# VB.NET Programming Project Report

This document serves as the official report for our project in the Concepts of Programming Languages course. The selected programming language for this project is VB.NET. Below, we outline the project goals, comparison with Java, key programming constructs, and our implementation details.

## Project Overview

This project is a desktop application developed using VB.NET and integrated with a relational database system, designed to assist users in managing their personal finances effectively. The application offers functionalities for tracking income, expenses, and budgets, enabling users to maintain financial discipline. By leveraging a clean and intuitive interface paired with reliable database management, the system ensures secure and organized storage of financial records.

The project highlights key programming constructs, demonstrating proficiency in database interaction, object-oriented programming, and user interface design, making it a robust tool for personal financial management.

## VB.NET LANGUAGE

A straightforward and easy-to-learn programming language, VB.NET is perfect for developing Windows applications. It is easy-to-understand rules make it great for learning and building software quickly. It integrates with databases with ease, which is beneficial for programs that require data management or storage. Additionally, VB.NET offers powerful error-detection and correction tools that increase the dependability of programs. It allows developers to build applications faster by providing many built-in tools and features. It is a fantastic option for developing various types of software because it is a component of the.NET framework, which provides flexibility and access to strong libraries.

## Problem Statement

Many people find it difficult to manage their personal finances. It's difficult to stay on track. Many people find it difficult to manage their personal finances. It's difficult to stay on track. Many people find it difficult to manage their personal finances. Financial disarray results from the difficulty of manually tracking income, expenses, and budgets. Financial disarray as a result of manually managing budgets, income, and expenses. Financial disarray as a result of manually managing budgets, income, and expenses. Analyzing their financial habits makes it harder to maintain financial discipline. This makes it more difficult to save money and maintain financial discipline.

## Problem Solution

To address these problems, our project offers a desktop application developed with VB.NET. The program to address these problems, our project offers a desktop application developed with VB.NET. The program to address these problems, our project offers a desktop application developed with VB.NET. By enabling users to track income, record expenses, and monitor their budget in real time, the application assists users in managing their finances. It provides features like real-time budgeting, categorized transactions, and dynamic balance updates. It provides features like real-time budgeting, categorized transactions, and dynamic balance updates.

It provides features like real-time budgeting, categorized transactions, and dynamic balance updates. It provides features like real-time budgeting, categorized transactions, and dynamic balance updates. For safe and orderly data storage, it provides features like dynamic balance updates, categorized transactions, and database integration. Its database integration and user-friendly interface allow for safe and well-organized data storage. Its database integration and user-friendly interface allow for safe and well-organized data storage. The application empowers users to take charge of their financial health by making personal finance management easy and accessible through its user-friendly interface. These features also empower users to take action. People might overspend, lose track of transactions, or find it difficult to People might overspend, lose track of transactions, or find it difficult to People might overspend, lose track of transactions, or find it difficult to examine their financial patterns if there is no clear system in place.

## Language Comparison: VB.NET vs. Java

1. Readability: VB.NET is known for its simple syntax, making it highly readable, especially for beginners. Java, while also readable, tends to be more verbose.
2. Writability: VB.NET includes many features, such as integrated support for Windows applications that enhance its writability. Java excels in writability due to its platform independence.
3. Reliability: Java's strong exception-handling model and JVM architecture offer robust reliability. VB.NET is reliable, but its heavy integration with Windows can limit portability.
4. Speed: Java tends to outperform VB.NET in execution speed due to the JVM optimizations, but VB.NET benefits from faster development cycles in Windows environments.

### ****Key Features****

1. **Income Tracking:**
   * Allows users to log income transactions with specific details such as amount, date, and remarks.
2. **Expense Tracking:**
   * Provides functionality to record expenses, categorizing them for better organization.
3. **Budget Balance Monitoring:**
   * Automatically calculates and displays the current balance after each transaction.
   * Updates balance dynamically for each new entry or modification.
4. **Search and Filter Transactions:**
   * Users can search for specific transactions or filter records by date range to analyze their financial history.
5. **Database Integration:**
   * Uses a SQL-based relational database for storing financial records securely.
   * Performs CRUD operations(Create, Read, Update, Delete) on transaction records.
6. **Dynamic Updates:**
   * Automatically updates all subsequent records when a transaction is edited or deleted, ensuring consistency in budget calculations.
7. **User-Friendly Interface:**
   * Simple and accessible interface developed using VB.NET Windows Forms, making it suitable for users with varying levels of technical proficiency.

### ****Technical Details****

1. **Programming Constructs Used:**
   * **Variables:** Global and local variables for tracking row counts, balances, and selected record IDs.
   * **Loops:** For and For Each loops to iterate through records and update balances.
   * **Conditionals:** If-Else and Select Case statements for input validation and dynamic query execution.
   * **Functions:** Modular functions such as clear() to reset forms and refresh data.
   * **Object-Oriented Design:** Classes and methods encapsulate functionality, ensuring a modular and maintainable structure.
2. **Database Operations:**
   * The database schema includes tables for transactions with fields like ID, income, expense, balance, date, and remarks.
   * SQL queries are used extensively for inserting, updating, deleting, and retrieving data.
   * Transactions are handled dynamically to maintain data integrity.
3. **Error Handling:**
   * Uses Try-Catch blocks to gracefully handle errors during database operations, ensuring application reliability.
4. **Tools and Technologies:**
   * **Frontend:** VB.NET Windows Forms for a graphical user interface.
   * **Backend:** SQL database for data persistence.
   * **Development Environment:** Visual Studio 2015 for coding and debugging.

## Implementation Details

### ****1. Variables****

* **Global Variables:**
  + maxrow, balance, id - Used to store the maximum rows, current balance, and selected record ID, respectively.
  + These are declared at the class level and are accessible across the class.

### ****2. Event Handlers****

* Handles user actions like button clicks and cell clicks:
  + **Example:**
    - btnSave\_Click for saving or updating records.
    - Form1\_Load for initializing the form and loading data.
    - btnSearch\_Click for searching records within a date range.
    - btnDelete\_Click for deleting a record.

### ****3. Conditionals****

* **If...Then...Else:**
  + Used for input validation and conditional logic.
  + **Example:** Validating fields like txtMoney and cboType before performing database operations.
  + **Example:** Differentiating actions based on cboType.Text ("Received" vs. "Withdraw").
* **Select Case:**
  + Used for cleaner multi-condition checks.
  + **Example:** Determining SQL queries based on transaction type.

### ****4. Loops****

* **For Each Loop:**
  + Iterates through rows of data for balance updates.
  + **Example:** Updating subsequent records' balances in the database.
* **For Loop:**
  + Used to iterate over dtglist rows to calculate totals (received, withdrawn, and balance).

### ****5. Functions****

* **Modular Design:** Functions encapsulate repetitive tasks.
  + **Example:** clear() resets fields and reloads data, reducing code duplication.

### ****6. Object-Oriented Programming (OOP)****

* The project uses classes and objects:
  + **Class:** Form1 encapsulates all form-related logic.
  + **Objects:** Components like txtMoney, dtglist, and cboType are instantiated objects representing UI elements.

### ****7. Database Operations****

* **SQL Queries:**
  + CRUD (Create, Read, Update, Delete) operations are performed on the tblbudget database.
  + Queries include INSERT, SELECT, UPDATE, and DELETE.
  + SQL queries are executed using helper functions like executeQuery().

### ****8. Error Handling****

* **Try...Catch Block:**
  + Used for exception handling during SQL query execution.
  + **Example:** Catching errors when updating balances for subsequent records.

### ****9. Data Binding and Manipulation****

* **DataGridView (dtglist):**
  + Used to display transaction records.
  + Cell clicks load data into form fields for editing.

### ****10. UI Interaction****

* Text box and combo box interactions:
  + **Example:** txtMoney\_Click and txtRemarks\_Click select all text when clicked.
* **Buttons:** Trigger specific functions (e.g., btnSave, btnSearch, btnDelete).

### ****11. Date and Time Manipulation****

* Uses dtpTransDate for handling transaction dates.
* Searches records by a date range using btnSearch\_Click.

## Individual Responsibilities

Each team member contributed to the project as follows:  
- Ali Moallem: Contributed to implementing functions event handlers.  
- Darine Chames: Worked on the user interface and form design, and database operations.  
- Moamen Hamdan: Focused on database integration and SQL queries.

## Conclusion

This project allowed us to explore VB.NET's capabilities and understand its strengths and limitations in comparison to Java, and important programming concepts like object-oriented design, database integration, and error handling were clarified for us. Through hands-on experience, we enhanced our programming skills and deepened our understanding of language design principles. All things considered, we learned a lot about VB.NET and how to develop a useful application that addresses a real-world issue.