CSS: 6](#_Toc182814759)

[Working with Forms: 7](#_Toc182814760)

[Building Layouts with Tailwind CSS: 7](#_Toc182814761)

[Tailwind CSS and React Integration: 7](#_Toc182814762)

[Challenges and Solutions: 7](#_Toc182814763)

[Hands-On Projects Using Tailwind CSS: 7](#_Toc182814764)

[CHAPTER FIVE 9](#_Toc182814765)

[GRAPHQL: QUERYING AND MUTATING DATA IN REACT 9](#_Toc182814766)

[Introduction to GraphQL: 9](#_Toc182814767)

[Integrating GraphQL with React: 9](#_Toc182814768)

[State Management with GraphQL: 9](#_Toc182814769)

[Hands-On GraphQL Projects: 10](#_Toc182814770)

[GraphQL with Django: 10](#_Toc182814771)

[Challenges and Solutions: 10](#_Toc182814772)

[CHAPTER SIX 11](#_Toc182814773)

[CHAPTER SIX: BUILDING THE IPT SYSTEM WITH REACT, TAILWIND CSS, GRAPHQL, DJANGO, AND JWT AUTHETICATION 11](#_Toc182814774)

[Project Overview 11](#_Toc182814775)

[Backend Development with Django and GraphQL 11](#_Toc182814776)

[GraphQL with Django 11](#_Toc182814777)

[JWT Authentication 12](#_Toc182814778)

[Frontend Development with React, GraphQL, and JWT Handling 13](#_Toc182814779)

[Protecting Routes with JWT in React 14](#_Toc182814780)

[Results and Achievements with JWT Integration 15](#_Toc182814781)

[CHAPTER SEVEN 16](#_Toc182814782)

[BASIC FIGMA CONCEPTS 16](#_Toc182814783)

[Introduction to Figma 16](#_Toc182814784)

[Understanding the Figma Interface 16](#_Toc182814785)

[Frames and Artboards 16](#_Toc182814786)

[Shapes, Text, and Images 16](#_Toc182814787)

[CHAPTER EIGHT 18](#_Toc182814788)

[COMPUTER MAINTAINANCE AND TRANSCRIPTING 18](#_Toc182814789)

[Computer Maintenance 18](#_Toc182814790)

[Transcription Experience 19](#_Toc182814791)

[SKILLS ACQUIRED DURING FIELDWORK 21](#_Toc182814792)

[Frontend Development 21](#_Toc182814793)

[Backend Development with Django 21](#_Toc182814794)

[GraphQL API Integration 21](#_Toc182814795)

[Design with Figma 21](#_Toc182814796)

[Transcription and Documentation 21](#_Toc182814797)

[Computer Maintenance 21](#_Toc182814798)

[CONCLUSION AND RECOMMENDATIONS 21](#_Toc182814799)

[CONCLUSION 21](#_Toc182814800)

[RECOMMENDATIONS 21](#_Toc182814801)

[REFERENCES 22](#_Toc182814802)

[APPENDICES 22](#_Toc182814803)

# LIST OF ABBREVIATION

**IPT** Industrial Practical Training

**UDOM** University of Dodoma.

**CSS** Cascading Style Sheet

**UI** User Interface.

**UX** User Experience.

**ICT** Information and Computer Technology

**JWT** JSON Web Token  
**BRELA** Business Registrations and Licensing Agency  
**HTML** HyperText Markup Language  
**JSX** JavaScript XML  
**DOM** Document Object Model  
**QR** Quick Response  
**API** Application Programming Interface  
**JSON** JavaScript Object Notation  
**USB** Universal Serial Bus  
**DVD** Digital Versatile Disc  
**BIOS** Basic Input/Output System  
**UEFI** Unified Extensible Firmware Interface  
**CPU** Central Processing Unit  
**GPU** Graphics Processing Unit  
**RAM** Random Access Memory  
**PC** Personal Computer

# LIST OF FIGURES

[Figure 1 Telesoft Organization structure 3](#_Toc182487793)

[Figure 2 Showing graphQL query for Logbook details 12](#_Toc182487794)

[Figure 3 Showing JWT authentication setting 12](#_Toc182487795)

[Figure 4 Showing the login system interface 13](#_Toc182487796)

[Figure 5 checking the token to authenticate the user 13](#_Toc182487797)

[Figure 6 Showing the student dashboard 14](#_Toc182487798)

[Figure 7 Protecting Routes with JWT 14](#_Toc182487799)

[Figure 8 Showing the code for managing the session on logged in user 15](#_Toc182487800)

[Figure 9 showing example of designed page cards on phone layout 17](#_Toc182487801)

[Figure 10 Showing example of texts that are Tran scripted 20](#_Toc182487802)

# ACKNOWLEDGEMENT

We thank God for giving us this opportunity to study at the University of Dodoma and participate in this industrial training. Through this experience, we have learned many valuable skills and gained insights that will be helpful for our future careers and our role in contributing to our country's development. We extend our sincere gratitude to UDOM’s leadership for allowing us to be part of this training program.

We are also deeply grateful to our supervisors and instructors for their trust, guidance, support, and generosity in sharing their knowledge. Their provision of sufficient learning materials and a conducive learning environment greatly contributed to our understanding and comfort throughout the program. May God bless each of you abundantly.

In addition, we would like to express our appreciation to all the members of the industrial training program for their camaraderie, support, and contributions, which enriched every moment of our learning journey. Although it is impossible to thank everyone individually, we are especially thankful for the encouragement and support from our friends, relatives, and beloved parents. May God grant them health and happiness.

# SUMMARY

This report provides an overview of the skills and knowledge I gained during the IPT – 3 period for the academic year 2023/2024 conducted at Telesoft Enterprises Limited. My training focused on key areas in modern web development and other essential IT skills.

Primarily, I studied React and Tailwind CSS, gaining practical experience in creating responsive and dynamic user interfaces. I also explored GraphQL, learning how to efficiently query and manage data for frontend applications. Additionally, I acquired basic concepts of Figma for UI/UX design, which enhanced my ability to create and visualize user-friendly interfaces.

Apart from web development, I delved into computer maintenance, where I learned about different computer components, their functions, potential issues that could arise from malfunctioning parts, and basic maintenance techniques. I also practiced transcription as a valuable skill for accurately capturing spoken content into written format.

This report provides detailed insights into each of these areas, reflecting the practical applications and theoretical knowledge I acquired throughout the training perio

# CHAPTER ONE

## INTRODUCTION

During my training period at Telesoft Enterprises, I was able to engage in various tasks that enhanced my technical and practical skills. The training program involved working on projects that utilized React, Tailwind CSS, Django, GraphQL, Figma, and computer maintenance. Some of the key tasks I performed included:

1. React Development: I studied advanced JavaScript concepts, including array methods, asynchronous programming, and state management. I also worked on creating dynamic and interactive applications using React and integrated them with a backend built on Django and GraphQL for efficient data handling.
2. Tailwind CSS: I learned how to style responsive web interfaces and create flexible layouts using Tailwind CSS.
3. GraphQL: I gained experience in working with GraphQL for querying and mutating data, particularly integrating JWT-based authentication for secure data transactions.
4. Figma: I explored designing user interfaces using Figma, practicing creating prototypes, wireframes, and improving UI/UX design skills.
5. Computer Maintenance: I also performed regular computer maintenance tasks, including software installation, system cleaning using blowers, and troubleshooting common issues.

Through this training, I was able to apply my theoretical knowledge in practical scenarios, developing skills that are crucial for future professional endeavors in web development, UI/UX design, and IT maintenance.

# CHAPTER TWO

ABOUT THE COMPANY  
Telesoft Enterprises Ltd. is a technology company headquartered in Dodoma, the capital city of Tanzania, specializing in consumer software and hardware solutions, including design, installation, and maintenance. Telesoft is dedicated to leveraging ICT knowledge to simplify life, not only within Tanzania but also globally. Despite challenges in the employment sector, Telesoft is committed to fostering a mindset of innovation and entrepreneurship, aiming to be a creator of opportunities rather than merely seeking them.

Background  
Founded on August 4, 2019, by John Furaha John and Baraka Magulumali Malongo, Telesoft was officially registered as Telesoft Enterprises Ltd. in 2020 under the Business Registration and Licensing Agency (BRELA) in Tanzania.

* **Motto**: *Simplifying Life*
* **Vision**: To become a leading enterprise in ICT innovation and service delivery.
* **Mission**: To excel in ICT research, innovation, creativity, sales, and marketing, with a strong focus on quality customer and stakeholder management.

### Goals:

* Conduct research on ICT challenges faced by communities.
* Utilize ICT knowledge to address these challenges.
* Develop high-quality software solutions (web and mobile applications).
* Offer ICT consultancy, computer maintenance, and training services.
* Provide network installation and graphic design services.

### Core Values

Integrity, Innovation, Creativity, and Commitment

### Activities Performed by the Organization and Experience

Telesoft Enterprises Ltd offers a compressive range of ICT services, including:

* Network design, installation, and management
* Graphic design and printing
* Computer hardware troubleshooting, maintenance, and training
* Software installation, troubleshooting, and consultancy
* Mobile application development
* Website development: designing and building responsive, user-friendly websites for various clients

### Organization Structure

The organization structure of Telesoft can be summarized as shown in the figure below

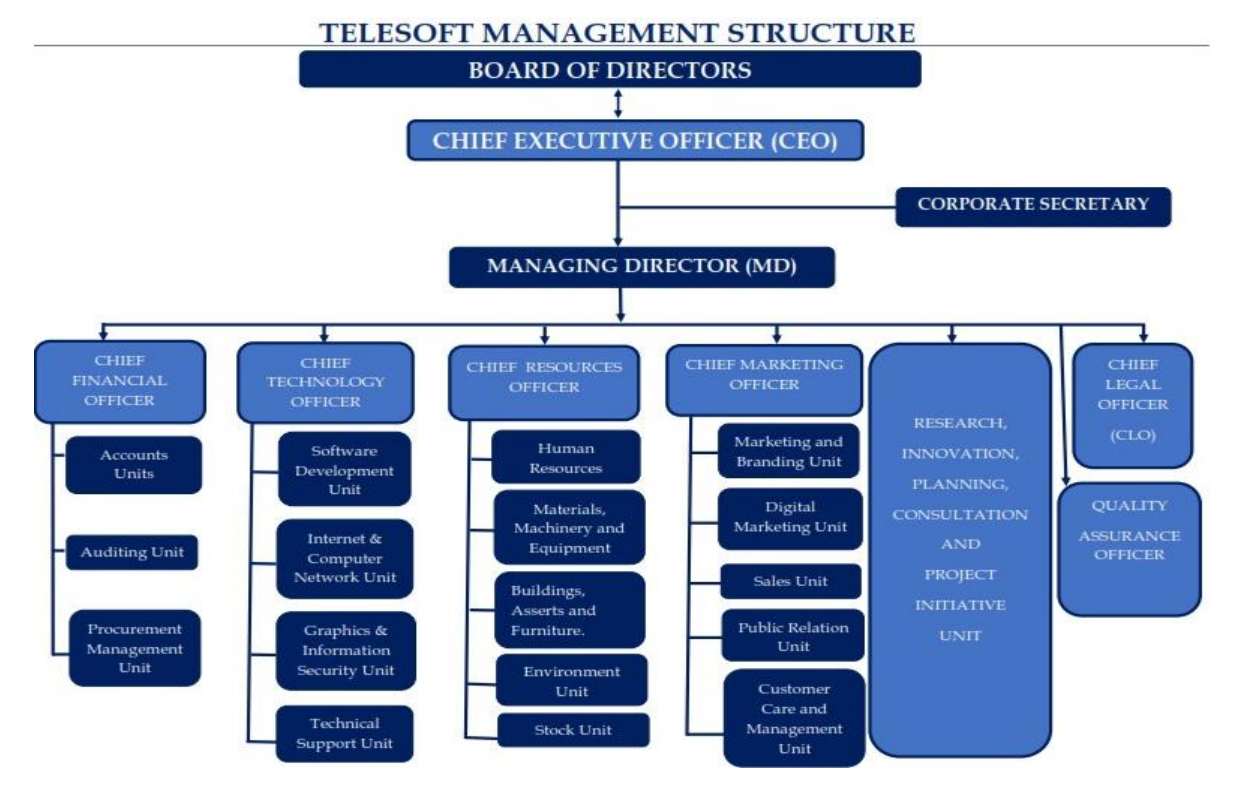


Figure 1 Telesoft Organization structure

# 

# CHAPTER THREE

## REACT

I focused on refining my knowledge and skills through a combination of theoretical learning and hands-on practice. Below is a detailed overview of the concepts and projects I studied:

### JavaScript Concepts and React Fundamentals:

I deepened my understanding of JavaScript features that are crucial for React development, such as:

* **Short-circuit and Logical Operators**: I explored how these operators help control flow and handle conditions in a more concise manner.
* **Nullish Coalescing and Optional Chaining**: These features helped me safely handle null or undefined values, improving the robustness of my code.
* **Array Methods**: I worked with map, filter, reduce, and sort to manipulate data in immutable arrays. These methods are essential for working with data in React components.
* **Asynchronous Methods**: I learned how to work with fetch and await to handle API calls asynchronously, which is a key skill when dealing with external data in React applications.

These JavaScript concepts are foundational for working with React efficiently, especially in more advanced scenarios where dynamic data and complex state management are involved.

### Rendering Components and React Structure:

* **Components as Building Blocks**: I studied how components form the foundation of React applications. I practiced creating and reusing components, making my code modular and maintainable.
* **Rendering the Root Component**: I learned how to render the root component in react application and how to use React Strict Mode to identify potential problems in my code
* **JSX and Props**: I delved into the rules of JSX, which is a syntax extension for JavaScript that allows writing HTML-like code in JavaScript. I explored passing and receiving props, understanding their immutability, and how data flows through components.
* **Conditional Rendering**: I practiced using logical operators like && and ternary operators to conditionally render elements, making components more dynamic based on state and props.
* **React Fragments**: I studied how React Fragments allow grouping multiple elements without adding extra nodes to the DOM, making my component structure cleaner.

### State Management and React Lifecycle:

* **State Mechanics**: I learned the importance of state in React and how to manage and update state using useState. Understanding how to manipulate state based on the current state is essential for handling dynamic user interfaces.
* **Controlled Components**: I studied how to create controlled components in React, where form inputs are controlled by the state, ensuring predictable and consistent behavior.
* **State Management Tools**: I explored React development tools to debug and optimize my applications.
* **Local vs Global State**: I learned the differences between local and global state, which is key to managing data at different levels of the component hierarchy.
* **Lifting State Up**: I practiced lifting state up to share data between components and ensure that they remain in sync.

### Hands-On Projects:

To solidify my understanding of React concepts, I worked on several hands-on projects:

1. **Accordion, Tabs, Modal, and Image Slider**: These simple interactive components helped me practice core React concepts and state management.
2. **Star-Rating App and Random Color Generator**: I added dark and light theme support, making the app more dynamic and user-friendly. I also implemented a random color generator for additional interactivity.
3. **GitHub Profile Finder and QR Code Generator**: These projects allowed me to work with external APIs and build applications that interact with real-world data.
4. **Scroll Indicators and Scroll-to-Top/Bottom Features**: I worked on enhancing user experience with scroll-based features.
5. **Custom Hooks**: I created custom hooks like useFetch to fetch data asynchronously and useWindowResize for responsive layouts.
6. **Weather App and Tic-Tac-Toe Game**: These apps provided an opportunity to work with more complex state management and implement interactive user interfaces.
7. **Shopping Cart with Online API Integration**: This project gave me a real-world experience of building a dynamic app that integrates with external APIs for product information.

# CHAPTER FOUR

## TAILWIND CSS: STYLING REACT APPLICATONS

In addition to React, I focused on learning and implementing **Tailwind CSS**, a utility-first CSS framework, to style my applications. Tailwind CSS allows for highly customizable designs without writing custom CSS. Below are the main concepts I explored:

### Introduction to Tailwind CSS:

* **Utility-First Approach**: I learned that Tailwind CSS is based on utility classes, where I can directly apply styles like text-center, bg-blue-500, or p-4 to elements in HTML or JSX. This makes styling quicker and more maintainable, as it avoids writing custom CSS for each component.
* **Customization**: I explored how to customize the default Tailwind CSS configuration, adjusting colors, spacing, and breakpoints to fit the design requirements of my React applications.

### Responsive Design:

* **Responsive Utilities**: Tailwind CSS uses breakpoints for building responsive layouts. I learned how to apply classes like sm:, md:, lg:, and xl: to make elements responsive based on different screen sizes. For example, md:w-1/2 sets an element to half width on medium screens and larger.
* **Flexbox and Grid Layouts**: I practiced using Tailwind's **Flexbox** and **Grid** utilities to create complex layouts, such as sidebars, content areas, and grids for displaying products or images. For example, using flex, flex-row, justify-between, and items-center to align content efficiently.

### Styling Components with Tailwind CSS:

* **Text and Typography**: I applied Tailwind’s text utilities like text-xl, font-bold, text-center, and text-gray-800 to style text and headings within React components.
* **Spacing and Layout**: I used spacing utilities (p-4, m-4, pt-2, space-x-4) to manage padding and margin, ensuring consistent spacing between elements.
* **Backgrounds and Borders**: I explored background utilities like bg-blue-500, bg-gradient-to-r, and border classes like border-2, border-gray-200 to create visually appealing backgrounds and borders for components.
* **Hover and Focus States**: I studied how to apply interactive styles, such as hover:bg-blue-400 and focus:ring-2, to improve the user experience and make components interactive.

### Working with Forms:

* **Form Inputs**: I used Tailwind’s form input classes like form-input, input, and rounded-lg to style text fields, checkboxes, and radio buttons. These utilities helped me create clean, user-friendly forms in React.
* **Form Validation**: Tailwind helped me style error messages and focus states in forms, ensuring that form validation was visually clear to users.

### Building Layouts with Tailwind CSS:

* **Sidebar and Navbar**: I built flexible sidebars and top navigation bars using Tailwind's grid and flex utilities. For instance, I used w-1/4 for fixed sidebar width and flex-1 for the main content area, ensuring the layout adjusted well on different screen sizes.
* **Card Layouts**: I implemented card designs using bg-white, shadow-lg, rounded-lg, and p-6 classes to display product details and other content, ensuring the layout was visually appealing and well-organized.

### Tailwind CSS and React Integration:

* **Dynamic Classes**: I explored using Tailwind with React by dynamically applying classes based on component state or props. This allowed me to create dynamic, interactive user interfaces, such as toggling a class when a modal is open or switching themes between light and dark modes.
* **Tailwind with JSX**: I became proficient at writing Tailwind CSS classes directly in JSX, understanding the best practices for applying utility classes without cluttering the component code.

### Challenges and Solutions:

* **Class Management**: Managing long lists of Tailwind classes in JSX components was challenging, so I learned to use **classnames** and other utility libraries to handle conditional classes more efficiently.
* **Customization**: In some cases, I needed more customized styles. I explored extending Tailwind's default configuration by modifying the tailwind.config.js file to include custom colors, fonts, and breakpoints, ensuring the designs were unique to the project’s needs.

### Hands-On Projects Using Tailwind CSS:

I applied Tailwind CSS in several projects to reinforce my learning:

1. **Product Grid**: I used Tailwind’s grid layout and responsive classes to display product listings in a grid format, adjusting the number of columns based on screen size.
2. **Modal and Pop-ups**: I created reusable modal components with Tailwind’s utilities for alignment, padding, and transitions.
3. **Card Layouts for Blogs**: I built blog post cards with images, titles, and descriptions, styled using Tailwind’s card utilities and hover effects.
4. **Navbar and Sidebar**: I styled responsive navigation bars and sidebars that collapsed into a hamburger menu on smaller screens.
5. **Responsive Form**: I created a responsive contact form with Tailwind’s form utilities, adding responsiveness for different screen sizes.

# CHAPTER FIVE

## GRAPHQL: QUERYING AND MUTATING DATA IN REACT

Alongside my work with React and Tailwind CSS, I focused on learning **GraphQL**, a powerful query language for APIs, and how to integrate it with React for efficient data fetching and mutation. Below are the key concepts and practices I explored:

### Introduction to GraphQL:

* **Basic GraphQL Concepts**: I began by learning the core concepts of GraphQL, including **queries**, **mutations**. Queries are used to fetch data, mutations are used to modify data, and subscriptions enable real-time updates.
* **GraphQL Schema**: I studied how to define and work with GraphQL schemas. A schema defines the types of data that can be queried or mutated, including fields, relationships, and constraints. This is the foundation for all GraphQL operations.

### Integrating GraphQL with React:

* **Apollo Client**: I explored the **Apollo Client**, a popular tool for managing data in React applications. I learned how to set up Apollo Client and integrate it with React components to fetch data and handle mutations.
  + I used the ApolloProvider to wrap my React application, enabling Apollo Client to manage the state of my GraphQL queries.
  + I practiced using **Apollo’s useQuery hook** to fetch data and **useMutation hook** for performing mutations in my React components.
* **Handling Queries**: I learned how to send queries to a GraphQL server to retrieve specific pieces of data.
* **Handling Mutations**: I explored how to modify data using GraphQL mutations. For instance, I worked on adding and updating todos in my application using mutations:

### State Management with GraphQL:

* **Local State Management**: I studied how to manage local state with GraphQL, leveraging Apollo Client’s built-in capabilities to store and update client-side data.
* **Caching**: One of the most powerful features of Apollo Client is its **cache**. I learned how Apollo Client automatically caches query responses, which improves performance by preventing redundant network requests.
* **Optimistic UI**: I explored how to use **optimistic UI** to provide a smoother user experience. This approach updates the UI immediately with the expected result of a mutation, even before the server responds.

### Hands-On GraphQL Projects:

I applied my knowledge of GraphQL through several hands-on projects in React:

1. **Todo Application**: I built a **Todo List** application that allowed users to add, update, and delete tasks using GraphQL mutations and queries. This helped me solidify the process of sending queries and mutations in React components.

### GraphQL with Django:

As part of my learning, I integrated **GraphQL with Django** using the **Graphene** library. This allowed me to set up a GraphQL API on the backend and connect it with my React frontend.

* I set up the Django backend with **Graphene** to handle GraphQL queries and mutations.
* I defined GraphQL schemas for my models in Django, creating types and resolvers to manage data.

### Challenges and Solutions:

* **Setting Up Apollo Client**: Initially, I encountered difficulties setting up Apollo Client in a React project. I resolved this by following detailed tutorials and reading the Apollo documentation to correctly set up and configure the client.
* **GraphQL Query Structure**: Understanding how to structure complex queries with multiple fields, nested objects, and relationships was challenging at first. However, through practice and debugging, I became more proficient in constructing optimized queries.
* **Error Management**: Handling GraphQL errors in a structured manner took some time, especially with mutations. I improved my approach by using the error handling patterns provided by Apollo Client.

# 

# CHAPTER SIX

## CHAPTER SIX: BUILDING THE IPT SYSTEM WITH REACT, TAILWIND CSS, GRAPHQL, DJANGO, AND JWT AUTHETICATION

In this chapter, I will discuss how I successfully built the **IPT System** by integrating **React** for the frontend, **Tailwind CSS** for styling, **GraphQL** for data querying, **Django** as the backend technology to handle business logic, and **JWT (JSON Web Tokens)** for secure user authentication.

### Project Overview

The **IPT System** is designed to manage and track information effectively. Users can interact with records through a seamless interface, performing tasks such as viewing, creating, and updating records. The system leverages **React** for dynamic user interactions, **Tailwind CSS** for efficient styling, **GraphQL** for efficient data fetching, **Django** to handle the backend logic, and **JWT** to manage authentication securely.

### Backend Development with Django and GraphQL

**Django** served as the backend framework, providing a secure and scalable solution for managing the database and handling requests. I chose Django for its simplicity and the ability to integrate well with **GraphQL** and **JWT** though I had the prior knowledge with it.

### GraphQL with Django

For querying and manipulating data, I integrated **GraphQL** with **Django** using **Graphene-Django**. This setup allowed me to create efficient and flexible data fetching mechanisms while maintaining a structured and scalable backend:

* **GraphQL Queries and Mutations**: I used **GraphQL queries** to fetch data and **mutations** for creating and updating records. This approach enables precise and efficient data fetching, allowing the frontend to request only the necessary data.
* **Protected GraphQL Endpoints**: All GraphQL queries and mutations were protected using **JWT authentication**. The **JWT token** was included in the request headers, ensuring that only authenticated users could access and modify the data.

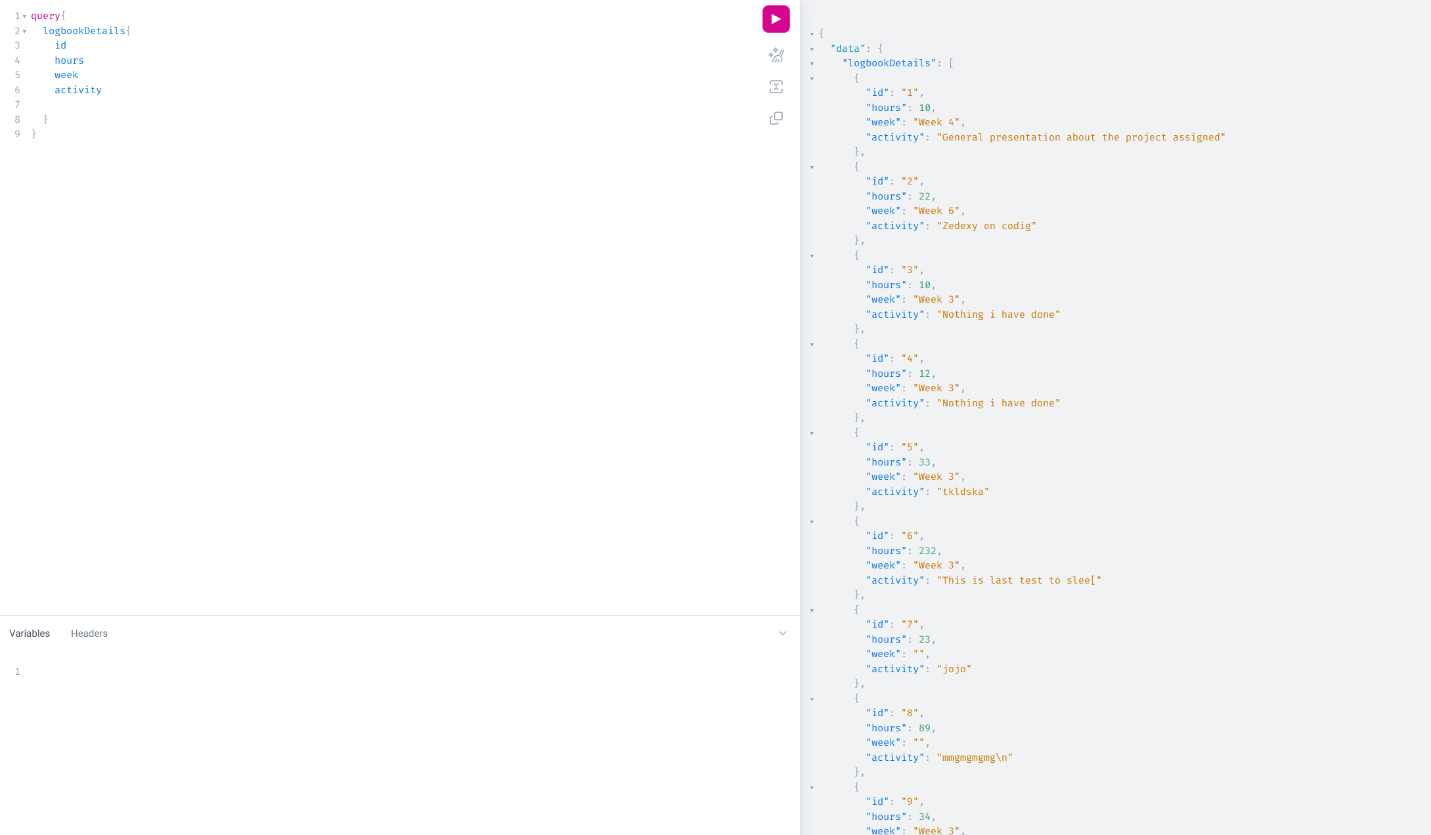


Figure 2 Showing graphQL query for Logbook details

### JWT Authentication

To secure the system, I used **JWT** for authentication. JWT is an open standard for securely transmitting information between parties as a JSON object. It is commonly used for managing user sessions and authorizing API requests. Here's how I implemented it:

* **JWT Token Generation**: The backend generates a **JWT token** upon successful login. The token is sent to the frontend and used for authenticating subsequent requests.
* **JWT Token Validation**: On the backend, the token is validated before processing any protected GraphQL request. This ensures that only valid, authenticated users can access specific data or perform certain actions.

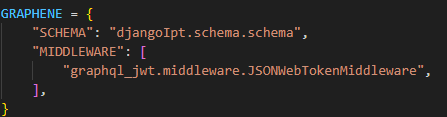


Figure 3 Showing JWT authentication setting

### Frontend Development with React, GraphQL, and JWT Handling

On the frontend, **React** handled the user interface, while **JWT** was used to manage user authentication. **GraphQL** was used to fetch and manipulate data from the backend efficiently.

**JWT Handling in React**

I implemented **JWT authentication** on the React frontend to securely manage user sessions:

* **Login Process**: When a user logs in, their credentials are sent to the backend via a **GraphQL mutation**. Upon successful authentication, the backend sends back the JWT token, which is then stored in **localStorage**.

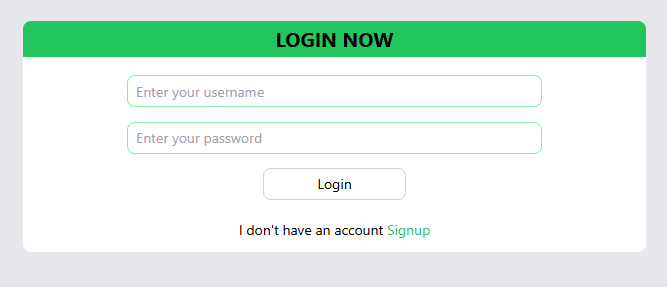


Figure 4 Showing the login system interface



Figure 5 checking the token to authenticate the user

* **Storing the JWT Token**: The token is stored in **localStorage** to persist the user session. Every subsequent GraphQL request includes the token in the **Authorization** header to authenticate the user.
* **Logout Process**: When the user logs out, the token is removed from **localStorage**, and the user is redirected to the login page.

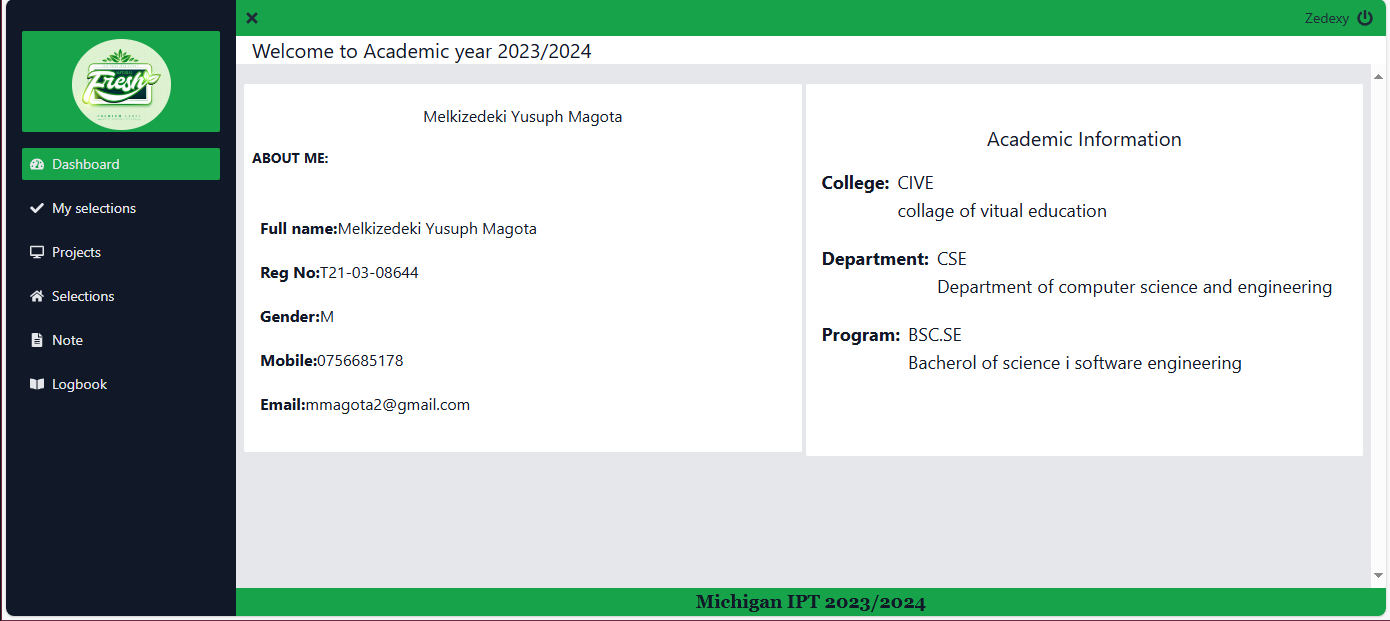


Figure 6 Showing the student dashboard

export default Login;

### Protecting Routes with JWT in React

To protect certain routes in the application, I used **React Router** and checked whether the JWT token is available in **localStorage**. If the token is not present or invalid, users are redirected to the login page.



Figure 7 Protecting Routes with JWT

### Results and Achievements with JWT Integration

By integrating **JWT authentication** with **GraphQL**, I was able to:

* **Secure the Application**: Authentication and authorization were handled securely using JWT, ensuring that only authenticated users could access protected GraphQL resources.
* **Manage User Sessions**: The frontend React app effectively managed user sessions, allowing users to stay logged in across page reloads.



Figure 8 Showing the code for managing the session on logged in user

* **Efficient Communication**: JWT tokens allowed for stateless communication between the frontend and backend, which is a lightweight and scalable solution for managing authentication.

The integration of **JWT** in the **IPT System** ensures that the application is secure, user-friendly, and efficient, while providing a seamless authentication flow.

# CHAPTER SEVEN

## BASIC FIGMA CONCEPTS

### Introduction to Figma

* Briefly introduce Figma as a versatile, cloud-based design tool widely used for UI/UX design and prototyping. Highlight that it supports collaborative workflows, allowing multiple team members to work on the same design file in real time.

### Understanding the Figma Interface

* **Canvas**: Describe the main workspace where all designs are created. Mention how the infinite canvas allows for creating multiple artboards and organizing various screens of the application.
* **Layers Panel**: Discuss the importance of the Layers panel in organizing elements, making it easy to select and manage different parts of the design.
* **Properties Panel**: Explain how the Properties panel provides settings to adjust the properties of selected objects, like colors, dimensions, alignment, and effects.

### Frames and Artboards

* **Frames**: Describe frames as containers for UI elements, used to organize layouts and define screen sizes. Mention how frames act as artboards and can be adjusted for different screen resolutions.
* **Artboards**: Explain that artboards in Figma are synonymous with frames when setting up designs for specific devices (desktop, tablet, mobile).

### Shapes, Text, and Images

* **Shapes**: Describe the basic shape tools (rectangles, circles, lines) used to create UI components like buttons, icons, and backgrounds.
* **Text**: Explain how to add text elements, adjust typography (font, size, weight), and use alignment and spacing to improve readability.
* **Images**: Discuss how images can be imported and managed within frames, resized, and styled to suit the design.

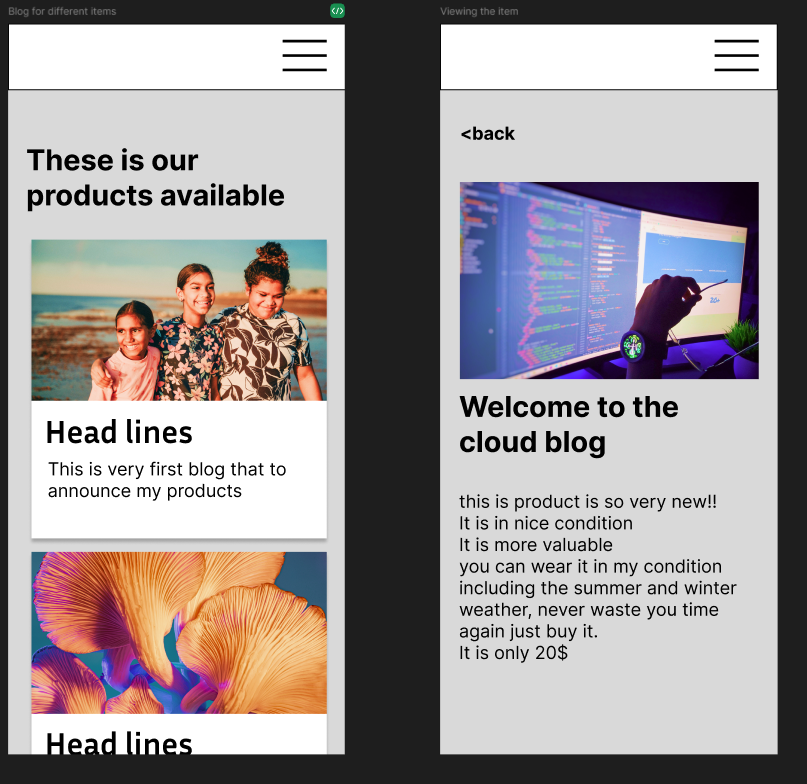


Figure 9 showing example of designed page cards on phone layout

# CHAPTER EIGHT

## COMPUTER MAINTAINANCE AND TRANSCRIPTING

### Computer Maintenance

#### Installing Windows Operating System

* **Preparing for Installation**: Outline the steps to ensure a smooth installation process, such as:
  + Backing up important files if re-installing.
  + Checking system requirements to ensure compatibility with the Windows version.
  + Creating a bootable USB drive or using a Windows installation disc.
* **Installation Steps**:
  + **Booting from USB/DVD**: Describe how to access the BIOS or UEFI settings to set the USB or DVD as the primary boot device.
  + **Selecting Install Options**: Go through the steps of choosing installation preferences (language, region, and edition) and the installation type (upgrade or clean installation).
  + **Partitioning the Hard Drive**: Explain how to select the drive partition for Windows installation, either creating new partitions or using existing ones.
  + **Completing Installation**: Describe the process of configuring Windows settings like account creation, network setup, and basic personalization.
* **Post-Installation Setup**:
  + **Drivers**: Emphasize installing essential drivers for components like graphics, sound, and network adapters to ensure all hardware functions properly.
  + **Updates**: Mention checking for and installing Windows updates to keep the OS secure and up-to-date.

#### Installing Essential Software

* **Anti-Virus Software**: Highlight the importance of installing antivirus software to protect the system from malware and viruses.
* **Productivity Software**: Describe installing software like Microsoft Office or Google Workspace to enable document editing, spreadsheet management, and presentations.
* **Web Browsers**: Mention installing updated web browsers like Chrome, Firefox, or Edge for internet access.
* **Utilities and Drivers**: Discuss installing software utilities for tasks like PDF reading, media playback, and compression tools (e.g., WinRAR or 7-Zip).

#### Cleaning the PC Using Blowers

* **Preparation for Cleaning**:
  + **Power Off and Unplug**: Stress the importance of disconnecting the PC from power sources and removing all cables.
  + **Gathering Cleaning Materials**: List items such as a blower, microfiber cloth, and isopropyl alcohol for safely cleaning components.
* **Using a Blower for Dust Removal**:
  + **Opening the PC Case**: Explain how to carefully remove the case panel to access internal components.
  + **Target Areas for Blowing**: Describe the areas most prone to dust buildup, such as:
    - **Fans and Heatsinks**: Mention blowing dust out of CPU and GPU fans and heatsinks to improve cooling.
    - **Power Supply Unit (PSU)**: Explain how to clean dust out of the PSU vents.
    - **Memory and Expansion Slots**: Briefly mention blowing dust out of RAM and PCI

#### Regular Maintenance Schedule

* **Software Updates**: Recommend checking for software updates regularly, including operating system, drivers, and installed applications.
* **Disk Cleanup and Defragmentation**: Explain how to use built-in tools for cleaning up junk files and defragmenting the hard drive to optimize performance.
* **Physical Cleaning**: Suggest a schedule for using blowers every few months to keep dust levels low, which helps in maintaining the PC's cooling efficiency and longevity.

### Transcription Experience

During my fieldwork, I engaged in several transcription tasks, which involved converting audio recordings into written text. This experience helped me understand the importance of accuracy and attention to detail in documentation. Through this process, I improved my listening skills, learned to handle various accents, and gained insights into effective documentation practices essential for accurate data recording.

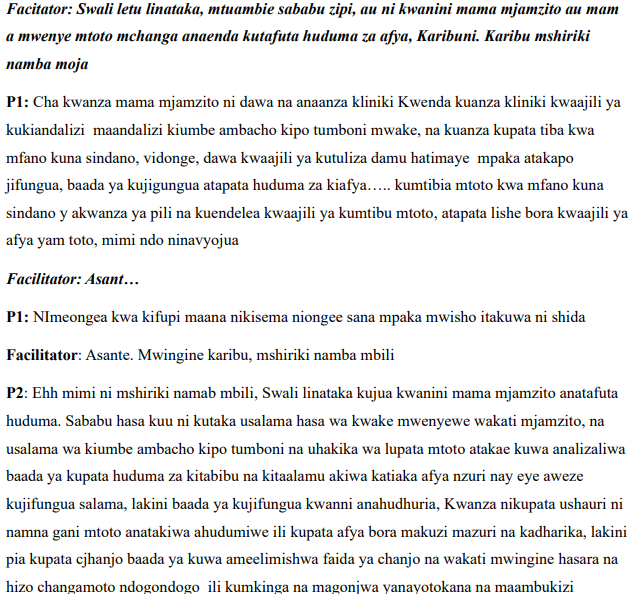


Figure 10 Showing example of texts that are Tran scripted

# SKILLS ACQUIRED DURING FIELDWORK

## Frontend Development

* + **React**: Mastered component-based design, state management, and hooks.
  + **Tailwind CSS**: Built responsive designs and efficient layouts within React.

## Backend Development with Django

* + **Django**: Developed backend architecture, handling models, views, and URLs.
  + **JWT Authentication**: Secured user access with JWT-based login and logout.

## GraphQL API Integration

* + Gained proficiency in querying/mutating data, integrating GraphQL with React.

## Design with Figma

* + Created wireframes, prototypes, and user flows, focusing on UI/UX principles.

## Transcription and Documentation

* + Transcribed recordings accurately and improved technical documentation skills.

## Computer Maintenance

* + Installed Windows OS, essential software, and maintained hardware by cleaning PCs.

# CONCLUSION AND RECOMMENDATIONS

## CONCLUSION

In conclusion, my fieldwork experience was highly enriching, providing me with valuable skills in frontend and backend development, UI/UX design, and computer maintenance. Despite challenges, I was able to accomplish key tasks and apply my knowledge practically. I am confident that the skills I gained will help me contribute meaningfully to my community and further my career.

## RECOMMENDATIONS

In my recommendation, I encourage the university to adopt a more practical approach to teaching, as hands-on experience plays a crucial role in developing problem-solving skills. By providing students with opportunities to apply theoretical knowledge in real-world scenarios, they will not only deepen their understanding but also enhance their technical abilities. A practical-based curriculum will better equip students to meet industry demands and contribute effectively to their communities.

# REFERENCES

1. **React Documentation**. (n.d.). *React – A JavaScript library for building user interfaces*. Retrieved from https://reactjs.org/docs/getting-started.html
2. **Tailwind CSS Documentation**. (n.d.). *Tailwind CSS – Utility-first CSS framework*. Retrieved from <https://tailwindcss.com/docs>
3. **Django Documentation**. (n.d.). *Django documentation*. Retrieved from <https://docs.djangoproject.com/en/stable/>
4. **GraphQL Documentation**. (n.d.). *GraphQL – A query language for your API*. Retrieved from https://graphql.org/learn/
5. **JWT.io Documentation**. (n.d.). *JSON Web Tokens*. Retrieved from https://jwt.io/introduction/
6. **Figma Documentation**. (n.d.). *Figma – The collaborative interface design tool*. Retrieved from https://www.figma.com/resources/learn/
7. **Computer Maintenance Guide**. (2020). *How to Maintain and Clean Your Computer*. Retrieved from https://www.computerhope.com/issues/ch000291.htm
8. **Technical Transcription**. (n.d.). *Transcription Guide – Tips for Efficient and Accurate Transcription*. Retrieved from https://www.rev.com/blog/transcription-guide

# APPENDICES

* Logbook