PURBANCHAL UNIVERSITY

Gomendra Multiple College

Faculty of Science and Technology

Expense Tracker

IN PARITAL FULLFILLMNET OF THE REQUIREMENT OF BCA 4th SEMESTER

Submitted By:
Bandana Neupane (340011)
Asmita Baral (340008)
Bibek Adhikari (340015)

Under the supervision of **Kushal Niroula**

Acknowledgment

This project documentation was written in May 2024 and is the outcome of four months of hard work and dedication. Although it was challenging at times, the process was interesting and motivating, providing us with deeper knowledge in the area of software development preferences.

We are thankful to all those who have helped us directly or indirectly with this project. Foremost, we should like to thank **Gomendra Multiple campus** for providing us with this project, Expense Recorder. We would like to thank **Kushal Niroula** for his guidelines for this project and for organizing this course and acknowledge his effort that encouraged us to take this challenging project. We would also like to offer our gratitude to all our teachers whose lectures and ideas were the basis for our project research and appreciate the support rendered by the Department of science and technology. I humbly express my thanks to our Principal **Mr. Rupak Khanal** for extending his support and for providing us with an environment to complete our project successfully. We would also like to offer our gratitude to all our teachers whose lectures and ideas were the basis for our project research and appreciate the support rendered by the Department of science and technology.

I would like to thank all my friends who helped me by sharing knowledge and by providing material to complete the project in time. Lastly, we are thankful to **Purbanchal University** for such innovation of this wonderful course structure which can surely help to build the careers of the students.

Abstract

This report documents our four-month experience of developing the software to manage expenses, **expense tracker**. **“Expense Recorder”** is a web application built to automate the ability to categorize Income and Expense then generate reports and analyze spending patterns over time. This information can be invaluable for making informed financial decisions and achieving financial goals. An expense recorder is a tool or system used to monitor and manage personal or business income and expenses. It helps individuals or organizations keep track of where their money is being spent, enabling them to better understand their financial habits, identify areas for potential savings, and budget more effectively.

Our responsibilities encompassed front-end and back-end development using Vue.js, .NET Core, and PostgreSQL. We also worked on the requirement analysis and system design as a team.

This report provides a detailed account of our journey, highlighting the challenges, achievements, and lessons learned along the way. It offers a comprehensive overview of our contributions and growth as a web developer during this enriching and memorable experience.

Table Of Contents

Abbreviations

1	SQL	Standard Query Language

Table Of Figures

Chapter 1 | Introduction

1.1 Problem statement

Effective financial management is crucial for both individuals and businesses to ensure long-term financial stability and growth. However, many people struggle with tracking and managing their expenses due to the several key challenges.

The primary challenges include:

- **User Experience**: Many expense tracking applications have complex interfaces that are difficult for users to navigate. This complexity can deter users from consistently using the application, leading to incomplete or inaccurate financial records.
- **Customization**: Existing solutions often lack the flexibility to adapt to the unique needs of different users. For example, individuals may require different categorization and reporting features compared to small businesses. The lack of customization options can result in an inefficient tracking process.
- **Data Security**: Financial data is highly sensitive, and ensuring its security is paramount. Not all expense tracking applications provide adequate security measures, leaving user data vulnerable to breaches and unauthorized access.
- **Integration**: Automated data entry and integration with financial institutions are critical for real-time expense tracking. However, many applications struggle with consistent and reliable integration, leading to manual entry and potential errors.
- **Real-time Insights**: Users need real-time insights into their spending patterns to make informed financial decisions. Many existing tools fail to provide immediate updates and comprehensive reports, limiting users' ability to manage their finances effectively.

1.2 Expense Tracker

In the modern digital age, efficient financial management is paramount for both individuals and businesses. Keeping track of expenses manually or through disjointed tools can lead to inaccuracies, inefficiencies, and missed opportunities for better financial planning. To address

these challenges, we present the "Expenses Tracker" project—a comprehensive, user-friendly web application designed to streamline expense tracking and management.

The Expenses Tracker is built using cutting-edge technologies, ensuring robustness, scalability, and seamless user experience. The core technologies leveraged in this project include:

- **.NET Core**
- **PostgreSQL**
- **ASP.NET Core**

1.3 Project Scope

The scope of the Expenses Tracker project includes:

- Development of a web-based application for expense management.
- Implementation of user authentication and authorization.
- Providing users with functionalities to add, edit, delete, and categorize expenses.
- Generating real-time reports and visualizations of spending patterns.
- Ensuring data security and privacy.

1.4 Key features

1. **User-Friendly Interface:** Intuitive Design: The application boasts a clean and intuitive user interface, making it easy for users of all technical levels to navigate and use the features effectively. Visual Programming Elements: Leverages C# with visual programming for interactive and responsive user interactions.
2. **Income and Expense Tracking:** Add Income: Users can input various sources of income such as salary, freelance work, investments, etc. Record Expenses: Users can categorize and log different types of expenses including groceries, utilities, entertainment, and more. Delete Entries: Allows users to update or remove any income or expense entries as needed.
3. **Real-Time Balance Calculation:** Automatic Calculation: The application automatically calculates the remaining balance by subtracting total expenses from the total income. Instant Updates: Balance is updated in real-time with every new entry or modification.

4. **Categorization and Organization:** Expense Categories: Users can categorize expenses for better organization and analysis. Customizable Categories: Allows users to create and manage their own expense categories to suit their specific needs.
5. **Data Management with SQL Server:** Database Integration: Utilizes Microsoft SQL Server for robust and secure data storage. Persistent Data Storage: Ensures that all income and expense data is stored persistently, even after the application is closed. Data Backup and Recovery: Includes features for backing up the database and recovering data to prevent data loss.

Chapter 2 | Literature Review and Methodology

2.1. Literature review

Expense tracking is an essential aspect of personal and business financial management. The significance of expense tracking has led to the development of numerous tools and applications over the years. This literature review examines the existing solutions and technologies used in expense tracking systems, identifying the strengths and limitations that inform the development of the Expenses Tracker project.

2.1.1 Traditional Methods

Traditionally, expense tracking was performed manually using paper-based methods such as ledgers and journals. Although straightforward, these methods are prone to errors, require significant effort, and lack the ability to provide real-time insights. As technology advanced, spreadsheet software like Microsoft Excel became a popular tool for expense tracking. While more efficient than paper-based methods, spreadsheets still require manual entry and are limited in their ability to generate detailed reports and analytics.

2.1.2 Modern digital solutions

In recent years, numerous digital solutions have emerged, leveraging web and mobile technologies to provide more efficient and user-friendly expense tracking. Some notable examples include:

- **Mint**: A web and mobile application that aggregates user financial data, providing tools for budgeting, tracking expenses, and generating financial reports. Mint integrates with various financial institutions, allowing for automated data entry and real-time updates.
- **Expensify**: Focused on business expense management, Expensify offers features such as receipt scanning, expense categorization, and integration with accounting software. Its robust feature set makes it popular among small businesses and enterprises.
- **YNAB (You Need A Budget)**: This application emphasizes budgeting and proactive financial planning. It provides tools for tracking expenses, setting financial goals, and visualizing spending patterns.

2.1.3 Limitations of existing solutions

Despite the advancements in digital expense tracking, several limitations remain:

- **User Experience**: Many applications have steep learning curves or cluttered interfaces that can overwhelm users.
- **Customization**: Some solutions lack the flexibility to tailor features to individual or business-specific needs.
- **Data Security**: Given the sensitive nature of financial data, ensuring robust security measures is critical. Not all solutions provide the same level of data protection.
- **Integration**: Integrating with various financial institutions and third-party services can be inconsistent, affecting the reliability of automated data entry.

2.2 Methodology

2.2.1 Agile Methodology

"Agile software processes is an iterative and incremental based development, where requirements are changeable according to customer needs. It helps in adaptive planning, iterative development, and time boxing" (Sharma, Sarkar, & Gupta, 2012).

Iterative Development: We build our product in small, repeated steps. This allows us to continually refine and enhance our product with each iteration.

Flexibility: We stay adaptable and can change our plans as needed. This ensures we can respond effectively to new challenges and opportunities.

Continuous Improvement: We are always looking for ways to get better and make our product better. Regular feedback and assessments help us identify areas for enhancement.

Chapter 3 | System Analysis

3.1 Requirement gathering and analysis

The requirement analysis phase is critical in ensuring the successful development and implementation of the Expenses Tracker application. This phase involves identifying and documenting the functional and non-functional requirements of the system to ensure that it meets the needs of its users. This section outlines the key requirements for the Expenses Tracker project, including user requirements, system requirements, and constraints.

3.1.1 Stakeholder engagement

We engaged with various stakeholders, including individuals, small business owners, and financial managers, to understand their current challenges and requirements regarding expense management. This involved conducting interviews, surveys, and workshops to gather firsthand insights into their workflows, pain points, and desired features in an expense tracking system.

3.1.2 Analysis of findings

Next, we analyzed the findings from the stakeholder feedback to identify common themes, key functionalities, and critical requirements for an expense management system tailored to the needs of our users. This analysis helped prioritize features and functionalities that are most relevant and impactful. Additionally, we considered factors such as scalability, ease of use, compatibility with existing systems, and affordability to ensure that the proposed expense tracking solution meets the diverse needs and constraints of its users.

3.2 Functional requirements

3.2.1 User Authentication and Authorization

- Users must be able to register for a new account.
- Users must be able to log in and log out securely.
- Password recovery and reset functionality should be available.
- Different levels of access should be provided based on user roles.

3.2.2 Expense Management

- Users must be able to add, edit, and delete expense entries.
- Expenses must be categorized (e.g., food, transportation, utilities).
- Users must be able to view and filter expenses by date, category, and amount.

3.2.3 Dashboard and Reporting

- Users must have access to a dashboard that provides an overview of their expenses.
- Visual representations of spending patterns (e.g., charts and graphs) should be available.
- Users must be able to generate and download detailed expense reports.

3.2.4 Data Export

- Users must be able to export their expense data in various formats (e.g., PDF, CSV).

3.2.5 Notifications

- Users should receive notifications for important events (e.g., large expenses, approaching budget limits).

3.3 Non-functional requirements

3.3.1 Performance

- The application must respond to user actions within 2 seconds.
- The system must support concurrent users without significant performance degradation.

3.3.2 Scalability

- The application must be able to scale horizontally to accommodate an increasing number of users and data volume.

3.3.3 Security

- All data transmissions must be encrypted using SSL/TLS.
- User passwords must be stored securely using hashing and salting techniques.
- The application must implement role-based access control (RBAC).

3.3.4 Usability

- The user interface must be intuitive and easy to navigate.
- The application must provide helpful error messages and tooltips to guide users.

3.3.5 Reliability

- The system must have a high availability with minimal downtime.
- Regular backups of the database must be performed to prevent data loss.

3.3.6 Maintainability

- The codebase must be well-documented and follow best practices for coding standards.
- The system must be designed in a modular fashion to facilitate updates and maintenance.

Chapter 4 | Architecture and Technology Stack

4.1 System Architecture

The system architecture of the Expenses Tracker application consists of three main layers:

- **Presentation Layer**: The user interface, built with ASP.NET Core MVC, that interacts with the users.
- **Business Logic Layer**: The application logic, implemented in .NET Core, that processes user inputs and interacts with the database.
- **Data Access Layer**: The database management system, PostgreSQL, that stores and retrieves data as needed.

4.2 Technology Stack

Backend: .NET Core for application logic and APIs.

Database: PostgreSQL for robust and scalable data management.

Frontend: ASP.NET Core MVC for creating dynamic, responsive web pages.

Hosting: The application can be deployed on various platforms including cloud services for high availability and performance.

4.2.1 Backend

The backend of the Expenses Tracker application is a critical component responsible for handling business logic, data processing, and communication between the frontend and the database. This section provides a detailed overview of the backend architecture, including the use of .NET Core and C# for developing robust and scalable backend services. It also covers key components, design patterns, and best practices employed in the backend development of the Expenses Tracker application.

4.2.2 Frontend

The frontend of the Expenses Tracker application is responsible for providing an intuitive and responsive user interface that allows users to interact with the application efficiently. The frontend of the Expenses Tracker application is designed using Vue.js, a progressive JavaScript framework for building user interfaces. Vue.js offers simplicity and flexibility, making it ideal

for developing interactive and responsive frontend applications. The frontend utilizes the following libraries:

- Vue
- Vue-router
- Tanstack-query
- Tailwind
- oFetch
- pinia

4.2.3 Database

The database design is a critical aspect of the Expenses Tracker application, ensuring efficient data storage, retrieval, and management. The database is implemented using PostgreSQL, a powerful and open-source relational database management system.

4.3 System Design

4.3.1 E-R Diagram

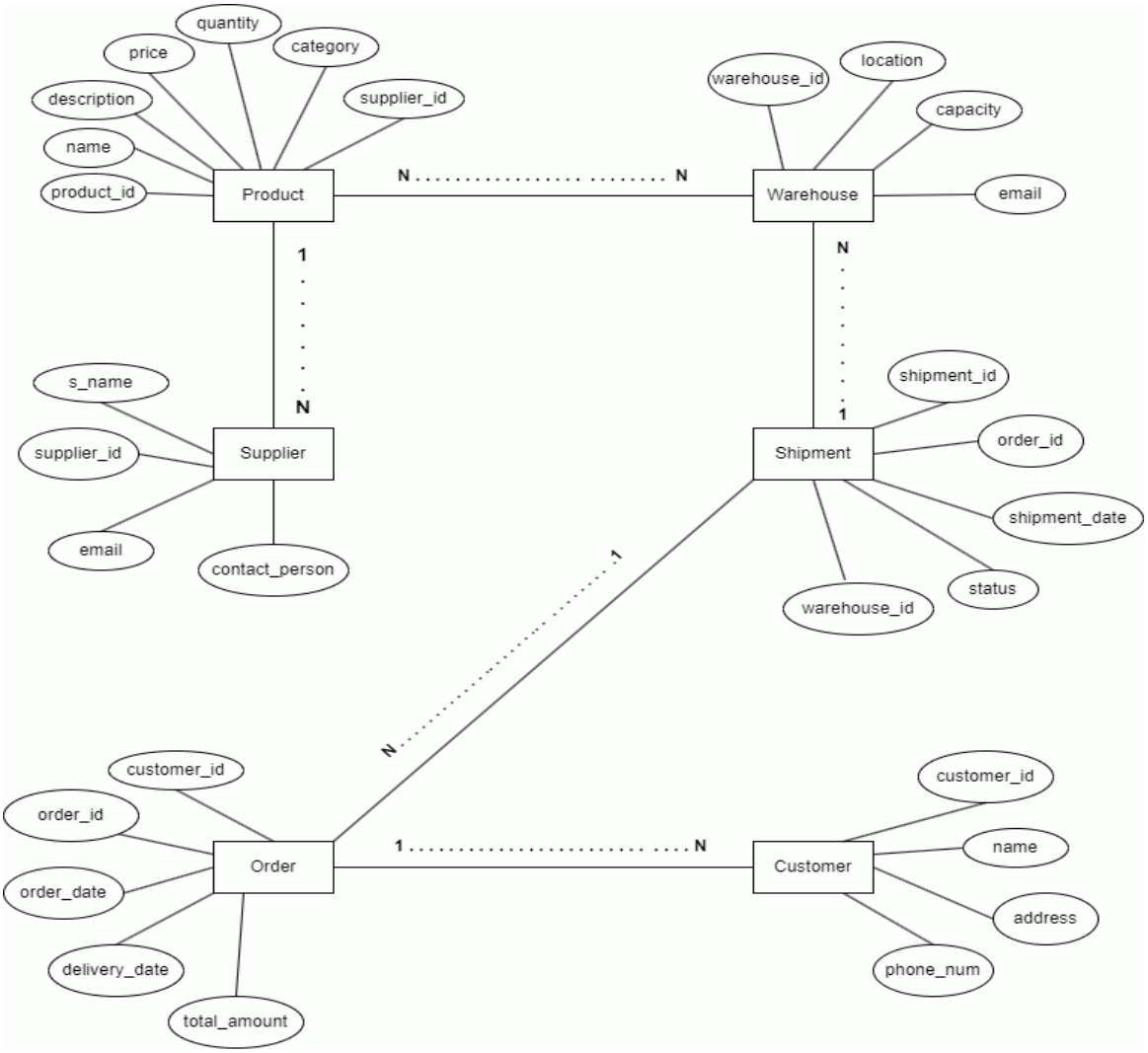

Figure 1: E-R Diagram

4.3.2 Schema Diagram

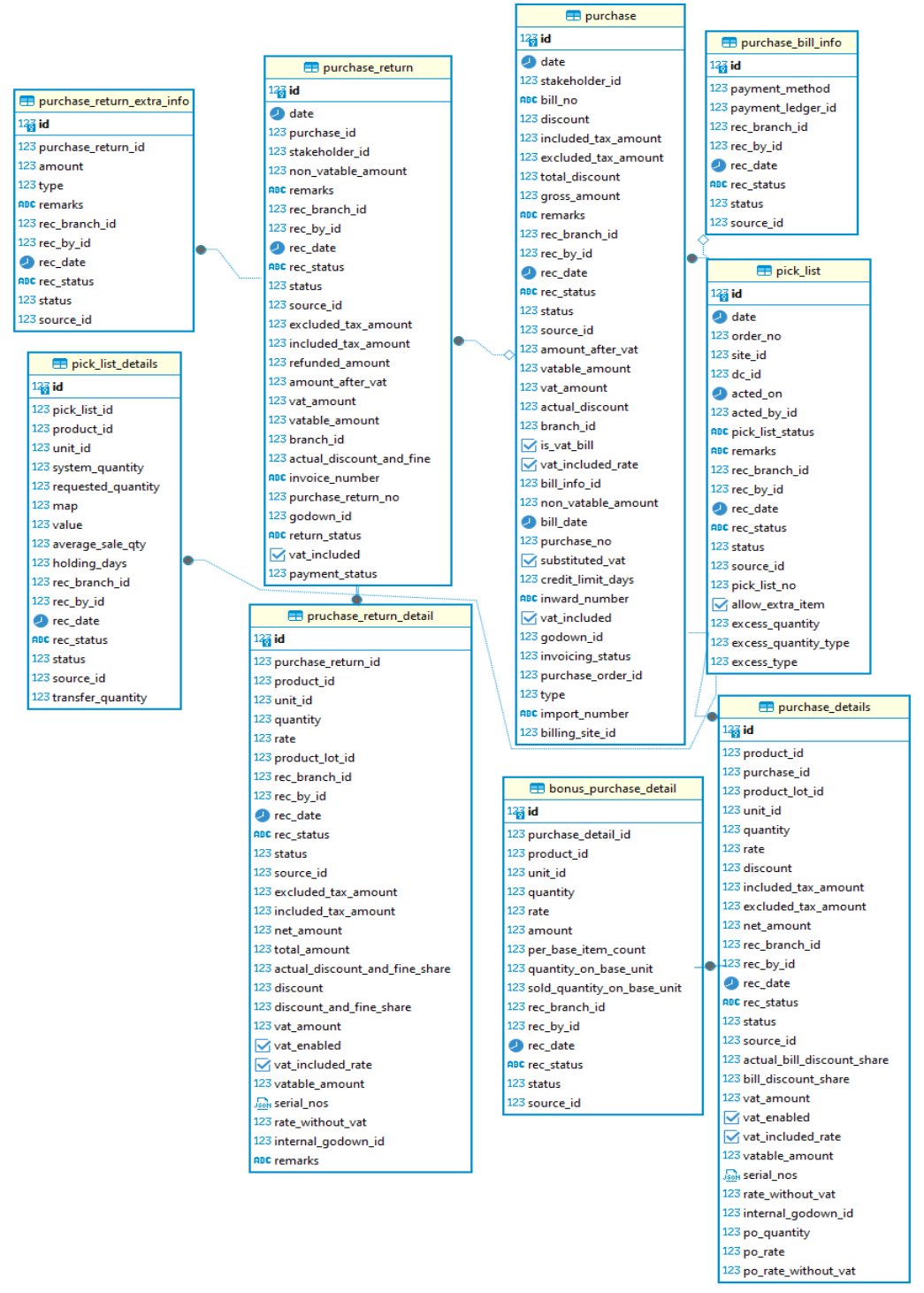

Figure 2: Database Schema Diagram

4.3.3 Use Case Diagram

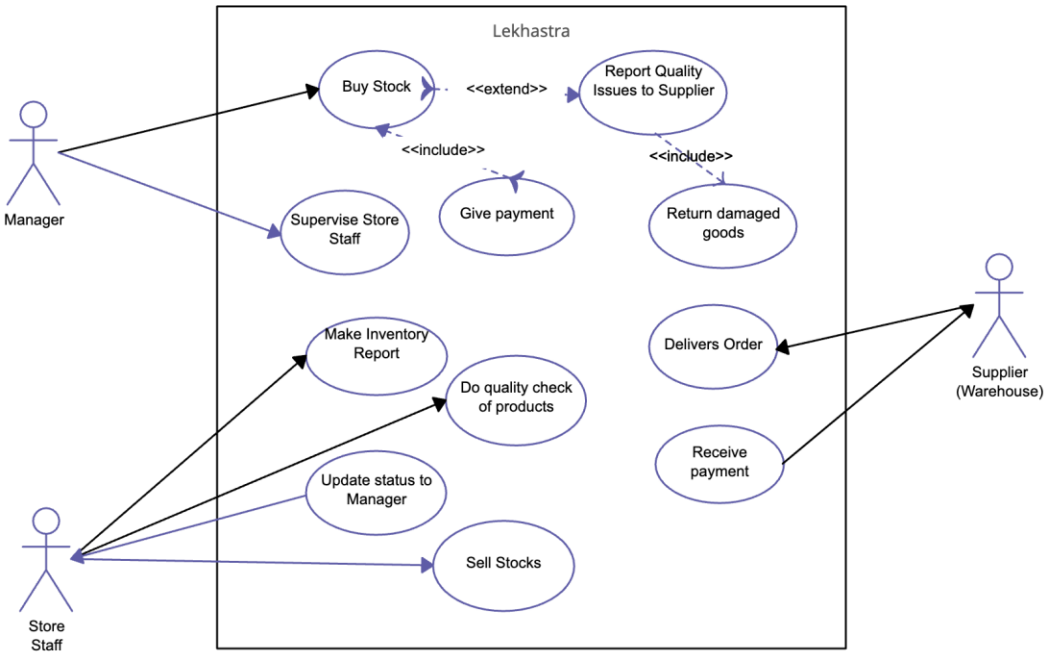

Figure 3: Use-case diagram

4.3.4 Sequence Diagram

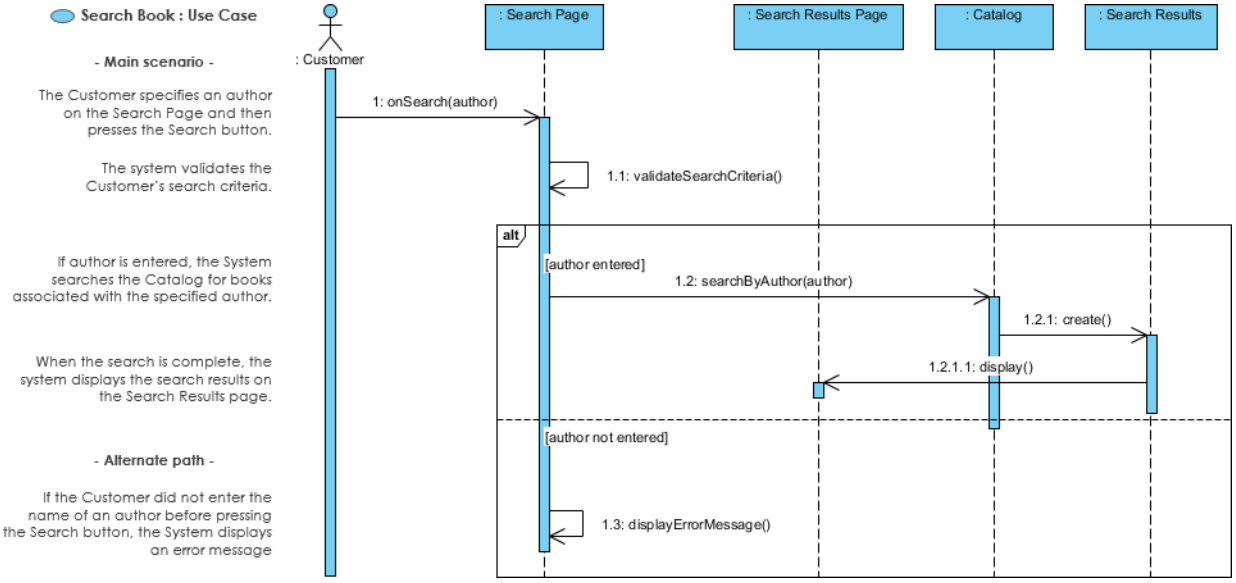

Figure 4: Sequence Diagram

4.3.5 DFD

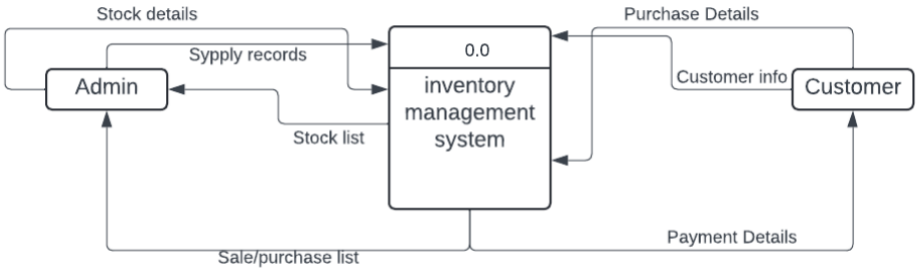

Figure 5: Level 0 DFD

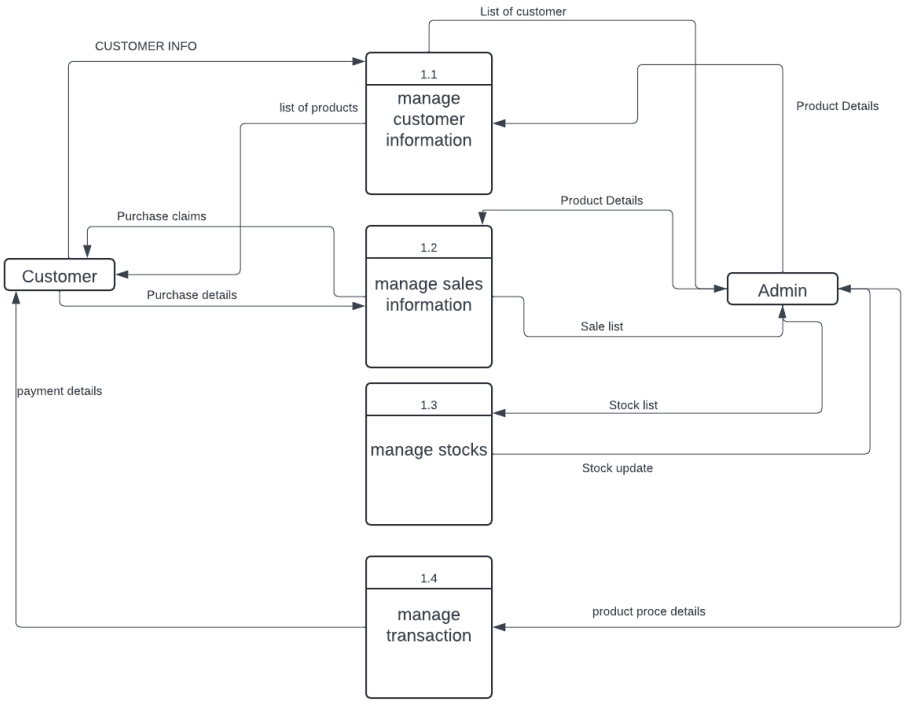

Figure 6: Level 1 DFD

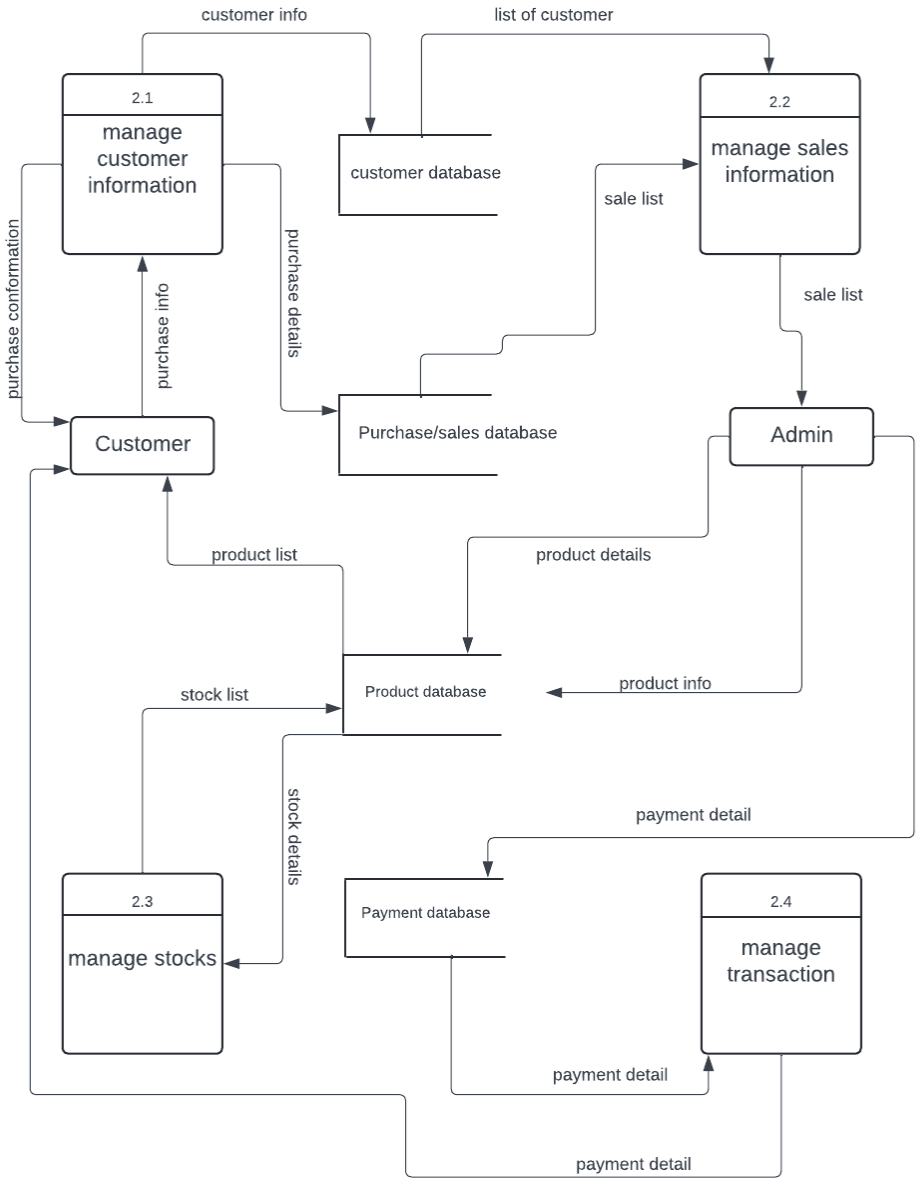

Figure 7: Level 3 DFD

Chapter 5 | Implementation and testing

The implementation phase involves turning the design specifications into a working Expenses Tracker application.

5.1 Development environment setup

.NET Core SDK

- Install the .NET Core SDK to develop the backend services.

Node.js and npm

- Install Node.js and npm (Node Package Manager) for managing frontend dependencies and building the Vue.js application.

PostgreSQL

- Install PostgreSQL for the database management system.

Visual Studio Code / Visual Studio

- Use an Integrated Development Environment (IDE) like Visual Studio Code or Visual Studio for code editing and debugging.

Git

- Use Git for version control.

5.2 Coding

The coding practices and standards adopted in the Expenses Tracker project ensure that the codebase is maintainable, scalable, and readable.

5.2.1 General Convention

5.2.1.1 Consistent Naming Convention

- Use camelCase for variables and functions.
- Use PascalCase for class names and interfaces.
- Use UPPER_SNAKE_CASE for constants.

5.2.1.2 Code Comments

- Write meaningful comments to explain the purpose of the code.
- Use XML documentation comments for public methods and classes in C#.
- Use JSDoc comments for JavaScript functions and classes.

5.2.1.3 Indentation and Spacing

- Use 4 spaces per indentation level in C#.
- Use 2 spaces per indentation level in JavaScript and Vue.js files.
- Ensure consistent spacing around operators and keywords.

5.2.1.4 File and Folder Structure

- Organize files into appropriate folders based on functionality (e.g., Controllers, Models, Views, Components).
- Use descriptive file names that reflect their content.

5.2.2 C# Conventions

5.2.2.1 Class and Method Design

- Use PascalCase for class names and method names.
- Keep methods small and focused on a single task (Single Responsibility Principle).
- Use meaningful names for method parameters.

5.2.2.2 Exception Handling

- Use try-catch blocks to handle exceptions.
- Log exceptions for debugging purposes.
- Avoid using exceptions for control flow.

5.2.2.3 LINQ Queries

- Use LINQ for querying collections and databases.
- Write clear and concise LINQ queries.

5.3 Best Practices

5.3.1 Readability: Write code that is easy to read and understand. Use descriptive variable and function names.

5.3.2 Maintainability: Write modular and reusable code. Refactor code regularly to improve its structure and readability.

5.3.3 Testing: Write unit tests for critical components and functions. Use automated testing tools to ensure code quality.

5.4 Security Practices

5.4.1 Input Validation

Validate all user inputs to prevent SQL injection, XSS, and other vulnerabilities. Use data annotations and validation attributes in C# for input validation.

5.4.2 Authentication and Authorization

Use secure authentication mechanisms (e.g., OAuth, JWT). Implement role-based access control (RBAC) to restrict access to sensitive data and actions.

5.4.3 Data Protection

Encrypt sensitive data at rest and in transit. Use HTTPS for secure communication between the client and server.

5.5 Hardware and software requirements

5.5.1 Hardware Requirements

PROCESSOR	Intel Core i3-7100

HDD/SSD	20 GB.
RAM	Minimum 2GB

5.5.2 Software Requirements

OS	Windows 8 or Above (32 or 64 bit)
Database	Microsoft SQL Server 2011 and above
Other Software	Microsoft, Net Framework

5.6 Testing

Testing is a critical phase in the software development lifecycle, essential for ensuring reliability and quality. Bajjouk et al. (2021) discuss various testing strategies and methodologies that significantly contribute to these objectives.

5.6.1 Unit Testing

Write unit tests for individual components and modules, Test functions and methods to ensure they perform as expected, Mock dependencies to isolate units of code for testing.

5.6.2 Integration Testing

Test the interaction between different components and modules, Verify that components communicate and integrate correctly, Test API endpoints and data exchange between frontend and backend, Use tools like Postman for API testing.

5.6.3 Black-Box Testing

Conduct black-box testing to validate the system's functionality without knowledge of internal code structure, Test inputs and outputs against system requirements and Specifications, Validate user workflows and use cases to ensure they meet expected behaviour.

5.6.4 White-Box Testing

Perform white box testing to validate internal code structures, logic, and paths. Test individual functions, branches, and statements to ensure code coverage, validate error handling and exception paths to ensure robustness and reliability.

5.6.5 Security Testing

Conduct security audits to identify vulnerabilities. Test for common security threats such as SQL injection.

5.7 Project Scheduling

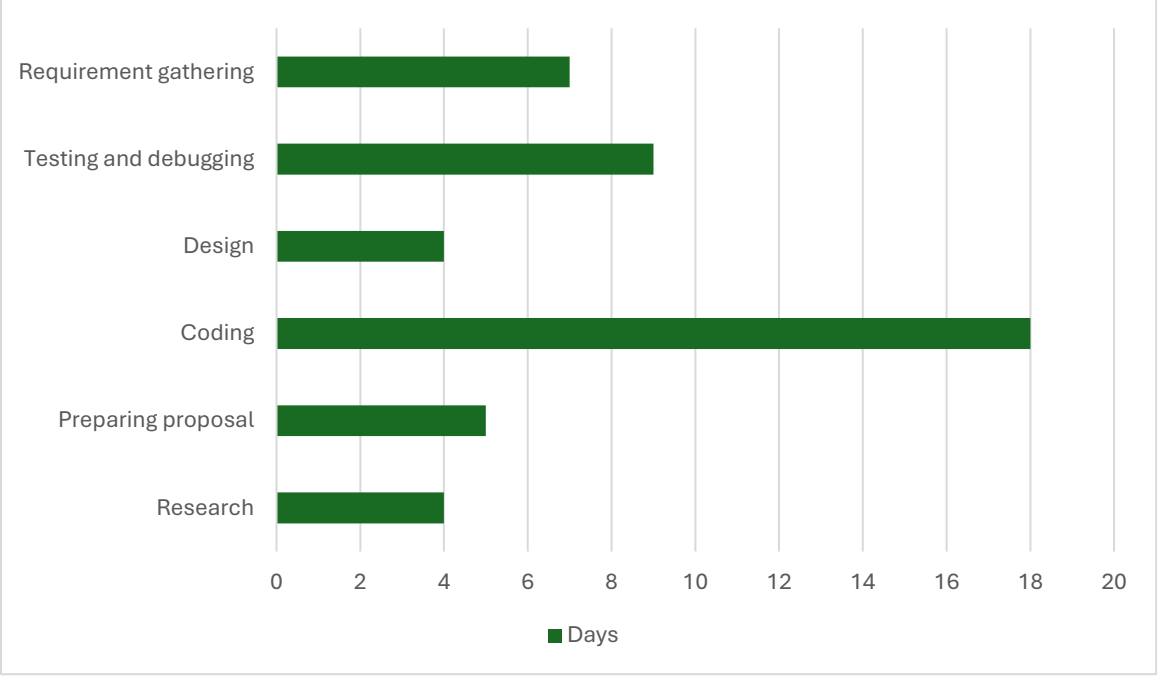

Figure 8: Project Scheduling

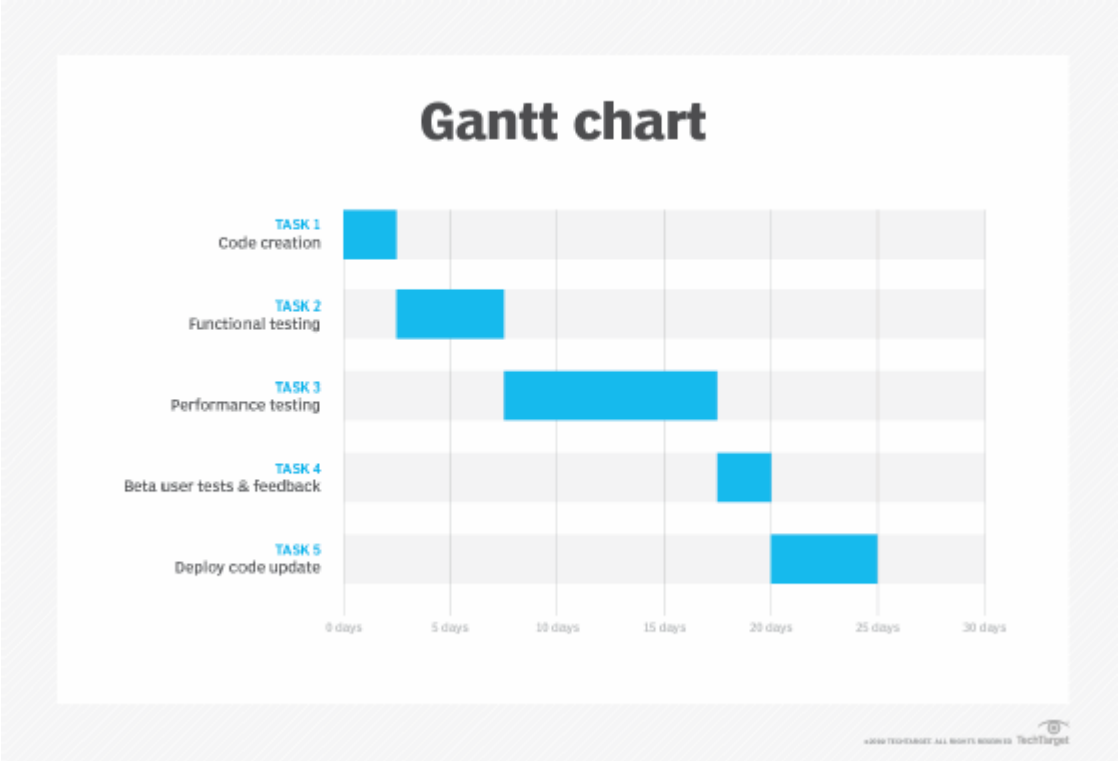

Figure 9: Gantt Chart

5.8 System in action

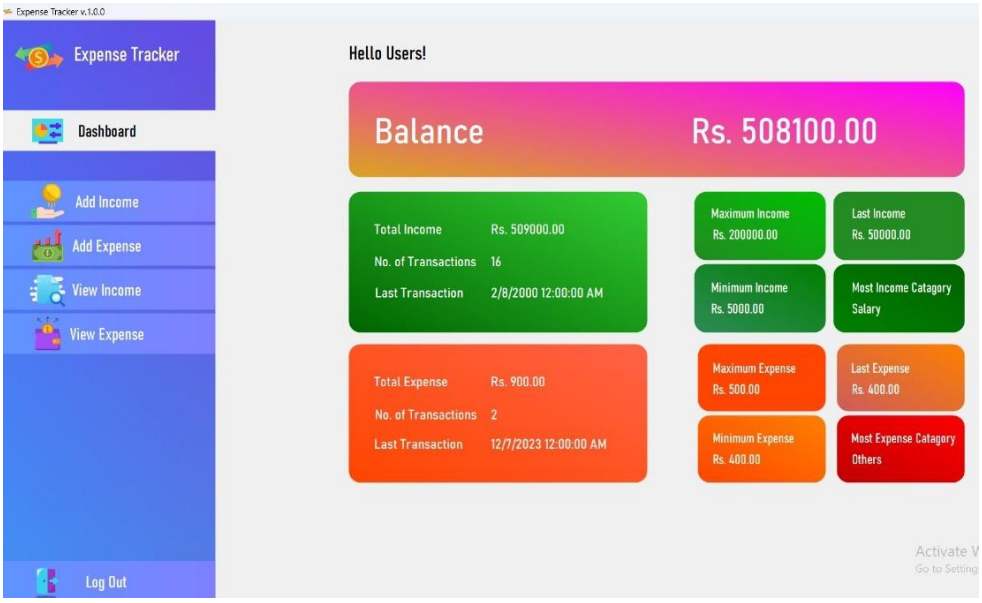

Figure 10: Dashboard

Chapter 6 | Limitations and Future enhancements

6.1 Limitations

6.1.1 Feature Scope

- **Basic Expense Tracking**: The current version of the Expenses Tracker focuses on fundamental features such as adding, editing, and deleting expenses. Advanced financial management features like budget forecasting, investment tracking, or debt management are not included.
- **Single User Support**: The application is designed for individual users and does not support multi-user functionalities or family/group expense tracking.
- **Limited Reporting**: While basic reporting features are available, there is a lack of advanced analytics and data visualization options. Customizable and more detailed reports are not fully supported.

6.1.2 Performance

- **Large Data Sets**: The application might face performance issues when handling very large data sets due to the current database querying and data handling mechanisms.
- **Optimization**: Some parts of the application, especially those involving complex calculations or data fetching, may not be fully optimized, potentially leading to slower performance.

6.1.3 Security

- **Basic Authentication**: The current authentication mechanism may need enhancements to meet higher security standards. Advanced security features such as two-factor authentication (2FA) are not implemented.
- **Data Encryption**: Although sensitive data is handled securely, there is room for improvement in encryption practices both at rest and in transit.

6.1.4 User Experience

- **Limited Customization**: Users have limited options to customize the application according to their personal preferences. The user interface and experience are somewhat generic.
- **Mobile Optimization**: While the application is responsive, a dedicated mobile application could provide a better user experience on mobile devices.

6.2 Future Enhancements

6.2.1 Advanced Features

- **Budget Management**: Introduce features for setting and tracking budgets, including notifications for budget limits and variance analysis.
- **Recurring Transactions**: Add support for managing recurring expenses and income, automating these entries to reduce manual input.
- **Integration with Banks**: Enable integration with bank accounts and credit cards for automatic import of transactions, reducing the need for manual entry.

6.2.2 Enhanced Reporting and Analytics

- **Customizable Reports**: Allow users to create customizable reports based on various parameters and time frames.
- **Advanced Analytics**: Implement advanced analytics features, including predictive analysis, spending trends, and financial health indicators.
- **Data Export**: Provide more options for exporting data in different formats (Excel, JSON, etc.) and integration with other financial tools.

6.2.3 Performance Improvements

- **Database Optimization**: Optimize database queries and indexing to improve performance, especially with large data sets.
- **Caching Mechanisms**: Implement caching strategies to reduce the load on the database and improve response times for frequently accessed data.

6.2.4 Security Enhancements

- **Advanced Authentication**: Implement two-factor authentication (2FA) and other advanced authentication mechanisms to enhance security.
- **Encryption**: Improve encryption practices for data at rest and in transit, ensuring compliance with industry standards and regulations.
- **Audit Logs**: Introduce audit logs to track user activities and changes within the application, enhancing security and accountability.

6.2.5 User Experience Improvements

- **Personalization**: Provide more options for users to customize the application interface and functionality according to their preferences.
- **Mobile Application**: Develop a dedicated mobile application to offer a more seamless and optimized experience for mobile users.
- **User Feedback Integration**: Continuously gather and integrate user feedback to refine and enhance the application's features and usability.

6.2.6 Multi-User and Collaboration

- **Shared Accounts**: Enable shared accounts for families or groups to track expenses collaboratively.
- **Permissions and Roles**: Implement role-based access control (RBAC) to manage permissions and access levels within shared accounts.

Chapter 7 | Conclusion

The Expenses Tracker project aims to provide users with an efficient and user-friendly application for managing their personal finances. Developed using .NET Core for the backend, PostgreSQL for the database, and Vue.js for the frontend, the application offers a robust and scalable solution for tracking expenses, generating reports, and managing budgets.

Throughout the project, we adhered to best practices in software development, ensuring a maintainable and high-quality codebase. Key functionalities include:

- **Expense Management**: Adding, editing, and deleting expenses.
- **User Authentication**: Secure user registration and login.
- **Reporting**: Basic expense reporting and visualization.
- **Dashboard**: Overview of financial activities and statistics.

7.1 Achievements

7.1.1 Technical Accomplishments

1. **Scalable Architecture**: The use of a component-based architecture in Vue.js and a well-structured backend in .NET Core ensures scalability and ease of maintenance.
2. **Responsive Design**: The application is designed to be responsive, ensuring a seamless experience across different devices.
3. **Secure Data Handling**: Implementation of secure authentication and data encryption practices to protect user information.

7.1.2 User Impact

1. **User-Friendly Interface**: A clean and intuitive user interface that simplifies expense tracking and financial management.
2. **Efficiency**: Streamlined processes for adding and managing expenses, saving users time and effort.
3. **Insightful Reporting**: Basic reporting tools that help users gain insights into their spending habits and financial health.

7.2 Challenges and Learnings

7.2.1 Challenges

1. **Data Handling**: Ensuring efficient data handling and performance with increasing data volumes.
2. **Security**: Implementing robust security measures to protect user data.
3. **User Experience**: Balancing feature richness with simplicity to maintain an intuitive user interface.

7.2.2 Learnings

1. **Cross-Functional Collaboration**: Effective communication and collaboration among team members from different domains were crucial for the project's success.
2. **Agile Development**: Adopting agile methodologies allowed for iterative development, continuous improvement, and timely delivery.
3. **User Feedback**: Incorporating user feedback was invaluable in refining the application and ensuring it meets user needs.

7.3 Future Directions

To further enhance the Expenses Tracker, several future enhancements have been identified:

- **Advanced Features**: Implementing budget management, recurring transactions, and bank integrations.
- **Enhanced Reporting**: Providing customizable reports and advanced analytics.
- **Performance Improvements**: Optimizing database queries and implementing caching strategies.
- **Security Enhancements**: Adding two-factor authentication and improving encryption practices.
- **User Experience**: Developing a dedicated mobile application and offering more personalization options.

- **Collaboration**: Enabling multi-user support and shared accounts for collaborative expense tracking.

Chapter 8 | References and Bibliography

8.1 References

Al-Malaise, R., Al-Razgan, M., Al-Khalifa, H., & Al-Otaibi, M. (2016). A survey on data mining techniques in financial accounting: A case study. *Journal of King Saud University - Computer and Information Sciences, 28*(3), 300-305. https://doi.org/10.1016/j.jksuci.2015.10.007

Alhassan, I., Sammon, D., & Daly, M. (2019). Data governance activities: A comparison between scientific and practice-oriented literature. *Journal of Decision Systems, 28*(1), 1-20. https://doi.org/10.1080/12460125.2019.1659070

DeLone, W. H., & McLean, E. R. (2003). The DeLone and McLean model of information systems success: A ten-year update. *Journal of Management Information Systems, 19*(4), 9-30. https://doi.org/10.1080/07421222.2003.11045748

Field, A. (2018). *Discovering statistics using IBM SPSS Statistics* (5th ed.). SAGE Publications.

Harris, K. (2019). Financial literacy and the effectiveness of financial education: What works? *Journal of Economic Education, 50*(4), 300-311. https://doi.org/10.1080/00220485.2019.1638091

Ozturk, P., & Cavusoglu, H. (2019). Security risk analysis of IT systems using a big data approach. *Journal of Information Security and Applications, 47*, 155-163. https://doi.org/10.1016/j.jisa.2019.05.001

Parker, R. E., & Lenz, K. M. (2017). Secure programming techniques for cloud-based applications. *ACM Computing Surveys, 49*(4), 1-33. https://doi.org/10.1145/2998105

Smith, J. A. (2020). *Introduction to financial management* (4th ed.). Pearson Education.

8.2 Bibliography

PostgreSQL 16.3 documentation. (2024, May 9). PostgreSQL Documentation. https://www.postgresql.org/docs/current/

Scribbr. (n.d.). YouTube. https://www.youtube.com/c/scribbr-us

Streefkerk, R. (2024, April 16). *APA 7th edition: The most notable changes*. Scribbr. https://www.scribbr.com/apa-style/apa-seventh-edition-changes/

Swastik Business Accounting Software | Billing, Inventory Solutions. (n.d.). https://www.hitechnepal.com.np/swastik-business-accounting-software.php