

**TRIBHUVAN UNIVERSITY**

**FACULTY OF HUMANITIES AND SOCIAL SCIENCE**

**A Project Report**

**On**

**"On Check: Accounting System"**

**Submitted to**

**Department of Computer Application**

**National College of Computer Studies**

In partial fulfillment of the requirements for Bachelor Degree in Computer Application

**Submitted By:**

Prince Raj Shrestha (6-2-551-67-2022)

Ronak Maharjan (6-2-551-81-2022)

**Under the Supervision of**

**Mr. Nabraj Bahadur Negi**



**TRIBHUVAN UNIVERSITY**

**FACULTY OF HUMANITIES AND SOCIAL SCIENCE**

**National College of Computer Studies**

**SUPERVISOR’S RECOMMENDATION**

I hereby recommend that this project prepared under my supervision by Prince Raj Shrestha and Ronak Maharjan entitled “Accounting System” in partial fulfillment of the requirements 4th degree of Bachelor of Computer Application is recommended for the final evaluation.

Signature of the Supervisor

Mr. Nabraj Bahadur Negi

Faculty

Member

Department of Computer Application

National College of Computer Studies

Paknajol, Kathmandu

****

**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

**National College of Computer Studies**

**LETTER OF APPROVAL**

This is to certify that this project prepared by Prince Raj Shrestha and Ronak Maharjan entitled “**On Check: Accounting System**” in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| **SIGNATURE of Supervisor**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Mr. Nabraj Bahadur Negi Faculty  Department of Computer Application National College of Computer Studies Paknajol, Kathmandu | **SIGNATURE of HOD/ Coordinator**    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Mr. Rajan Poudel Faculty Member  Department of Computer Application National College of Computer Studies Paknajol, Kathmandu |
| **SIGNATURE of Internal Examiner**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **SIGNATURE of External Examiner**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

# ABSTRACT

The “On Check: Accounting System” project aims to develop an efficient and user-friendly Desktop application for facilitating various key process on accounts like Recording, Classifying, Summarizing and Financial Report with the help of JAVA. The system provides a comprehensive solution to address the challenges associated with traditional or manual way of accounting system and makes the accounting process faster, accurate and easier. The main characteristics of this System are covered in this document, such as user registration and authentication, record of transactions, various classification like balance sheets and profit and loss account, secure data storage, and lastly financial report or history. This project's development process followed the RAD model. It also showcases the development stack's components, including database management systems with MySQL and Xammp, spring framework, and other pre-defined frameworks. The suggested the application will have an easy-to-use interface, simple navigation, and strong with secured backend features to guarantee effective records of transaction, and user accounts.

***Keywords: JAVA, RAD model, MySQL, Xampp***

# ACKNOWLEDGEMENT

This Report entitled “On Check” is prepared in the partial fulfillment of the requirements of Bachelor in Computer Application. It is the result of cooperation and support of our supervisor and teacher. Most importantly, we would like to give thanks to the Santosh sir for providing the opportunity to provide the guidance for making the project. We would like to express my sincere thanks to our supervisor, Mr. Nabraj Bahadur Negi, for providing suggestions and guidance for preparing this report. Your precious time has certainly helped me in giving the meaning to our project work. We highly appreciate the opinion and constructive suggestions from the teachers and intellectuals.

Prince Raj Shrestha

Ronak Maharjan

**Table of Contents**

[ABSTRACT iv](#_Toc190969202)

[ACKNOWLEDGEMENT v](#_Toc190969203)

[LIST OF ABBREVIATIONS viii](#_Toc190969204)

[LIST OF FIGURES ix](#_Toc190969205)

[LIST OF TABLES x](#_Toc190969206)

[Chapter 1. Introduction 1](#_Toc190969207)

[1.1 Introduction to Project 1](#_Toc190969208)

[1.2 Problem Statement 1](#_Toc190969209)

[1.3 Objectives 2](#_Toc190969210)

[1.4 Scope and Limitation 2](#_Toc190969211)

[1.5 Report Organization 3](#_Toc190969212)

[Unit 1. Introduction 3](#_Toc190969213)

[Unit 2. Background study and literature review 3](#_Toc190969214)

[Unit 3. System analysis and design 3](#_Toc190969215)

[Unit 4. Implementation and testing 3](#_Toc190969216)

[Unit 5. Conclusion and future recommendation 3](#_Toc190969217)

[Chapter 2. Background Study and Literature Review 4](#_Toc190969218)

[2.1 Background Study 4](#_Toc190969219)

[2.1.1 Study of existing system 4](#_Toc190969220)

[2.2 Literature Review 4](#_Toc190969221)

[Chapter 3. System Analysis and Design 7](#_Toc190969222)

[3.1 System Analysis 7](#_Toc190969223)

[3.1.1 Requirement Analysis 7](#_Toc190969224)

[3.1.2 Feasibility Analysis 9](#_Toc190969225)

[3.1.3 ER-Model Diagram 10](#_Toc190969226)

[3.1.4 Data Flow Diagram 10](#_Toc190969227)

[3.2 System design 13](#_Toc190969228)

[3.2.1 Architectural Design 13](#_Toc190969229)

[3.2.2 Flowchart 14](#_Toc190969230)

[3.2.3 Database Schema 14](#_Toc190969231)

[3.2.4 Interface Design 15](#_Toc190969232)

[3.2.5 Physical DFD 16](#_Toc190969233)

[Chapter 4. Implementation and Testing 17](#_Toc190969234)

[4.1 Implementation 17](#_Toc190969235)

[4.1.1 Tools used 17](#_Toc190969236)

[4.1.2 Implementation of Modules 17](#_Toc190969237)

[4.2 Testing 47](#_Toc190969238)

[4.2.1 Testing case for unit testing 47](#_Toc190969239)

[4.2.2 Testing case for System testing 48](#_Toc190969240)

[Chapter 5. Conclusion and Future Recommendation 49](#_Toc190969241)

[5.1 Lesson learned 49](#_Toc190969242)

[5.2 Conclusion 49](#_Toc190969243)

[5.3 Future Recommendation 50](#_Toc190969244)

[References 51](#_Toc190969245)

[Appendix 52](#_Toc190969246)

# LIST OF ABBREVIATIONS

AI: Artificial Intelligence

BEA: Business Entity Assumptions

DFD-Data Flow Diagram

ER-Model: Entity Relation Model

GCA: Going Concern Assumption

PA: Accounting Period or Periodicity Assumption

CSS-Cascading Style Sheet

# LIST OF FIGURES

[Fig 3.1: Use case diagram of Oncheck 8](#_Toc190969357)

[Fig 3.2: E-R Diagram of OnCheck System 10](#_Toc190969358)

[Fig 3.3: DFD level 0 of OnCheck System 10](#_Toc190969359)

[Fig 3.4: DFD level 1 of OnCheck System 11](#_Toc190969360)

[Fig 3.5: DFD level 2 of OnCheck System 12](#_Toc190969361)

[Fig 3.6: Server Client model of Oncheck System 13](#_Toc190969362)

[Fig 3.7: Flowchart of Oncheck system 14](#_Toc190969363)

[Fig 3.8: Database Schema of OnCheck System 14](#_Toc190969364)

[Fig 3.9: Interface design for Accountant of OnCheck 15](#_Toc190969365)

[Fig 3.10: Interface Design for Viewer of OnCheck 15](#_Toc190969366)

[Fig 3.11: Physical DFD of OnCheck System 16](#_Toc190969367)

[Fig 4.1: login module 19](#_Toc190969368)

[Fig 4.2:Register module 19](#_Toc190969369)

[Fig 4.3: journal entry module 25](#_Toc190969370)

[Fig 4.4: Payment module 27](#_Toc190969371)

[Fig 4.5: Receipt Module 29](#_Toc190969372)

[Fig 4.6: Jourmal Report Module 32](#_Toc190969373)

[Fig 4.7: Legder main menu module 37](#_Toc190969374)

[Fig 4.8: Income Expenses Report module 41](#_Toc190969375)

[Fig 4.9: Trial Balance module 47](#_Toc190969376)

[Fig 7.1:Main Menu 52](#_Toc190969377)

[Fig 7.2: Display Menu 52](#_Toc190969378)

[Fig7.3: Ledger Report 53](#_Toc190969379)

# LIST OF TABLES

[Table 4.1: Tools Used for OnCheck System 17](#_Toc183880078)

[Table 4.2: Unit testing of OnCheck 47](#_Toc183880079)

[Table 4.3:System testing of Oncheck 48](#_Toc183880080)

# Introduction

## Introduction to Project

A business's accounting system is its brains and heart. “On Check” is a system that enables the company to record and track expenses and manage transaction, balance sheet, p/l account and so on. Businesses may find it challenging to document daily transactions, which makes it challenging for the accounting department to monitor the company. An accounting system optimizes all of those processes for the department and creates a smooth flow of business accounts, which serve as the foundation of the organization. This system monitors both single-entry and double-entry accounting. Single-entry accounting records all basic accounting transactions, while double-entry accounting records all necessary information for items such as balance sheets and profit and loss accounts that impact both parties.

It is a computerized system that runs on a server. To store every record, the user has to sign in, which makes the connection and data transmission and storage relatively secure. It's a secure, non-profitable system that records transactions accurately and efficiently and aids in the processing of financial reports. It keeps to the latest accounting practices yet preserves the viability of the single entry system, which aids in any organization's analysis of revenue development and expansion for future reference and potential corporate growth.

## Problem Statement

In today’s competitive business landscape, many organizations still rely on traditional, manual accounting systems that are time-consuming, prone to human error, and inefficient in handling the complexities of modern financial operations. These outdated systems often result in inaccurate financial reporting, delayed decision-making, and compliance risks due to missed deadlines or improper documentation. Additionally, the lack of real-time financial data makes it difficult for businesses to monitor their financial health and make informed strategic decisions. There is a clear need for an advanced, automated solution that not only streamlines accounting processes but also provides accurate, up-to-date financial information. The ON CHECK Accounting System addresses these challenges by offering an integrated platform that reduces human errors, ensures compliance, and enables businesses to manage their finances more effectively. By providing real-time financial insights and automating routine tasks, ON CHECK allows organizations to focus on growth and decision-making rather than time-consuming administrative tasks.

## Objectives

The main objective of ON CHECK is:

1. To identify the limitations of traditional accounting methods and highlight the necessity of implementing an automated system to improve financial accuracy and efficiency.
2. To provide a detailed overview of the ON CHECK Accounting System’s features, explaining how it streamlines accounting tasks, reduces errors, and improves decision-making through real-time financial insights.

To outline the potential cost savings associated with adopting the ON CHECK system, such as reduced labor costs, minimized errors, and improved operational efficiency.

## Scope and Limitation

Accounting System is a useful tool to manage user’s transactional records and analyze it for further future prediction and to provide financial status of the business. It also helps in decision making of the business as it helps in knowing where business stands in right now at time. It has many scopes which it functions in such as:

1. Recording Information: It is used for storing transactional process of a user or a whole organization, which are measurable in terms of money.
2. Analyzing transaction: It is mostly used to analyze the transaction that have been made to such as balance sheet and p/l account for office and users for making the user understand their financial situation.
3. Classification of Data: The recorded data is arranged in a manner so as to group the transactions of similar nature at one place so that full information of these items may be collected under different heads.

Accounting System is concerned with the preparation of final account. At the most the system can reveal what has happened so far, but it cannot exercise any control over the past happenings. Some of the limitation of it are as:

1. It can only record quantitative records.
2. It does not take into account of price level change.
3. It does not provide the cost of products manufactured.
4. There is No Future Assessment and further decision should be made of future by the user and the organization.

## Report Organization

### Introduction

Chapter One introduce the concept of the project. It explains the issue and the need for it. It also discusses the goals behind the creation of the ON CHECK. It also illustrates the limitations and the extent to which it can assist us.

### Background study and literature review

It generally optimizes the basic explanation of the project's wide concept, supporting theories, and terminology. It additionally provides a few reviews and theories from other researchers that are comparable.

### System analysis and design

This chapter covers the process of gathering requirements, including functional and non-functional project requirements. It also covers the feasibility study, which is primarily based on technical, operational, economical, and schedule studies, as well as project design. Diagrams, functionality analysis, a method for obtaining requirements, and a process model are all included.

### Implementation and testing

This chapter attempts to provide details regarding the project's implementation, the variety of CASE tools that were used, and the kinds of testing that the project underwent during the system creation phase.

### Conclusion and future recommendation

This chapter includes the possible outcome of this project, conclusion and future recommendations.

# Background Study and Literature Review

## Background Study

The history of accounting and money itself date back thousands of years. In contrast, the [1](double entry bookkeeping system and the professionalization of accounting) were the results of centuries of thinking, practice, and action, which gave rise to the present accounting system, which has been in place almost since the [2]1800s(19th century)AD. Large-scale businesses and organizations may now easily manage and keep their records using the current system, which offers a comfortable accounting interface. It facilitates the smooth and ongoing growth of enterprises by simplifying decision-making processes and enhancing the effectiveness of their expert systems. The demand for simpler systems with user-friendly interfaces has increased as technology has developed. These systems should enable both small and large organizations to profit from them and utilize them for their advantage.

### Study of existing system

In order to conduct a system study, numerous programs and websites were examined and tested. These websites consist of [3]tally, [4]Quick Books, and so forth. These companies' websites and software offer an extensive range of capabilities that are simple to use and interpret. These websites enable users to save recordings on servers with more secure cloud systems, access them, and enlarge them as needed. Users can add transactions using these systems in accordance with their needs, and they also support double entry systems for businesses.

## Literature Review

An exploratory review of the body of existing accounting literature is the research methodology used. It has been noted that the development of these ideas aims to reduce corporate asset theft, fraud, mistakes, and misappropriations [5]. To prevent unclear and scandalous financial statement reporting, it is advised that the International Financial Reporting Standard Committee's and practicing accountants' implementation guidelines for these theories be followed and made simpler.

As technology has advanced, accounting systems have evolved from traditional to modern and from modern to virtual in order to handle all accounting tasks. These days, it is a widely utilized system all over the world that helps users perform bookkeeping more quickly and easily. There are underlying theories that play a significant role in how income is defined and structured. [5] The theories of real income can be seen as BEA which state business and owner are separate which means system records the only part of business seems owner as a part of a business, GCA which state the ongoing business unless it is terminated, PA which state that the continuity of time can be sub divided into number of discrete part. Accounting systems have a significant advantage over all other systems since they increase productivity, decrease human error, increase accuracy, can be scaled, and, depending on the system being used, offer an adequate to strong security system. Even yet, there are still issues that an accounting system must deal with, including implementation costs, data mitigation, system integration, risk related to cyber security, and integration with multiple devices.

An accounting system's procedures, usually, this procedure involves inputting data into the accounting system, which keeps it in a database, such as sales invoices and receipts.[4] An accounting system consists of five essential processes:

1. Financial activities like as sales, expenses, equipment purchases, loans, and asset sales must be recorded as the initial step in an accounting system.
2. Classifying: In this step, you will use categories such as revenue, expenses, assets, and liabilities to group the transactions into different groups.
3. Summarizing: Usually done in the form of financial statements, this is the third phase in the accounting system process.
4. Analyzing: These reports need to be examined once the transactions have been
5. Financial reporting: This is the final step in the procedure. Relevant stakeholders will get financial information from you, including financial statements and analysis.

These fundamental categories are essential for evaluating an accounting system's features in order to make sure the program satisfies the requirements of all sizes of firms. [3]Below is a summary of every feature:

1. Accounting is essential for keeping track of every transaction and preserving the trial balance, income statement, and balance sheet.
2. Standard reports like as cash flow, income, and balance sheets are part of financial management and controls.
3. Costing: keeping an eye on cost centers and spotting deviations from projected expenses.
4. Sales & Receivables: Producing reports on customers' past transactions and outstanding balances.
5. Purchases & Payable: managing vendor invoices and keeping track of payments due to vendors.
6. Access control and security: fine-grained management of who can view and alter financial data.

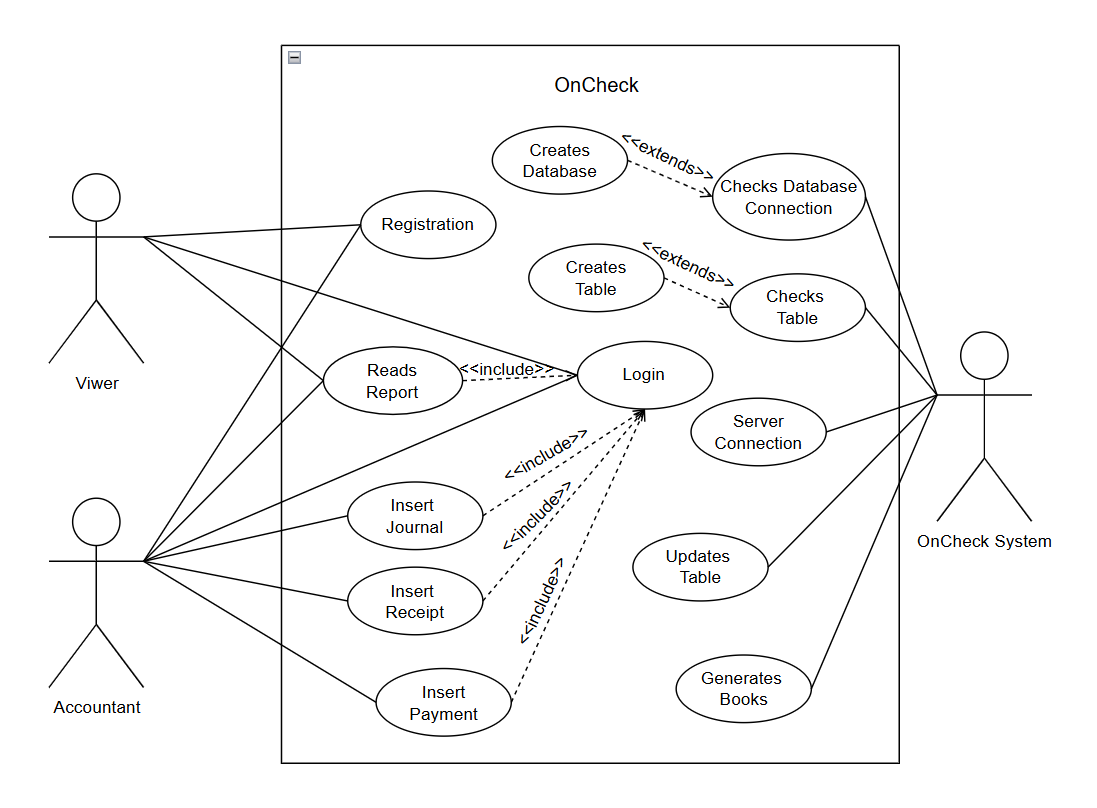
The amount of research on accounting systems emphasizes how essential they are to present business operations, providing advantages like accuracy, efficiency, and regulatory compliance. Cost, integration, and security issues, however, need to be properly handled. Accounting systems will become more efficient and flexible in the future because to emerging technologies like AI and cloud computing.

# System Analysis and Design

## System Analysis

### Requirement Analysis

1. **Functional Requirement:**
   1. User authentication module:
      * Login: Verify the user's login information (password and username). Permit access according to the role of the user (Accountant, User).
      * Registration: Permit new users to register by giving them the required information. Make sure user credentials are stored securely in the respective databases.
   2. Accounting operation:
      * Journal management: Create, access, and change journal entries. For every transaction, record the account type, amount, credit, and debit.
      * Payment management: Record each payment together with the date, amount, and description.
      * Receipt management: Create and store transaction receipts.
      * Financial reports: Create reports for the Trial Balance Income and Expense Statement, Journal Book, and Ledger Book. Permit users to designate a range of dates for the creation of reports.
   3. Database management:
      * Server Connection: Before utilizing the system, make sure the server is available.
      * Table management: As per needed, create, modify, and validate database and its tables.
      * Data Integrity: All user and transaction data are securely stored in the database.
   4. Administrative control:
      * Set up database for user to connect.
      * Update or delete existing data.
      * Resolve and check for database connection issues.
2. **Use Case Diagram**



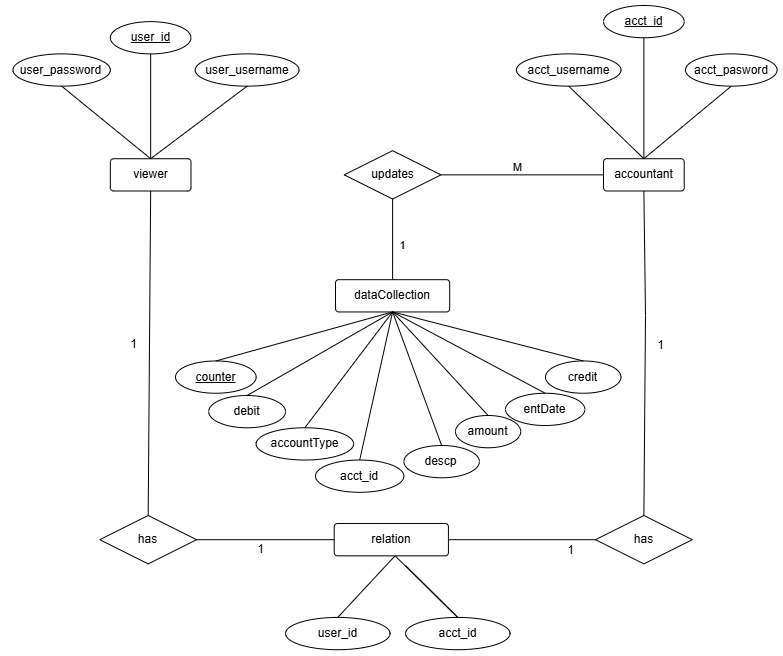
**Fig 3.1: Use case diagram of Oncheck**

1. **Non-Functional Requirement:**
   1. Performance: The system should be able to support several users at once without experiencing any performance issues. Report-generating queries should run in reasonable amounts of time.
   2. Security: Use a simple database security mechanism to store all passwords. Separate roles (Accountant, User) to prevent unwanted access.
   3. Usability: User-friendly interface that makes it easy for users to access modules including transaction viewing, report production, and journal writing. Clear error messages and notifications for specific issues.
   4. Maintainability: Future modules or improvements, like support for more financial reports or external integrations, ought to be supported by the system.

### Feasibility Analysis

1. Technical Feasibility: Standard technologies like database systems (like MySQL) and programming languages (like Java) must be installed and are widely supported for the system to function. Java, SQL, JavaFX, and CSS are examples of modern app development tools that offer enough support to create an interface that is easy to use. Tools for monitoring, backups, and validation can help reduce the risk of database corruption and server connectivity problems.
2. Operational Feasibility: Important user requirements including journal administration, payment tracking, and report generating are met by the system. The software may achieve high user acceptance with proper training and an intuitive UI.
3. Economic Feasibility: Hardware and Infrastructure Costs: Server hosting, database storage, and workstations for librarians. Software Development Costs: Developer salaries, software licenses, and integration with barcode scanners (if required). Maintenance Costs: System updates, bug fixes, and data backups.
4. Schedule Feasibility: Requirement Analysis (2–3 weeks): Gathering and documenting system needs. System Design (3–4 weeks): Creating UI/UX mockups, database structures, and system flow diagrams. Development Phase (6–8 weeks): Implementing core features like book cataloging, user registration, and transaction handling. Testing Phase (3–4 weeks): Conducting unit testing, system testing, and user acceptance testing (UAT). Deployment and Training (2–3 weeks): Installing the system, training staff, and addressing initial feedback.

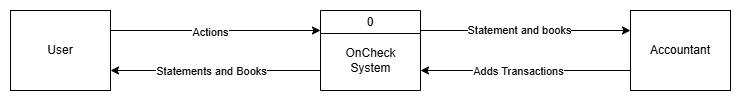
### ER-Model Diagram



**Fig 3.2: E-R Diagram of OnCheck System**

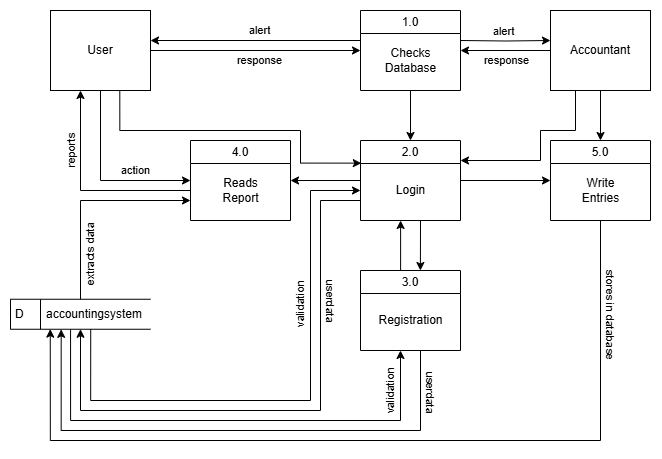
### Data Flow Diagram

Level 0 DFD:



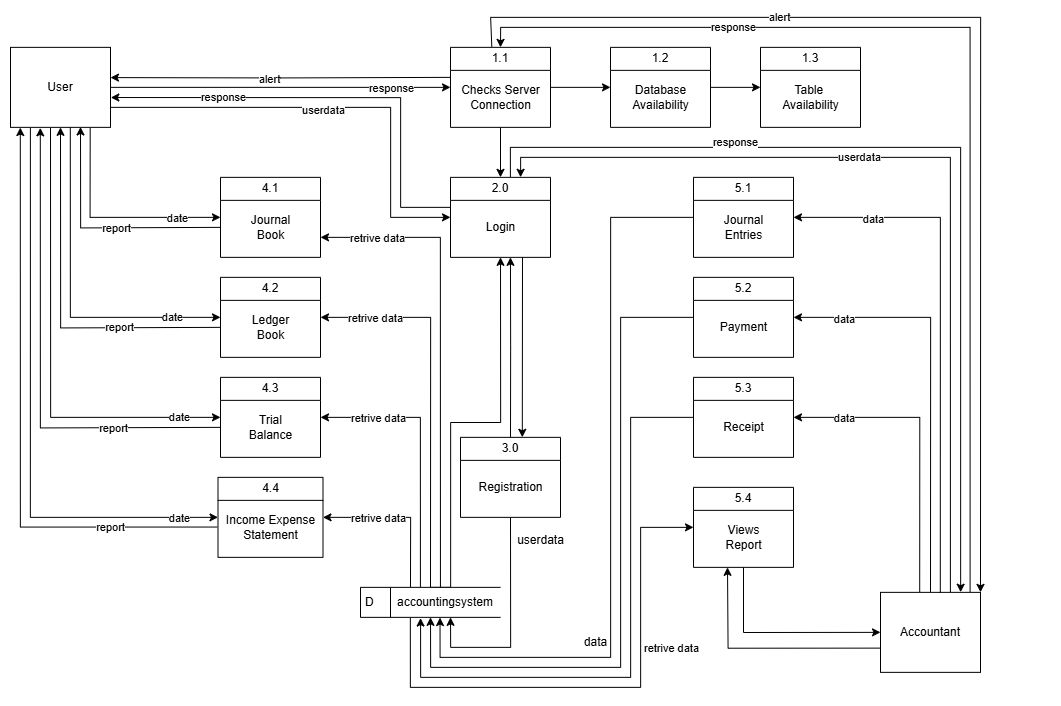
**Fig 3.3: DFD level 0 of OnCheck System**

Level 1 DFD:



**Fig 3.4: DFD level 1 of OnCheck System**

Level 2 DFD:



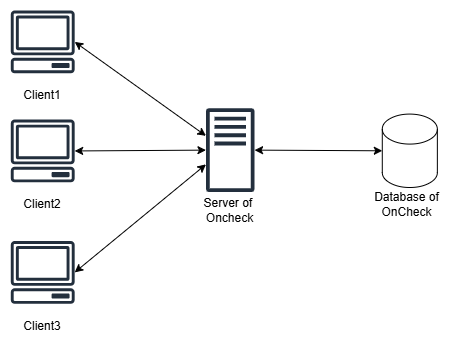
**Fig 3.5: DFD level 2 of OnCheck System**

## System design

### Architectural Design

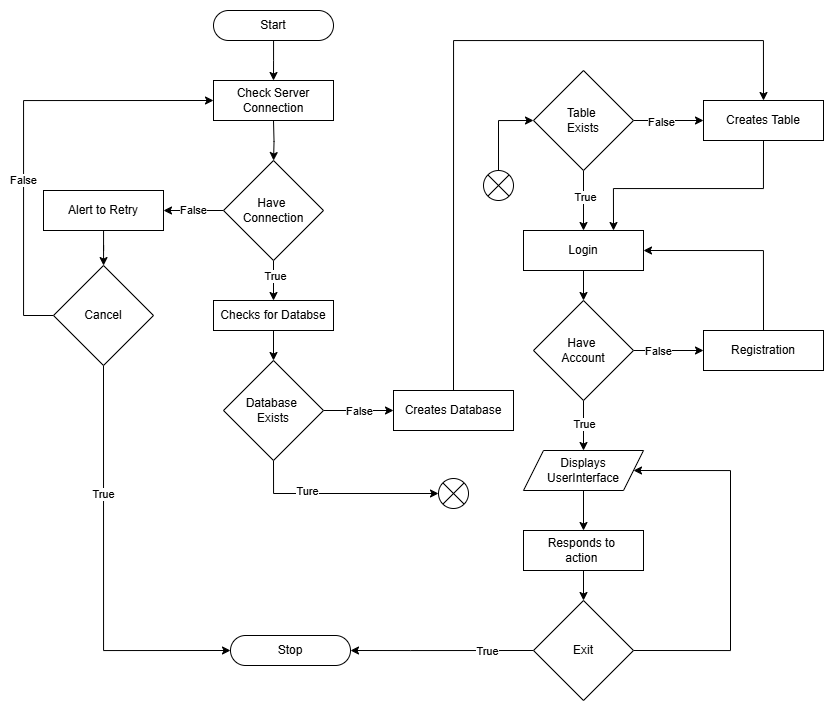
The client-server architecture of the OnCheck Accounting System allows for effective accounting process management while maintaining scalability and security. The database and application logic are hosted by the server, which serves as the system's central component and enables the safe processing and storing of financial data, including journal entries, payments, receipts, and reports. The client, on the other hand, gives users an easy-to-use interface via which they can submit queries to the server and get prompt answers. As the company expands, this paradigm fosters collaboration and scalability by supporting numerous users at once. Role-based access controls safeguard sensitive data, and centralized data management makes backups, recovery, and regulatory compliance simple.

Since updates are managed on the server, there are less interruptions for the clients. Additionally, the system facilitates real-time cooperation, guaranteeing smooth communication between accountants, and other users.



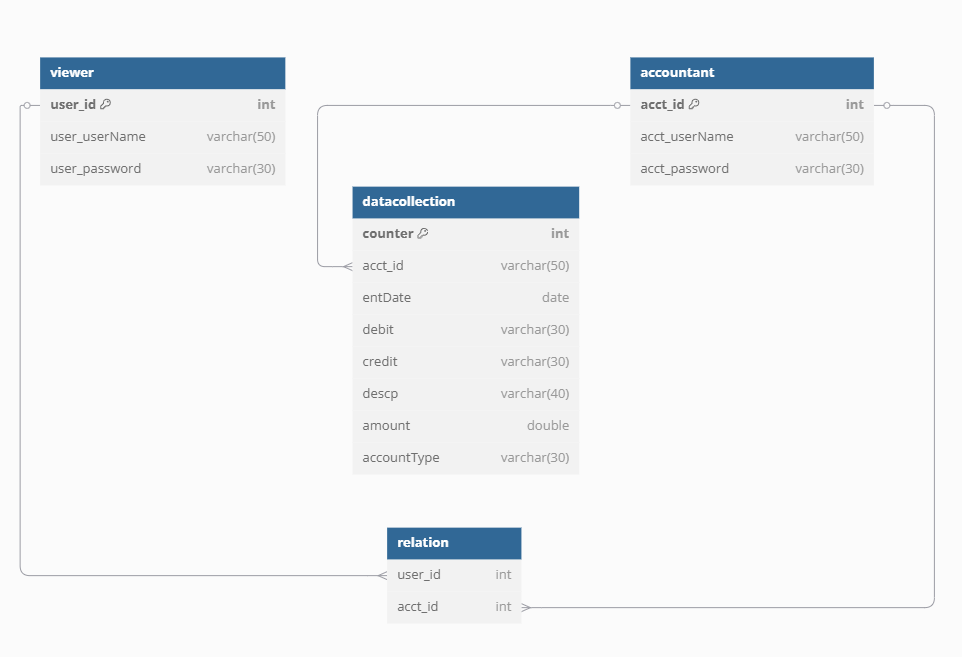
**Fig 3.6: Server Client model of Oncheck System**

### Flowchart



**Fig 3.7: Flowchart of Oncheck system**

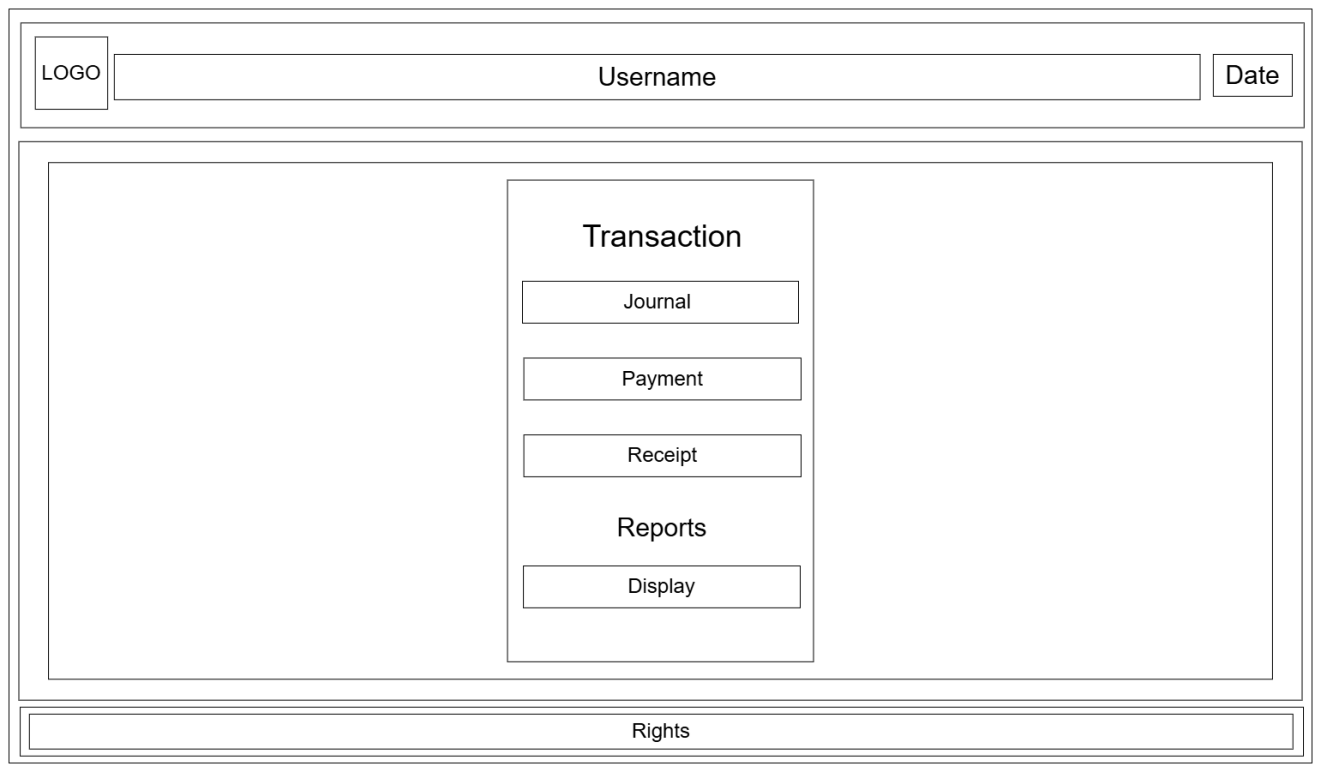
### Database Schema



**Fig 3.8: Database Schema of OnCheck System**

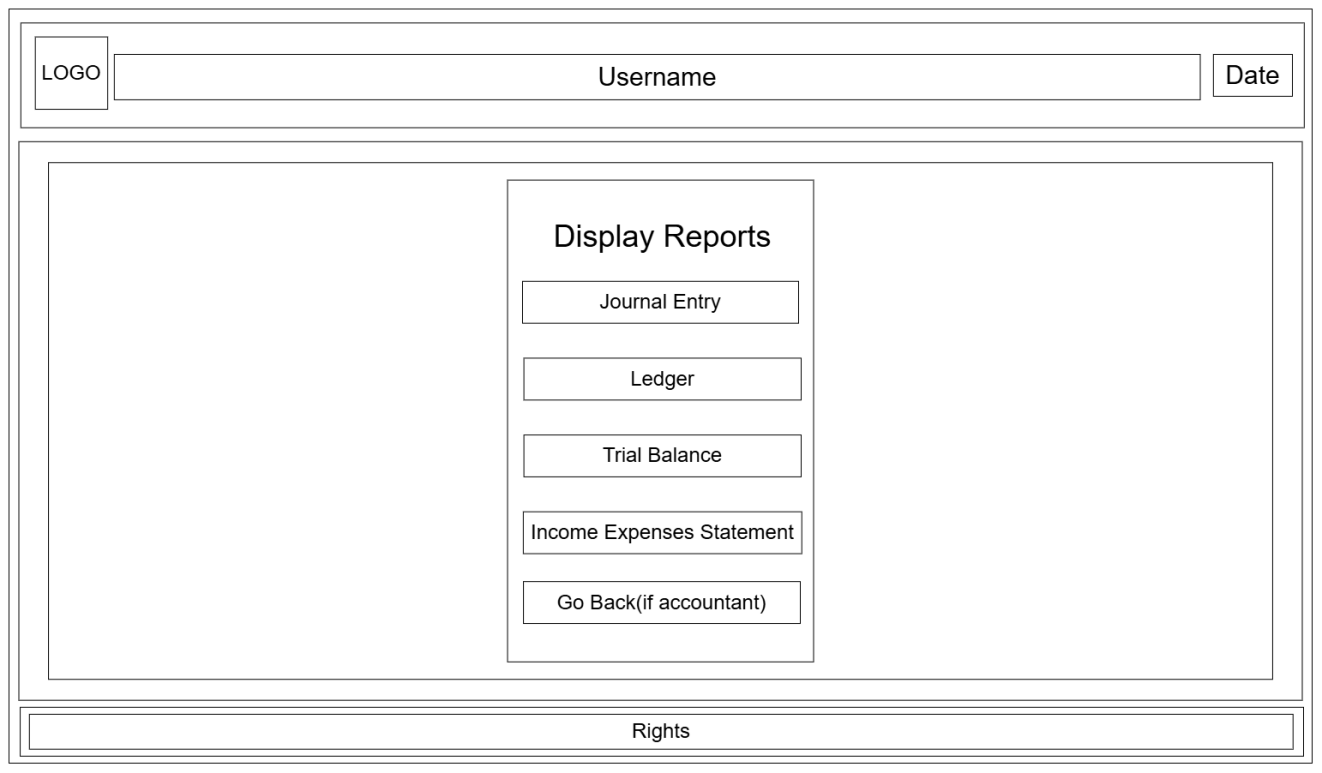
### Interface Design

1. Interface Design for Accountant:



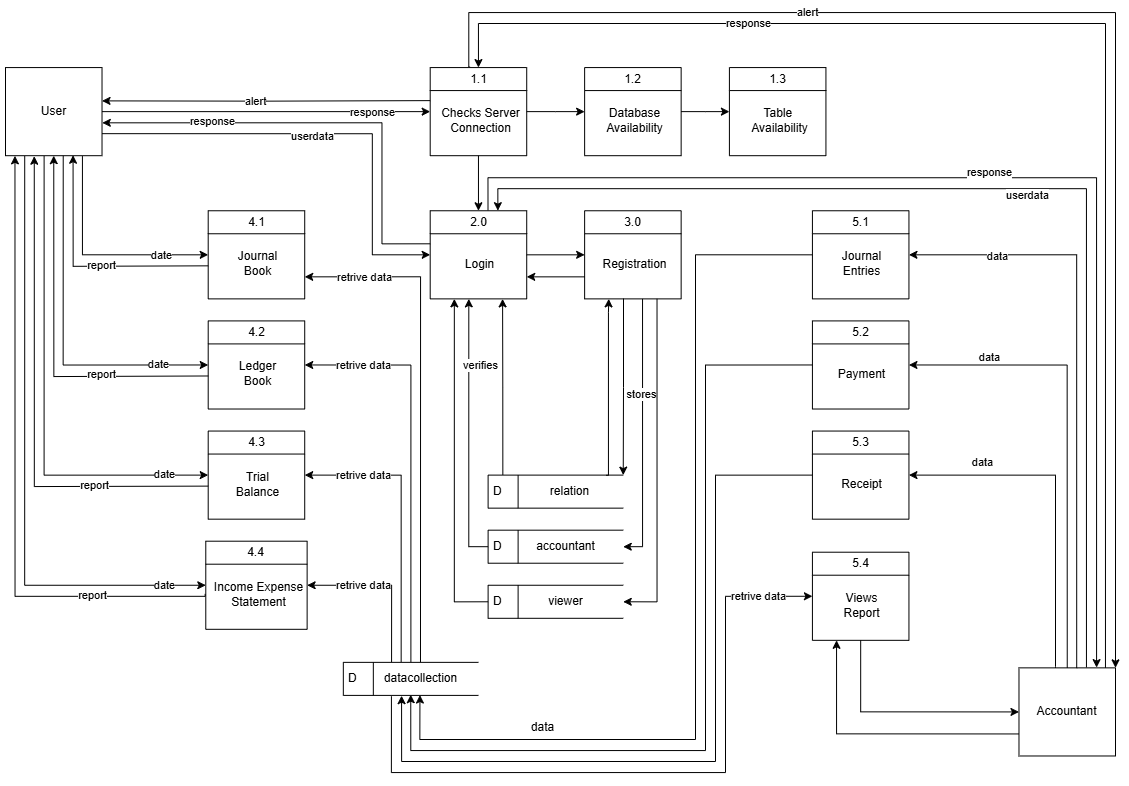
**Fig 3.9: Interface design for Accountant of OnCheck**

1. Interface Design for Viewer:



**Fig 3.10: Interface Design for Viewer of OnCheck**

### Physical DFD



**Fig 3.11: Physical DFD of OnCheck System**

# Implementation and Testing

## Implementation

### Tools used

**Table 4.1: Tools Used for OnCheck System**

|  |  |  |  |
| --- | --- | --- | --- |
| Frontend | Backend | Case Tool | Database Platforms |
| Javafx, Css | Java, MySQL | Vs-code, Draw.io, dbdiagram.io, Word, PowerPoint | Xampp, workbench |

### Implementation of Modules

1. **Login/Registration Module:**

This module is Responsible for registering or Login of the user or accountant respectively.

@SuppressWarnings("unused")

private void checkAccount(ActionEvent event) {

action = new databaseAction();

String username = userNameField.getText().trim();

String password = passwordField.getText().trim();

userNameField.setOnMouseClicked(event2 ->{

if(errorMessage.isVisible()){

errorMessage.setVisible(false);

}

});

passwordField.setOnMouseClicked(event2 ->{

if(errorMessage.isVisible()){

errorMessage.setVisible(false);

}

});

if (username.isEmpty()) {

errorMessage.setVisible(true);

errorMessage.setText("Username is empty.");

return;

}else if(password.isEmpty()){

errorMessage.setVisible(true);

errorMessage.setText("Password is empty.");

return;

}

LinkedNode.setacctLoginStat(action.readAccountant(username, password));

if (LinkedNode.getAcctLoginStat()) {

LinkedNode.setUserName(username);

goMain(event);

} else {

LinkedNode.setuserLoginStat(action.readUser(username, password));

if (LinkedNode.getUserLoginStat()) {

LinkedNode.setUserName(username);

goMain(event);

} else {

errorMessage.setVisible(true);

errorMessage.setText("The credentials are invalid! Please try again.");

}

}

}

public boolean readAccountant(String username, String password) {

String sql = "SELECT acct\_password FROM accountant WHERE acct\_userName = ? AND acct\_password = ?";

try (Connection connection = databaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(sql)) {

statement.setString(1, username);

statement.setString(2, password);

try (ResultSet result = statement.executeQuery()) {

return result.next();

}

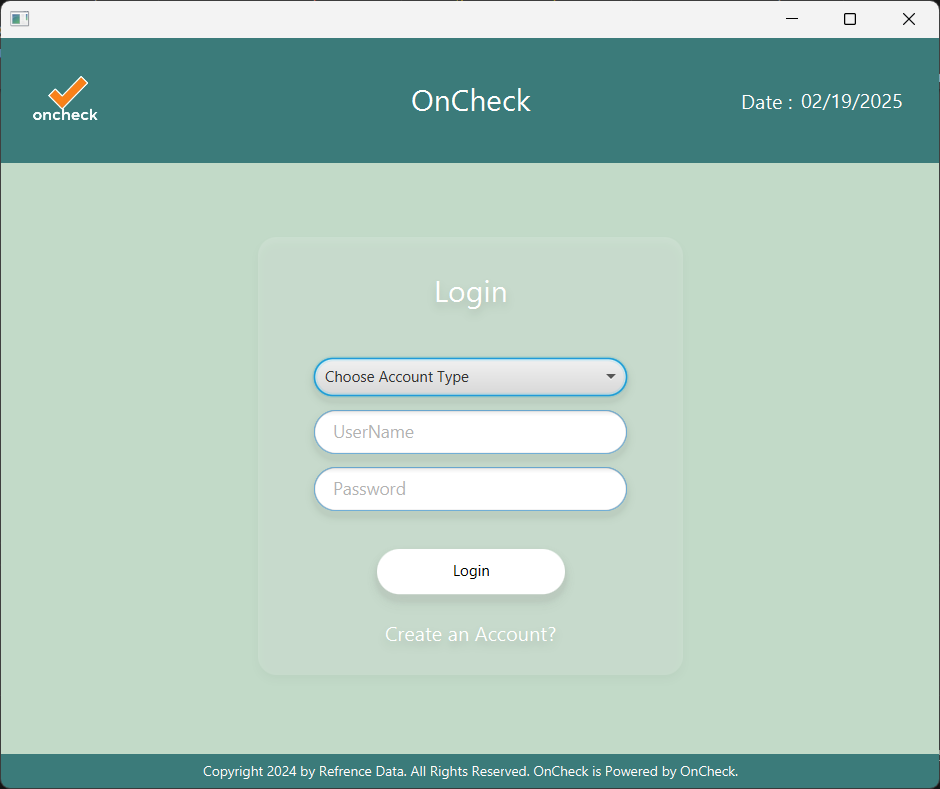
} catch (SQLException e) {

e.printStackTrace();

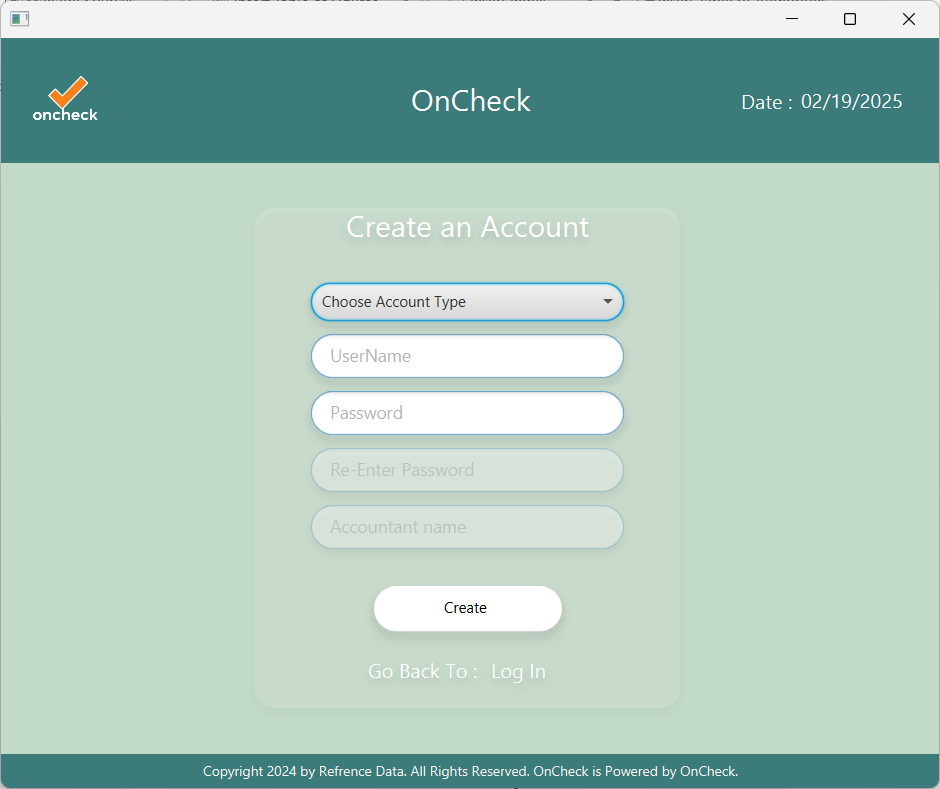
}

return false;

}



**Fig 4.1: login module**



**Fig 4.2:Register module**

1. **Operation Module:**
   1. **Journal Entry:**

This module handles the operation for all journal entries.

private void addJournalEntry(){

// journalEntry.setVisible(true);

account.setVisible(false);

chooseAccount.setVisible(false);

accountType.setLayoutY(12);

accountTypeLabel.setLayoutY(12);

addTransaction();

}

public void addTransaction() {

initial();

ComboBox<String> debitBox = new ComboBox<>();

debitBox.setDisable(true);

debitBox.setMaxWidth(Double.MAX\_VALUE);

debitBox.setPrefWidth(250);

ComboBox<String> creditBox = new ComboBox<>();

creditBox.setMaxWidth(Double.MAX\_VALUE);

creditBox.setPrefWidth(250);

creditBox.getItems().addAll(accounts);

VBox particularBox = new VBox();

particularBox.setSpacing(5);

// particularBox.setPadding(new Insets(0, 5, 5, 5));

HBox debitRow = new HBox(2,new Label("By, "),debitBox);

HBox creditRow= new HBox(2,new Label("To, "),creditBox);

particularBox.getChildren().addAll(debitRow,creditRow);

TextField descp = new TextField();

descp.setEditable(false);

TextField debitamount = new TextField();

debitamount.setEditable(false);

VBox debitamountBox = new VBox();

debitamountBox.setSpacing(5);

debitamountBox.setPadding(new Insets(5, 5, 5, 5));

debitamountBox.getChildren().add(debitamount); // Add amount to debit box

TextField creditamount = new TextField();

creditamount.setEditable(false);

VBox creditamountBox = new VBox();

creditamountBox.setSpacing(5);

creditamountBox.setPadding(new Insets(5, 5, 5, 5));

TextField decoy = new TextField();

decoy.setVisible(false);

decoy.setDisable(true);

creditamountBox.getChildren().addAll(decoy,creditamount);

addEventDynamic(creditamount,debitamount);

creditBox.valueProperty().addListener((observable, oldValue, credit) -> {

debitBox.setDisable(false);

debitBox.getItems().clear(); // Clear previous accountType items

descp.setEditable(false); // Reset descp

descp.clear(); // Clear descp

debitamount.setEditable(false); // Reset amount

debitamount.clear();

creditamount.setEditable(false); // Reset amount

creditamount.clear();

// Set the accountType options based on particulaBox selection

if ("Capital".equals(credit)) {

debitBox.getItems().addAll("Cash", "Bank");

} else if ("Sales".equals(credit)) {

debitBox.getItems().addAll("Cash", "Bank", "Debtors");

} else if ("Creditor".equals(credit)) {

debitBox.getItems().addAll("Purchase","Wages","Rent","Salary");

} else if ("Purchase Return".equals(credit)) {

debitBox.getItems().addAll("Cash", "Bank","Creditor");

}

// else if("Sales Return".equals(credit)){

// debitBox.getItems().addAll("Cash", "Bank","Debtors");

// }

else if ("Cash".equals(credit)) {

debitBox.getItems().addAll("Bank", "Purchase", "Sales Return","Creditor","Wages","Rent","Salary");

} else if ("Bank".equals(credit)) {

debitBox.getItems().addAll("Cash", "Purchase", "Sales Return","Creditor","Wages","Rent","Salary");

} else if ("Debtors".equals(credit)) {

debitBox.getItems().addAll("Cash", "Bank","Sales Return");

} else {

debitBox.getItems().clear(); // Clear items if no match

}

// Listener for accountType changes

debitBox.valueProperty().addListener((observable1, oldValue1, debit) -> {

if (credit != null && !credit.trim().isEmpty()) {

amountFlag = true; // Set the flag to true if an account type is selected

} else {

amountFlag = false; // Reset flag if no account type is selected

}

// Enable the amount field if accountType is selected

if (debitBox.getValue() != null) {

debitamount.setEditable(true);

}

// Enable descp for specific account types

if ("Debtors".equals(debit) || "Creditor".equals(credit)|| ("Creditor".equals(credit) && "Purchase".equals(debit)) || "Capital".equals(credit) || ("Debtors".equals(credit) && "Purchase".equals(debit))) {

descp.setEditable(true); // Enable descp for specific conditions

descp.setPromptText("Enter a Name");

} else {

descp.setEditable(false); // Lock descp otherwise

descp.setPromptText("");

}

});

});

descp.addEventFilter(KeyEvent.KEY\_TYPED, event->{

String character = event.getCharacter();

if(!character.matches("[a-zA-Z ]")){

event.consume();

}

});

RowConstraints newRowConstraints = new RowConstraints();

newRowConstraints.setMinHeight(65);

newRowConstraints.setPrefHeight(65);

newRowConstraints.setVgrow(Priority.NEVER);

journalGridPane.add(particularBox, 0, rowCounter);

journalGridPane.add(descp, 1, rowCounter);

journalGridPane.add(debitamountBox, 2, rowCounter);

journalGridPane.add(creditamountBox, 3, rowCounter);

journalGridPane.getRowConstraints().add(newRowConstraints);

Region separatorRegion = new Region();

separatorRegion.setMinHeight(1);

separatorRegion.setPrefHeight(1);

separatorRegion.setMaxHeight(1);

separatorRegion.setStyle("-fx-background-color: #b2c8b7;");

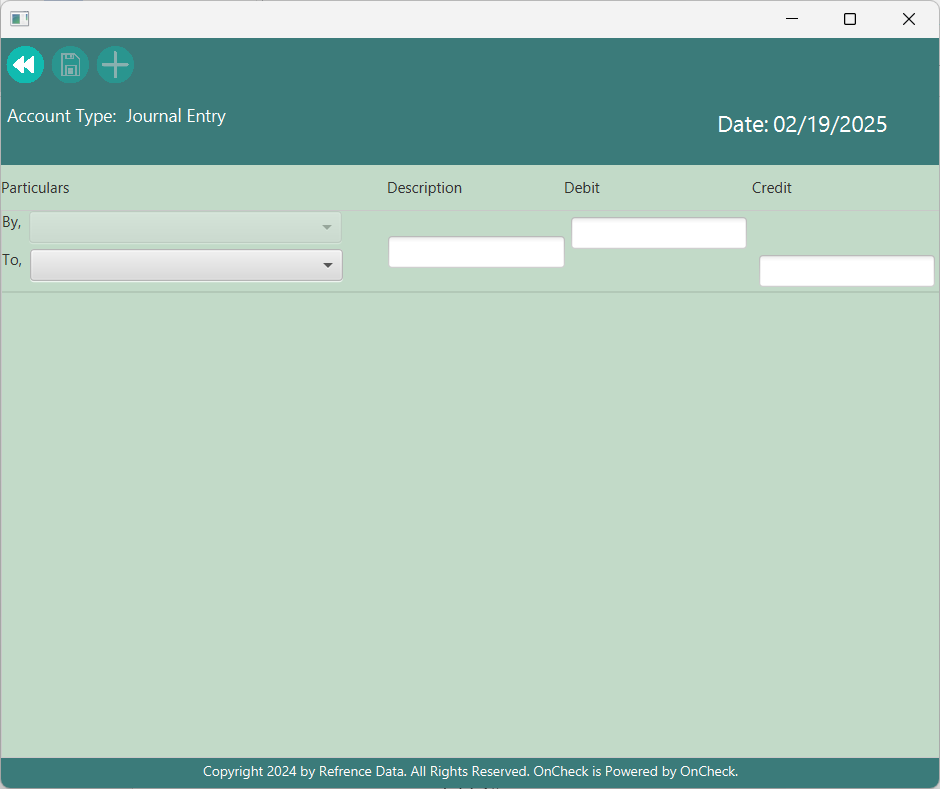
GridPane.setColumnSpan(separatorRegion, journalGridPane.getColumnCount());

journalGridPane.add(separatorRegion, 0, rowCounter );

GridPane.setValignment(separatorRegion, VPos.BOTTOM);

rowCounter++;

}



**Fig 4.3: journal entry module**

* 1. **Payment Entry:**

This module handles the operation for all payment transaction.

private void addPaymentTransaction() {

initial();

ComboBox<String> debitBox = new ComboBox<>();

debitBox.setDisable(true);

// debitBox.setMaxWidth(Double.MAX\_VALUE);

debitBox.setPrefWidth(250);

debitBox.getItems().addAll(debit);

TextField descp = new TextField();

descp.setEditable(false);

TextField amount = new TextField();

amount.setEditable(false);

// Set margins

GridPane.setMargin(debitBox, new Insets(10, 10, 10, 10));

GridPane.setMargin(descp, new Insets(10, 10, 10, 10));

GridPane.setMargin(amount, new Insets(10, 10, 10, 10));

addEventDynamic(amount);

gridPane.add(debitBox, 0, rowCounter);

gridPane.add(amount, 1, rowCounter);

chooseAccount.valueProperty().addListener((observable,oldValue,value)->{

LinkedNode.setSave(false);

});

if(chooseAccountFlag==false){

chooseAccount.setOnAction(event->{

if(chooseAccount.getSelectionModel().getSelectedItem() != null){

debitBox.setDisable(false);

chooseAccountFlag=true;

}else{

debitBox.setDisable(true);

chooseAccountFlag=true;

}

});

}else{

debitBox.setDisable(false);

}

debitBox.valueProperty().addListener((observable, oldValue, credit) -> {

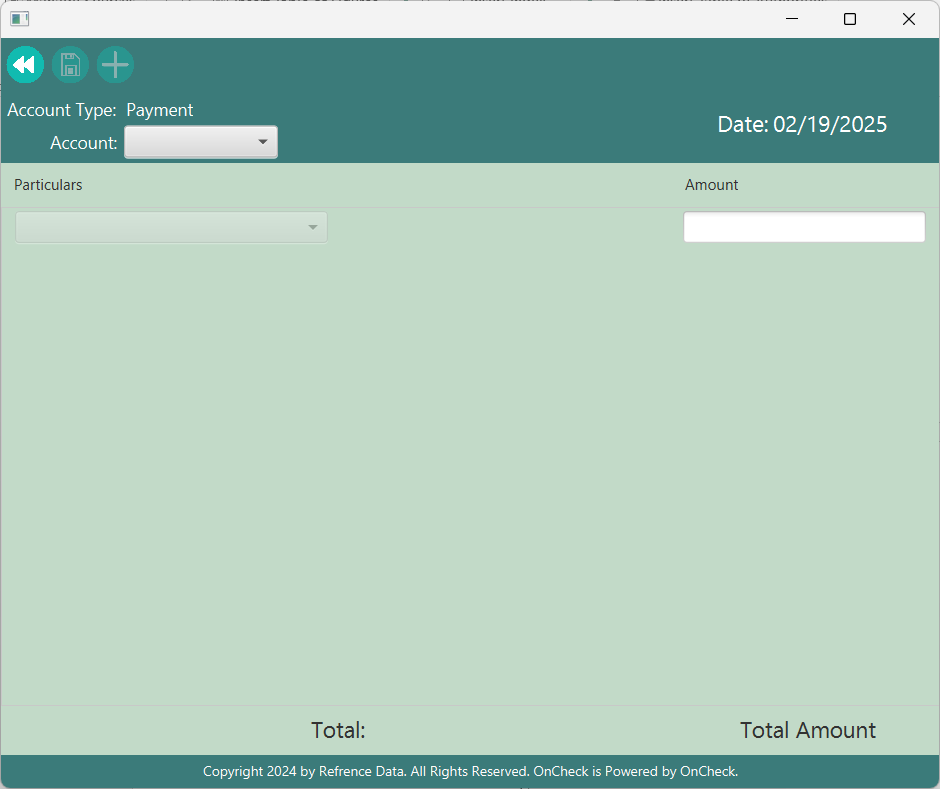
amount.setEditable(true);

amountFlag = true;

});

rowCounter++;

}



**Fig 4.4: Payment module**

* 1. **Receipt Entry:**

This module handles the operation for all receipt transaction.

private void addRecepit(){

initial();

//This is debit which means that we will receive it.

ComboBox<String> creditBox = new ComboBox<>();

creditBox.setDisable(true);

// creditBox.setMaxWidth(Double.MAX\_VALUE);

creditBox.setPrefWidth(250);

creditBox.getItems().addAll(credit);

TextField descp = new TextField();

descp.setEditable(false);

TextField amount = new TextField();

amount.setEditable(false);

// Set margins

GridPane.setMargin(creditBox, new Insets(10, 10, 10, 10));

GridPane.setMargin(descp, new Insets(10, 10, 10, 10));

GridPane.setMargin(amount, new Insets(10, 10, 10, 10));

addEventDynamic(amount);

gridPane.add(creditBox, 0, rowCounter);

gridPane.add(amount, 1, rowCounter);

chooseAccount.valueProperty().addListener((observable,oldValue,value)->{

LinkedNode.setSave(false);

});

if(chooseAccountFlag==false){

chooseAccount.setOnAction(event->{

if(chooseAccount.getSelectionModel().getSelectedItem() != null){

creditBox.setDisable(false);

chooseAccountFlag=true;

}else{

creditBox.setDisable(true);

chooseAccountFlag=true;

}

});

}else{

creditBox.setDisable(false);

}

creditBox.valueProperty().addListener((observable, oldValue, credit) -> {

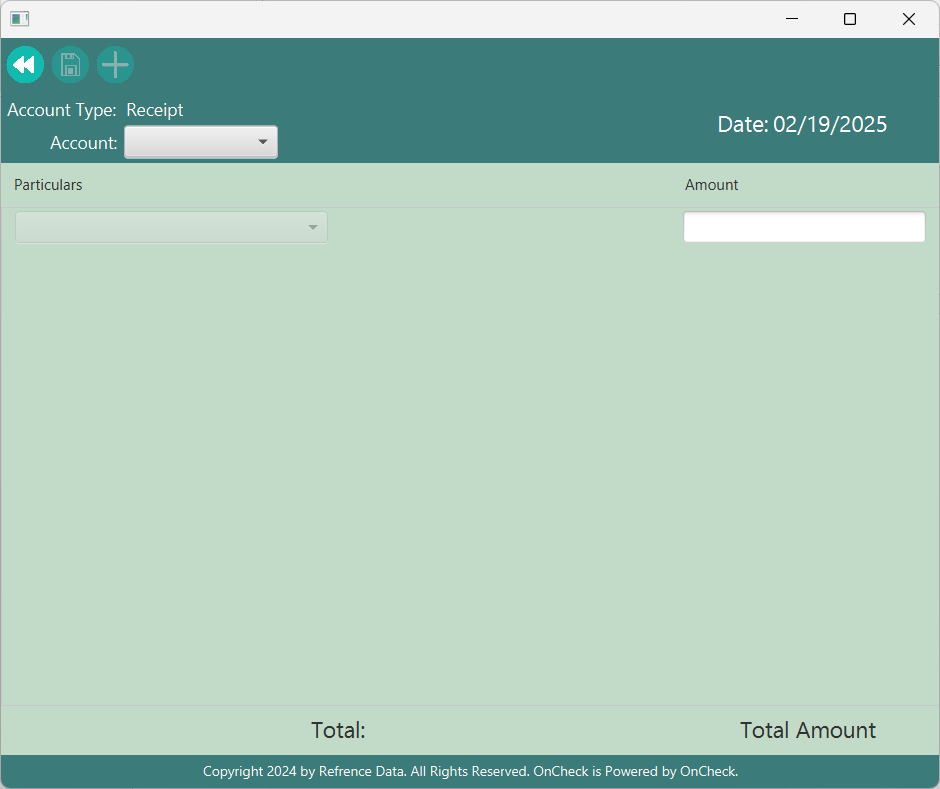
amount.setEditable(true);

amountFlag = true;

});

rowCounter++;

}



**Fig 4.5: Receipt Module**

1. **Report Module**
   1. **Journal Entry:**

public void showData() {

gridPane.getChildren().clear(); // Clear previous data entries

rowCounter = 0; // Reset rowCounter to start fresh

totalamt = 0; // Reset total amount

int dataIndex = action.countWithDate(getDate());

node = action.read(dataIndex, getDate());

for (int i = 0; i < dataIndex; i++) {

VBox particularBox = new VBox();

particularBox.setSpacing(5);

// Create labels for credit and debit

Label credit = new Label();

credit.setMaxWidth(Double.MAX\_VALUE);

if(node.credit[i].equals("Creditor")){

credit.setText("\tTo," +node.descp[i]+"'s" + "\t A/c");

}else{

credit.setText("\tTo," + node.credit[i] + "\t A/c");

}

Label debit = new Label();

debit.setMaxWidth(Double.MAX\_VALUE);

if(node.debit[i].equals("Debtors")){

debit.setText(node.descp[i]+"'s"+ "\t A/c");

}else{

debit.setText(node.debit[i] + "\t A/c");

}

Label descp = new Label();

descp.setMaxWidth(Double.MAX\_VALUE);

descp.setText(node.descp[i]);

// Create an amount label

Label amount = new Label();

amount.setMaxWidth(Double.MAX\_VALUE);

amount.setText(String.valueOf(node.amount[i]));

totalamt += node.amount[i];

// Add debit and credit labels to the particular box

particularBox.getChildren().add(debit);

particularBox.getChildren().add(credit);

// Create and set debit amount box

VBox debitamountBox = new VBox();

debitamountBox.setSpacing(5);

Label debitAmountLabel = new Label(amount.getText());

debitamountBox.getChildren().add(debitAmountLabel); // Add amount to debit box

// Create and set credit amount box

VBox creditamountBox = new VBox();

creditamountBox.setSpacing(5);

creditamountBox.getChildren().add(new Label(""));

Label creditAmountLabel = new Label(amount.getText());

debitamountBox.getChildren().add(creditAmountLabel);

creditamountBox.getChildren().add(creditAmountLabel); // Add same amount to credit box

// Set margins for the boxes

GridPane.setMargin(particularBox, new Insets(10, 10, 20, 10));

GridPane.setMargin(debitamountBox, new Insets(10, 10, 20, 10));

GridPane.setMargin(creditamountBox, new Insets(10, 10, 20, 10));

// Add boxes to the grid pane

gridPane.add(particularBox, 0, rowCounter);

gridPane.add(debitamountBox, 2, rowCounter);

gridPane.add(creditamountBox, 3, rowCounter);

Region separatorRegion = new Region();

separatorRegion.setMinHeight(1);

separatorRegion.setPrefHeight(1);

separatorRegion.setMaxHeight(1);

separatorRegion.setStyle("-fx-background-color: #b2c8b7;");

GridPane.setColumnSpan(separatorRegion, gridPane.getColumnCount());

gridPane.add(separatorRegion, 0, rowCounter );

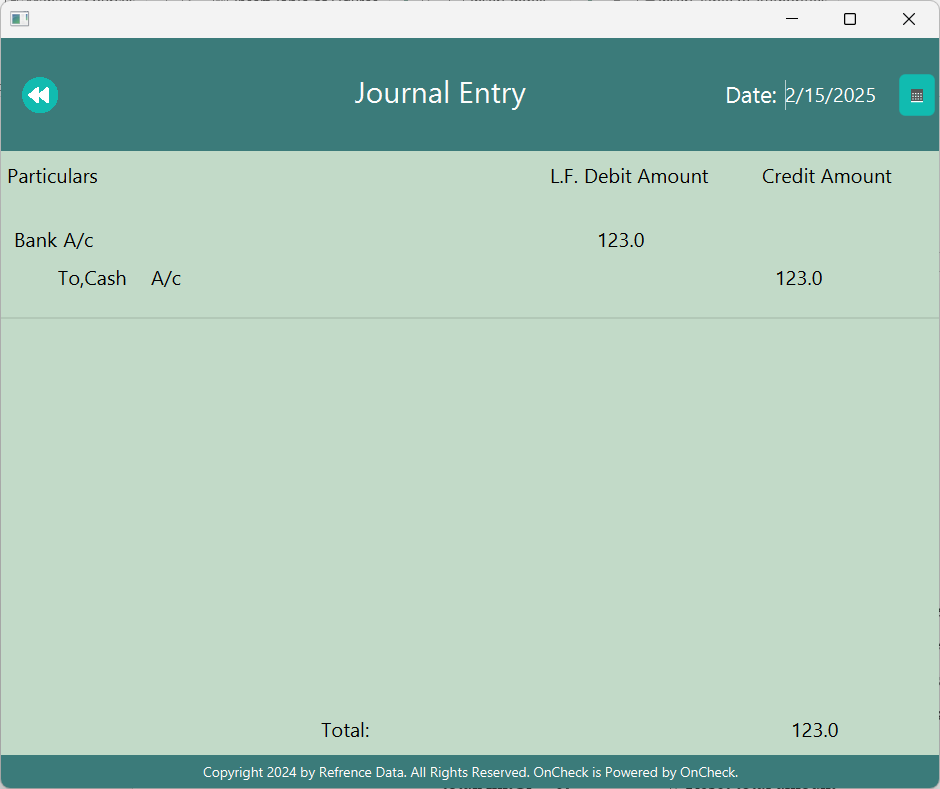
GridPane.setValignment(separatorRegion, VPos.BOTTOM);

rowCounter++;

}

totalAmount.setText(String.valueOf(totalamt));

}



**Fig 4.6: Jourmal Report Module**

* 1. **Ledger:**

private void initial(){

drGridPane.getChildren().clear(); // Clear previous data entries

crGridPane.getChildren().clear(); // Clear previous data entries

rowCounterCr = 0; // Reset rowCounter to start fresh

rowCounterDr = 0; // Reset rowCounter to start fresh

totalAmtDr = 0; // Reset total amount

totalAmtCr = 0; // Reset total amount

dataIndex = action.countWithDate(getDate());

node = action.read(dataIndex, getDate());

}

private void addBalance(){

if(totalAmtCr>totalAmtDr){

double total;

total = totalAmtCr-totalAmtDr;

Label balancLabel = new Label();

balancLabel.setMaxWidth(Double.MAX\_VALUE);

balancLabel.setText("Balance c/d");

Label balanceAmount = new Label();

balanceAmount.setMaxWidth(Double.MAX\_VALUE);

balanceAmount.setText(String.valueOf(total));

GridPane.setMargin(balancLabel, new Insets(10, 10, 10, 10));

GridPane.setMargin(balanceAmount, new Insets(10, 10, 10, 10));

drGridPane.add(balancLabel,0,rowCounterDr);

drGridPane.add(balanceAmount,2,rowCounterDr);

rowCounterDr++;

}else if(totalAmtCr<totalAmtDr){

double total;

total = totalAmtDr-totalAmtCr;

Label balancLabel = new Label();

balancLabel.setStyle("-fx-font-size: 16px;");

balancLabel.setMaxWidth(Double.MAX\_VALUE);

balancLabel.setText("Balance b/d");

Label balanceAmount = new Label();

balanceAmount.setStyle("-fx-font-size: 16px;");

balanceAmount.setMaxWidth(Double.MAX\_VALUE);

balanceAmount.setText(String.valueOf(total));

GridPane.setMargin(balancLabel, new Insets(10, 10, 10, 10));

GridPane.setMargin(balanceAmount, new Insets(10, 10, 10, 10));

crGridPane.add(balancLabel,0,rowCounterCr);

crGridPane.add(balanceAmount,2,rowCounterCr);

rowCounterCr++;

}

}

private void showData(String titleCompare){

initial();

for(int i =0;i<dataIndex;i++){

if(node.debit[i].equals(titleCompare)){

Label credit = new Label();

credit.setStyle("-fx-font-size: 16px;");

credit.setMaxWidth(Double.MAX\_VALUE);

if(node.credit[i].equals("Creditor")){

credit.setText("To, "+node.descp[i]);

}else{

credit.setText("To, "+node.credit[i]);

}

Label amount = new Label();

amount.setStyle("-fx-font-size:16;");

amount.setMaxWidth(Double.MAX\_VALUE);

amount.setText(String.valueOf(node.amount[i]));

totalAmtDr += node.amount[i];

GridPane.setMargin(credit, new Insets(10, 10, 10, 10));

GridPane.setMargin(amount, new Insets(10, 10, 10, 10));

drGridPane.add(credit, 0, rowCounterDr);

drGridPane.add(amount, 2, rowCounterDr);

rowCounterDr++;

}

if(node.credit[i].equals(titleCompare)){

Label debit = new Label();

debit.setStyle("-fx-font-size: 16px;");

debit.setMaxWidth(Double.MAX\_VALUE);

if(node.debit[i].equals("Debtors")){

debit.setText("By, "+node.descp[i]);

}else{

debit.setText("By, "+node.debit[i]);

}

Label amount = new Label();

amount.setStyle("-fx-font-size:16;");

amount.setMaxWidth(Double.MAX\_VALUE);

amount.setText(String.valueOf(node.amount[i]));

totalAmtCr += node.amount[i];

GridPane.setMargin(debit, new Insets(10, 10, 10, 10));

GridPane.setMargin(amount, new Insets(10, 10, 10, 10));

crGridPane.add(debit, 0, rowCounterCr);

crGridPane.add(amount, 2, rowCounterCr);

rowCounterCr++;

}

}

addBalance();

if(totalAmtCr>totalAmtDr){

totalAmtDr = totalAmtDr+(totalAmtCr-totalAmtDr);

}else{

totalAmtCr = totalAmtCr+(totalAmtDr-totalAmtCr);

}

totalAmountCr.setText(String.valueOf(totalAmtCr));

totalAmountDr.setText(String.valueOf(totalAmtDr));

}

private void chooseAccount(){

if (titleLabel.getText().equals("Capital Account")) {

showData("Capital");

}else if(titleLabel.getText().equals("Purchase Account")){

showData("Purchase");

}else if(titleLabel.getText().equals("Cash Account")){

showData("Cash");

}else if(titleLabel.getText().equals("Bank Account")){

showData("Bank");

}else if(titleLabel.getText().equals("Sale's Account")){

showData("Sales");

}else if(titleLabel.getText().equals("Purchase Return Account")){

showData("Purchase Return");

}else if(titleLabel.getText().equals("Sale's Return Account")){

showData("Sales Return");

}else if(titleLabel.getText().equals("Debtor's Account")){

showData("Debtors");

}else if(titleLabel.getText().equals("Creditor's Account")){

showData("Creditor");

}else if(titleLabel.getText().equals("Wage Account")){

showData("Wages");

}else if(titleLabel.getText().equals("Rent Account")){

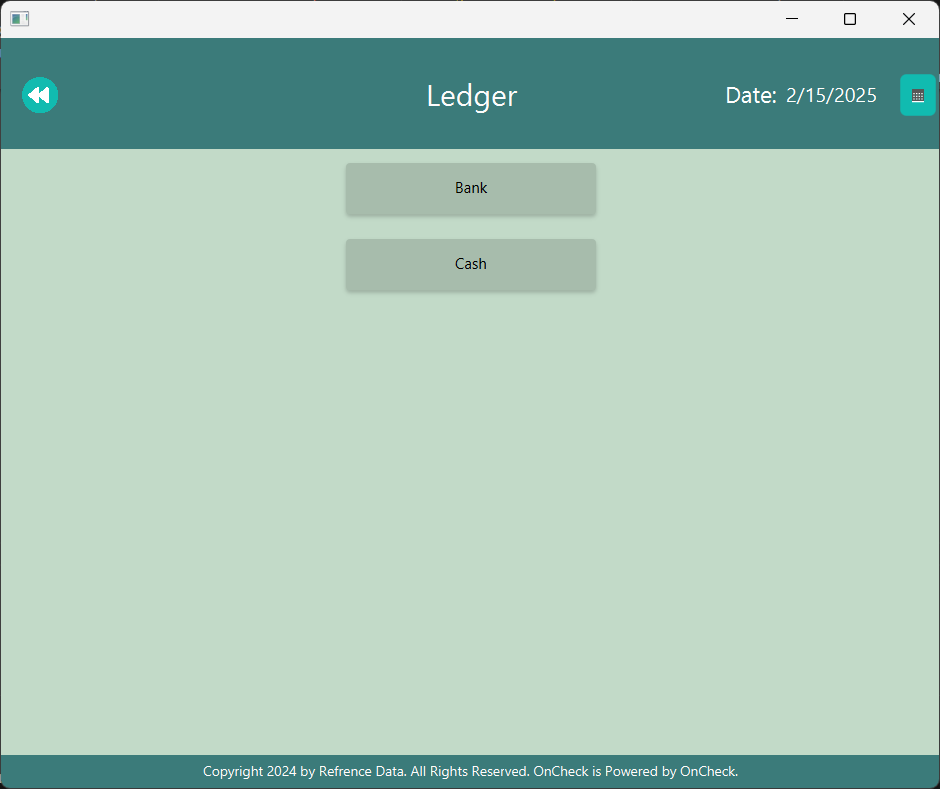
showData("Rent");

}else if(titleLabel.getText().equals("Salary Account")){

showData("Salary");

}

}



**Fig 4.7: Legder main menu module**

* 1. **Income/Expense Statement:**

private void initial() {

incomeGridPane.getChildren().clear();

expenseGridPane.getChildren().clear();

rowCounterInc = 0; // Reset rowCounter to start fresh

rowCounterExp = 0; // Reset rowCounter to start fresh

totalAmntInc = 0; // Reset total amount

totalAmntExp = 0; // Reset total amount

dataIndex = action.countWithDate(getDate());

node = action.read(dataIndex, getDate());

}

private void setBalance(){

totalInc=0;

totalExp=0;

if(amtIncome>amtExpense){

totalInc = amtIncome-amtExpense;

totalAmntInc+=totalInc;

}else if(amtExpense>amtIncome){

totalExp = amtExpense-amtIncome;

totalAmntExp+=totalExp;

}

}

private void calculateData(String title){

int i =0;

amtExpense =0;

amtIncome =0;

//first ma debit side ko cash ko total amount;

//credit side ko cash ko total amount;

//add garni balance;

while(i<dataIndex){

if(node.debit[i].equals(title)){

amtExpense += node.amount[i];

}

if(node.credit[i].equals(title)){

amtIncome += node.amount[i];

}

i++;

}

setBalance();

}

private void createLabelIncome(String lableTitle){

Label particular = new Label();

particular.setMaxWidth(Double.MAX\_VALUE);

particular.setText(lableTitle);

Label amount = new Label();

amount.setMaxWidth(Double.MAX\_VALUE);

amount.setText(String.valueOf(totalInc));

GridPane.setMargin(particular, new Insets(10, 10, 10, 10));

GridPane.setMargin(amount, new Insets(10, 10, 10, 10));

incomeGridPane.add(particular,0,rowCounterInc);

incomeGridPane.add(amount,1,rowCounterInc);

rowCounterInc++;

}

private void createLabelExpenses(String lableTitle){

Label particular = new Label();

particular.setMaxWidth(Double.MAX\_VALUE);

particular.setText(lableTitle);

Label amount = new Label();

amount.setMaxWidth(Double.MAX\_VALUE);

amount.setText(String.valueOf(totalExp));

GridPane.setMargin(particular, new Insets(10, 10, 10, 10));

GridPane.setMargin(amount, new Insets(10, 10, 10, 10));

expenseGridPane.add(particular,0,rowCounterExp);

expenseGridPane.add(amount,1,rowCounterExp);

rowCounterExp++;

}

public void check(){

Boolean prucahse=false,sales=false,wages=false,rent=false,salary=false;

initial();

for(int i = 0;i<dataIndex; i++){

if(node.debit[i].equals("Purchase")&& prucahse == false){ // Expense

prucahse=true;

calculateData("Purchase");

createLabelExpenses("Purchase");

}

if(node.credit[i].equals("Sales") && sales == false){ // Income

sales=true;

calculateData("Sales");

createLabelIncome("Sales");

}

if(node.debit[i].equals("Wages") && wages == false) { // Expenses

wages=true;

calculateData("Wages");

createLabelExpenses("Wages");

}

if(node.debit[i].equals("Rent") && rent == false) { // Expenses

rent=true;

calculateData("Rent");

createLabelExpenses("Rent");

}

if(node.debit[i].equals("Salary") && salary == false) { // Expenses

salary=true;

calculateData("Salary");

createLabelExpenses("Salary");

}

}

totalIncome.setText(String.valueOf(totalAmntInc));

totalExpenses.setText(String.valueOf(totalAmntExp));

}



**Fig 4.8: Income Expenses Report module**

* 1. **Trial Book:**

private void initial(){

gridPane.getChildren().clear(); // Clear previous data entries

rowCounter = 0; // Reset rowCounter to start fresh

totalAmntDr=0; // Reset total amount

totalAmntCr=0; // Reset total amount

action = new databaseAction();

dataIndex = action.countWithDate(getDate());

node = action.read(dataIndex, getDate());

}

private void setBalance(){

if(amtDr>amtCr){

totalDr=0;

totalDr = amtDr-amtCr;

totalAmntDr+=totalDr;

}else if(amtCr>amtDr){

totalCr=0;

totalCr = amtCr-amtDr;

totalAmntCr+=totalCr;

}

}

private void calculateData(String title){

int i =0;

amtCr=0;

amtDr=0;

//first ma debit side ko cash ko total amount;

//credit side ko cash ko total amount;

//add garni balance;

while(i<dataIndex){

if(node.debit[i].equals(title)){

amtDr+=node.amount[i];

}

if(node.credit[i].equals(title)){

amtCr+=node.amount[i];

}

i++;

}

setBalance();

}

private void createLabe(String lableTitle){

Label particular = new Label();

particular.setMaxWidth(Double.MAX\_VALUE);

particular.setText(lableTitle);

Label amount = new Label();

amount.setMaxWidth(Double.MAX\_VALUE);

if(amtDr>amtCr){

amount.setText(String.valueOf(totalDr));

}else if(amtCr>amtDr){

amount.setText(String.valueOf(totalCr));

}

GridPane.setMargin(particular, new Insets(10, 10, 10, 10));

GridPane.setMargin(amount, new Insets(10, 10, 10, 10));

gridPane.add(particular,0,rowCounter);

Region separatorRegion = new Region();

separatorRegion.setMinHeight(1);

separatorRegion.setPrefHeight(1);

separatorRegion.setMaxHeight(1);

separatorRegion.setStyle("-fx-background-color: #b2c8b7;");

GridPane.setColumnSpan(separatorRegion, gridPane.getColumnCount());

gridPane.add(separatorRegion, 0, rowCounter );

GridPane.setValignment(separatorRegion, VPos.BOTTOM);

if(amtDr>amtCr){

gridPane.add(amount,2,rowCounter);

}else if(amtCr>amtDr){

gridPane.add(amount,3,rowCounter);

}

rowCounter++;

}

public void check(){

Boolean cash=false,bank=false,prucahse=false,capital=false,sales=false,pruchaseReturn=false,salesReturn=false,debtors=false,creditors=false,wages=false,rent=false,salary=false;

initial();

for(int i = 0;i<dataIndex; i++){

if((node.credit[i].equals("Bank")||node.debit[i].equals("Bank")) && bank == false){

bank=true;

calculateData("Bank");

createLabe("Bank");

}

if(node.credit[i].equals("Capital") && capital == false){

capital=true;

calculateData("Capital");

createLabe("Capital");

}

if((node.credit[i].equals("Cash")||node.debit[i].equals("Cash")) && cash == false){

cash=true;

calculateData("Cash");

createLabe("Cash");

}

if(node.debit[i].equals("Purchase")&& prucahse == false){

prucahse=true;

calculateData("Purchase");

createLabe("Purchase");

}

if(node.credit[i].equals("Sales") && sales == false){

sales=true;

calculateData("Sales");

createLabe("Sales");

}

if(node.credit[i].equals("Purchase Return") && pruchaseReturn == false){

pruchaseReturn=true;

calculateData("Purchase Return");

createLabe("Purchase Return");

}

if(node.debit[i].equals("Sales Return") && salesReturn == false){

salesReturn=true;

calculateData("Sales Return");

createLabe("Sales Return");

}

if((node.credit[i].equals("Debtors")||node.debit[i].equals("Debtors")) && debtors == false) {

debtors=true;

calculateData("Debtors");

createLabe("Debtors");

}

if((node.credit[i].equals("Creditor")||node.debit[i].equals("Creditor")) && creditors == false) {

creditors=true;

calculateData("Creditor");

createLabe("Creditor");

}

if(node.debit[i].equals("Wages") && wages == false) {

wages=true;

calculateData("Wages");

createLabe("Wages");

}

if(node.debit[i].equals("Rent") && rent == false) {

rent=true;

calculateData("Rent");

createLabe("Rent");

}

if(node.debit[i].equals("Salary") && salary == false) {

salary=true;

calculateData("Salary");

createLabe("Salary");

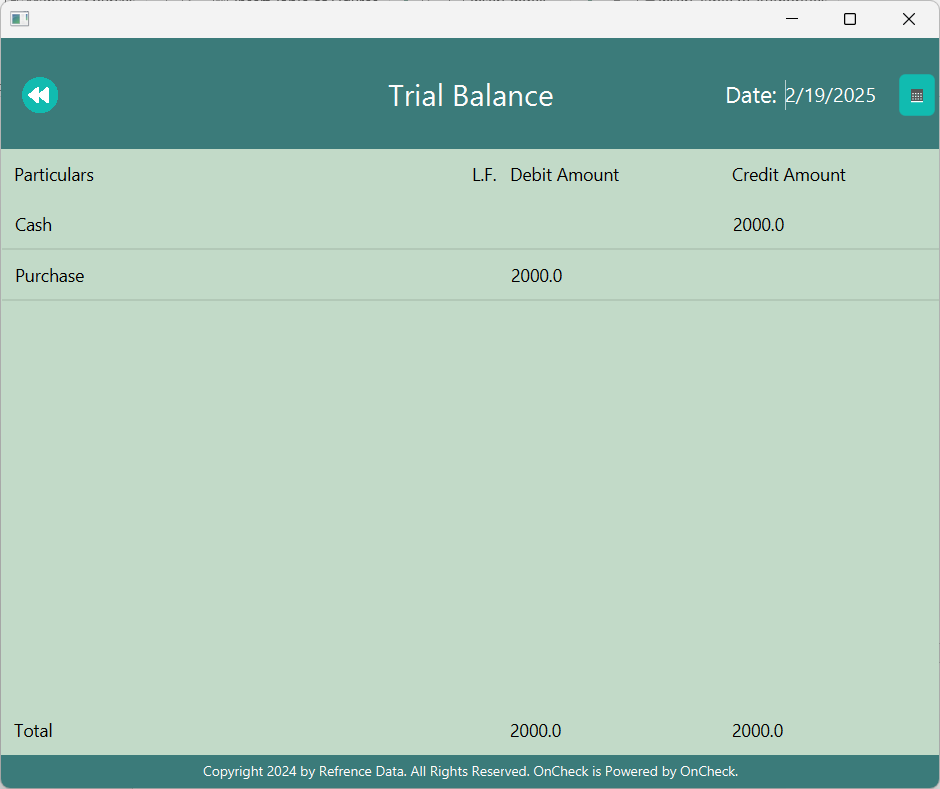
}

}

totalAmountDr.setText(String.valueOf(totalAmntDr));

totalAmountCr.setText(String.valueOf(totalAmntCr));

}



**Fig 4.9: Trial Balance module**

## Testing

### Testing case for unit testing

**Table 4.2: Unit testing of OnCheck**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Test case(s) | Steps | Expected result | Status |
| 1. | Login | Enter username, enter password and press login | The user should be redirected to the main page | Pass |
| 2. | Verify user | Enter an invalid username and password then press login | The user should be displayed with wrong credential | Pass |
| 3. | Operation:  🡪 Journal Entry,  🡪 Payment,  🡪 Receipt | User should perform various data entries in different format such as journal, payment or receipt and save them | Then all the entries are saved when the user exits | Pass. |
| 4. | Display:  🡪Journal Book  🡪Ledger  🡪Income/Expense  🡪Trial | User should view all the reports and check if the data is correct. | All the entered data should be displayed according to the view. | Pass |

### Testing case for System testing

**Table 4.3: System testing of Oncheck**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Test case(s) | Steps | Expected results | Status |
| 1. | Login and verify | Register as either the user or accountant then login and verify with right and wrong credential | If wrong input or no input designated errors should be displayed else main panel should be opened with username | Pass |
| 2. | Data entries and save | Data should be entered as per restricted by the system and must be exited by saving and without saving for conformation of saving | If not save then all data that was inputted must be lost and should not be saved in the database, but if all the data was saved the database must be updated with new inputs | Pass |
| 3. | Viewing records according to the dates | View records in display panel by choosing multiple dates where different date should provide different records as per saved by the user | If a particular date was selected all the records from the selected date must be shown in the respective format | Pass |

# Conclusion and Future Recommendation

## Lesson learned

Journal administration, payment recording, receipt creation, and financial reporting are just a few of the crucial accounting tasks that the suggested Accounting Administration System is intended to automate and simplify. This solution gives customers real-time insights into their financial data, increases efficiency, and lowers manual errors.  
According to the analysis, the system is technically feasible and can accommodate a variety of user needs by utilizing strong database management and user-friendly interfaces. It ensures quicker and more accurate processing of financial records and is operationally beneficial in enhancing the workflows of administrators, accountants, and regular users. Additionally, the system complies with legal and regulatory requirements, guaranteeing adherence to accounting standards and data protection legislation.

To conclude, the Accounting Management System is a workable, effective, and safe way to deal with important financial management issues. Its use is highly advised in order to improve productivity, streamline accounting procedures, and facilitate well-informed decision-making.

## Conclusion

Each project forces us to learn and improve our knowledge in different fields. We learned a great deal of ability to solve problems from the following project, including collaboration, individual problem-solving, proper application of instructions, communication and writing abilities, and managing a team.

1. Team work:

As a collaborative project, it teaches how to work together with others and create the system as its entirety. We now know how to divide our work in Front-end, Back-end, and Document and cooperate as a team. We have communicated with our partner and supervisor using present platforms like social media

1. Problem-solving skills

We acquired a great deal of solving skills from this project, especially the ability to recognize and resolve different issues that occur in this system. We have resolved other errors, such as logical and syntactic ones, after discussing with our supervisor.

1. Documentation:

Our ability to effectively organize and share ideas has grown as a result of the valuable knowledge we have gained developing project-related proposals and paperwork. During the process, we learnt how to create use case diagrams, schema diagrams, data flow diagrams, ER diagrams, and other important visual representations using a variety of case tools. We enhanced the overall quality of our project documentation by working on several drafts and iterations, guaranteeing accuracy and clarity.

1. Time and team management:

Effective time management, particularly in light of the intricacy of the system's components, was the most significant lesson we took away. Maintaining development and guaranteeing the system's quality required knowing which elements to focus. One of our greatest obstacles was time management, but we overcome it with commitment and persistent work. We used social media sites to communicate with our supervisor and team members even over the holidays in order to post updates, discuss progress, and resolve issues.

We learned the need of efficient team management in addition to time management. To make sure that each team member participated to the project in accordance with their areas of strength, clear communication, work delegation, and collaboration were essential.

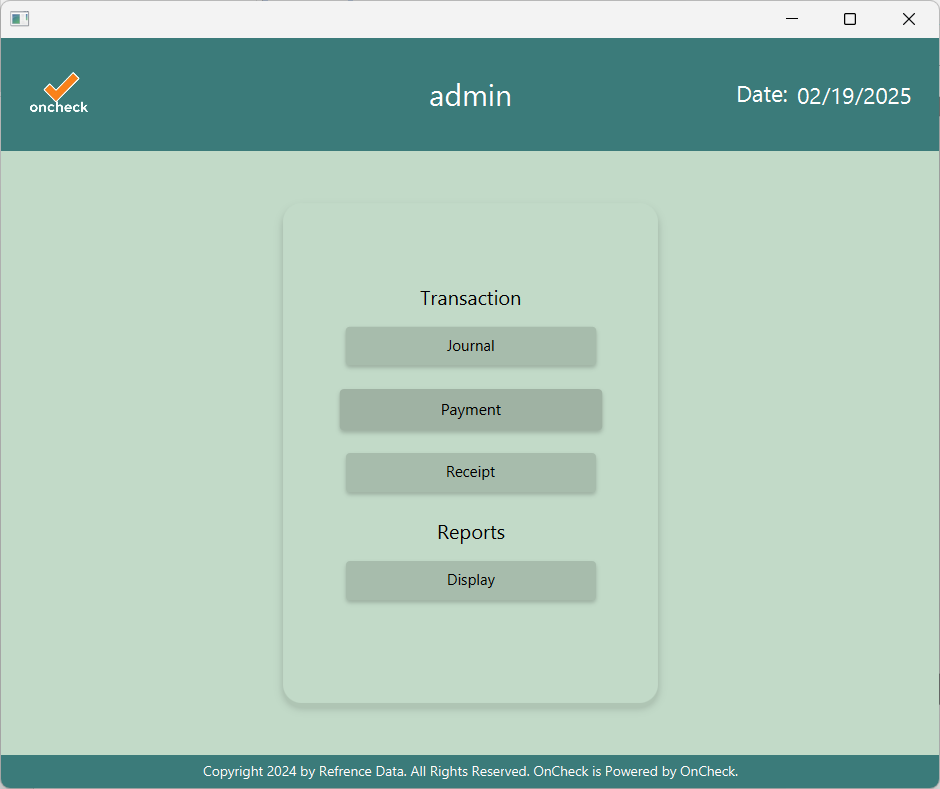
## Future Recommendation

The On Check: Accounting System's future goals call for preserving scalability, getting input from users for continuous enhancements, and keeping up with new technologies. In the future, the group aims to include further services including sophisticated financial analytics and effortless tax calculations. Implementing agile project management techniques, guaranteeing data security through frequent updates, and offering thorough user support are further considerations. By focusing on these elements, the system will keep on to be reliable, efficient, and adaptable in response to changing user demands and market conditions, promoting long-term success and user happiness.

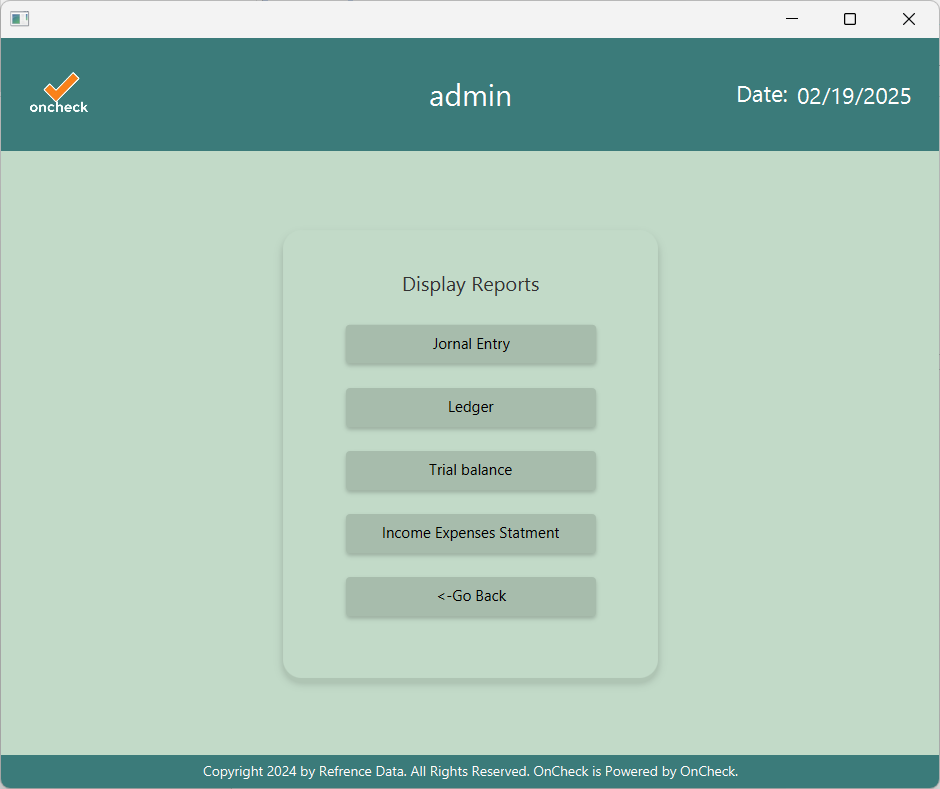
# References

|  |  |
| --- | --- |
| [1] | "wikepedia," [Online]. Available: https://en.wikipedia.org/wiki/History\_of\_accounting. [Accessed 19 9 2024]. |
| [2] | J. Fernando, "Investopedia," [Online]. Available: https://www.investopedia.com/terms/a/accounting.asp#:~:text=The%20history%20of%20accounting%20has%20been%20around%20almost,only%20been%20around%20since%20the%20early%2019th%20century.. [Accessed 19 9 2024]. |
| [3] | "Tallysolution," Tally, [Online]. Available: https://tallysolutions.com/accounting/accounting-system/. [Accessed 19 9 2024]. |
| [4] | "Intuit quickbooks," Intuit, [Online]. Available: https://quickbooks.intuit.com/r/accounting/accounting-system-definition/?msockid=27258816b4cb629a36cd9cdeb5666373. [Accessed 19 9 2024]. |
| [5] | A. O. Unegbu, "Theories of Accounting," Research Journal of Finance and Accounting, ,University of Kurdistan Hewler, 2014 . |

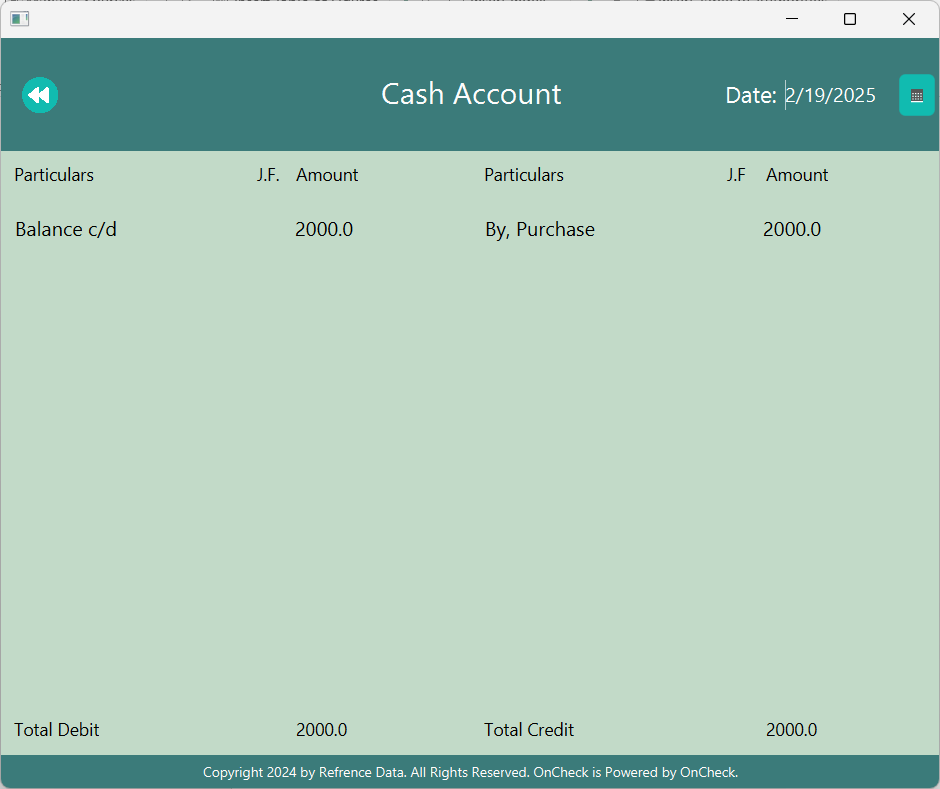
# Appendix



**Fig 7.1:Main Menu**



**Fig 7.2: Display Menu**



**Fig7.3: Ledger Report**