Software Requirements Specification

Finance Tracker App

Prepared by Dajana Lelaj

Elja Dalipaj

Irva Sula

Kostandina Zhupaj

Kristiana Mullaj

20 May 2024

Table of Contents

Table of Contents **2**
Revision History **2**
1. Introduction **1**
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
1.5 References 1
2. Overall Description **2**
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 2
2.5 Design and Implementation Constraints 2
2.6 User Documentation 2
2.7 Assumptions and Dependencies 3
3. External Interface Requirements **3**
3.1 User Interfaces 3
3.2 Hardware Interfaces 3
3.3 Software Interfaces 3
3.4 Communications Interfaces 3
4. System Features **4**
4.1 System Feature 1 4
4.2 System Feature 2 (and so on) 4
5. Other Nonfunctional Requirements **4**
5.1 Performance Requirements 4
5.2 Safety Requirements 5
5.3 Security Requirements 5
5.4 Software Quality Attributes 5
5.5 Business Rules 5
6. Other Requirements **5**
Appendix A: Glossary **5**
Appendix B: Analysis Models **5**
Appendix C: To Be Determined List **6**

1. Introduction

1.1 Purpose

The purpose of this presentation is to introduce and demonstrate the features and benefits of the finance tracker application. The purpose of this paper is to demonstrate how this app will help the users do more things with their finances, ranging from the more general tracking of expenses to more specific areas of budgeting and data analysis. To demonstrate the app's functions, we want to show how the app can make a user's life easier when it comes to finances, help to develop financial literacy, and contribute to the formation of desirable financial behaviors.

1.2 Document Conventions

The following conventions will be used:

1. *Title: Each section will be written in bold font*
2. *Body Text: Standard text will be written in italic font.*
3. *Notes: Important additional information will be highlighted with a "Note" label.*
4. *Figures will be numbered sequentially (e.g Figure 1).*

By following these conventions, the presentation will be more clear, and easy to follow for all audience members.

1.3 Intended Audience and Reading Suggestions

This presentation documentation is targeted at a different group of users: potential customers who want to handle their finances better, people who might be interested in investing in the app or even becoming business partners with the app creator, developers and the tech staff involved in app creation and development, marketing and sales specialists who study the app's value proposition.
To understand the app properly, it is preferable to read its introduction and purpose sections first to clarify the application's goals and mission to continue research on features and functionality to master the application.
Therefore the next section which is the user interface and experience should be read thoroughly in order to understand how the interface is designed to be user friendly.
Then continue with future developments in order to find useful information about updates and other future features and improvements.
Finally, go to non functional requirements and experience should be read thoroughly in order to understand how the interface is designed to be user friendly followed by the security section where the readers will be able to learn about the methods used to protect the data.

1.4 Product Scope

Our finance tracker app is an all in one solution to manage the short and long-term finances effectively. The software is also user-friendly to ensure that its users can monitor their income, expenses, and savings goals with effortless and efficient real-time tracking. Also, it provides integrated privacy and security along with synching of data across various devices for constant access of information.
The application is developed using JAVASCRIPT, CSS, PHP, SQL . Also users have the option to connect with Github.

2. Overall Description

2.1 Product Perspective

The Finance Tracker application assists people in solving the problem of how to manage their finances in the best way possible. The app has created a platform that will help the user to make transactions, monitor and manage money and ensure that they can securely oversee the process. It is a tool for analyzing users' financial health and enhancing it.

2.2 Product Functions

The main functions of the program include:

1. *User Registration: Make an account available for users to begin to use the app.*
2. *Expense Tracking: Create a platform where people can keep track of what they spend.*
3. *Income Tracking: Permit users to input their salary information*

2.3 User Classes and Characteristics

The application is targeted and designed for various user classes even individuals, small business, freelancers, students, family as well the retirees affects the app in a variety of ways. These products and services offer effective financial management and planning for the smartphone users.

2.4 Operating Environment

The program operates as a web based appliaction on both mobile devices, such as smartphones and tablets. Also through a web browser on various operating systems.

2.5 Design and Implementation Constraints

The application is developed by integrating the following technologies JAVASCRIPT, CSS, PHP and MYSQL. It corresponds to the current standards and rules of web applications programming.

2.6 User Documentation

The program provides user documentation for the finance tracker app including user manual or a tutorial to assist users in understanding its features and functionalities. The documentation explains how to do the registration and then how to put expense and income tracking.

2.7 Assumptions and Dependencies

The application includes assumptions of financial data delivery for real-time updates, user engagement assumptions such as regular usage to track expenses and income.
Stability of the connection that is necessary for the data synchronization provides reliability for different types of the devices and operation systems that are used to run the system. Stakeholders will need careful explanation of these assumptions and interdependencies to understand how the app fits the scenario and what its use-case boundaries are.

3. External Interface Requirements

3.1 User Interfaces

General GUI Standards

- *Consistency: All screens will follow a consistent layout and design theme. This includes font styles, colors, button designs, and navigation elements.*
- *Responsiveness: The UI will be responsive, ensuring usability across various devices (desktop, tablet, mobile).*
- ***Navigation bar:** Located on the left side of the screen. Includes connections to the major sections: Dashboard Expenses, Budget, and Support.Includes user account administration options (e.g., profile, settings,logout,change password).*
- ***Header:** Displays the application's name and logo. Includes dashboard for quick access to various sections and transactions. Notification icons for notifications and messages.*
- ***Help:** Available on navigation bar and directs users to the assistance department.*
- ***Save:** Used to save changes to forms.*
- ***Cancel:** Discards changes and returns to the prior state.*
- ***Delete:** To remove entries, usually accompanied by a confirmation box.*
- ***User Authentication Interface Components:***
- *Login Form: Includes email and password fields, as well as "Remember Me" and password recovery options.*
- *Sign-Up Form: User information (name, email, password) is requested, followed by a confirmation email.*
- *Password Recovery: An email-based password reset feature.*
- ***Error Message Display Standards***
- *Error Positioning: Show error warnings near the relevant input fields or at the top of the form.*
- *Error Style: To emphasize errors, use a separate color (for example, red) and an icon.*
- *Error messages: Give precise and brief descriptions of the error, and, if possible, offer corrective activities.Modal dialogs should be used to capture the user's attention when there are severe issues.*

3.2 Hardware Interfaces

Logical Characteristics

- *User Input Devices:*

    - *Keyboard: Used for text input, navigation, and keyboard shortcuts.*
    - *Mouse/Touchpad: Used for navigation, selection, and interaction with the UI.*
    - *Touchscreen: For touch-enabled devices, supporting tap, swipe, and other touch gestures.*
- *Output Devices:*
    - *Display: The UI will adapt to various screen sizes and resolutions.*
    - *Speakers: For devices with audio capabilities, used for notifications and alerts.*

Data and Control Interactions

- *Data Input:*
    - *Form Submissions: Data is entered through forms and submitted via HTTP/HTTPS requests.*
    - *Keyboard Shortcuts: Allow users to perform actions quickly using keyboard combinations.*

Communication Protocols

- *HTTP/HTTPS:*
    - *The primary protocol for data exchange between the client (user device) and the server. Ensures secure communication, especially for sensitive financial data.*

Device-Specific Considerations

- *Desktop and Laptop:*
    - *Operating Systems: Windows, macOS, Linux.*
    - *Browsers: Chrome, Firefox, Safari, Edge.*
    - *Peripherals: External keyboards, mice, and monitors.*
- *Tablets and Smartphones:*
    - *Operating Systems: iOS, Android.*
    - *Browsers: Safari (iOS), Chrome (Android), and other mobile browsers.*

3.3 Software Interfaces

Database Management System: MySQL: *The goal is to save user data, financial transactions, budget information, and system settings.*
Data In: User registrations, login information, income and expense records, and budget settings.
Data output includes user authentication details, transaction histories, budget summaries, and report data.
Communications: SQL queries for data processing and retrieval are done using the backend API.

Development Tools and Libraries for Frontend:

- *React.js: For creating the user interface.*
- *Bootstrap : A framework for creating responsive designs and styles.*
- *Backend:*
- *Node.js: A runtime environment for server-side programs.*
- *JSON Web Token: Used for user authentication.*

Data Items and Messages

- *User Data:*
  - *Incoming: Registration details, login credentials, profile updates.*
  - *Outgoing: Authentication tokens, user profile information.*
  - *Purpose: Manage user accounts and authentication.*
- *Financial Transactions:*
  - *Incoming: Income entries, expense records, transaction categories.*
  - *Outgoing: Transaction lists, filtered financial records.*
  - *Purpose: Track and manage user finances.*
- *Budget Details:*
  - *Incoming: Budget creation and updates, savings goals.*
  - *Outgoing: Budget summaries, alerts, and notifications.*
  - *Purpose: Help users manage and monitor their budgets.*

Services and Communication

- *Web Server:*
  - *Service: Serve the frontend application and handle API requests.*
  - *Communication: HTTP/HTTPS requests and responses.*
- *Authentication Service:*
  - *Service: Authenticate users and manage sessions.*
  - *Communication: JWT for secure token-based authentication.*
- *Database Service:*
  - *Service: Store and retrieve application data.*
  - *Communication: SQL queries.*

3.4 Communications Interfaces

- ***Web Browser:***

Access the web application using common browsers.
Browsers supported include Chrome, Firefox, Safari, and Edge.
HTTP and HTTPS are the protocols used.

- ***Requirements:***

HTML5/CSS3: Used to render the user interface.
JavaScript is used to create dynamic content and implement client-side logic.
Web Storage: Used to store session data and user preferences.

- ***Network Server Communications***

Function: To facilitate communication between the client (web browser) and the server.

- ***Protocols:***

HTTP/HTTPS: HTTPS is essential for all data transfers in order to maintain security.
WebSockets: Used to send real-time data updates and notifications.

- ***Email Notifications***

Function: Send emails for user registration, password recovery, and notice.

4. Finance Tracking Web application features

4.1 User Registration

4.1.1 Description and Priority

Description: This feature allows new users to create an account by providing their personal details. It includes email verification to activate the account.
Priority: High
Priority Component Ratings:

- *Benefit: 9*
- *Penalty: 8*
- *Cost: 4*
- *Risk: 3*

4.1.2 Stimulus/Response Sequences

1. *User Action: The user navigates to the registration page and fills in their name, email, and password.*
    - *System Response: The system validates the input fields.*
2. *User Action: The user submits the registration form.*
    - *System Response: The system creates a new user account and sends a verification email.*
3. *User Action: The user clicks on the verification link in the email.*
    - *System Response: The system verifies the email and activates the user account.*

4.1.3 Functional Requirements

REQ-1: The system must provide a registration form that includes fields for name, email, and password.
REQ-2: The system must validate that the email address is in the correct format and that the password meets the security requirements (e.g., minimum length, complexity).
REQ-3: The system must check if the email address is already in use and notify the user if it is.
REQ-4: The system must send a verification email to the provided email address upon successful form submission.
REQ-5: The system must provide a mechanism to verify the user's email address via a unique verification link.

4.2 Expense Tracking

4.2.1 Description and Priority

Description: This feature allows users to track their expenses by entering detailed information such as amount, date, category, description, and payment method. It helps users monitor their spending habits and manage their budgets effectively.
Priority: High
Priority Component Ratings:

- *Benefit: 9*
- *Penalty: 7*
- *Cost: 5*
- *Risk: 4*

4.2.2 Stimulus/Response Sequences

1. *User Action: The user selects the "Add Expense" option from the dashboard or navigation menu.*
    - *System Response: The system displays the expense entry form.*
2. *User Action: The user fills in the expense details such as amount, date, category, description, and payment method.*
    - *System Response: The system validates the input fields and saves the expense data.*
3. *User Action: The user submits the expense entry form.*
    - *System Response: The system updates the user's expense records and recalculates budget metrics if necessary*

4.2.3 Functional Requirements

REQ-1: The system must provide an expense entry form with fields for amount, date, category, description, and payment method.
REQ-2: The system must validate that the amount field is numeric and non-negative, and that the date field is in the correct format.
REQ-3: The system must allow users to select from predefined expense categories or add custom categories.
REQ-4: The system must provide options for users to specify the payment method used for the expense.
REQ-5: The system must save the expense data to the user's account upon submission of the expense entry form.

4.3 Budget Management

4.3.1 Description and Priority

Description: This feature enables users to create and manage budgets for different expense categories and overall monthly budgets. It helps users set financial goals and track their spending against these goals.
Priority: High
Priority Component Ratings:

- *Benefit: 8*

- *Penalty: 7*
- *Cost: 6*
- *Risk: 5*

4.3.2 Stimulus/Response Sequences

1. *User Action: The user navigates to the budget management section of the website.*
    - *System Response: The system displays the budget creation form and existing budget summaries.*
2. *User Action: The user fills in the budget creation form, specifying budget amounts for various expense categories and an overall monthly budget.*
    - *System Response: The system validates the input fields and saves the budget data.*
3. *User Action: The user submits the budget creation form.*
    - *System Response: The system updates the user's budget records and recalculates budget metrics if necessary.*

4.3.3 Functional Requirements

REQ-1: The system must provide a budget creation form with fields for specifying budget amounts for various expense categories and an overall monthly budget.

REQ-2: The system must validate that the budget amounts are numeric and non-negative.

REQ-3: The system must allow users to specify budget amounts for pre-defined expense categories or add custom categories.

REQ-4: The system must save the budget data to the user's account upon submission of the budget creation form.

REQ-5: The system must recalculate budget metrics (e.g., total budget, remaining budget) based on the newly created budget.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Quick response times are essential for the finance tracking app to give users a smooth, efficient experience. A respectable total of simultaneous users should not significantly impair the program's performance. Efficient data retrieval and computation are essential for timely budget computations, even when dealing with extensive datasets.

5.2 Safety Requirements

User data security should be given top priority in the program, and precautions should be taken to safeguard sensitive data. Mechanisms for authentication and authorization of users ought to be in place to stop illegal access to the application and user information. To reduce potential risks, the program should follow business standard procedures and security standards. To ensure strong protection, regular security inspections and patches should be carried out. To protect privacy, user data should also be encrypted when it's in transit and also at rest.

5.3 Security Requirements

Through the use of encryption methods, the software should provide secure interaction between the browser used by the user and the server. Sensitive data, such as user passwords, should be encrypted and hashed appropriately before being kept. To fix problems with security, the program should update its components and software dependencies on a regular basis. To find and address any security issues, penetration tests should be done on a regular basis. Furthermore, the least privilege principle ought to be used to limit access to sensitive data.

5.4 Software Quality Attributes

The program should be written using best practices to ensure that it remains clean, modular, and appropriately commented. It should be tested for correctness and reliability to the highest extent plausible. It should therefore be in a form that is easily extended and easily maintained as appropriate.

5.5 Business Rules

The program should ensure any conditions required for the handling of income and expenses are catered for. It should be used to make sure that input data is accurate and secure.
It should handle exceptions and prompt clear error messages when inputs are invalid or any exceptional situations.

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Other Requirements

Database Requirements

- *Data Integrity: Utilize relationships and restrictions in the database structure to guarantee data integrity. When appropriate, primary keys, foreign keys, and specific limitations ought to be employed.*
- *Data Backup: To avoid data loss, regularly schedule automated database backups. Daily backups should be performed, and retention rules should be set up to save backups for at least 30 days.*
- *Scalability: Without appreciable performance deterioration, the structure of the database should be able to handle increases in the number of users and data volume. It is advisable to take into account partitioning, splitting, and indexing techniques.*
- *Data Migration: Offer procedures and instruments for transferring data from current financial monitoring systems to this programme. Achieve data integrity and compatibility while migrating.*

Internationalization Requirements

- *Language Support: In order to serve a worldwide user base, the programme should support a number of languages. English, Albanian, French, and German should be the starting languages supported, with the option to add additional as required.*
- *Currency Support: Enable users to track their earnings and expenditures in the currency of their choice. Conversion rates and the ability to provide financial information in the user's selected currency should be supported by the application.*
- *Date and Number forms: Depending on the user's locale preferences, the programme should adjust to local time and number forms.*
- *Localized Content: Make sure that every piece of user-facing content, such as alerts, error messages, and support materials, is completely localized.*

Legal Requirements

- *Data Protection: Obey to data protection laws, such as the General Data Protection Regulation (GDPR) for users in the European Union.*
- *Privacy Policy: Provide a clear description of the procedures for gathering, using, storing, and sharing user data. When required, get the user's permission.*
- *Terms of Service: Before users may use the programme, they must accept the terms of service agreement. The usage of the programme, user obligations, and liability disclaimers should all be included in this agreement.*
- *Financial rules: Verify that the application complies with all applicable financial rules, including KYC (Know Your Customer) standards and AML (Anti-Money Laundering) legislation, in the locations where it is accessible.*

Reuse Objectives

- *Modular Design: To encourage component reuse across various application sections and in subsequent projects, develop the application utilizing a modular design.*
- *API Reuse: Create reusable APIs to enable smooth integration of the financial tracker with other programmes or services.*
- *Code reuse: To make sure that code is readily maintained and reused, adhere to coding norms and best practices. When it makes sense, apply design patterns to encourage efficiency and uniformity.*
- *Documentation: To promote reuse and simplicity of integration, provide thorough documentation for each module, API, and service.*

Accessibility Requirements

- *Web Content Accessibility Guidelines (WCAG): To make an application accessible to people with disabilities, make sure it conforms with WCAG 2.1 AA criteria.*
- *Keyboard Navigation: Keyboard navigation ought to provide access to all functions and functionalities.*
- *Support for Screen Readers: Assure interaction with screen readers so as to help users who are blind or visually challenged.*
- *Color Contrast: To help people who are visually impaired, employ color schemes with great contrast.*

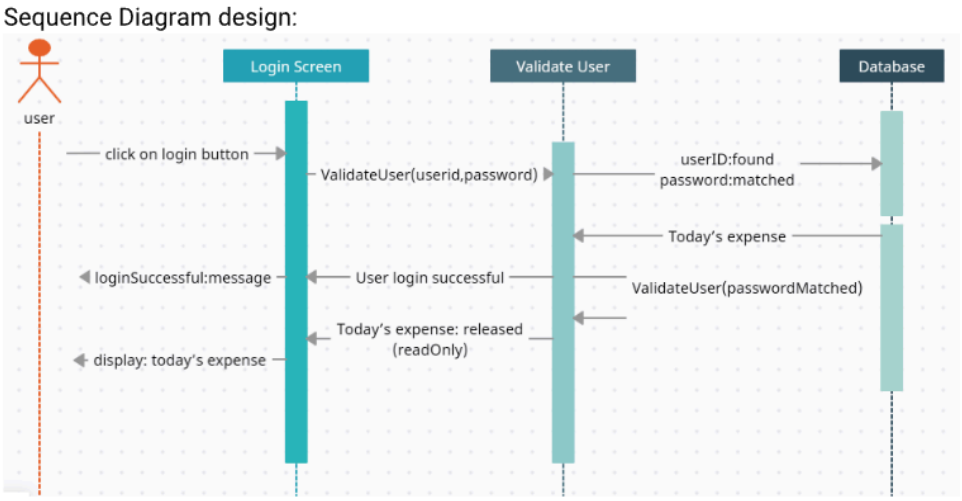
Performance Requirements

- *Response Time: For 95% of visitor engagements, the application's maximum time to respond should be 2 seconds.*
- *Test the application's load to make sure it can withstand a peak number of users at once without experiencing performance issues.*
- *Efficiency: To reduce resource consumption and enhance performance, optimise database queries and backend activities.*

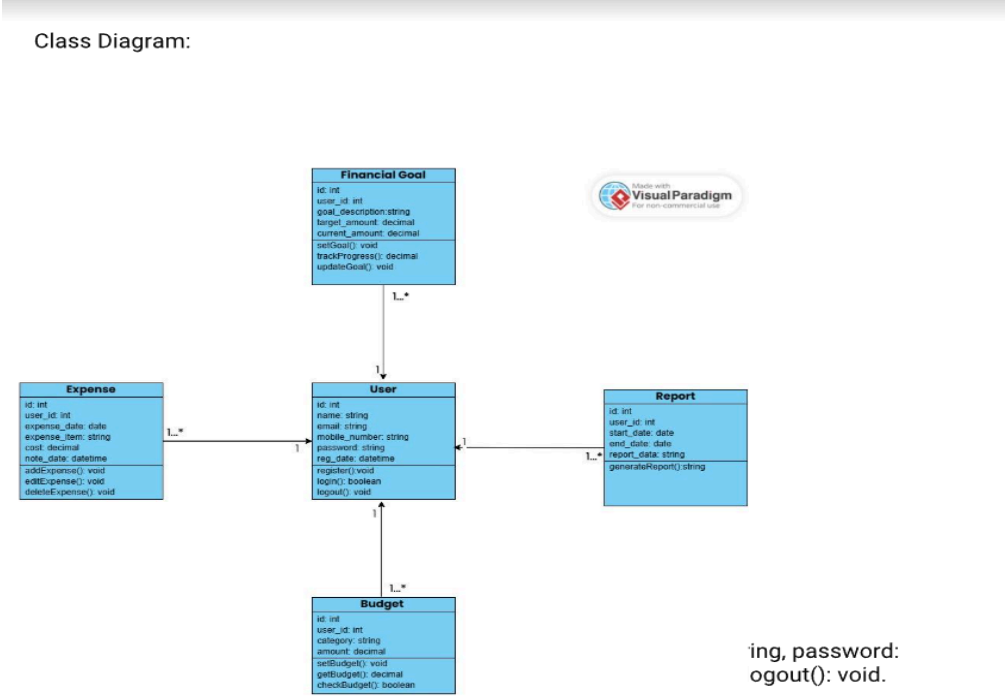
Appendix A: Glossary

- *SRS - Software Requirements Specification: A document that describes the software product to be developed, including functional and non-functional requirements.*
- *UI/UX - User Interface/User Experience: Refers to the design and interaction aspects of the software*
- *that users experience.*
- *API - Application Programming Interface: Defines methods for software components to*
- *communicate and interact with each other.*
- *HTTPS - Hypertext Transfer Protocol Secure: Protocol for secure communication over a computernetwork, commonly used for web browsing.*

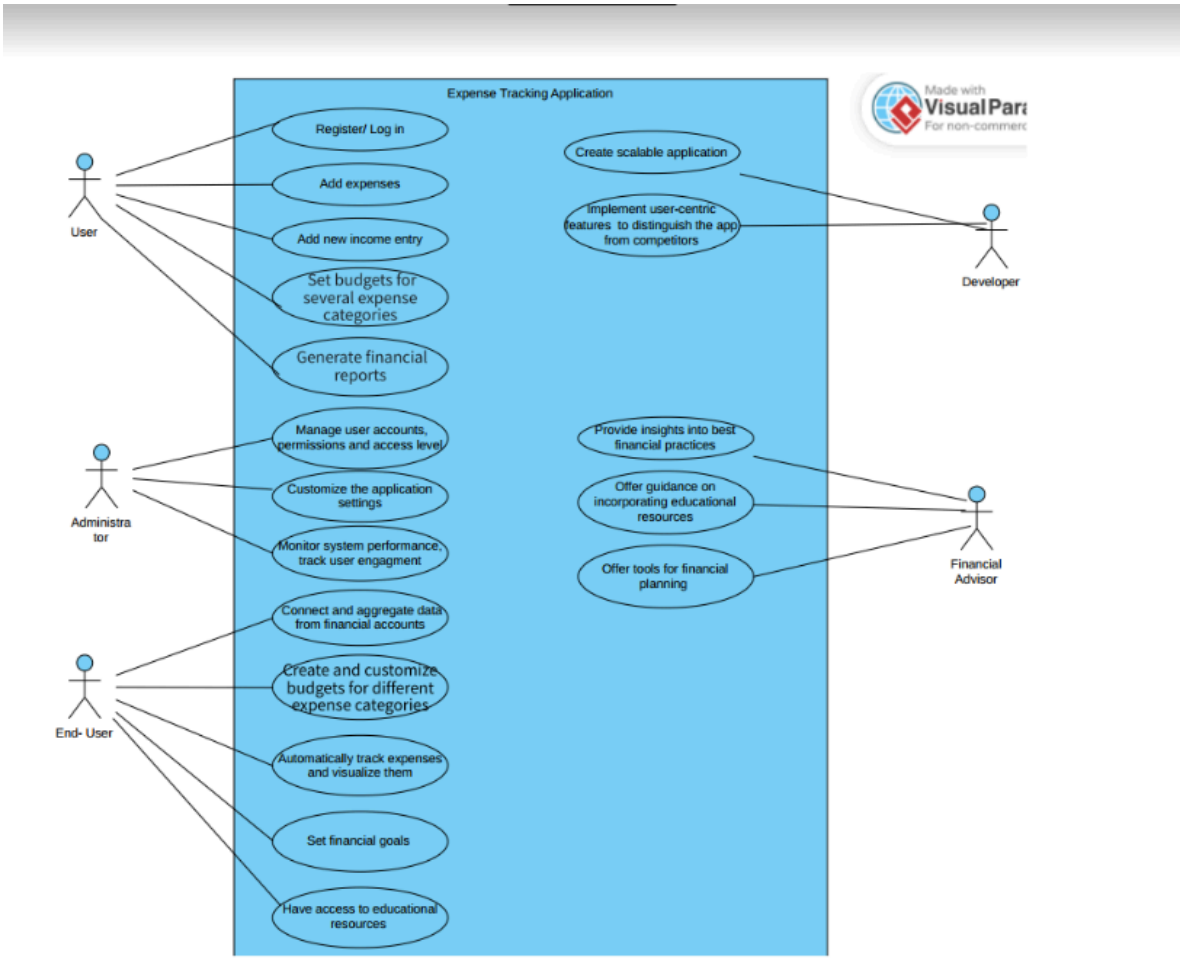
Appendix B: Analysis Models



Picture 1: Sequence diagram

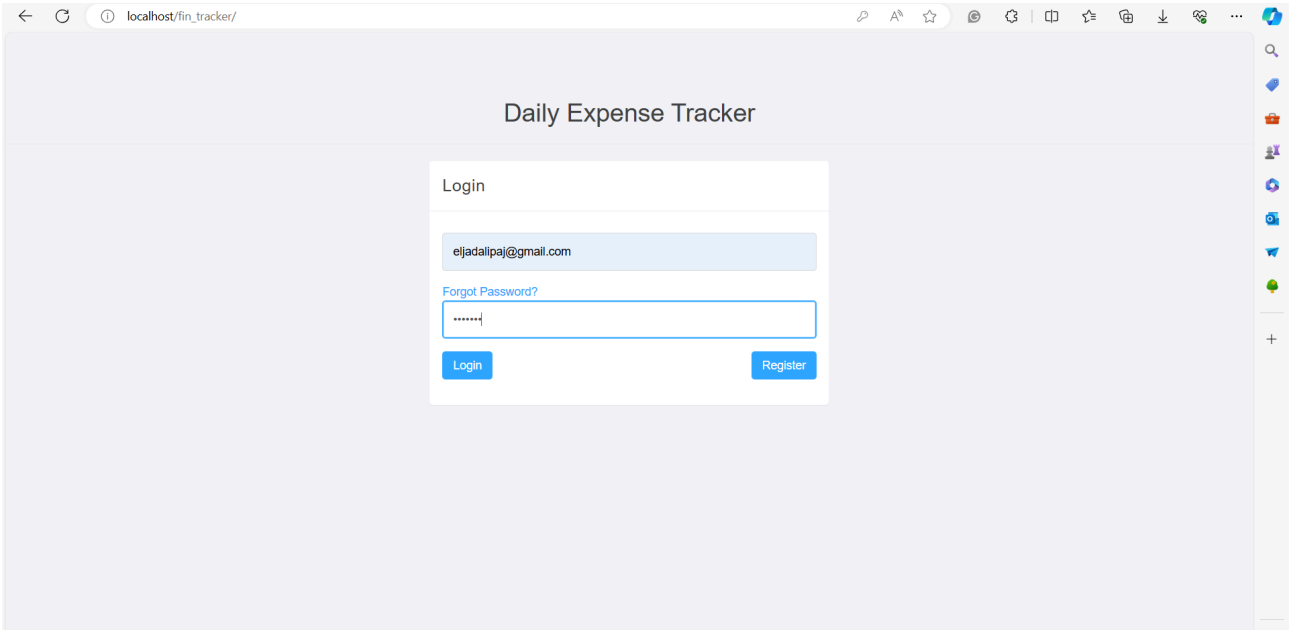


Picture 2: Class diagram

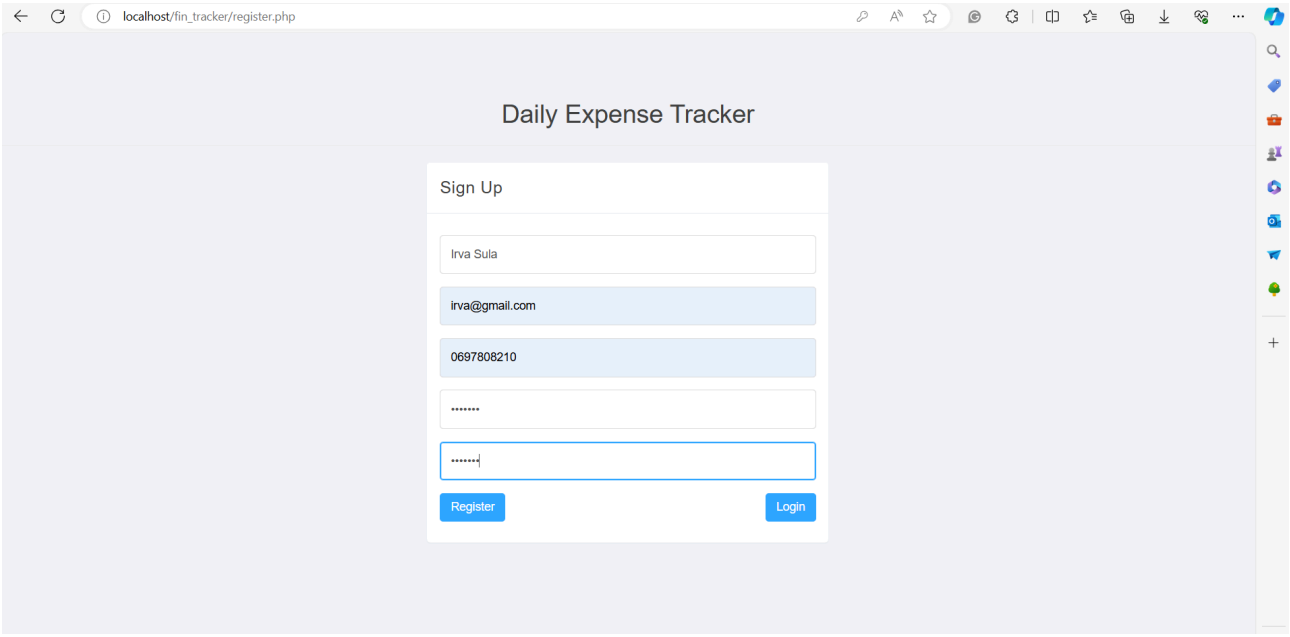


Picture 3: Use-case diagram

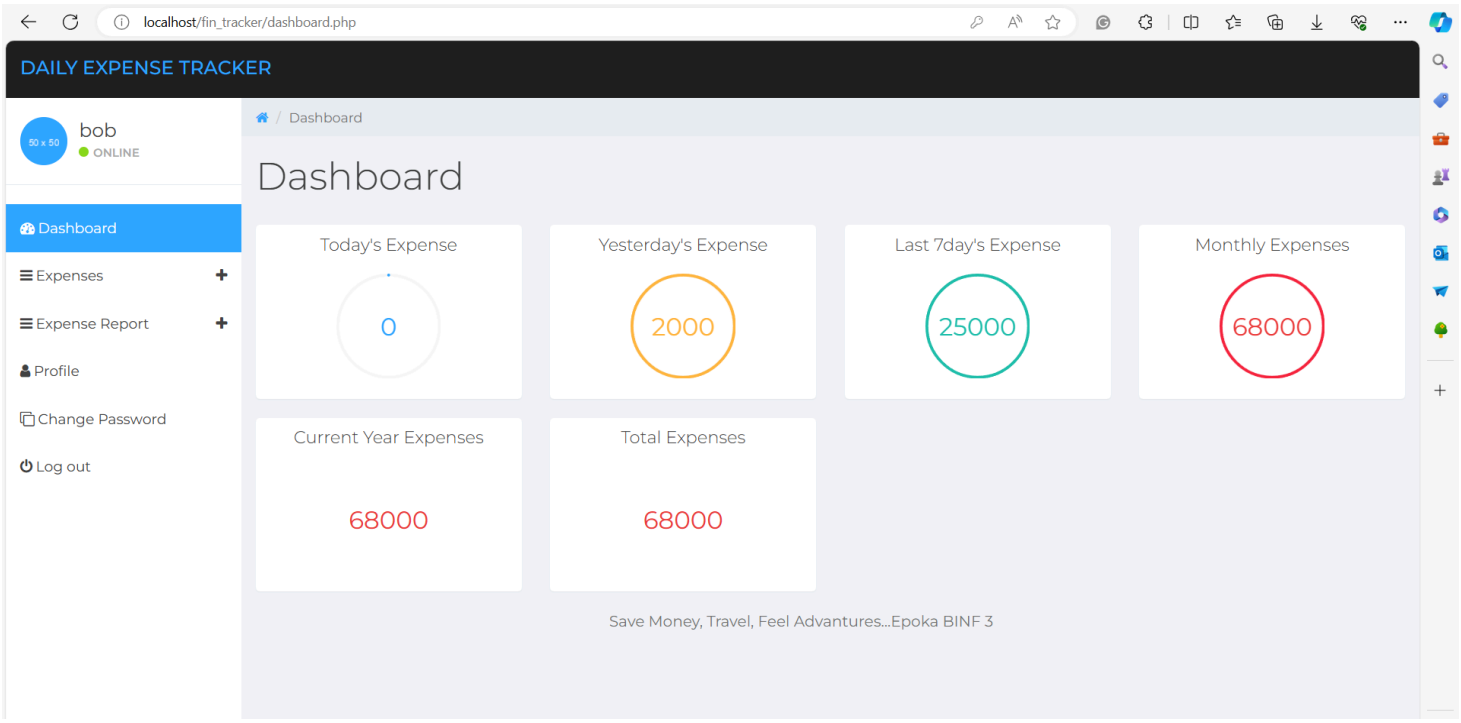
Appendix C: Project Demonstration



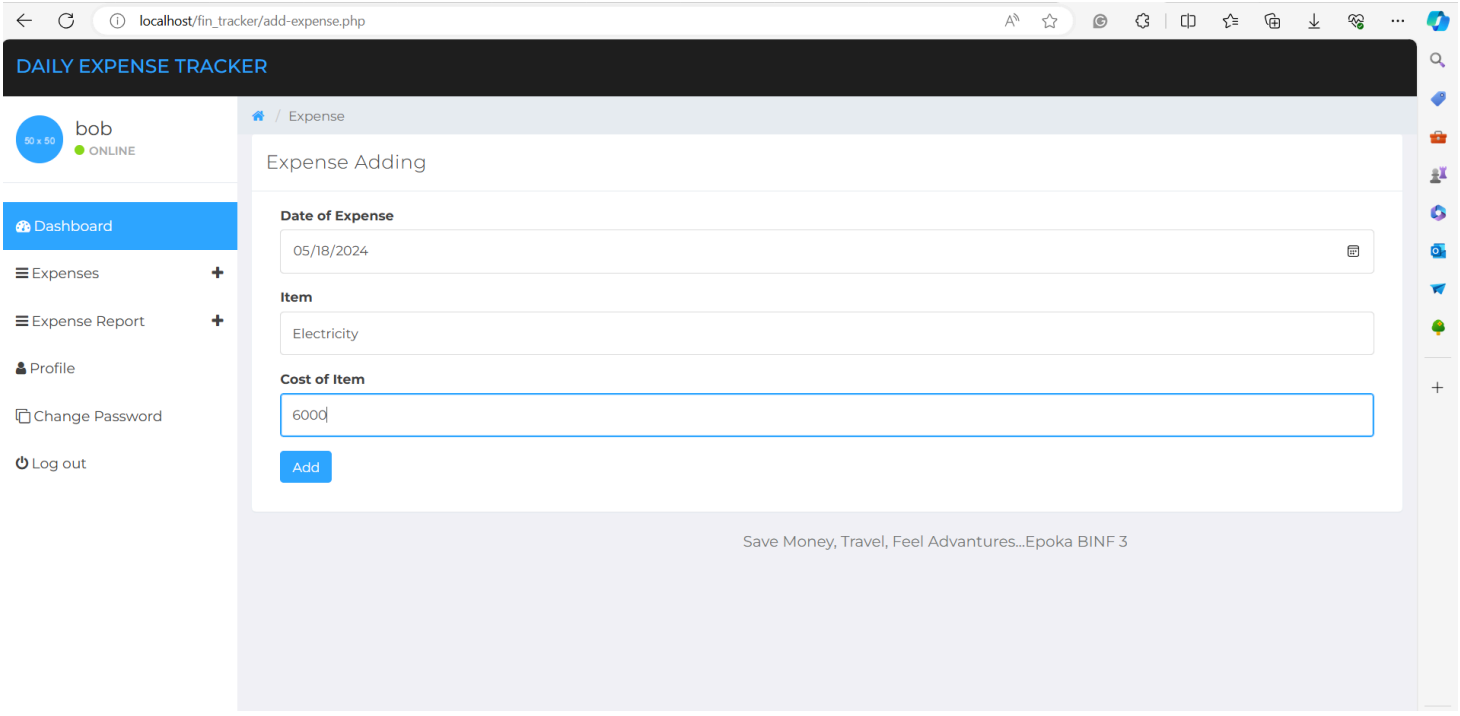
Picture 1: Log in to website; enter email address and password



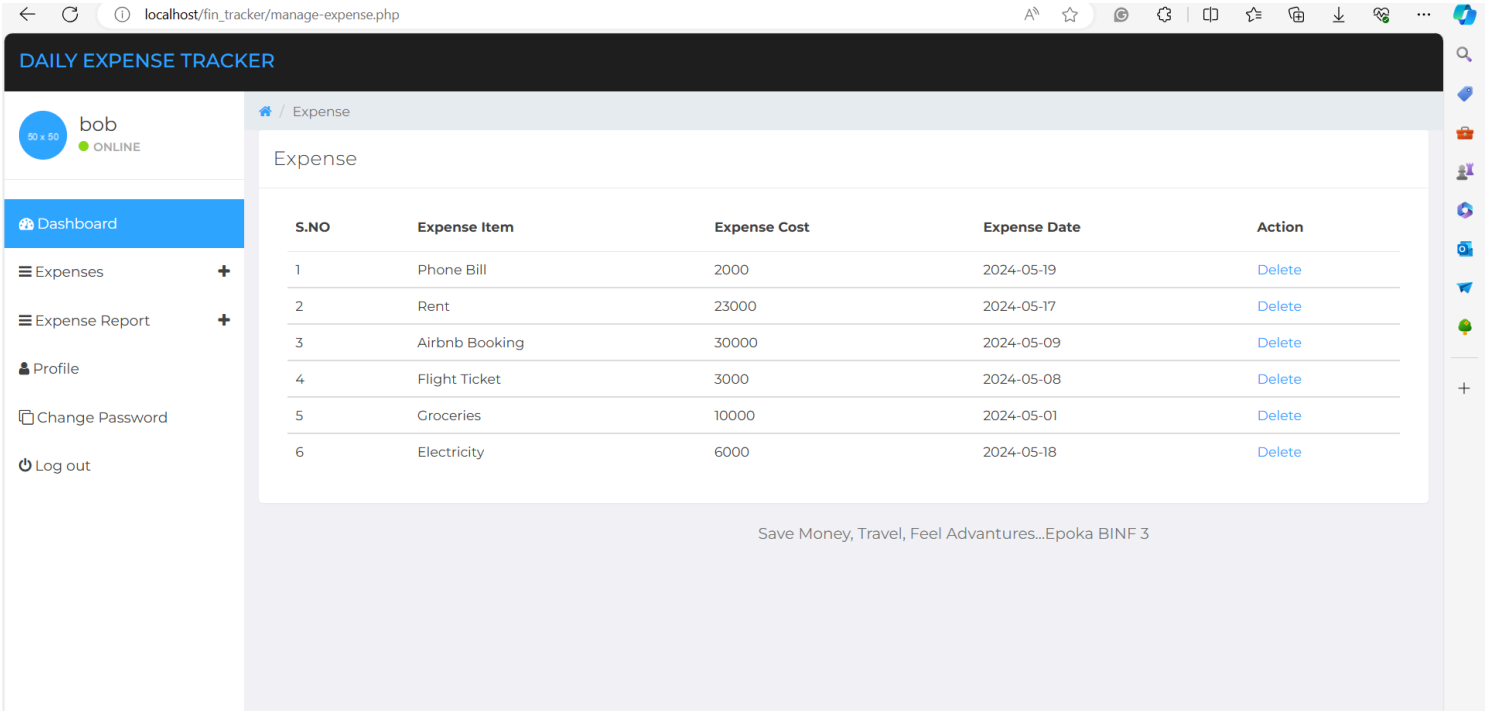
Picture 2: Register as a new user



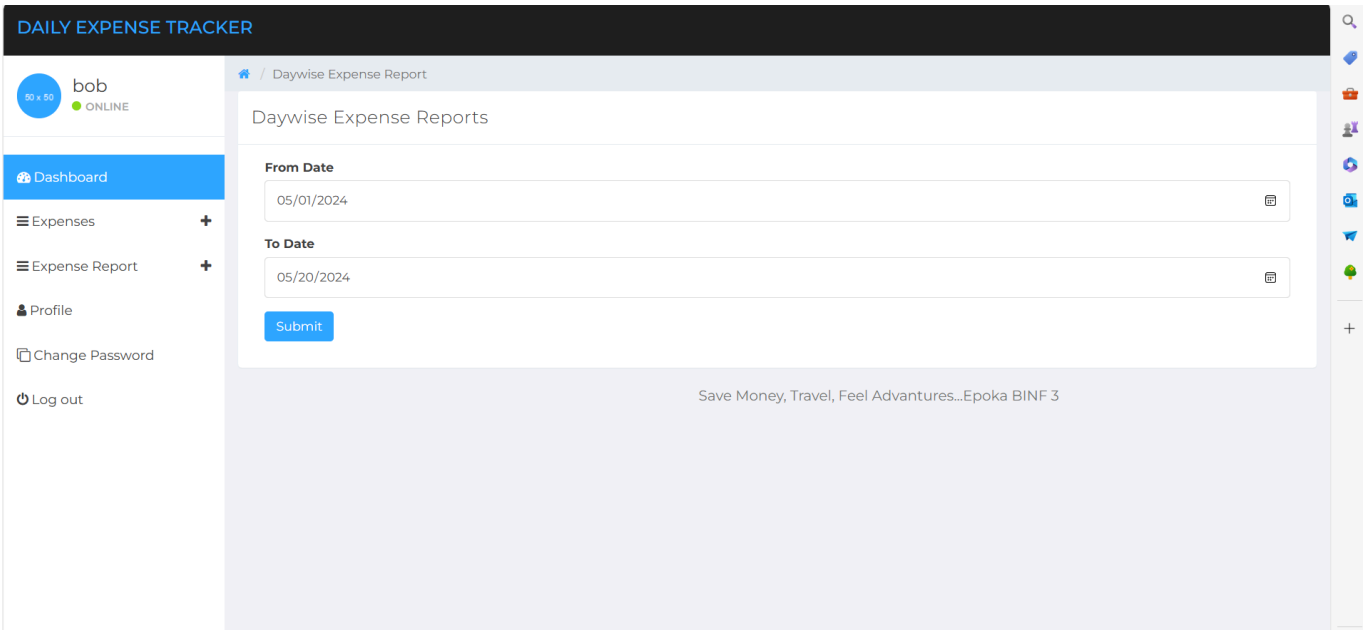
Picture 3: Dashboard where expenses are displayed



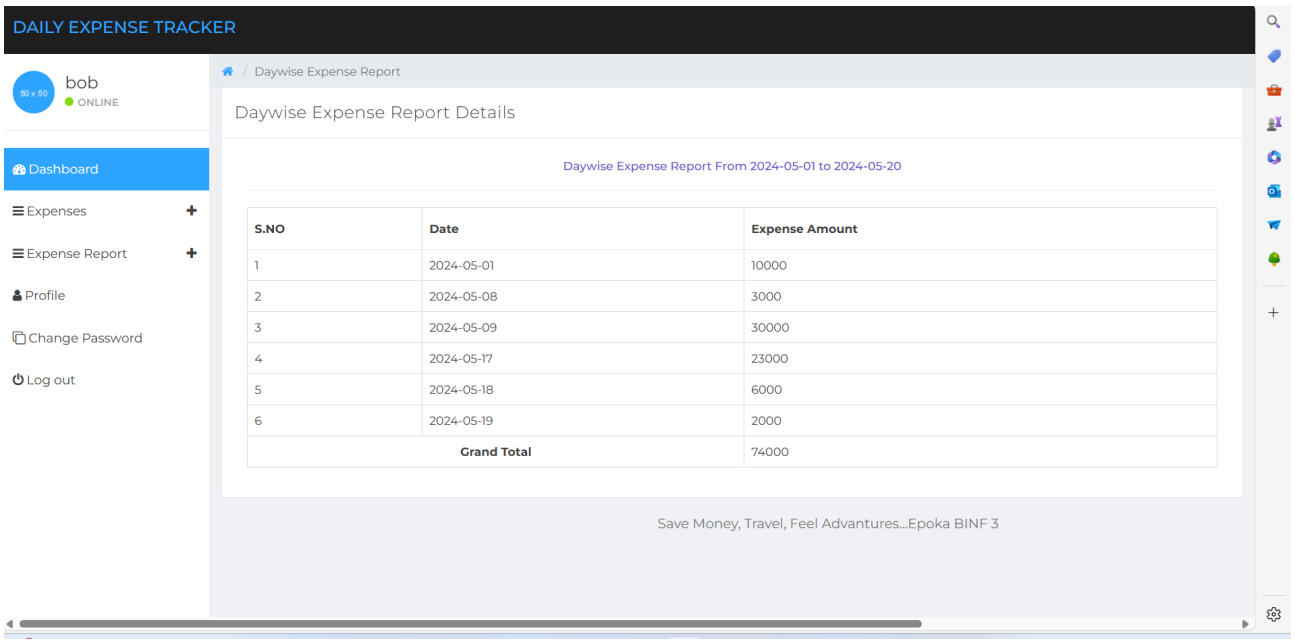
Picture 4: Adding a new expense



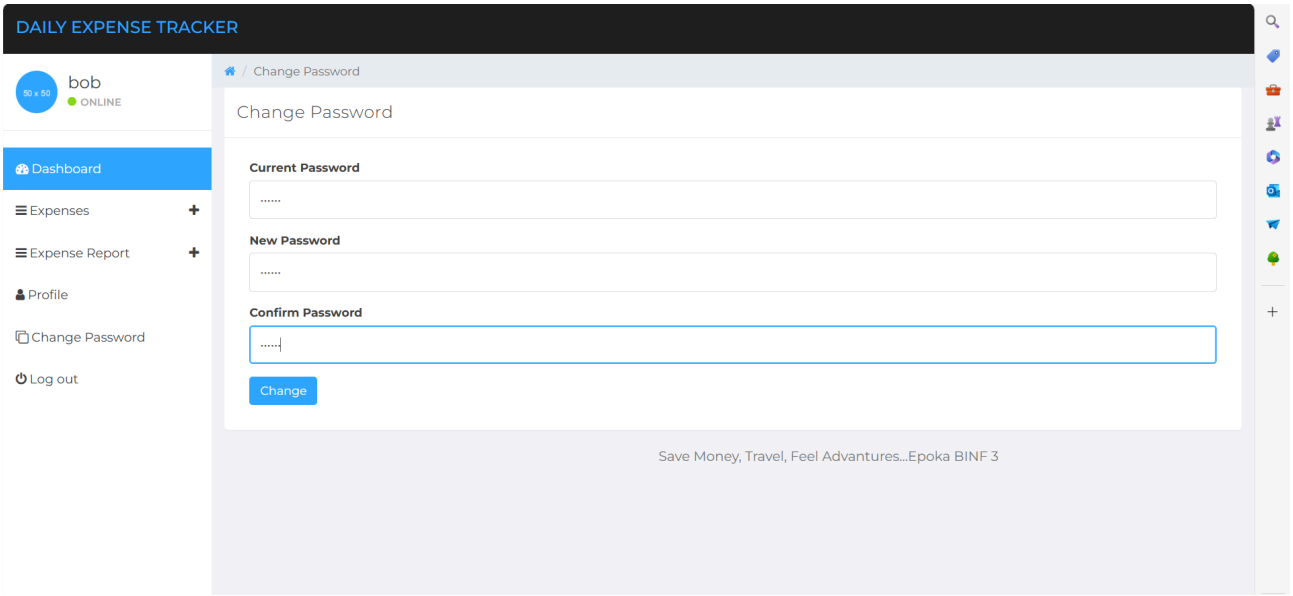
Picture 5: Listing the expenses. They can also be deleted.



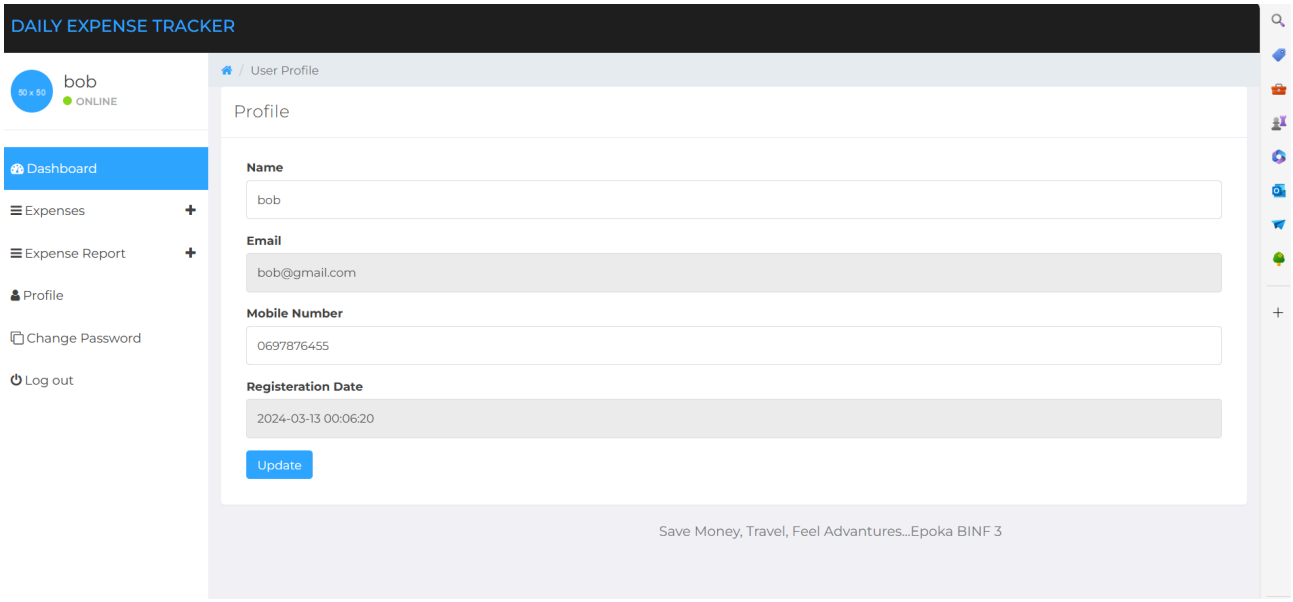
Picture 6: Calculating daywise expenses from date- to date. Can also calculate monthwise and yearwise expenses.



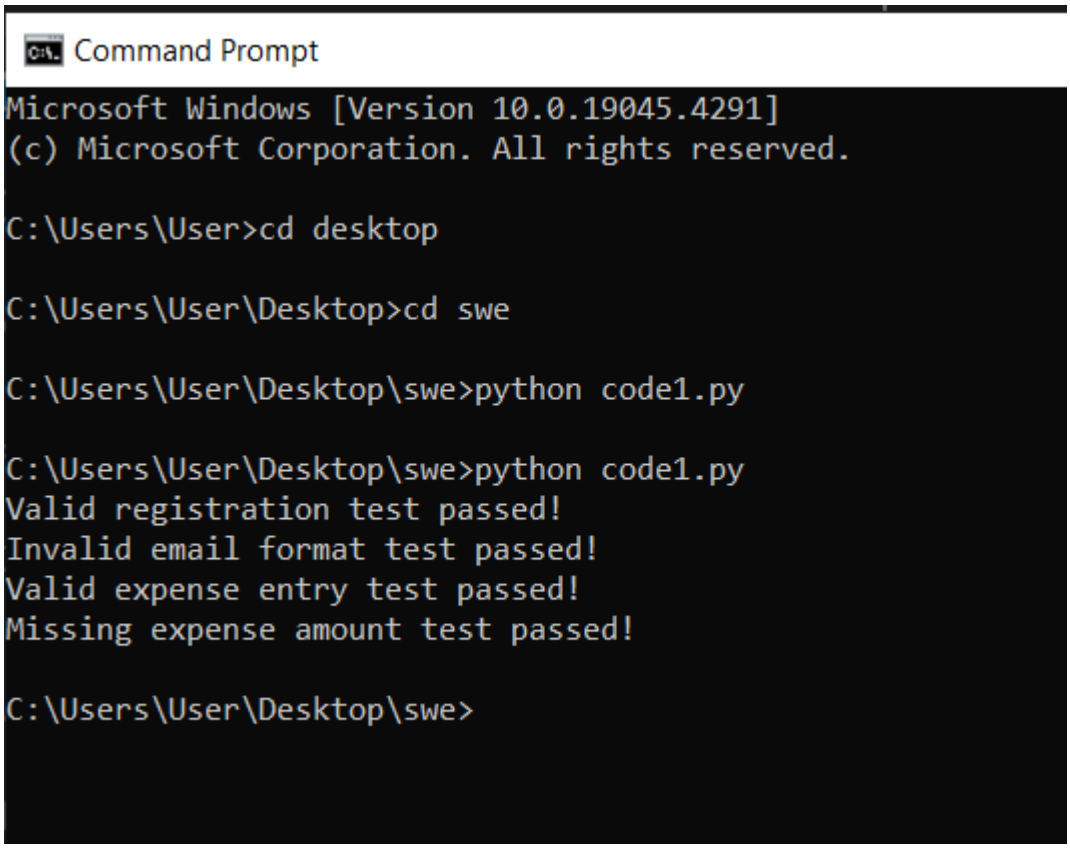
Picture 7: Display of daywise expense report details. Can also be done with monthwise and yearwise expenses.



Picture 8: Password changes in the account.



Picture 9: Account details



Picture 10: Prompt of the testing code

code1.py > ...

```python
# Mock functions for user registration and expense entry
def register_user(user_data):
    # Simulate user registration process
    if 'email' not in user_data or '@' not in user_data['email']:
        return 'invalid_email_format_error'
    else:
        # Register user in the database (not implemented in this example)
        return 'success'

def enter_expense(expense_data):
    # Simulate expense entry process
    if 'amount' not in expense_data:
        return 'missing_amount_error'
    else:
        # Add expense to the database (not implemented in this example)
        return 'success'

# Test cases for user registration
def test_valid_registration():
    # Simulate user input
    user_data = {
        'name': 'John Doe',
        'email': 'john@example.com',
        'password': 'securepassword'
    }

    # Call the registration function with user data
    registration_status = register_user(user_data)

    # Assert that registration is successful
    assert registration_status == 'success'
    print("Valid registration test passed!")

def test_invalid_email_format():
    # Simulate user input with invalid email format
    user_data = {
        'name': 'Jane Smith',
```

code1.py > ...

```python
def test_invalid_email_format():
    user_data = {
        'name': 'Jane Smith',
        'email': 'invalidemail',
        'password': 'password123'
    }

    # Call the registration function with user data
    registration_status = register_user(user_data)

    # Assert that registration fails due to invalid email format
    assert registration_status == 'invalid_email_format_error'
    print("Invalid email format test passed!")

# Test cases for expense tracking
def test_valid_expense_entry():
    # Simulate user input
    expense_data = {
        'amount': 50.00,
        'date': '2024-05-25',
        'category': 'Groceries',
        'description': 'Weekly grocery shopping',
        'payment_method': 'Credit Card'
    }

    # Call the expense entry function with expense data
    expense_entry_status = enter_expense(expense_data)

    # Assert that expense entry is successful
    assert expense_entry_status == 'success'
    print("Valid expense entry test passed!")

def test_missing_expense_amount():
    # Simulate user input with missing amount
    expense_data = {
        'date': '2024-05-25',
        'category': 'Groceries',
```

code1.py > ...

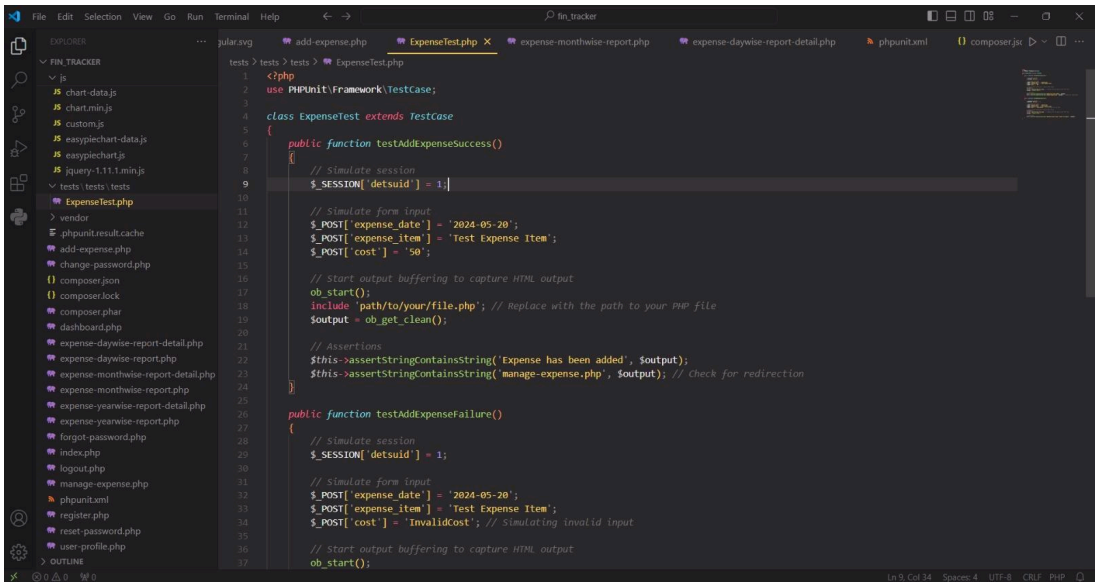
```python
def test_missing_expense_amount():
    # Simulate user input with missing amount
    expense_data = {
        'date': '2024-05-25',
        'category': 'Groceries',
        'description': 'Weekly grocery shopping',
        'payment_method': 'Credit Card'
    }

    # Call the expense entry function with incomplete expense data
    expense_entry_status = enter_expense(expense_data)

    # Assert that expense entry fails due to missing amount
    assert expense_entry_status == 'missing_amount_error'
    print("Missing expense amount test passed!")

# Run the test cases
if __name__ == "__main__":
    test_valid_registration()
    test_invalid_email_format()
    test_valid_expense_entry()
    test_missing_expense_amount()
```

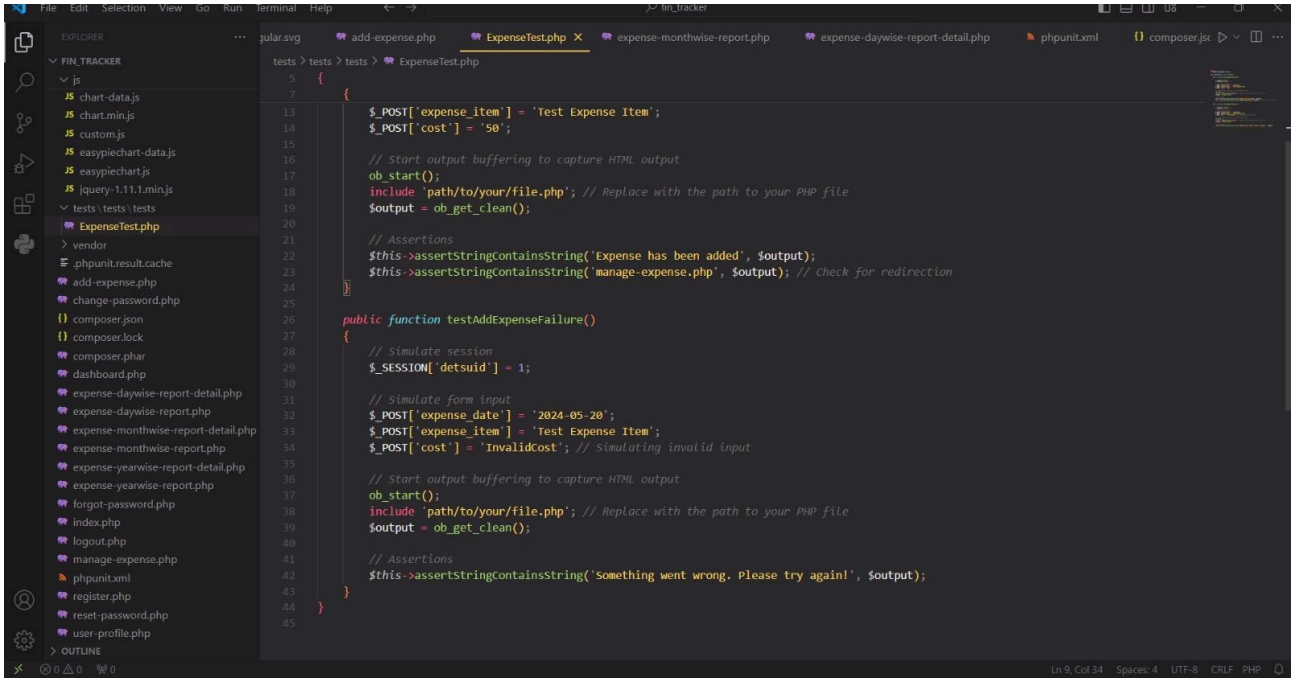
Picture 11: Software testing code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console

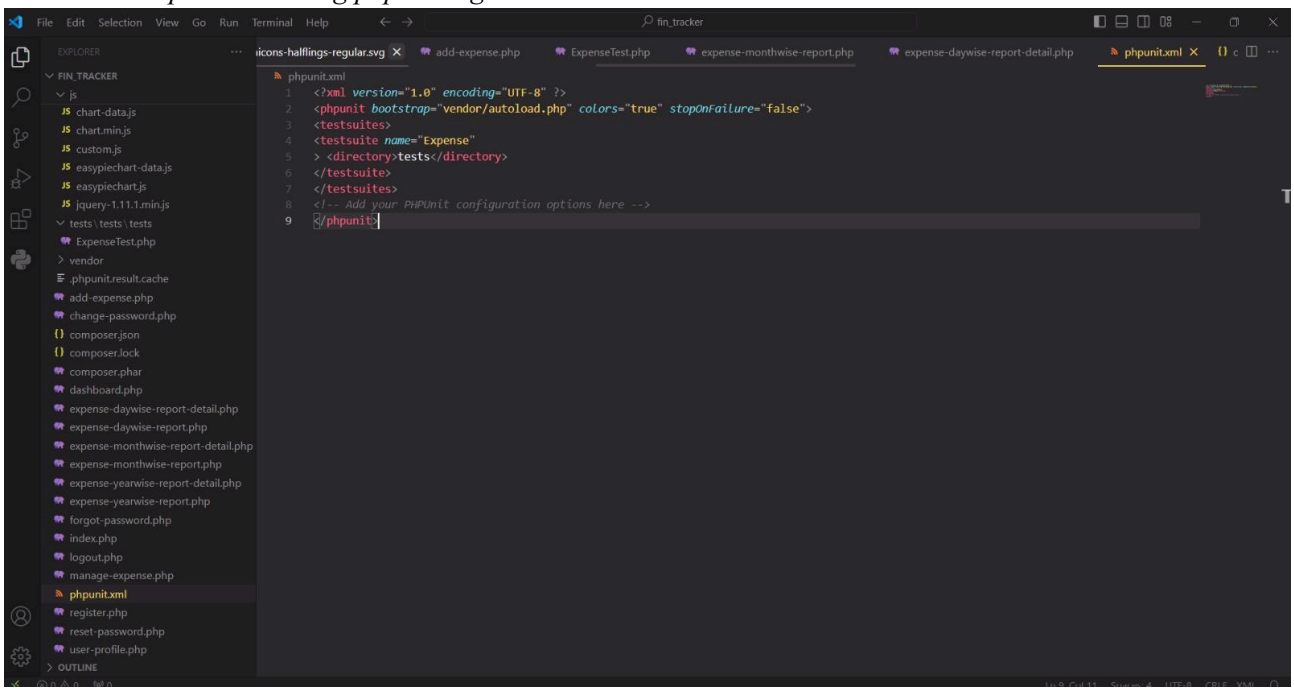
```
PS C:\Users\User\Desktop\swe>  & 'c:\Program Files\Python312\python.exe' 'c:\Users\User\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '62838' '--' 'C:\Users\User\Desktop\swe\code1.py'
PS C:\Users\User\Desktop\swe> ^C
PS C:\Users\User\Desktop\swe>
PS C:\Users\User\Desktop\swe>  c:; cd 'c:\Users\User\Desktop\swe'; & 'c:\Program Files\Python312\python.exe' 'c:\Users\User\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '63002' '--' 'c:\Users\User\Desktop\swe\code1.py'
Valid registration test passed!
Invalid email format test passed!
Valid expense entry test passed!
Missing expense amount test passed!
PS C:\Users\User\Desktop\swe>
```

Picture 12: Terminal of the successful code runned in VS Code

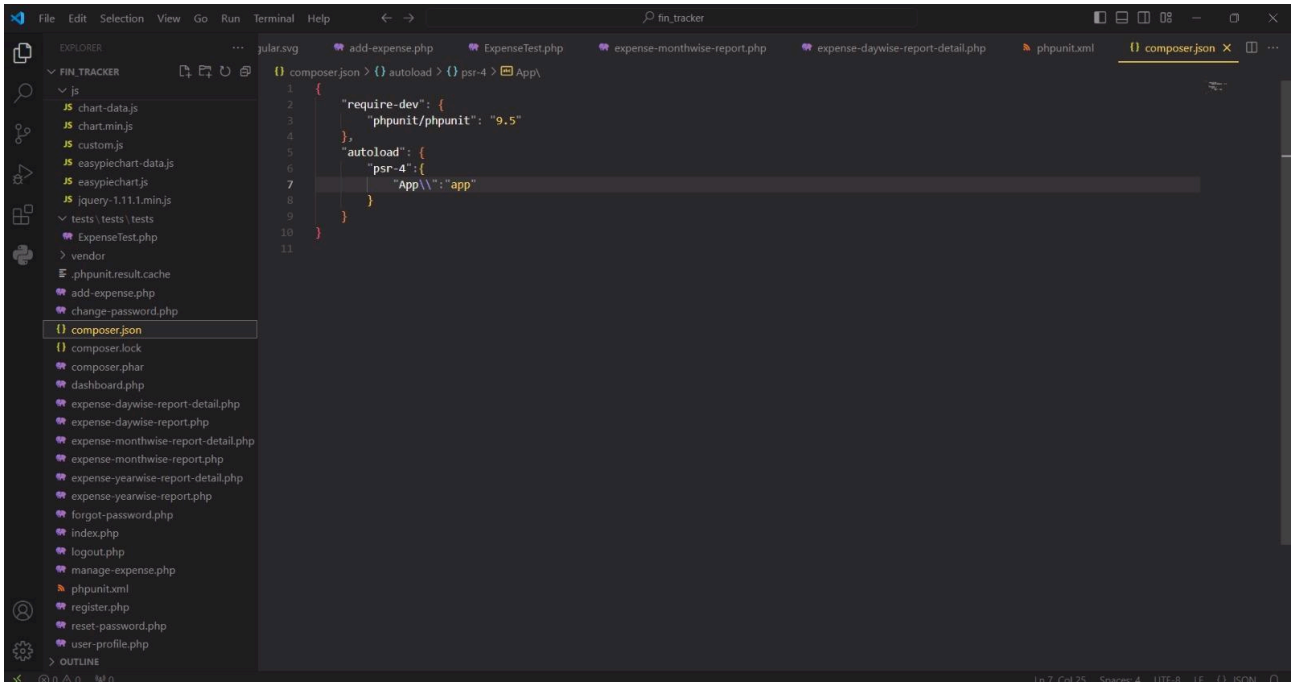




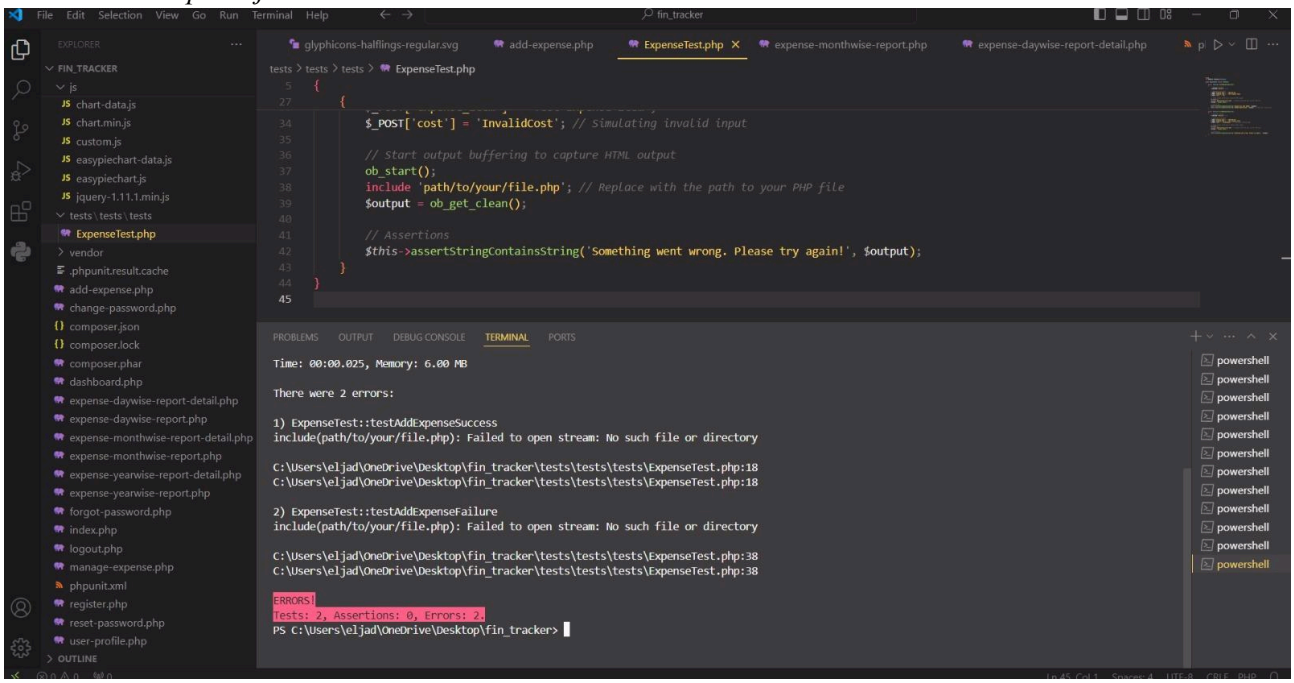
Picture 13: Expense Tracking php testing code



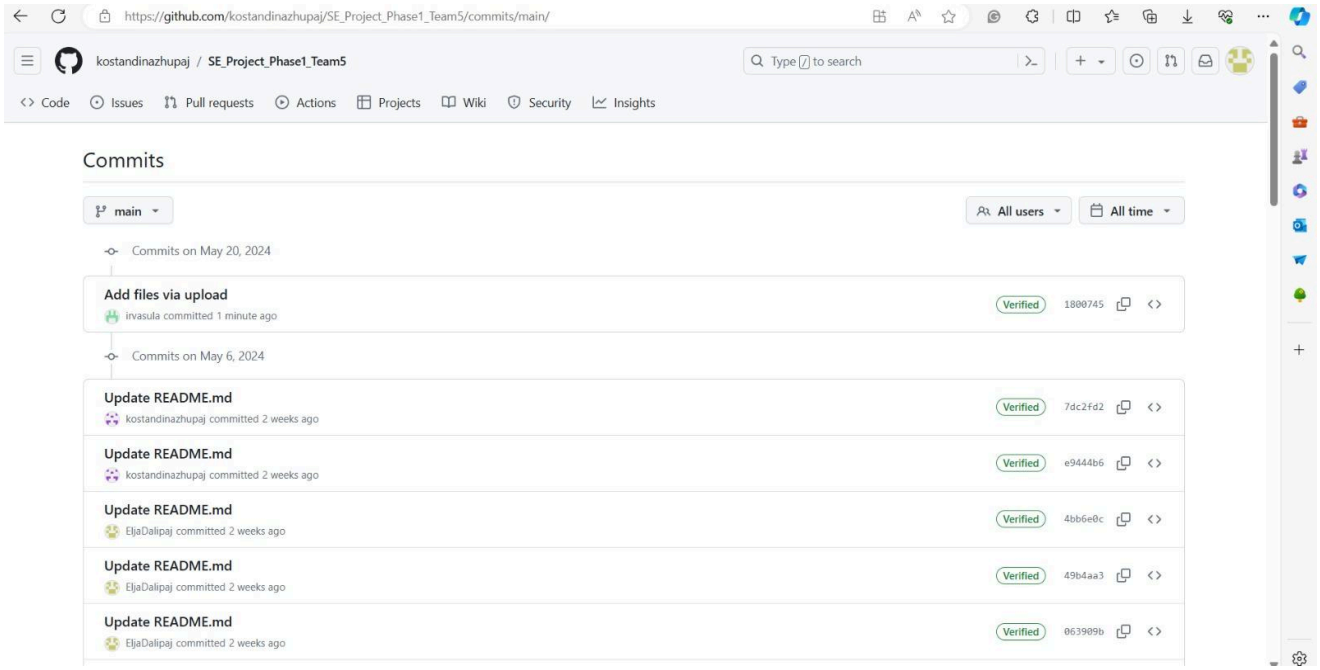
Picture 14: phpunit.xml

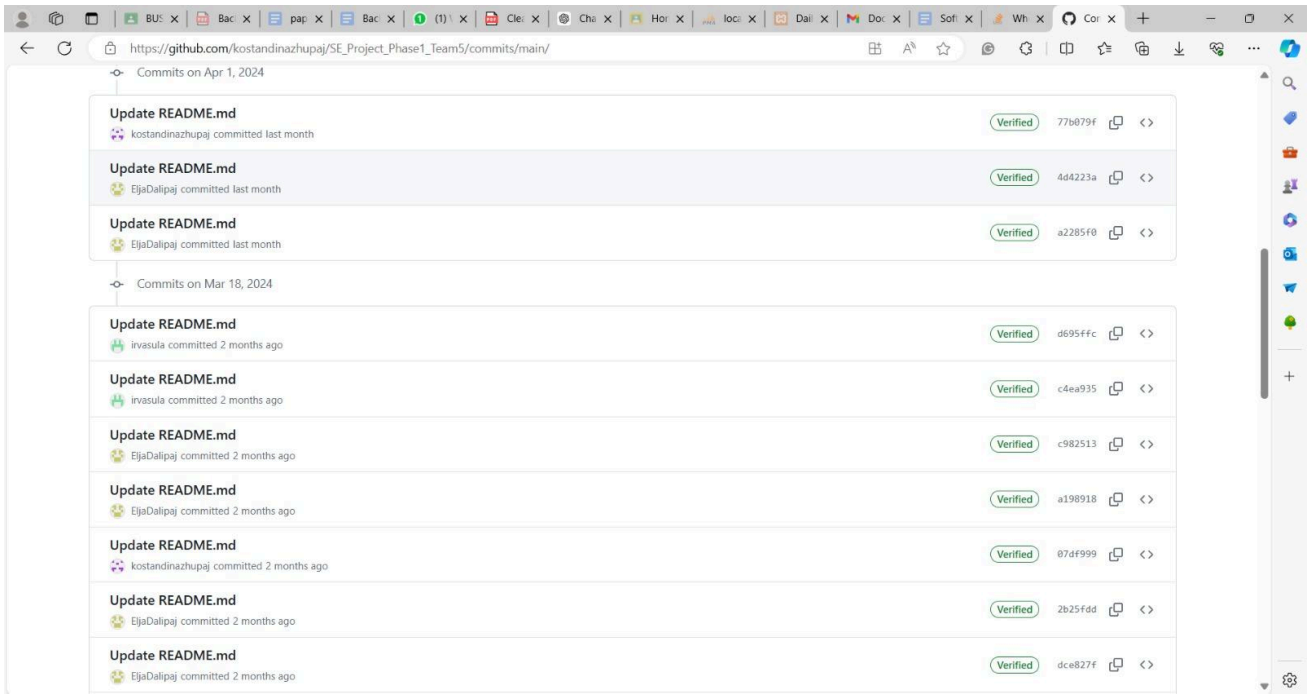


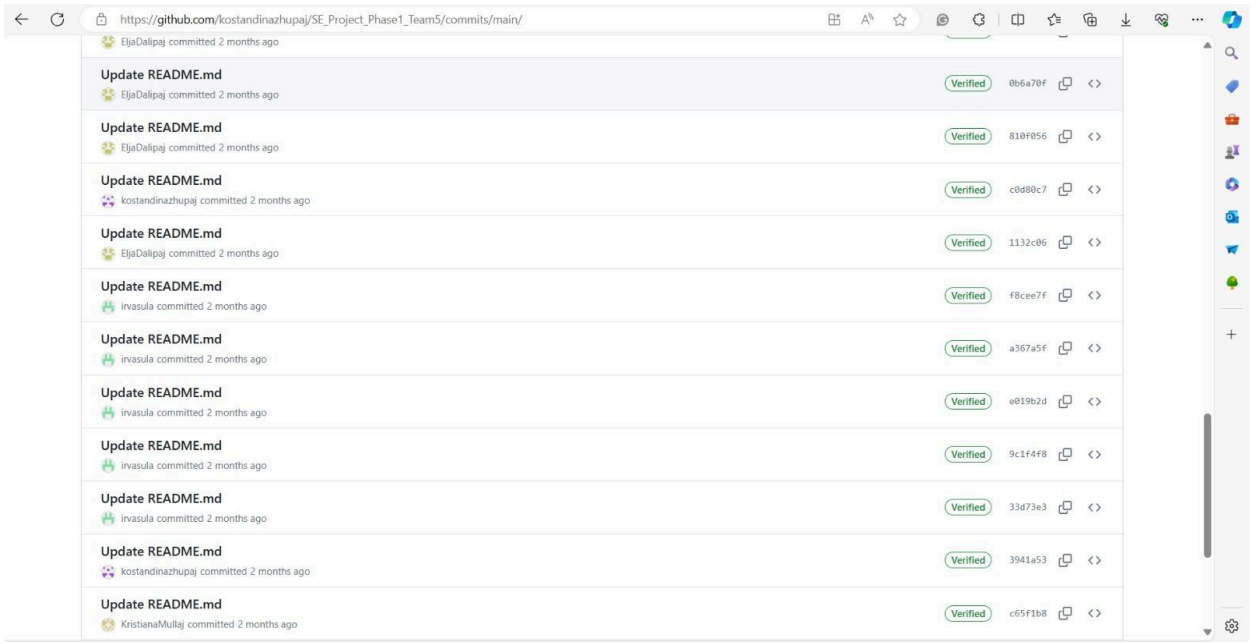
Picture 15: composer.json



Picture 16: Terminal of passing successfully the testing code


https://github.com/kostandinazhupaj/SE_Project_Phase1_Team5/commits/main/
kostandinazhupaj / SE_Project_Phase1_Team5
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Commits
main
All users
All time
Commits on May 20, 2024
Add files via upload
irvasula committed 1 minute ago
Verified
1800745
Commits on May 6, 2024
Update README.md
kostandinazhupaj committed 2 weeks ago
Verified
7dc2fd2
Update README.md
kostandinazhupaj committed 2 weeks ago
Verified
e9444b6
Update README.md
EljaDalipaj committed 2 weeks ago
Verified
4bb6e0c
Update README.md
EljaDalipaj committed 2 weeks ago
Verified
49b4aa3
Update README.md
EljaDalipaj committed 2 weeks ago
Verified
063909b


https://github.com/kostandinazhupaj/SE_Project_Phase1_Team5/commits/main/
Commits on Apr 1, 2024
Update README.md
kostandinazhupaj committed last month
Verified
77b079f
Update README.md
EljaDalipaj committed last month
Verified
4d4223a
Update README.md
EljaDalipaj committed last month
Verified
a2285f0
Commits on Mar 18, 2024
Update README.md
irvasula committed 2 months ago
Verified
d695ffc
Update README.md
irvasula committed 2 months ago
Verified
c4ea935
Update README.md
EljaDalipaj committed 2 months ago
Verified
c982513
Update README.md
EljaDalipaj committed 2 months ago
Verified
a198918
Update README.md
kostandinazhupaj committed 2 months ago
Verified
07df999
Update README.md
EljaDalipaj committed 2 months ago
Verified
2b25fdd
Update README.md
EljaDalipaj committed 2 months ago
Verified
dce827f



Picture 17: Git hub history commitment