DEBRE MARKOS UNIVERSITY

DEBRE MARKOS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COMPUTER ENGINEERING STREAM

BSC PROJECT

TITLE

BUDGET EXPENSE MANAGEMENT SYSTEM FOR DMU INSTITUTE OF TECHNOLOGY

BY

NAME	ID
NATHAN TADESSE---	1419\12
SELAMSEW ALEMU---	1507\12
MEHARI NADEW---	2104\12

ADVISOR: ENEDEG MOLLA (MSC.)

June 28, 2024

Debre Markos, Ethiopia

TITLE: Budget Expense Management System for DMU Institute of Technology

BY

Nathan Tadesse

Selamsew Alemu

Mehari Nadew

A thesis submitted to the Department of Electrical and Computer Engineering, Debre Markos Institute of Technology, Debre Markos University, in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Computer Engineering.

Advisor: Enedeg Molla (MSc.)

June 28, 2024

Debre Markos, Ethiopia

DECLARATION

We undersigned, declare that the thesis comprises my own work. In compliance with internationally accepted practices, I have acknowledged and refereed all materials used in this work

Name	**Signature**	**Date**
Nathan Tadesse	______________	____________
Selamsew Alemu	______________	____________
Mehari Nadew	______________	____________

This project has been submitted for examination with my approval as university advisor.

Name	**Signature**	**Date**
______________	_____________	______________

DEBRE MARKOS UNIVERSITY

DEBRE MARKOS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

THESIS APPROVAL SHEET

Student: ______________________________ ______________ ____________________

Student: ______________________________ ______________ ____________________

Student: ______________________________ ______________ ____________________

Name Signature Date

The following graduate faculty members certify that this student has successfully completed the necessary written thesis and oral presentation for partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Computer Engineering.

Approved By:

Advisor: ______________________________ ______________ __________________

Name Signature Date

External Examiner: ________________ ______________ ________________

Name Signature Date

Internal Examiner: ________________ ______________ ________________

Name Signature Date

Chair Holder: ____________________ ______________ ________________

Name Signature Date

Department head: ____________________ ______________ ________________

Name Signature Date

Acknowledgment

First and for most and above all our biggest thanks would be to Almighty “GOD” because nothing could be possible without his free will and the completion of this project is supported by him.

Secondly, we extend our heartfelt gratitude to Mr. Enedeg Molla (MSc.) our advisor, for his invaluable guidance and support throughout this project. His expertise and encouragement have been instrumental in the successful completion of our work.

Thirdly, we are also profoundly thankful to the management and staff of Budget and finance Directorate of Debre Markos university for their cooperation and assistance and to all those who have contributed to the development of the Debre Markos University Institute of Technology (DMUIT) Budget Expense Management System. Special thanks go to the faculty members and administrative staff of DMUIT for their valuable insights and support throughout the project.

Lastly, we appreciate the collective efforts of our project team and the continuous support from our peers and family, who have been a constant source of motivation and assistance during this journey

Abstract

The DMUIT Budget Expense Management System is designed to streamline and enhance the financial management processes within the institute. This web-based application aims to provide a user-friendly interface for managing budget allocations, expense tracking, and financial reporting.

The system is built upon modern web technologies, utilizing HTML, CSS, and React.js for the front-end, and Node.js with Express for the back-end, ensuring scalability and performance. MongoDB is employed as the database to store and manage financial data securely. Key features include role-based access control to ensure data security, real-time expense tracking, comprehensive reporting functionalities, and intuitive user interfaces tailored to the needs of administrators, department heads, and financial officers.

By implementing the DMUIT Budget Expense Management System, the institute aims to improve transparency, efficiency, and accountability in financial operations, thereby supporting informed decision-making and sustainable financial management practices.

Table of Contents

List of Table

List of Figures

LIST OF ACRONYMS

DMUIT	Debremarkos university Institute of Technology
IBEX	Integrated budget and Expenditure System software
UC	Use Case
HTML	Hyper-Text Markup Language
CSS	Cascaded Style Sheet
UI	User Interfaces
UML	Unified Modeling Language
HTTP	Hyper Text Transfer Protocol
API	Application Programming Interface

CHAPTER ONE

1. INTRODUCTION

Budget management is a critical function in organizations, providing a structured approach to planning and controlling financial resources. It involves the allocation of funds to various activities and departments based on strategic priorities and expected outcomes. Effective budget management ensures that organizations can achieve their financial goals while maintaining fiscal responsibility and transparency.

Budget expense management systems automate and streamline the budgeting process, facilitating efficient allocation, monitoring, and utilization of financial resources. These systems play a crucial role in enhancing organizational efficiency, minimizing financial risks, and supporting informed decision-making by providing real-time insights into expenditure patterns and budget performance.

The DMUIT Budget Expense Management System represents a significant step forward in this endeavor. This comprehensive web-based application is tailored to meet the specific needs of DMUIT by providing a robust platform for budget allocation, expense tracking, and financial reporting. By integrating modern web technologies and secure database management, the system aims to streamline administrative workflows, improve transparency, and empower decision-makers with timely and accurate financial data.

This introduction outlines the rationale behind developing the DMUIT Budget Expense Management System, highlights its key objectives, and provides an overview of the structure and functionalities that will be detailed throughout this documentation. It serves as a foundational understanding of how this project addresses the institute's financial management challenges and contributes to its overarching goals of efficiency, accountability, and excellence.

1.1. Background

Debre Markos University Institute of Technology (DMUIT) is a prominent educational institution committed to fostering innovation and excellence in engineering and technology education. As the institute grows, managing its financial resources efficiently becomes increasingly critical. Traditional methods of budget management and expense tracking have proven to be cumbersome and prone to errors, leading to inefficiencies and a lack of transparency.

To address these challenges, DMUIT has initiated the development of the DMUIT Budget Expense Management System, a web-based application designed to streamline financial operations. This system leverages modern web technologies to provide a comprehensive solution for budget allocation, expense tracking, and financial reporting, ensuring data accuracy, security, and accessibility.

1. 2. Statement of the Problem

The current financial management processes at DMUIT Uses IBEX and Manual Processes, involving spreadsheets and paper-based records. These methods are not only time-consuming but also susceptible to human error, resulting in inaccurate financial data and delayed reporting. This lack of an integrated system hampers the institute's ability to make informed financial decisions, allocate resources effectively, and maintain accountability.

The absence of a centralized budget management system also poses challenges in monitoring and controlling expenses across various departments. This can lead to overspending, misallocation of funds, and difficulties in tracking financial performance against allocated budgets. Therefore, there is a pressing need for an automated, user-friendly system that can address these issues and support the institute's financial management objectives.

1.3. Objective

1.3.1. General Objective

The general objective of this project is to develop and implement the DMUIT Budget Expense Management System, a web-based application designed to improve the efficiency, accuracy, and transparency of financial management processes at Debre Markos University Institute of Technology.

1.3.2. Specific Objectives

- ❖ To design a user-friendly interface for managing budget allocations and expense tracking.
- ❖ To implement role-based access control to ensure data security and integrity.
- ❖ To develop real-time expense tracking and monitoring capabilities.
- ❖ To provide comprehensive financial reporting functionalities for informed decision-making.
- ❖ To integrate the system with existing financial workflows to enhance overall efficiency.

1.4. The Significance of the Project

The implementation of the DMUIT Budget Expense Management System holds significant potential benefits for DMUIT. By automating and centralizing financial management processes, the system will:

- ❖ Enhance the accuracy and reliability of financial data.
- ❖ Reduce the time and effort required for budget management and expense tracking.
- ❖ Improve transparency and accountability in financial operations.
- ❖ Enable timely and informed decision-making through comprehensive reporting.
- ❖ Support the efficient allocation and utilization of financial resources.

1.5. Scope of the project

This study focuses on the design, development, and implementation of the DMUIT Budget Expense Management System. The scope includes:

- ❖ Analysis of current financial management processes at DMUIT.
- ❖ Design and development of the web-based application using HTML, CSS, React.js, Node.js with Express, and MongoDB.
- ❖ Implementation of role-based access control and security features.
- ❖ Integration with existing workflows and systems.
- ❖ Testing and validation of the system's functionality and performance.

1.6. Limitation of the project

However, the study is subject to certain limitations:

- ❖ The project is limited to the financial management processes within DMUIT and does not extend to other departments or external entities.
- ❖ The development and implementation timeline may be constrained by resource availability and institutional schedules.
- ❖ The system does not support languages other than English language.
- ❖ The system cannot help people with visual impairment.
- ❖ The system is accessible when the internet is available.
- ❖ User adoption and training requirements may impact the initial effectiveness of the system.
- ❖ The System may face challenges related to memory usage, particularly when handling large datasets and concurrent user sessions.

CHAPTER TWO

2. LITERATURE REVIEW

Budget and expense management systems are integral components of organizational financial strategies. The ever-evolving landscape of technology has prompted a paradigm shift in how businesses approach these crucial functions. This literature review aims to synthesize existing research, exploring the historical context, technological advancements, benefits, challenges, and future directions in budget and expense management systems.

2.1.Historical Evolution of Budgeting Systems

The historical evolution of budgeting systems provides essential context for understanding the present landscape. Traditional approaches like zero-based budgeting and incremental budgeting have given way to more flexible models [1]. The adoption of technology in budgeting marks a significant shift, allowing for more dynamic and data-driven financial planning.

2.2.Technological Integration in Budget and Expense Management

The integration of technology, particularly Enterprise Resource Planning (ERP) systems and cloud-based solutions, has transformed the efficiency and effectiveness of budget and expense management (Smith & Jones, 2018; Brown et al., 2019) [3].These technological solutions offer real-time accessibility, improved collaboration, and streamlined processes.

2.3.Benefits of Automated Budgeting Systems

Automated budgeting systems have demonstrated numerous advantages. Chen et al. (2020) [4].emphasize the reduction of errors, enhanced data accuracy, and increased overall efficiency. The ability to track expenses in real-time is a key benefit, providing organizations with a more accurate and timelier financial picture (Jones & Williams, 2017). [5]

2.4.Challenges and Limitations

Despite the benefits, challenges persist in the adoption of budget and expense management systems. Security concerns, especially in cloud-based solutions, have been identified (Lee & Kim, 2016) [6]. Employee resistance during the implementation phase is also a noteworthy challenge (Harrison et al., 2015) [7]. Addressing these challenges is crucial to ensure successful adoption and sustained use.

2.5. Integration with Financial Reporting

The seamless integration of budgeting systems with financial reporting is imperative for organizations seeking transparency and accuracy in financial information (Davis & Miller, 2019) [8]. A well-integrated system facilitates more informed decision-making by stakeholders and ensures consistency in financial data across various organizational functions.

2.6. Mobile Applications in Expense Management

The role of mobile applications in expense management is gaining prominence. Brown and Smith (2021) [9] highlight the accessibility benefits of mobile apps, allowing employees to submit real-time expense reports. However, concerns related to security and policy compliance need careful consideration.

2.7. Future Directions and Research Gaps

The future of budget and expense management systems holds exciting possibilities. Researchers are increasingly exploring the development of intelligent systems using Artificial Intelligence (AI) and machine learning algorithms (Turing & Wallace, 2022) [10]. Additionally, there is a growing need for studies examining the long-term impacts of system implementations on organizational performance and culture.

2.8. Customization and Scalability Challenges

[11] Black et al. (2019) shed light on the challenges associated with customization and scalability. As organizations vary in size and structure, the effectiveness of budget and expense management systems hinges on their ability to adapt to diverse industry needs and organizational scales.

2.9. Organizational Change and Implementation Dynamics

Kim and Patel (2018) delve into the challenges associated with organizational change and the implementation of these advanced systems [12]. Resistance to change, the importance of adequate training, and the impact of organizational culture on adoption rates emerge as critical factors requiring attention.

2.10. Emerging Technologies and Future Prospects

Looking forward, there is immense potential for the integration of emerging technologies into budget and expense management. The literature hints at possibilities such as artificial intelligence for predictive financial analysis, blockchain for enhanced security, and sustainable financial practices as focal points for future exploration.

From the historical evolution to the current technological landscape, researchers have explored various facets of these systems. While automated solutions bring about numerous benefits, challenges remain, requiring continuous attention. The integration of technology, especially mobile applications, is reshaping how organizations approach expense management. As we look to the future, the development of intelligent systems and a deeper understanding of long-term impacts will shape the next phase of research in this critical area.

CHAPTER 3

3. METHODOLOGY

3.1.Data Collection Methods

Data source for this project is the office of Debre Markos Finance and Budget Administration Directorate. We have used different methods to collect data. Data collection is the most important part of the project to find the main requirement of the system and to understand how the system does.

- ❖ **Interviews:** -This is one of the methods used for the collection of data in which the project designers are asking different questions to Debre Markos Finance and Budget Administration Directorate Manager, Mr. Endale Tefera, and employees for obtaining the required information and data.
- ❖ **Observation:** -This is another type of method for collecting data and information in which could witness the actual events which will happen in the organization. In this method all team members have observed and note down the events from that observation.
- ❖ **Document Analysis:** -The team member also collected certain relevant information from written documents in the Finance and Budget Administration Directorate. Not only that but also, we tried to review other relevant documents to develop this project.

3.2.The system development model

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as “Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development. But the proposed system follows Waterfall Model. Because the waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap. In "The Waterfall" approach, the whole process of software development is divided into separate phases and, the outcome of one phase acts as the input for the next phase sequentially.

3.3.Feasibility study

Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Economical, Technical, Operational and Legal feasibility for adding new modules and debugging old running system. There are aspects in the feasibility study portion of the preliminary investigation:

- ❖ Technical Feasibility
- ❖ Operational Feasibility
- ❖ Economic Feasibility
- ❖ Legal feasibility

3.3.1. Technical Feasibility

The proposed system can be easily maintained and repaired; technically, the system will be powerful to be applied by low skilled users as much as possible. There is no need for the developer involvement in almost all implementation of the entire system. It is easily accessible by the people who can easily understand natural languages.

3.3.2. Operational Feasibility

➢ **User Adoption**

The system is designed to be user-friendly, with an intuitive interface that requires minimal training. Training sessions and user manuals will be provided to ensure smooth adoption by all users.

➢ **Integration with Existing Processes**

The system is designed to integrate seamlessly with existing financial processes at DMUIT. This includes compatibility with current workflows and data formats, minimizing disruption during the transition.

3.3.3. Economic Feasibility

➢ **Cost Analysis**

The cost analysis includes the following components:

- ❖ **Development Costs**: Salaries for the development team, software licenses, and other development tools.
- ❖ **Deployment Costs**: Server costs, hosting fees, and related expenses.
- ❖ **Maintenance Costs**: Ongoing support, updates, and troubleshooting.

The total estimated cost is within the budget allocated by DMUIT for the project.

➢ **Cost-Benefit Analysis**

The benefits of implementing the DMUIT Budget Expense Management System include:

- **Reduced Administrative Costs**: Automation of financial processes reduces the need for manual data entry and paperwork.
- **Improved Accuracy**: Minimization of errors associated with manual processes.
- **Enhanced Decision-Making**: Real-time access to financial data supports informed decision-making.

The anticipated benefits outweigh the costs, making the project economically feasible.

3.3.4. Legal Feasibility

➢ **Compliance with Regulations**

The system will comply with relevant financial regulations and standards, ensuring legal and regulatory adherence. This includes data protection laws and financial reporting standards.

➢ **Data Privacy and Security**

Robust security measures, including role-based access control and data encryption, will be implemented to protect sensitive financial data and ensure user privacy.

3.4.SYSTEM ANALYSIS

To obtain all the required information for the system development, interview, document review and observation were conducted. The UML modeling technique was used to model the analysis phase of the system. The system analysis, modeling deals with analyzing the proposed system. It includes the system use case diagrams, sequence diagrams, activity diagrams, analysis class diagram and their descriptions. After identifying the actors and use cases, the use cases are developed and textual descriptions are stated. The Sequence diagram depicted based on the use cases which are developed for the proposed system. Activities will be represented by the activity diagrams.

System analysis involves understanding and specifying in detail what the DMUIT Budget Expense Management System needs to accomplish. This chapter describes the existing system, identifies the requirements for the new system, and defines the functional and non-functional requirements.

3.5.Existing System Analysis

3.5.1. Current Financial Management Process

The current financial management process at DMUIT is primarily manual and involves the following steps:

- ❖ **Budget Allocation**: Budgets are allocated to various departments manually using spreadsheets and paper documents.
- ❖ **Expense Tracking**: Expenses are tracked manually, with records maintained in spreadsheets and paper files.
- ❖ **Financial Reporting**: Financial reports are generated manually by compiling data from various spreadsheets and documents.

3.5.2. Challenges with the Existing System

- ❖ **Inefficiency**: Manual processes are time-consuming and labor-intensive.
- ❖ **Inaccuracy**: Susceptibility to human errors in data entry and calculations.
- ❖ **Lack of Transparency**: Difficulty in tracking and auditing financial transactions.
- ❖ **Delayed Reporting**: Time lag in generating financial reports, impacting decision-making.
- ❖ **Data Security**: Paper documents and spreadsheets are vulnerable to unauthorized access and data loss.

3.5.3. Requirements for the New System

The new DMUIT Budget Expense Management System is designed to address the challenges of the existing system and meet the needs of its users. The requirements are divided into functional and non-functional requirements.

3.6.Functional Requirements

3.6.1. User Management

- ❖ **User Registration and Authentication**: Users should be able to register, log in, and log out securely.
- ❖ **Role-Based Access Control**: Different user roles (e.g., administrator, financial officer, department head) with specific permissions.

3.6.2. Budget Management

- ❖ **Budget Allocation**: Allocate budgets to various departments and projects.
- ❖ **Budget Adjustment**: Modify allocated budgets as needed.
- ❖ **Budget Overview**: View summaries and details of allocated budgets.

3.6.3. Expense Tracking

- ❖ **Expense Entry**: Record and categorize expenses.
- ❖ **Expense Approval**: Workflow for approving expenses by authorized personnel.
- ❖ **Expense Monitoring**: Real-time tracking of expenses against allocated budgets.

3.6.4. Financial Reporting

- ❖ **Report Generation**: Generate financial reports (e.g., budget vs. actual, expense summaries).
- ❖ **Custom Reports**: Create custom reports based on specific criteria.
- ❖ **Export Reports**: Export reports in various formats (e.g., PDF, Excel).

3.6.5. Notifications and Alerts

- ❖ **Email Notifications**: Notify users of important events (e.g., budget approval, expense submission).
- ❖ **Alerts**: Alerts for budget overruns or critical issues.

3.7. Non-Functional Requirements

Performance

- ❖ **Scalability**: The system should handle an increasing number of users and transactions.
- ❖ **Response Time**: The system should provide quick responses to user actions.

Usability

- ❖ **User Interface**: The interface should be intuitive and easy to navigate.
- ❖ **User Experience**: Provide a positive user experience with minimal training required.

Security

- ❖ **Data Protection**: Secure sensitive financial data through encryption.
- ❖ **Access Control**: Implement strict access control measures to prevent unauthorized access.
- ❖ **Data Integrity**: Ensure the integrity of data through validation and error-checking mechanisms.
- ❖ **System Availability**: The system should be available 24/7 with minimal downtime.
- ❖ **Error Handling**: Robust error handling to manage and log system errors gracefully.

Maintainability

- ❖ **Modular Design**: Design the system with a modular architecture for easy maintenance and updates.

- ❖ **Documentation**: Provide comprehensive documentation for users and developers.

3.8.Technical Requirement

Technical requirements are the technical issues that must be considered too successfully complete a project. These are aspects such as performance, reliability, availability that the project must meet on in order to proceed with a project.

Generally, these are technical requirements to complete the project.

- ❖ The interface of the system should be user friendly (easy to use).
- ❖ The interface should display error message if it detects invalid input.
- ❖ The system should deny unauthorized accesses to the system domain.
- ❖ The system should provide help for the user.
- ❖ Requires an expert to use the system.
- ❖ Training the users to access the system.

3.9.Change cases

Change cases are used to describe new potential requirements for a system or modifications to existing requirements. These are modeled in a simple manner. Describe the potential change to the existing requirements, indicate the likeliness of that change occurring, and indicate the potential impact of that change.

The system is ready to change if the organization has been open different branches and also if new material has been imported to the camp, in this case the system is easily scalable and ready to be changed.

Constraints

A constraint is a restriction on the degree of freedom you have in providing a solution. Constraints are effectively global requirements, such as limited development resources or a decision by senior management that restricts the way you develop a system. Constraints can be economic, political, technical, or environmental and pertain to the project resources, schedule, target environment, or to the system itself. The major constraint is:

- ❖ Lack of time to gathering a requirement and data.

Other constraints include: -

Technical Constraints

- **System Compatibility**: The system must be compatible with common web browsers (e.g., Chrome, Firefox, Safari) and operating systems (e.g., Windows, macOS, Linux).
- **Performance**: The system should handle up to 1,000 concurrent users without significant performance degradation.
- **Scalability**: The architecture must support future scalability to accommodate growth in user base and data volume.

Security Constraints

- **Data Encryption**: All sensitive data must be encrypted at rest and in transit using industry-standard encryption methods.
- **Authentication**: Strong authentication mechanisms (e.g., multi-factor authentication) must be implemented to ensure secure access.
- **Authorization**: Role-based access control (RBAC) must be strictly enforced to prevent unauthorized access to sensitive data and functions.

Compliance Constraints

- **Regulatory Compliance**: The system must comply with relevant financial regulations and standards (e.g., data protection laws, financial reporting standards).
- **Auditability**: The system must maintain detailed audit logs of all transactions and user activities to support auditing and compliance requirements.

Operational Constraints

- **Backup and Recovery**: Regular automated backups must be scheduled, and a robust data recovery plan must be in place.
- **Maintenance Windows**: Scheduled maintenance windows must be established to minimize downtime and ensure system updates and backups are performed without disrupting users.

3.10. Actor and Use case Identification

Based on the findings of existing system assessment, the system process is modeled and use cases and actors are identified. We relate actors with the corresponding use cases as shown in figure 1.

Actors of the system

Actors: An actor represents anything or anyone that interacts with the system. This may include people (not just the end user), external systems, and other organizations. Actors are always external to the system being modeled; they are never part of the system. The actors that will participate in the system are listed below:

Directory1

- ❖ View
- ❖ View expense
- ❖ Expense
- ❖ View budget
- ❖ Logout

Directory2

- ❖ login
- ❖ View expense
- ❖ Expense
- ❖ View budget
- ❖ Logout

Directory3

- ❖ login
- ❖ View expense

- Expense
- View budget
- Logout

Directory4

- login
- View expense
- Expense
- View budget
- Logout

Directory5

- login
- View expense
- Expense
- View budget
- Logout

Directory6

- login
- View other expense
- View your Expense
- Allocate budget
- budget
- remove user
- Register
- logout

Directory7

- login
- View other expense
- View your Expense
- expense
- view budget
- logout

Use cases of the system

The Use case (UC) represents functionality provided by a system unit and expressed by sequence of message exchange by the system unit and one or more actors of the system. The following use cases have been identified for the proposed system specification.

Table 1:Use case Identification

Use case name	Use case ID	Use/includes
Login	UC-01	---
Logout	UC-02	---
View expense	UC-03	Login
Expense	UC-04	Login
View budget	UC-05	Login
View other expense	UC-06	Login
View your expense	UC-07	Login
Allocate budget	UC-08	Login
Budget	UC-09	Login
Remove user	UC-10	Login
Register	UC-11	Login

Use case Diagram

The use case diagram is concerned with the interaction between the system and actors (objects outside the system that interact directly with it). It presents a collection of use cases and their corresponding external actors. A use case is a generic description of an entire transaction involving several objects of the system. Use cases are represented as ellipses, and actors are depicted as icons connected with solid lines to the use cases they interact with. A use case diagram is helpful in visualizing the context of a system and the boundaries of the system's behavior. Each use cases in the use case diagram can also be described using a narrative form.

A Use Case represents a discrete unit of interaction between a user and the system. A use case diagram contains four components.

1. **Boundary**: -which defines the system of interest in relation to the world around it.
2. **Actors**: -usually individuals involved with the system defined according to their roles.
3. **Use cases**: -which the specific roles are played by the actors within and around the system.
4. **The relationships** between the actors and the use cases as depicted in the following figure.

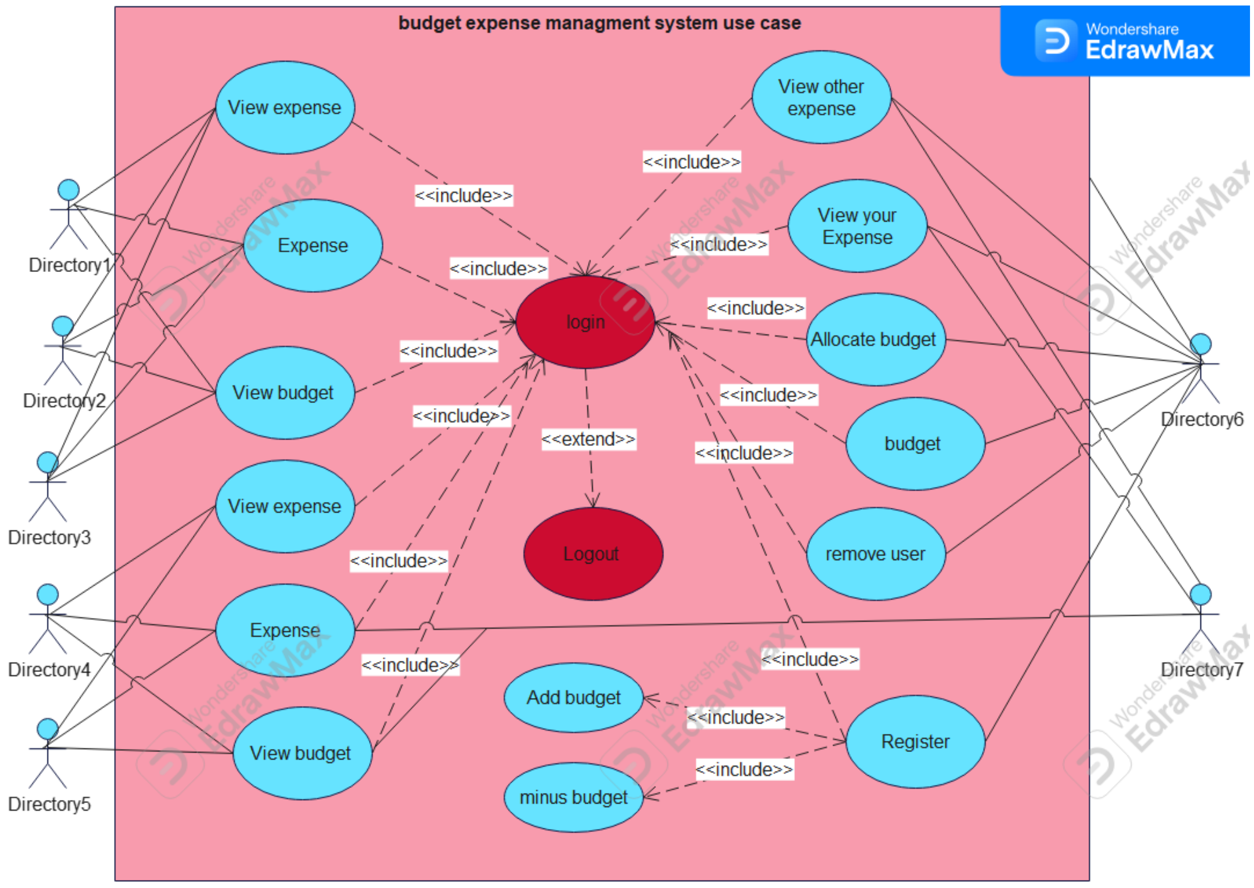


Figure 1:Use case Diagram

Use case description

The following consecutive tables show the use case description for each of the use cases. Each table contains the use case name, the actor which initiates and interacts with the use case, description of the use case and typical course of events that show the interaction between the actor and the use case which enable the team to easily depict the functions of the proposed system.

Table 2:Use case description for login

Name	Description
Login	UC-01
Goal	To authenticate and log the user into the system.
Actors	User
Preconditions	None
Postconditions	User is authenticated and logged into the system
Alternative Course of Action:	User navigates to the login page. User enters username and password System validates credentials. System logs user in and redirects to the dashboard
Alternative Course of Action Invalid username or password	System displays an error message. User re-enters credentials or clicks on "Forgot Password".

Table 3:Use case description for logout

Name	Description
Logout	Use case ID

	UC-02
Actors	User
Preconditions	User must be logged in
Postconditions	User is logged out of the system
Basic Course of Action:	User clicks on the logout button. System logs the user out. System redirects user to the login page
Alternative Course of Action	None

Table 4:Use case description for view expense

Name	Description
Use Case	View Expense
Use Case ID	UC-03
Goal	To allow the user to view their expenses.
Preconditions	User must be logged in
Postconditions	User can view their expenses.
Basic Course of Action:	navigates to the expense view page. System retrieves user's expense data. System displays the expense data.
Alternative Course of Action	System is unable to retrieve expense data: 1. System displays an error message. 2. User can retry or contact support.

Table 5:Use case description for expense

Use case	Expense
Use case ID	UC-04
Goal	To manage user expenses.
Actor	User
Precondition	User must be logged in.
Postcondition	User can manage their expenses.
Basic course of action	User navigates to the expense management page. User enters or edits expense details. System saves the expense data.
Alternative Course of Action	System is unable to save expense data: System displays an error message. User can retry saving or contact support.

Table 6:Use case description for view budget

Use case	View Budget
Use case ID	UC-05
Goal	To allow the user to view their budget.
Actors	User
Precondition	User must be logged in
Postcondition	User can view their budget.
Basic Course of Action:	ser navigates to the budget view page. System retrieves user's budget data. System displays the budget data.
Alternative Course of Action	System is unable to retrieve budget data: System displays an error message. User can retry or contact support.

Table 7:Use case description for View Other Expense

Use case	View Other Expense
Use case ID	UC-06
Goal	To allow the user to view other users' expenses.
Actor	User
Precondition	User must be logged in.
Postcondition	User can view other users' expenses.
Basic Course of Action:	User navigates to the other expenses view page. System retrieves other users' expense data. System displays the expense data.
Alternative Course of Action	System is unable to retrieve expense data System displays an error message. User can retry or contact support

Table 8:Use case description for View Your Expense

Use case	View Your Expense
Use case ID	UC-07
Goal	To allow the user to view their own expenses.
Actor	User
Precondition	User must be logged in
Postcondition	User can view their own expenses.
Basic Course of Action:	User navigates to their own expense view page. System retrieves user's expense data. System displays the expense data.

Table 9:Use case description for budget

Use case	Budget
Use case ID	UC-09
Goal	To manage user budge
Actors	User
Precondition	User must be logged in
Postcondition	User can manage their budget.
Basic Course of Action	User navigates to the budget management page. User enters or edits budget details. System saves the budget data.
Alternative Course of Action	System is unable to save allocation data: 1. System displays an error message. 2. User can retry saving or contact support

Table 10:Use case description for Remove user

Use case	Remove user
Use case ID	UC-10
Goal	To remove a user from the system
Actors	Admin
Precondition	Admin must be logged in.
Postcondition	User is removed from the system.
Basic Course of Action:	Admin navigates to the user management page. Admin selects a user to remove. System removes the user from the database.

Alternative Course of Action	System is unable to remove user: 1. System displays an error message. 2. Admin can retry or contact support.

Table 11:Use case description for register

Use case	Register
Use case ID	UC-11
Goal	To register a new user in the system.
Actors	User
Precondition	None
Postcondition	User is registered in the system.
Basic Course of Action:	User navigates to the registration page. User enters registration details. System saves the registration data and creates a new user account.
Alternative Course of Action	System is unable to save registration data: 1. System displays an error message. 2. User can retry registration or contact support.

3.11. Sequence Diagram

Sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom.

- ❖ **The boxes** across top of the diagram represent classifiers or their instances; typically use cases, objects, classes or actors.

- **The solid lines** hanging from the boxes are called objects lifelines, representing the life span of object during the scenario being modeled.
- **Messages** indicate as labeled arrows, when the source and the target of a message is an object or class label the signature of the method invoked in response to the message.

However, if either the source or target is the human actor, then the message is labeled with brief text describing the information is available.

Generally, a sequence diagram shows object interactions arranged in time sequence.

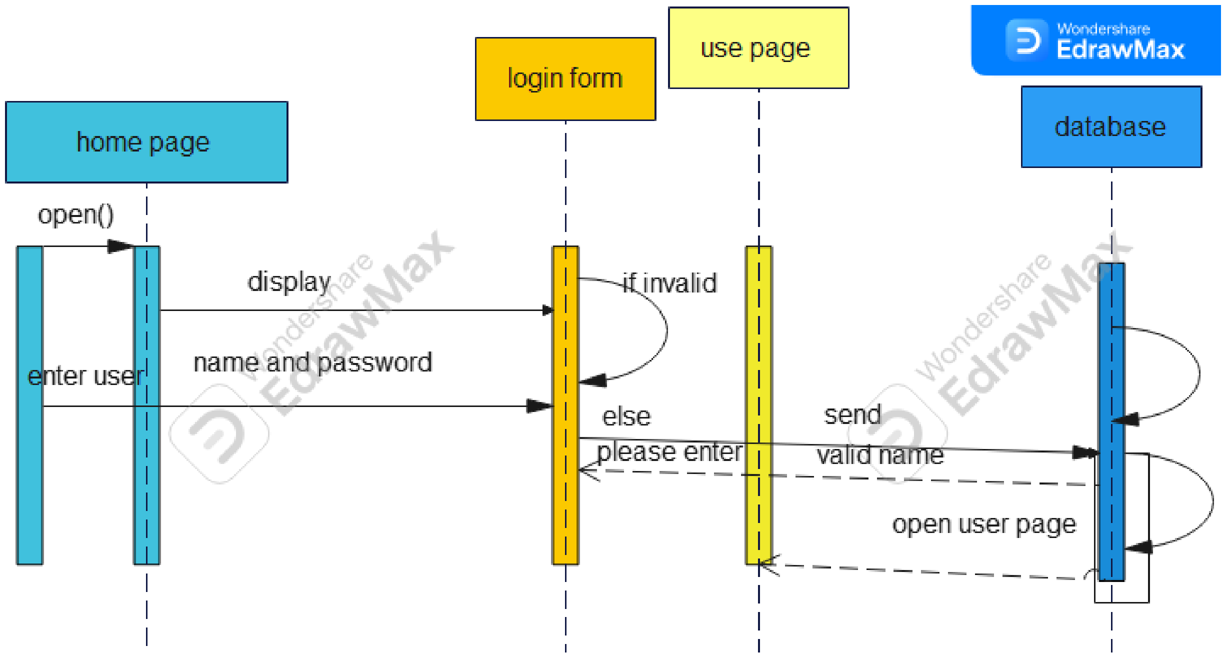


Figure 2:sequence home page

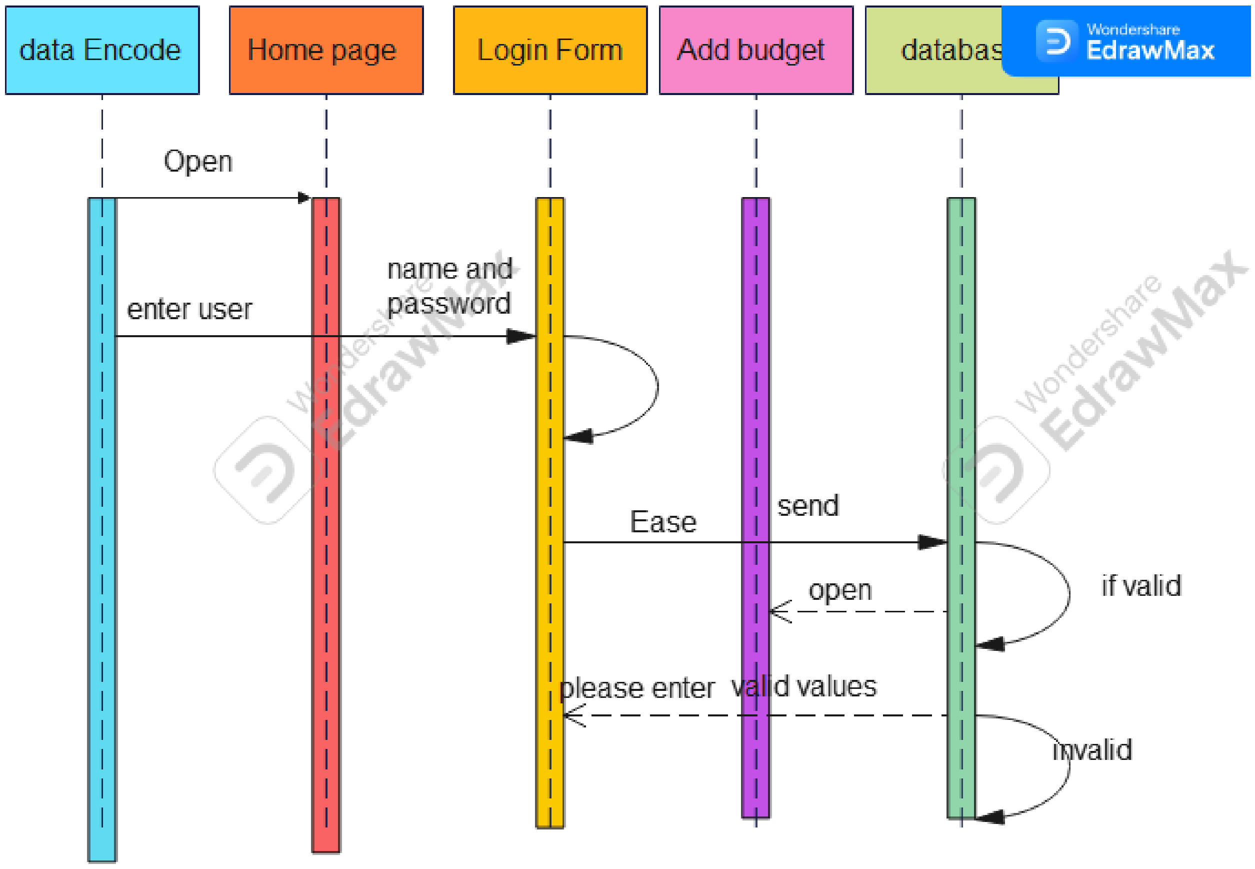
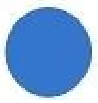
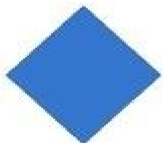
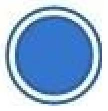
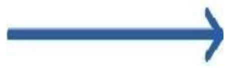
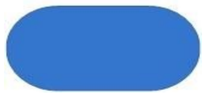


Figure 3: sequence diagram for login

Activity Diagram

Active diagrams are used to model the flow of an object as it moves from state to state at different points in the flow of control. It is essentially a flow chart that emphasizes the activity that takes place over time. Activity diagrams can be used to model higher-level business process at the business unit level, or to model low-level internal class actions. It is "Less technical" in appearance, compared to sequence diagrams, and business-minded people tend to understand them more quickly.

Table 12:Activity diagram Description

No	Symbol	Name	Function
1.		Initial state	To start the activity.
2.		Decision activity	To make validation.
3.		Final state	To terminate from the activity.
4.		Arrow	To show the flow of activity between two activities.
5.		Activity	Set of activities that the user and system performing.

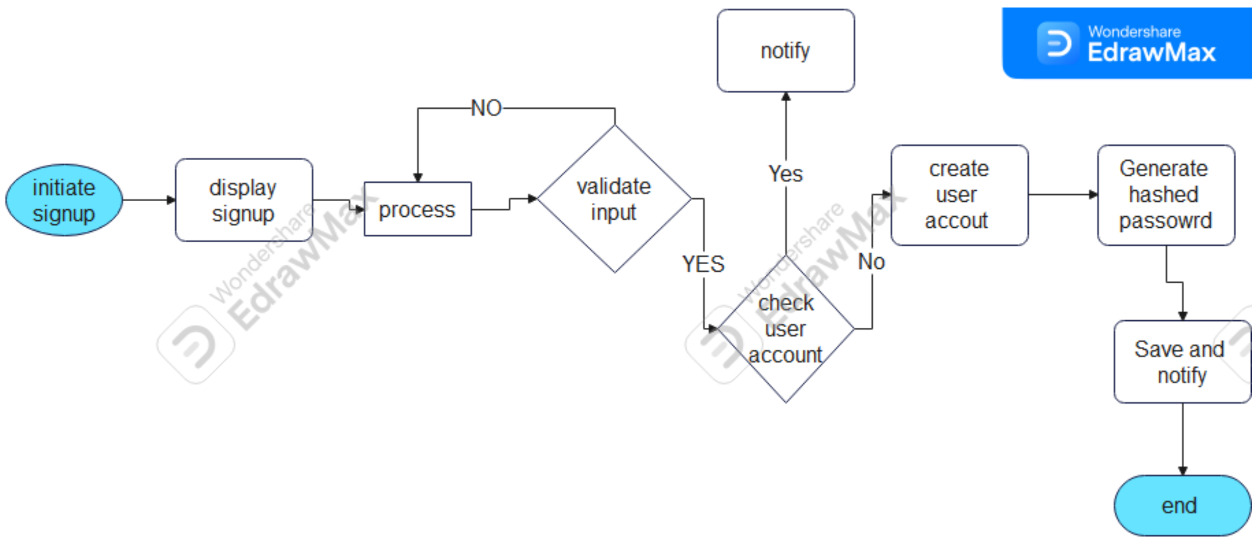


Figure 4:activity diagram for signup

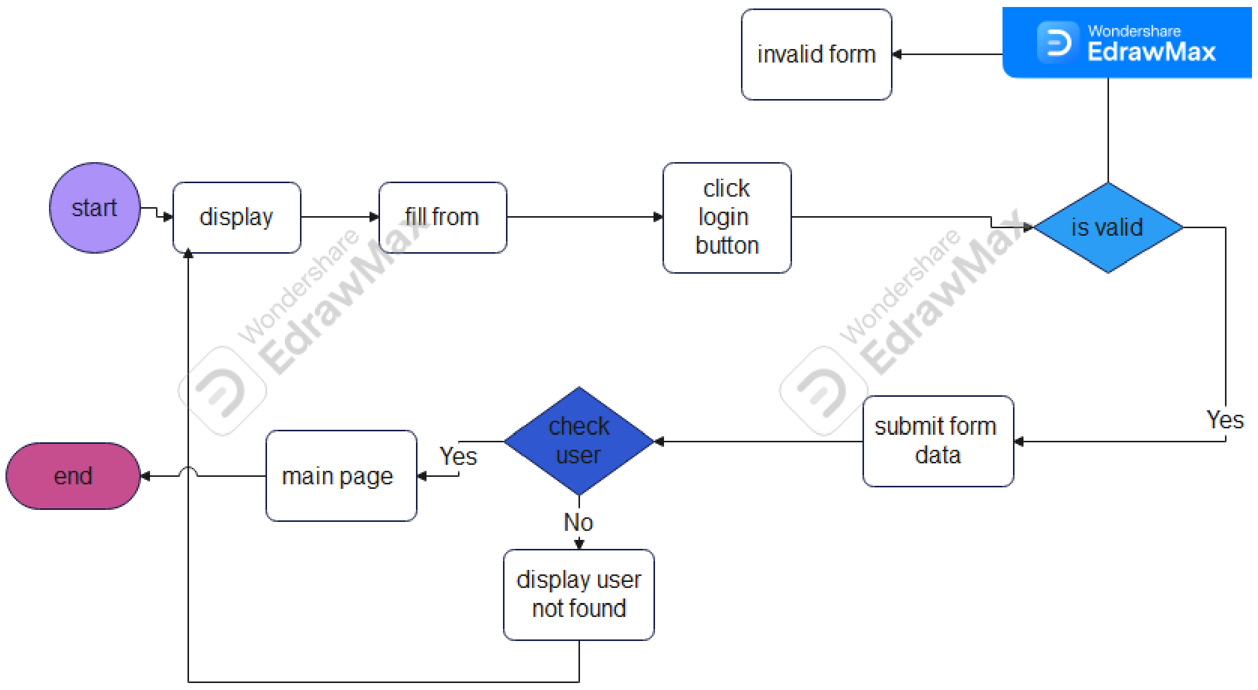


Figure 5:activity diagram for login

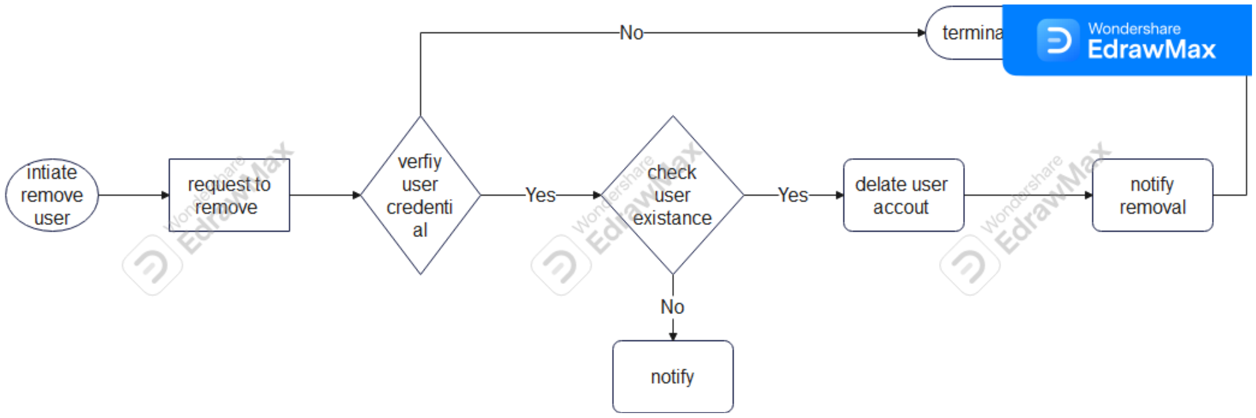


Figure 6:activity diagram for remove user

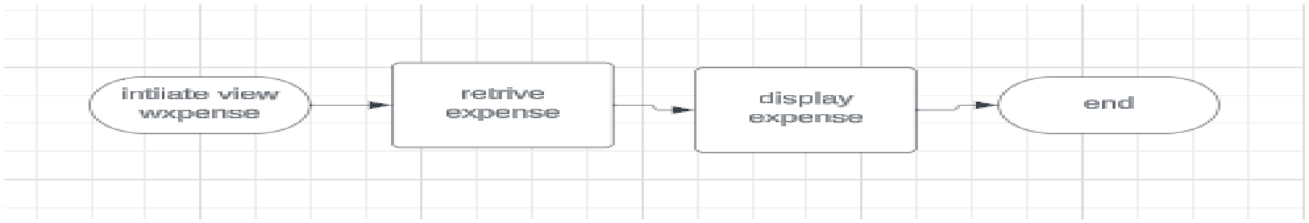


Figure 7:activity diagram for view expense

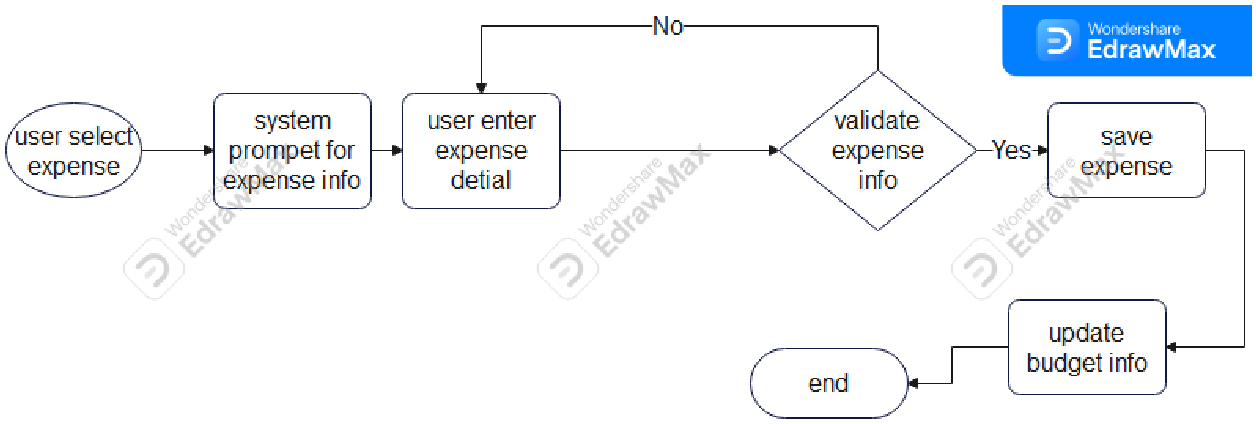


Figure 8:activity diagram create expense

3.12. Analysis Class Diagram

UML class diagrams are the mainstay of object-oriented modeling. Class models show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the

operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual/domain modeling and detailed structural design modeling.
Classes are depicted as boxes with three sections: the top one indicates the name of the class, the middle one lists the attributes of the class, and the third one lists the methods. By including both an attribute and a method box in the class. Another approach would be to have two sections, one for the name and one for the responsibilities. Donald Bell, IBM Global Services, UML basics: An introduction to the Unified Modeling Language, 2003 [13]

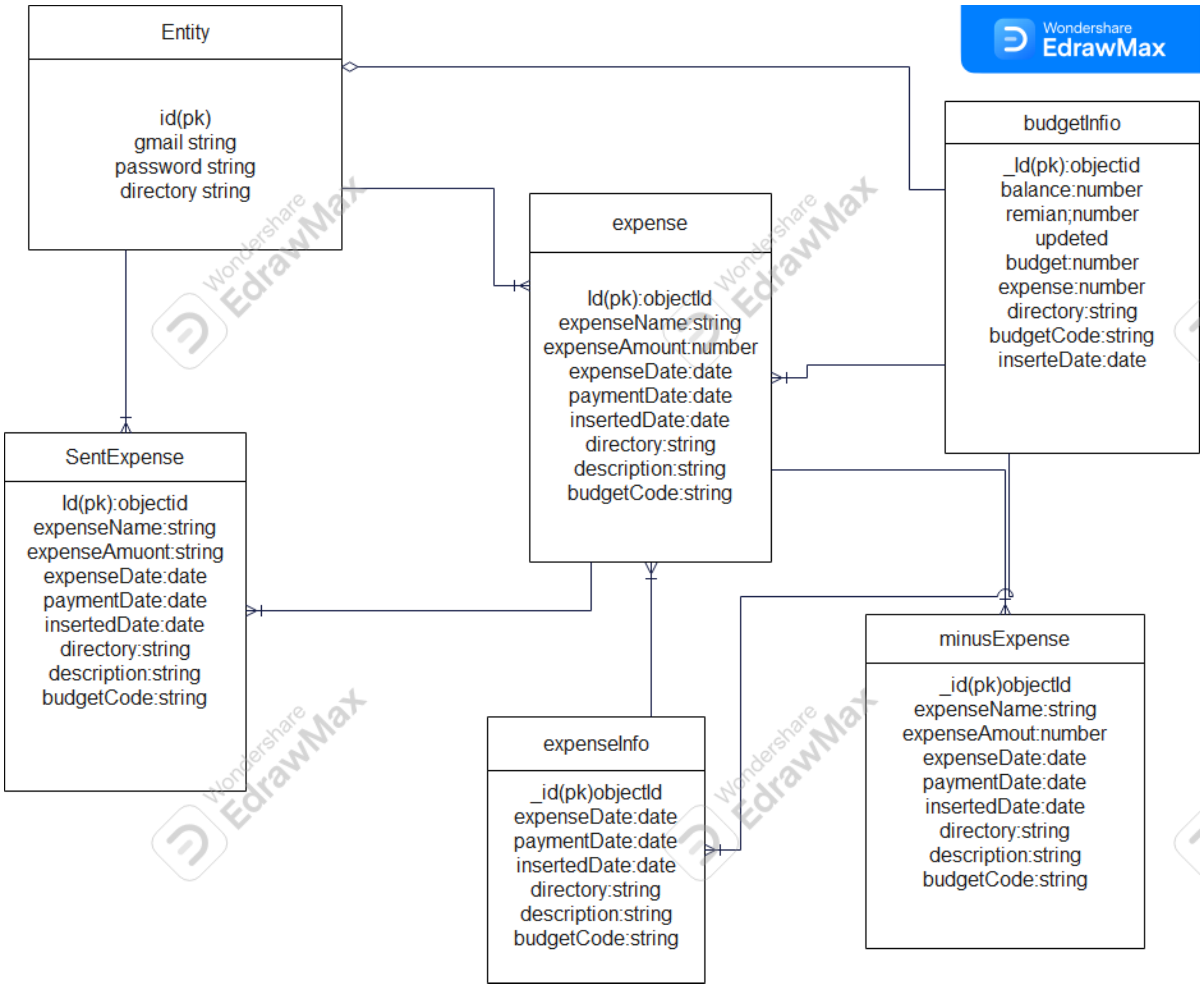


Figure 9:Analysis class diagram

3.13. SYSTEM DESIGN

The Design Phase seeks to develop detailed specifications that emphasize the physical solution to the user's information technology needs. The system requirements and logical description of the entities, relationships, and attributes of the data that were documented during the Requirements Analysis Phase are further refined and allocated into system and database design specifications that are organized in a way suitable for implementation within the constraints of a physical environment. (e.g., like computer, database, facilities etc.).

During the Design Phase, the initial strategy for any necessary training is also begun. Estimates of project expenses are updated to reflect actual costs and estimates for future phases. In addition, the work planned for future phases is redefined, if necessary, based on information acquired during the Design Phase.

System design translates the requirements and analysis into a blueprint for constructing the DMUIT Budget Expense Management System. This chapter covers the architectural design, detailed design of key components, data model, and interface design to ensure a cohesive and robust system.

3.14. Architectural Design

The system architecture is based on a three-tier model, consisting of the presentation layer, application layer, and data layer. This separation of concerns enhances modularity, scalability, and maintainability. [14] [15] [16]

Presentation Layer

- ➢ **Technologies**: HTML, CSS, React.js
- ➢ **Responsibilities**:
  - ❖ Render user interface components
  - ❖ Handle user interactions
  - ❖ Communicate with the application layer via API calls

Application Layer

- ➢ **Technologies**: Node.js, Express.js
- ➢ **Responsibilities**:
  - ❖ Implement business logic
  - ❖ Handle client requests and responses

- Interact with the data layer to perform CRUD operations

Data Layer

- **Technologies**: MongoDB
- **Responsibilities**:
    - Store and manage data
    - Ensure data integrity and security
    - Provide efficient data retrieval and update mechanisms

3.15. DESIGN GOALS OF THE SYSTEM

Design goals are essential for guiding the development process, ensuring that the system meets its intended objectives and provides value to its users. The design goals for the DMUIT Budget Expense Management System focus on functionality, usability, performance, scalability, security, and maintainability.

Functionality

Goal: Provide comprehensive budget and expense management capabilities to meet the needs of Debre Markos University Institute of Technology.

- **Accurate Budget Tracking**: Enable precise tracking of budget allocations, adjustments, and expenditures.
- **Expense Recording and Approval**: Facilitate the recording of expenses and ensure they go through a proper approval process.
- **Financial Reporting**: Generate detailed financial reports for analysis and decision-making.
- **User Role Management**: Support different user roles with specific permissions and access levels.

Usability

Goal: Ensure the system is user-friendly and accessible to all users, regardless of their technical proficiency.

- **Intuitive Interface**: Design an easy-to-navigate interface with clear instructions and helpful tooltips.
- **Responsive Design**: Ensure the system is accessible on various devices, including desktops, tablets, and smartphones.

- ❖ **Minimal Learning Curve**: Provide a system that requires minimal training, with most features being self-explanatory.

Performance

Goal: Achieve optimal performance to handle a large number of users and data efficiently.

- ❖ **Fast Response Times**: Ensure that all user interactions with the system are processed quickly.
- ❖ **Efficient Data Processing**: Implement efficient algorithms and data handling techniques to manage large volumes of budget and expense data.
- ❖ **Resource Optimization**: Optimize the use of server and database resources to support high performance.

Scalability

Goal: Design the system to accommodate future growth in terms of users, data, and functionality.

- ❖ **Modular Architecture**: Use a modular design that allows easy addition of new features and components.
- ❖ **Database Scalability**: Ensure the database can scale horizontally and vertically to handle increasing data volumes.
- ❖ **Cloud Readiness**: Design the system to be easily deployable on cloud platforms, allowing for scalable infrastructure.

Security

Goal: Implement robust security measures to protect sensitive financial data and ensure system integrity.

- ❖ **Data Encryption**: Encrypt sensitive data both at rest and in transit to protect against unauthorized access.
- ❖ **Access Control**: Use role-based access control (RBAC) to restrict access to sensitive functions and data.
- ❖ **Regular Security Audits**: Conduct regular security audits and vulnerability assessments to identify and mitigate potential threats.

Maintainability

Goal: Ensure the system is easy to maintain and update, reducing the cost and effort of long-term maintenance.

- **Clean Codebase**: Write clean, well-documented code that follows best practices and coding standards.
- **Automated Testing**: Implement automated tests to catch issues early and ensure the system remains stable after updates.
- **Continuous Integration/Continuous Deployment (CI/CD)**: Use CI/CD pipelines to automate the build, testing, and deployment processes, facilitating frequent and reliable updates.

Reliability

Goal: Design the system to be reliable and available, minimizing downtime and ensuring consistent operation.

- **Redundancy**: Implement redundant systems and data backups to ensure availability in case of hardware or software failures.
- **Error Handling**: Implement comprehensive error handling to gracefully manage unexpected issues and provide informative feedback to users.
- **Monitoring**: Use monitoring tools to continuously track system performance and detect issues before they affect users.

3.16. Design Class Diagram

The class diagram represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The classes diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams which can be mapped directly with object-oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as structural diagram for this the team developed the following class diagram.

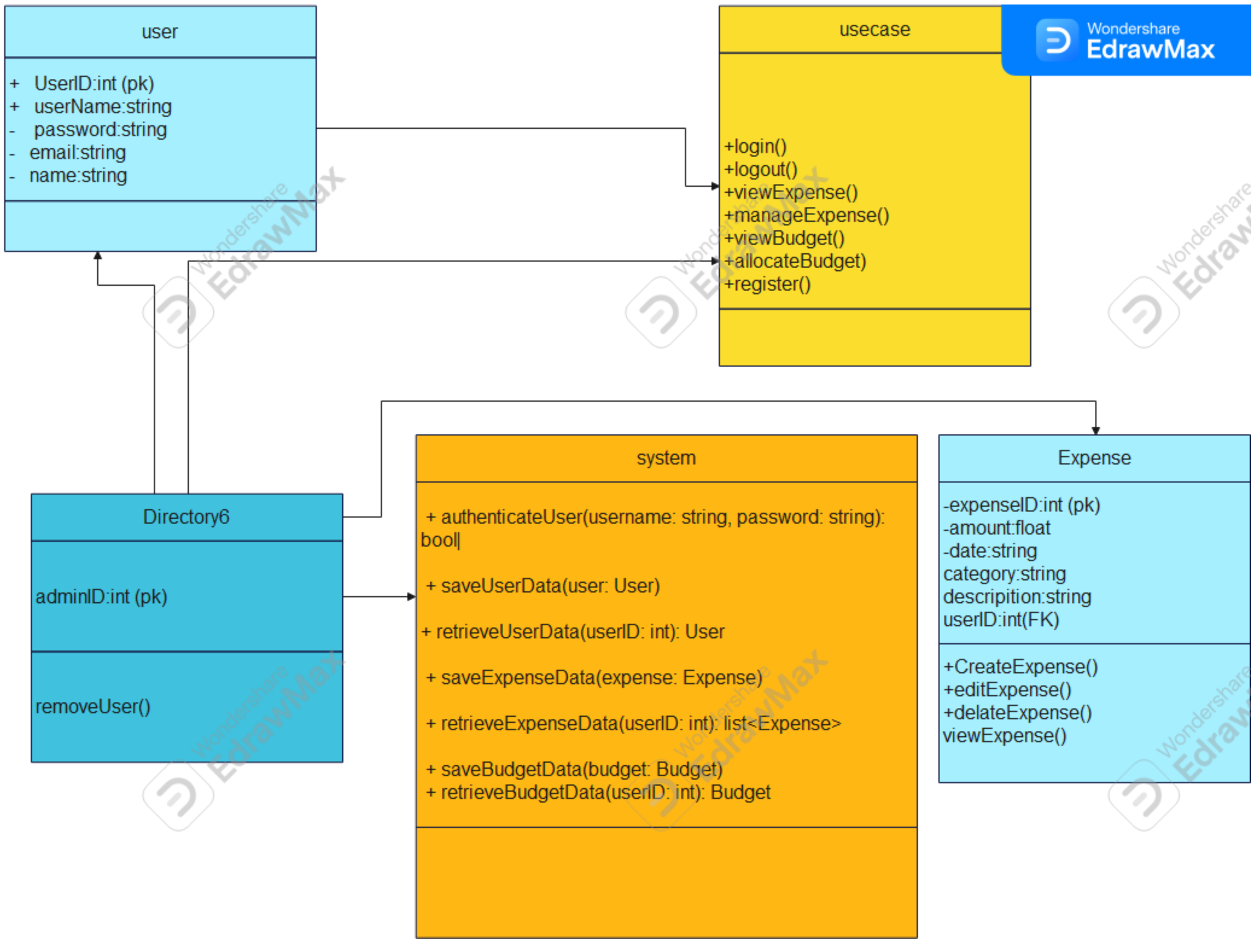


Figure 10:Design class diagram

Table 13:Description of design class diagram

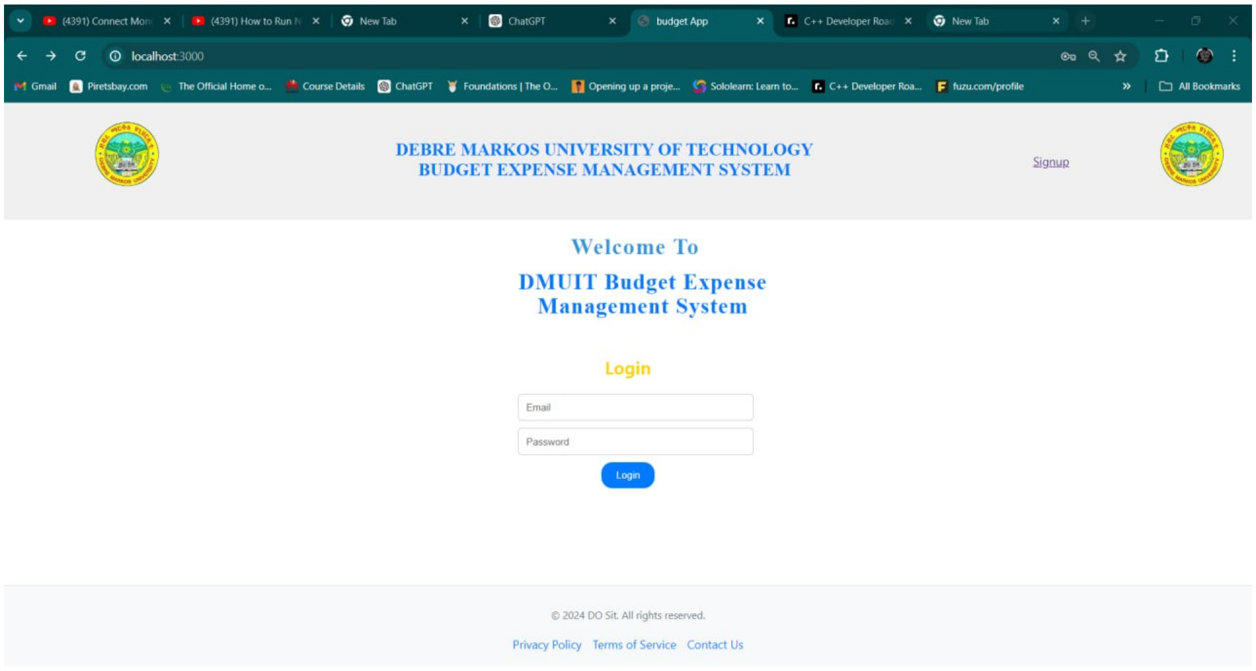
Class name	Attribute	Methods	Description
User	userId: String - username: String - password: String - role: String	- authenticate(password: String): Boolean - updatePassword(newPassword: String): void - getRole(): String	Represents a user in the system, responsible for authentication and managing user information.
Budget	- budgetId: String - department: String - fiscalYear: Int - amountAllocated: Double -	- allocateAmount(amount: Double): void - updateSpent(amount: Double): void - getRemainingAmount(): Double	Represents a budget record for a department within a fiscal year, managing allocation and expenses.

	amountSpent: Double		
Expense	- expenseId: String - budgetId: String - amount: Double - date: Date - category: String - description: String	- validate(): Boolean - save(): void - getCategory(): String	Represents an expense, linked to a budget, with methods to validate and save the expense.
Report	- reportId: String - department: String - fiscalYear: Int - reportType: String - data: List<Expense>	- generate(): void - format(): String - getReportData(): List<Expense>	Represents a report generated for a department and fiscal year, with methods to generate and format the report.
Session	- sessionId: String - userId: String - token: String - expiryDate: Date	- validateToken(token: String): Boolean - invalidate(): void - renew(): void	Manages user sessions, including token validation, invalidation, and renewal.

3.17. Sample user interface

The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals what is often called user centered design. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. The following interface design pictures describe the logical characteristics of some interfaces between the system and the users. So, the sample interfaces are shown as follows.

Figure: home page user interface



DEBRE MARKOS UNIVERSITY OF TECHNOLOGY
BUDGET EXPENSE MANAGEMENT SYSTEM

Signup

Welcome To

DMUIT Budget Expense Management System

Login

Email

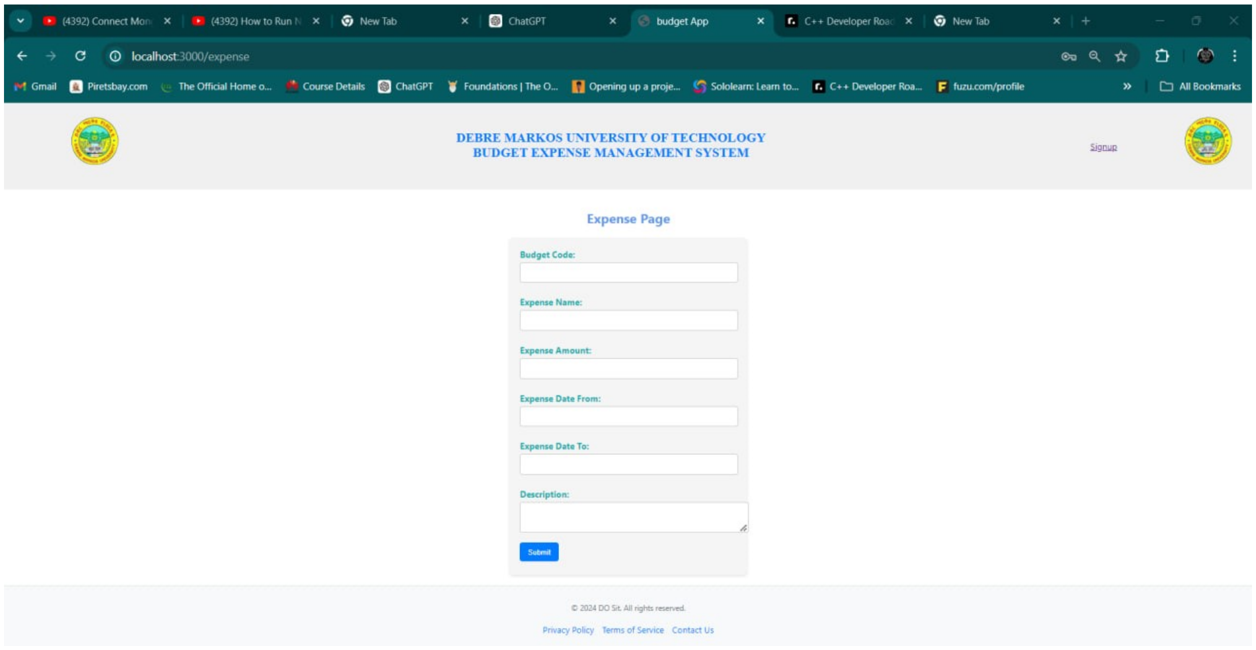
Password

Login

© 2024 DO Sit. All rights reserved.

Privacy Policy Terms of Service Contact Us

Figure 11:Landing page



DEBRE MARKOS UNIVERSITY OF TECHNOLOGY
BUDGET EXPENSE MANAGEMENT SYSTEM

Signup

Expense Page

Budget Code:

Expense Name:

Expense Amount:

Expense Date From:

Expense Date To:

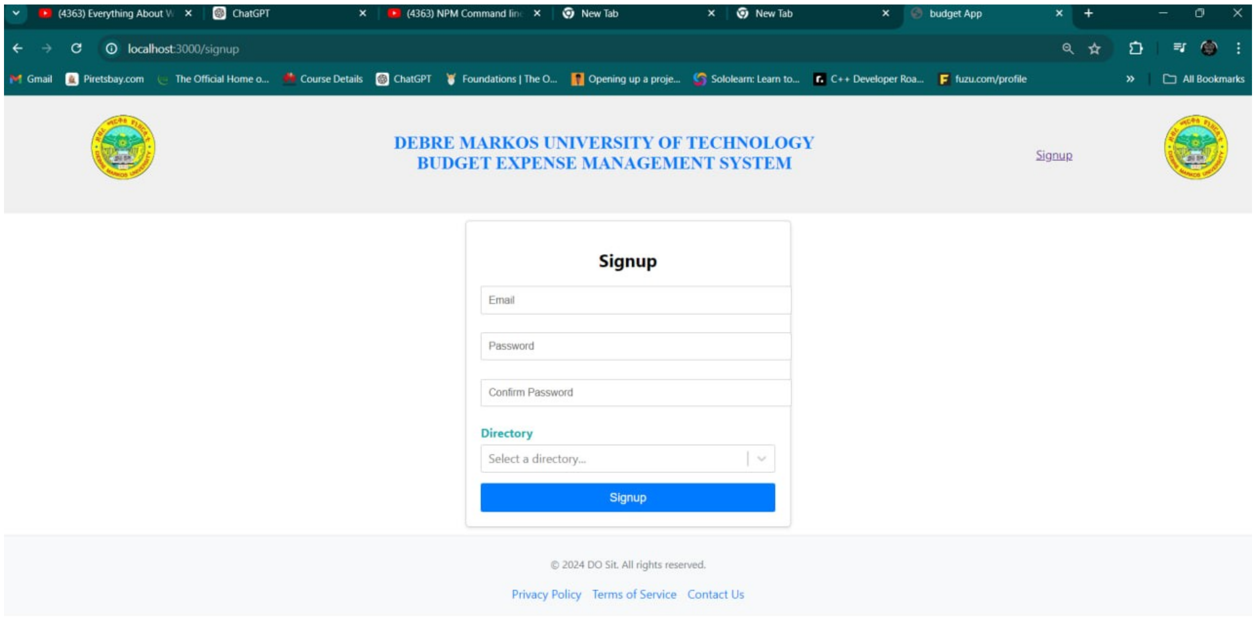
Description:

Submit

© 2024 DO Sit. All rights reserved.

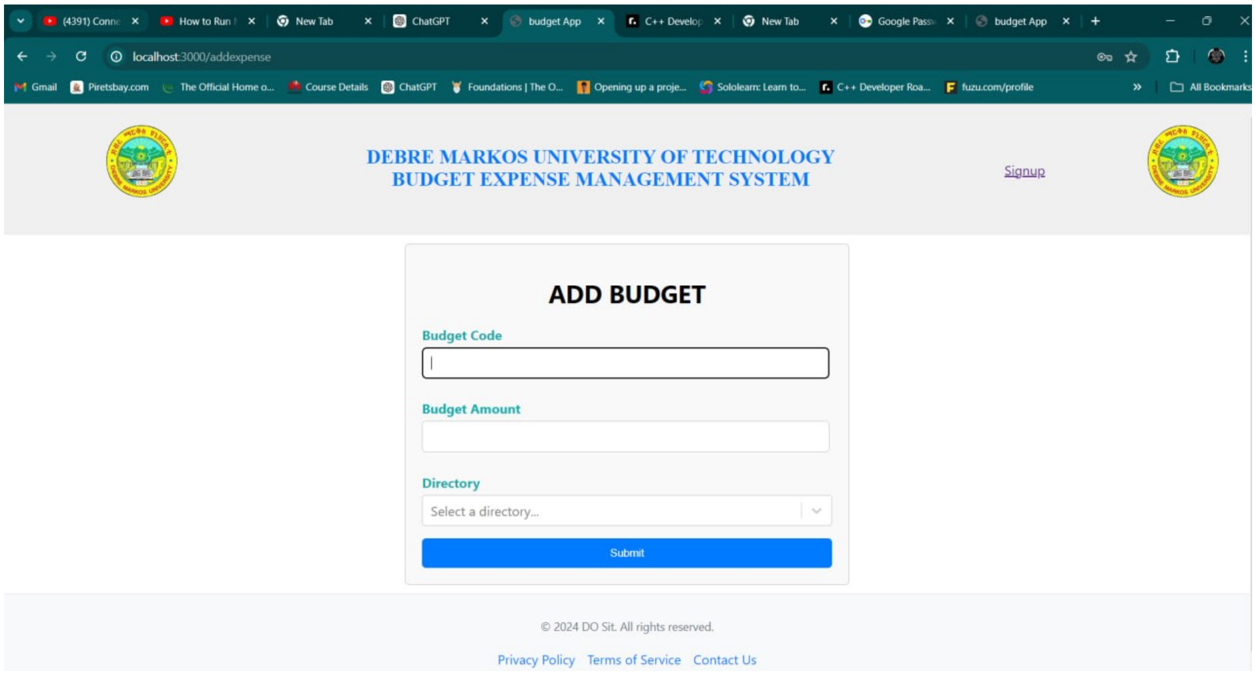
Privacy Policy Terms of Service Contact Us

Figure 12:sign up page user interface



localhost:3000/signup
DEBRE MARKOS UNIVERSITY OF TECHNOLOGY
BUDGET EXPENSE MANAGEMENT SYSTEM
Signup
Signup
Email
Password
Confirm Password
Directory
Select a directory...
Signup
© 2024 DO Sit. All rights reserved.
Privacy Policy Terms of Service Contact Us

Figure 13:signup page



localhost:3000/addexpense
DEBRE MARKOS UNIVERSITY OF TECHNOLOGY
BUDGET EXPENSE MANAGEMENT SYSTEM
Signup
ADD BUDGET
Budget Code
Budget Amount
Directory
Select a directory...
Submit
© 2024 DO Sit. All rights reserved.
Privacy Policy Terms of Service Contact Us

Figure 14: Add Budget Page

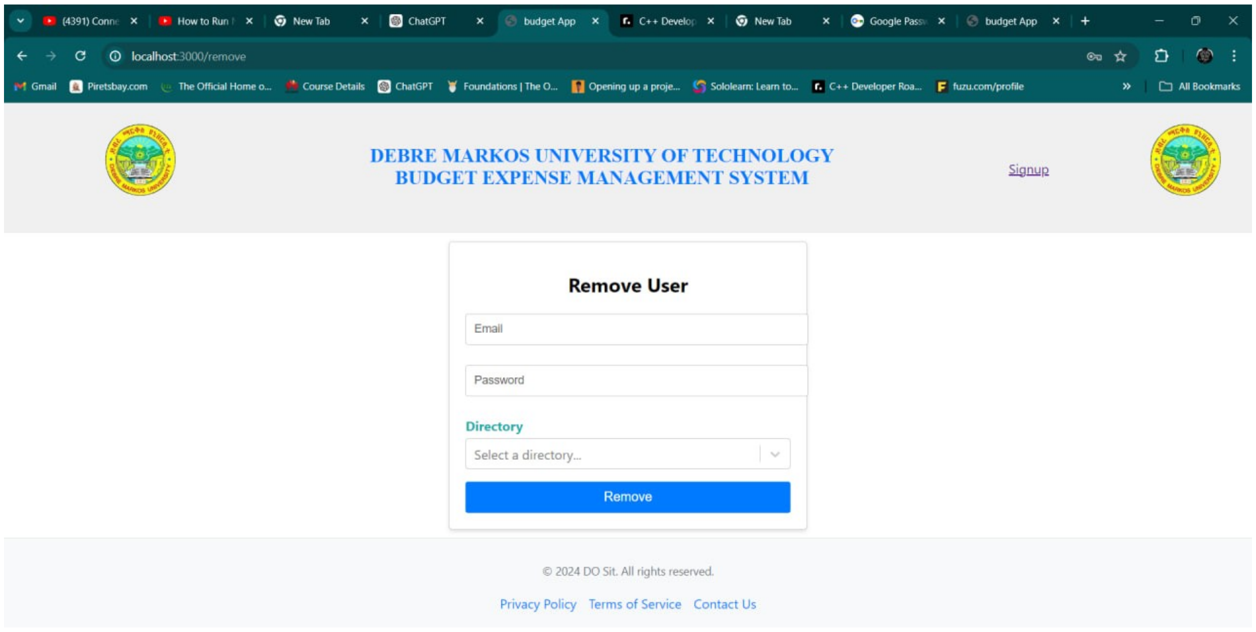


Figure 15:Remove User Page

CHAPTER FOUR

4. IMPLEMENTATION

Systems implementation is the final stage of system development process of defining how the information system should be built ensuring that the information system is operational and used and ensuring that the information system meets quality.

The implementation document helps users on how to work with the system. It acts as a user manual and it helps users not to be the system confused with. It includes sample forms and some selected fragment code. It gives the users a brief over view of the system.

This document describes the project implementation for developing the web-based budget expense management system. The project implements HTML, CSS, React.js, Node.js, MongoDB. The project will be capable of running on standard internet web browsers. The interface for the project will provide a good navigation to user of the system with nice look and feel.

4.1.Overview of the Programming Language

The combination of HTML, CSS, JavaScript, React.js, Node.js, Express.js, and MongoDB provides a comprehensive stack for building a robust, scalable, and user-friendly budget expense management system. Each technology plays a critical role in ensuring that the system meets the functional and non-functional requirements of the DMUIT Budget Expense Management System.

HTML (Hypertext Markup Language)

HTML is the standard markup language for creating web pages. It provides the structure of the webpage by using a series of elements that define different parts of the content, such as headings, paragraphs, links, images, and other multimedia. HTML is fundamental for web development as it forms the backbone of any webpage.

Key Features

- ❖ **Markup Language**: HTML is not a programming language but a markup language used to structure content on the web.
- ❖ **Elements and Tags**: Uses tags to define elements such as headings, paragraphs, links, images, and forms.
- ❖ **Standardized**: Defined by the World Wide Web Consortium (W3C) and WHATWG, ensuring consistency across different browsers.

- **Responsive Design**: Works with CSS to create responsive and adaptive web pages.

Why HTML?

HTML is fundamental for creating the structure and layout of web pages. It is essential for defining the content and organizing the user interface elements of the DMUIT Budget Expense Management System.

CSS (Cascading Style Sheets)

CSS is a stylesheet language used to describe the presentation of a document written in HTML. It allows developers to style and layout web pages, including aspects like colors, fonts, and spacing. CSS makes web pages visually appealing and ensures that they are responsive and accessible across different devices and screen sizes.

Key Features

- **Styling Language**: Used to describe the presentation of a document written in HTML.
- **Selectors and Properties**: Uses selectors to apply styles to elements and properties to define styles such as color, font, and layout.
- **Responsive Design**: Facilitates the creation of responsive web designs that adapt to different screen sizes and devices.
- **CSS Frameworks**: Supports frameworks like Bootstrap and Tailwind CSS that expedite development.

Why CSS?

CSS enhances the visual presentation and usability of the DMUIT Budget Expense Management System. It ensures that the system is visually appealing and accessible on various devices, providing a better user experience.

JavaScript

Key Features

- **Dynamic and Interactive**: Enables the creation of dynamic content that can interact with the user.
- **Event-Driven**: Supports event-driven programming, making it ideal for creating responsive interfaces.
- **Wide Usage**: Universally supported across all modern web browsers.

- **Versatile**: Can be used for both client-side and server-side development with environments like Node.js.

Why JavaScript?

JavaScript is crucial for adding interactivity and dynamic behavior to the DMUIT Budget Expense Management System. It allows for real-time updates, form validations, and asynchronous communication with the server.

React.js

React.js is a popular JavaScript library for building user interfaces, especially single-page applications. It allows developers to create reusable UI components, manage the state of an application efficiently, and handle the rendering of components dynamically. React.js simplifies the process of building complex user interfaces by using a component-based architecture. [17]

Key Features

- **Component-Based**: Allows for building encapsulated components that manage their own state and can be composed to create complex UIs.
- **Virtual DOM**: Improves performance by only updating the parts of the DOM that have changed.
- **Declarative**: Makes it easier to reason about and manage the state of the application.
- **Rich Ecosystem**: Supported by a vast ecosystem of libraries and tools.

Why React.js?

React.js is chosen for its ability to create efficient, scalable, and maintainable user interfaces. Its component-based architecture aligns well with the needs of the DMUIT Budget Expense Management System, enabling reusable and modular code.

Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser. It is designed for building scalable network applications. Node.js uses an event-driven, non-blocking I/O model, which makes it lightweight and efficient, perfect for data-intensive real-time applications.

Key Features

- **Asynchronous and Event-Driven**: Non-blocking I/O operations make it suitable for handling multiple concurrent connections.

- ❖ **Single Language**: Allows using JavaScript for both client-side and server-side development.
- ❖ **NPM (Node Package Manager)**: Provides access to a large repository of reusable modules.
- ❖ **Scalable**: Ideal for building scalable network applications.

Why Node.js?

Node.js is used for its performance and scalability in handling server-side operations. It allows for the creation of a robust backend that can manage numerous concurrent requests efficiently, which is essential for the DMUIT Budget Expense Management System.

Express.js

Key Features

- ❖ **Minimal and Flexible**: Provides a thin layer of fundamental web application features without obscuring Node.js features.
- ❖ **Middleware**: Uses middleware to handle HTTP requests and responses.
- ❖ **Routing**: Provides robust routing capabilities for managing different endpoints.
- ❖ **Extensible**: Can be easily extended with various middleware modules.

Why Express.js?

Express.js simplifies backend development by providing essential web application features. It is used to create RESTful APIs that facilitate communication between the front-end and the back-end of the DMUIT Budget Expense Management System.

MongoDB

MongoDB is a NoSQL, document-oriented database that stores data in JSON-like format. It is designed for scalability, flexibility, and performance. MongoDB allows for the storage of large volumes of data and provides powerful querying and indexing capabilities. It is particularly well-suited for applications that require a flexible schema design.

Key Features

- ❖ **Document-Oriented**: Stores data in flexible, JSON-like documents.
- ❖ **Scalable**: Designed to handle large volumes of data and high throughput.
- ❖ **Schema-Less**: Allows for a flexible schema design that can evolve with application needs.
- ❖ **Rich Query Language**: Supports powerful querying and indexing capabilities.

Why MongoDB?

MongoDB is chosen for its flexibility and scalability. Its document-oriented approach is well-suited for the dynamic and evolving data requirements of the DMUIT Budget Expense Management System. It allows for efficient storage and retrieval of complex data structures

CHAPTER FIVE

5. TESTING

Developing software is a complex process. No matter how hard we try to eliminate all faults simply by going through the development phases which is requirements elicitation, requirement analysis, system design, and implementation, however through good practice we can make sure that the most series fault does not occur in the first place. In addition, we need a separate testing phase, with the goal of elimination all remaining faults before release the system. Testing is the final phase of implementation. Testing is a process to show the correctness of the program. Testing is checking of the system workability in an attempt to discover errors and avoiding such errors from the system. In this the team members tested the entire system as a whole with all forms, code, modules. In this we tested all the functionalities in the System. All errors in the forms, functions, modules have been tested.

5.1. Sample Test

To simplify the testing process, the project team followed the different types of tests that break the testing process up into the distinct levels. These types testing is unit testing, integration testing, acceptance testing and system testing. We have seen the following different testing strategies

5.1.1. Unit Testing

Unit testing involves testing individual components or units of the system to ensure they function as expected. This is typically performed by developers during the coding phase.

Key Test Cases

1. **User Authentication**:
    - ❖ Verify that valid credentials allow access.
    - ❖ Check that invalid credentials are rejected.
    - ❖ Ensure session tokens are correctly generated and validated.
2. **Budget Allocation**:
    - ❖ Test the creation of new budget records.

- Verify updates to existing budget allocations.
- Ensure correct handling of invalid inputs (e.g., negative amounts).

3. **Expense Tracking**:

- Verify that expenses are correctly recorded.
- Ensure expenses are properly linked to the appropriate budget.
- Check validation of expense details (e.g., date, category).

5.1.2. Integration Testing

Integration testing focuses on verifying the interactions between different components of the system. This ensures that the integrated components work together as intended.

Key Test Cases

1. **API Endpoints**:

- Verify that endpoints return correct responses.
- Check error handling for invalid requests.
- Ensure correct data is returned and updated via the API.

2. **Front-End and Back-End Integration**:

- Test the interaction between the React.js front-end and the Node.js/Express.js back-end.
- Ensure data fetched from the back-end is correctly displayed in the front-end.

5.2. System Testing

System testing involves testing the entire system as a whole to ensure it meets the specified requirements. This includes functional and non-functional testing.

Key Areas

1. **Functional Testing**:

- ❖ Verify all use cases and user scenarios.
- ❖ Ensure all features work as expected.

2. **Performance Testing**:
   - ❖ Assess system performance under various load conditions.
   - ❖ Identify potential bottlenecks and optimize performance.
3. **Security Testing**:
   - ❖ Verify that the system is secure from common vulnerabilities (e.g., SQL injection, XSS).
   - ❖ Ensure proper authentication and authorization mechanisms.

5.3. User Acceptance Testing (UAT)

User Acceptance Testing is performed by the end-users to ensure the system meets their needs and requirements. This phase is crucial for gaining user approval before deployment.

Key Steps

1. **Test Plan Creation:**
   - ❖ Define test scenarios based on real-world use cases.
   - ❖ Prepare test data and environment.
2. **Execution:**
   - ❖ Users perform the tests according to the test plan.
   - ❖ Document any issues or feedback.
3. **Feedback Review:**
   - ❖ Review user feedback and address any issues.
   - ❖ Make necessary adjustments to the system.

CHAPTER SIX

CONCLUSIONS AND RECOMMENDATION

The DMUIT Budget Expense Management System project aimed to develop an efficient, user-friendly, and scalable solution to manage the budget and expenses of Debre Markos University Institute of Technology. This project involved the development of a web-based application using modern technologies and followed a structured methodology to ensure the system met its objectives.

Key Achievements

1. **User Authentication and Security**: The system provides robust user authentication and authorization mechanisms to ensure that only authorized personnel can access and manage budgetary data. The use of secure session tokens and hashed passwords enhances the security of the system.
2. **Budget and Expense Management**: The system allows for efficient allocation and tracking of budgets and expenses. Users can easily allocate funds to different departments, track expenditures, and generate detailed reports on financial activities.
3. **User-Friendly Interface**: With a responsive and interactive front-end developed using React.js, the system offers a seamless user experience. The interface is designed to be intuitive and accessible across various devices.
4. **Scalability and Performance**: The use of Node.js and Express.js for the back-end, along with MongoDB for data management, ensures that the system is scalable and can handle high volumes of data and concurrent users efficiently.
5. **Comprehensive Testing**: Extensive testing, including unit, integration, system, and user acceptance testing, was conducted to ensure the system is reliable and meets all specified requirements.

Challenges Encountered

1. **Data Integration**: Integrating data from various departments and ensuring data consistency posed a significant challenge. This was addressed by implementing robust validation and error-handling mechanisms.
2. **User Training**: Ensuring that users are comfortable with the new system required comprehensive training sessions and user documentation.
3. **Performance Optimization**: Optimizing the system to handle large datasets and multiple concurrent users required significant effort in fine-tuning the database queries and server configurations.

Recommendations

Based on the experience and findings from this project, the following recommendations are proposed for future improvements and extensions of the DMUIT Budget Expense Management System:

1. **Enhanced Reporting and Analytics**: Implement advanced analytics features to provide deeper insights into financial data. This could include trend analysis, predictive analytics, and more customizable reporting options.
2. **Mobile Application**: Develop a mobile application to complement the web-based system. This would provide users with greater flexibility and access to budget management tools on-the-go.
3. **Integration with Other Systems**: Integrate the budget expense management system with other institutional systems such as procurement, payroll, and academic management systems to streamline data flow and enhance overall efficiency.
4. **Automated Alerts and Notifications**: Implement automated alerts and notifications for budget thresholds, expense approvals, and other critical events. This would enhance proactive management and reduce the risk of budget overruns.
5. **Regular User Training and Support**: Conduct regular training sessions and provide continuous support to users to ensure they are well-versed with the system's features and functionalities. This will help in maximizing the system's usage and effectiveness.

6. **Periodic System Audits**: Conduct periodic audits of the system to identify any potential security vulnerabilities or performance issues. Regular updates and maintenance should be scheduled to keep the system secure and up-to-date with the latest technological advancements.

Future Work

Future work on the DMUIT Budget Expense Management System could focus on the following areas:

Machine Learning Integration: Explore the integration of machine learning algorithms to predict budget trends and anomalies, providing users with more intelligent and data-driven decision-making tools.

Cloud Deployment: Consider deploying the system on a cloud platform to leverage the benefits of scalability, flexibility, and reduced infrastructure costs. Cloud services also offer enhanced security and disaster recovery options.

User Feedback Incorporation: Continuously gather user feedback to identify areas for improvement. Incorporating user suggestions can lead to a more user-centric system that better meets the needs of its users.

Multilingual Support: Implement multilingual support to cater to a diverse user base, ensuring that the system is accessible to users who speak different languages.

References

[1] Anthony, Traditional approaches like zero-based budgeting and incremental budgeting have given way to more flexible models, 2003.

[2] young, Traditional approaches like zero-based budgeting and incremental budgeting have given way to more flexible models, 2003.

[3] Smith, The transformed the efficiency and effectiveness of budget and expense management (Smith & Jones, 2018; Brown et al., 2019)., 2018.

[4] C. e. al, Automated budgeting systems have demonstrated numerous advantages. Chen et al. (2020), 2020.

[5] Jone, The ability to track expenses in real-time is a key benefit, providing organizations with a more accurate and timelier financial picture (Jones & Williams, 2017)., 2017.

[6] Security concerns, especially in cloud-based solutions, have been identified (Lee & Kim, 2016)., 2016, Lee.

[7] Harrison, Employee resistance during the implementation phase is also a noteworthy challenge (Harrison et al., 2015), 2015.

[8] Davis, The seamless integration of budgeting systems with financial reporting is imperative for organizations seeking transparency and accuracy in financial information (Davis & Miller, 2019)., 2019.

[9] Brown, The role of mobile applications in expense management is gaining prominence. Brown and Smith (2021), 2021.

[10] Turing, Researchers are increasingly exploring the development of intelligent systems using Artificial Intelligence (AI) and machine learning algorithms (Turing & Wallace, 2022)., 2022.

[11] Black, Black et al. (2019) shed light on the challenges associated with customization and scalability., 2019.

[12] K. a. Patel, Kim and Patel (2018) delve into the challenges associated with organizational change and the implementation of these advanced systems., 2018.

[13] Donald, Donald Bell, IBM Global Services, UML basics: An introduction to the Unified Modeling.

[14] W3Schools., W3Schools. (2021, August 21). HTML. Retrieved September 11, 2022, from.

[15] Node.js Official Documentation. Retrieved from https://nodejs.

[16] MongoDB Official Documentation. Retrieved from https://docs.mongodb.com/.

[17] React Official Documentation. Retrieved from https://reactjs.org/docs/getting-started.html.

[18] (. &. Young)., 2003.

[19] it, rom OOSAD IT 2nd year handout.

APPENDIX

```
import "./Login.css";
import React, { useState } from "react";
import axios from "axios";
import Header from "./Header";
import { useNavigate } from "react-router-dom";
import Footer from "./Footer";
function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [directory, setDirectory] = useState("");
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [loginError, setLoginError] = useState("");
  const navigate = useNavigate();
  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post("http://localhost:8000", {
        email,
        password,});
      if (response && response.data) {
        const { directory } = response.data;
        // Store the token and directory in localStorage or use them for further API calls as needed
        console.log("Directory:", directory);
        setDirectory(directory);
        setIsLoggedIn(true); // Set isLoggedIn state to true upon successful login
      } else {
        console.error("Invalid response:", response);
        setLoginError("Invalid response received"); }
  } catch (error) {
    console.error(
      "Error during login:",
      error.response ? error.response.data.error : error.message
    );
    setLoginError("Username or password is incorrect");}  };
// Render the appropriate module (JavaScript file) based on the directory
const renderModule = () => {
  if (isLoggedIn) {
    switch (directory) {
      case "Software and Website Adminstrator Directorate":
        navigate("/software-and-website-adminstrator-directorate");
        break;
      case "ICT InfrastructureInstallation and Admnistration Directorate":
        navigate(
          "/ict-infrastructureinstallation-and-admnistration-directorate"
        );
        break;
      case "Science Technology Human Resource and Institute Capacity Directorate":
        navigate(
          "/science-technology-human-resource-and-institute-capacity-directorate"
        );
        break;
      case "Human Resource Admninstration Development Directorate":
        navigate("/human-resource-admninstration-development-directorate");
        break;
      case "Record and Folder Management Directorate":
        navigate("/record-and-folder-management-directorate");
        break;
      case "Budget Plan Preparation and Follow-Up Evaluation Directorate":
        navigate(
          "/budget-plan-preparation-and-follow-up-evaluation-directorate"
        );
        break;
      case "Science and Information Technology Head":
        navigate("p/science-and-information-technology-head");
        break;
      default:
        return <h2>Unknown Directory</h2>;
    }
  } else {
    return (
      <div className="login-container">
        <Header />
        {/* <div className="welcome-container">
          <h1>Welcome to DMUIT Budget Expense Management System</h1>
        </div> */}
        <div class="content">
          <h1>
            Welcome To
            <p className="para">DMUIT Budget Expense</p>
            <p>Management System</p>
          </h1>
        </div>
        <div className="login-form">
          <h1 className="login">Login</h1>
          <form className="form" onSubmit={handleLogin}>
            <input
```

```
            type="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            placeholder="Email"
            required
          />
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            placeholder="Password"
            required
          />
          <div className="btn_container">
            <button className="btn" type="submit">
              Login
            </button>
          </div>
        </form>
        {loginError && <p className="error-message">{loginError}</p>}
      </div>
      <Footer />
    </div>  );  } };
  return <div className="Login">{renderModule()}</div>;
}
export default Login;
```