EXPENSE TRACKER

Version 1

Functionality:

* Add Expenses
* View Expenses
* Add income
* View Income
* Edit Expenses and Income

Process:

Model:

What does ---- def \_\_str\_\_(self): return f"{self.get\_type\_display()} - {self.category}: ${self.amount}" ---- code do?

The `\_\_str\_\_` method in Django models is a special method that defines how the object is represented as a string. This is particularly useful when you’re inspecting objects in the Django admin or when debugging.

Breaking Down the `\_\_str\_\_` Method:

```python

def \_\_str\_\_(self):

return f"{self.get\_type\_display()} - {self.category}: ${self.amount}"

```

- `\_\_str\_\_(self)`: This is the method that gets called when you convert the object to a string, like when you print it or view it in the Django admin interface.

- `self.get\_type\_display()`:

- `get\_type\_display()` is a method automatically provided by Django for fields that use `choices`.

- It returns the human-readable name associated with the current value of the `type` field. For example, if `type` is `'income'`, `self.get\_type\_display()` would return `"Income"`.

- `self.category`: This accesses the `category` field of the model, which might be something like `"Salary"` or `"Food"`.

- `self.amount`: This accesses the `amount` field of the model, representing the transaction amount, like `100.00`.

- Return Value:

- The `return` statement combines these elements into a formatted string that gives a readable summary of the transaction.

- For instance, if the transaction is an income of $100.00 in the "Salary" category, this method would return `"Income - Salary: $100.00"`.

The f”” is an f-string.

Why It’s Useful:

- Readable Output: When you print or log an instance of the `Transaction` model, you’ll see a clear and informative string representation instead of something generic like `<Transaction object at 0x...>`.

- Django Admin: In the Django admin interface, the `\_\_str\_\_` method determines what text appears in lists and drop-downs, making it easier to identify objects.

This method is particularly helpful in making your objects more understandable when you work with them in your application.

Problem faced: Wasn’t able to do migrations and the error was that the app was not installed. FIX: Add the app name in settings.py in the Django project directory.

Views:

Templates:

Urls:

Other Features:

* User Authentication
  + Done on 19th August
  + Created user specific transactions to be seen by them only
* Expense and Income Categories
* Bootstrap 5 table used for interface: <https://bbbootstrap.com/snippets/bootstrap-5-table-search-and-checkboxes-10209122>
* SCSS Dashboard page Template: https://codepen.io/havardob/pen/ExvwGBrMonthly Summary/Report
* Homepage : https://codepen.io/rafaelavlucas/pen/QWwXKON
* Graphs and charts
* Recurring Transactions
* Search and Filter
* Budget Management
* Export to CSV/Excel
* Email Notifications
* Mobile Responsiveness
* Multi Currency Support
* Expense Splitting
* Tags for Transactions
* Transaction Attachment
* Dark Mode
* API Integration

To effectively document your Expense Tracker project, following a structured approach is crucial. Here’s a suggested document structure using Agile methodology principles:

1. Project Overview

- Project Name: Expense Tracker

- Objective: Develop a user-friendly application to track and manage personal expenses.

- Stakeholders: List the main stakeholders, including users, project owners, and team members.

- Scope: Define what will be included in the project (e.g., tracking transactions, generating reports, etc.) and what is out of scope.

2. Agile Methodology Overview

- Methodology Chosen: Agile (Scrum or Kanban)

- Reason for Choice: Explain why Agile is suitable for this project (flexibility, iterative development, user-centric approach).

- Development Approach: Brief description of how Agile will be applied, including sprint cycles, daily stand-ups, and retrospectives.

3. User Stories

- Template: "As a [type of user], I want to [action] so that [benefit]."

- Examples:

- As a user, I want to add a transaction so that I can track my expenses.

- As a user, I want to categorize my expenses so that I can analyze my spending patterns.

- As a user, I want to generate monthly reports so that I can see my financial progress.

4. Product Backlog

- List of Features:

- Transaction Management (Add, Edit, Delete transactions)

- Budget Management

- Report Generation (Daily, Weekly, Monthly)

- User Authentication

- Data Export (CSV, PDF)

- Prioritization: Rank features based on their importance to the user and business value.

5. Sprint Planning

- Sprint Goals: Define what will be achieved in each sprint.

- Sprint Backlog: Select user stories from the product backlog for the sprint.

- Estimated Effort: Estimate the time and resources required for each task.

6. Architecture and Design

- System Architecture: Diagram and description of the architecture (e.g., MVC framework, database design).

- Data Models: Define the data structures, including tables for transactions, users, categories, etc.

- User Interface Design: Wireframes or mockups for key screens (e.g., transaction entry, dashboard).

7. Development Process

- Coding Standards: Define the coding conventions and best practices to be followed.

- Version Control: Outline the version control system (e.g., Git) and branching strategy.

- Continuous Integration/Deployment: Describe the CI/CD pipeline, including tools used for automated testing and deployment.

8. Testing

- Test Plan: Outline the overall testing strategy.

- Unit Testing: Define the scope of unit tests and responsible team members.

- Integration Testing: Plan for testing how different parts of the system work together.

- User Acceptance Testing (UAT): Describe how the final product will be tested by users.

9. Documentation

- User Documentation: Guides and manuals for end-users.

- Developer Documentation: Technical documentation for developers, including API documentation.

- Change Log: Document all the changes, updates, and fixes made during development.

10. Deployment and Maintenance

- Deployment Strategy: Steps for deploying the application to production.

- Post-Deployment Support: Plan for addressing any issues after deployment, including bug tracking and updates.

- Maintenance Plan: Long-term strategy for maintaining the software, including regular updates and feature enhancements.

11. Project Retrospective

- Lessons Learned: Document the successes and challenges faced during the project.

- Process Improvement: Suggestions for improving the process in future projects.

12. Appendices

- Glossary: Define any technical terms or acronyms used in the documentation.

- References: List any external resources or references used.

13. Team and Contact Information

- Team Members: List of all team members with their roles.

- Contact Information: Contact details for project inquiries.

Tools to Consider:

- Asana: For managing tasks, user stories, and sprints.

- GitHub/GitLab: For version control and CI/CD pipelines.

- Jira/Confluence: Alternative to Asana, with powerful features for Agile management.

- Figma: For UI/UX design and wireframes.

- Swagger: For API documentation.

This structured documentation will help ensure that your project is well-organized, and that all team members and stakeholders have a clear understanding of the project’s progress, goals, and deliverables

Table of Contents

[Project Overview 9](#_Toc176287196)

[Objectives 9](#_Toc176287197)

[Scope 9](#_Toc176287198)

[Tools Used 9](#_Toc176287199)

[Other Applications References 9](#_Toc176287200)

[Software methodology 9](#_Toc176287201)

[Agile Methodology 9](#_Toc176287202)

[User Stories 9](#_Toc176287203)

[Product Backlog 9](#_Toc176287204)

[Architecture and Design 9](#_Toc176287205)

[System Architecture 9](#_Toc176287206)

[Data Models 9](#_Toc176287207)

[User Interface Design 9](#_Toc176287208)

[Development Process 9](#_Toc176287209)

[Coding Standards 9](#_Toc176287210)

[Version Control 9](#_Toc176287211)

[CI/CD Pipeline 9](#_Toc176287212)

[Testing 10](#_Toc176287213)

[Test Plan 10](#_Toc176287214)

[Integration Testing 10](#_Toc176287215)

[User Acceptance Testing 10](#_Toc176287216)

[User Documentation 10](#_Toc176287217)

[Developer Documentation 10](#_Toc176287218)

[Deployment 10](#_Toc176287219)

[Deployment Plan 10](#_Toc176287220)

[Deployment Strategy 10](#_Toc176287221)

[Post-Deployment Support 10](#_Toc176287222)

[Maintenance 10](#_Toc176287223)

[Project Retrospective 10](#_Toc176287224)

[Challenges 10](#_Toc176287225)

[Process Improvement 10](#_Toc176287226)

[Glossary 10](#_Toc176287227)

[References 10](#_Toc176287228)

# Project Overview

# Objectives

# Scope

# Tools Used

Visual Studio Code – For Coding and development environment

Git and GitHub – Version Control

# Other Applications References

## Global Top Money Management Applications

1. Mint
2. YNAB (You Need a Budget)
3. PocketGuard
4. GoodBudget
5. Personal Capital
6. Mvelopes
7. Wally
8. Emma (UK)
9. Money dashboard (UK)
10. Snoop (UK)
11. Cleo (UK)
12. Plum(UK)

## Key Features of the Applications

* Mint
  + Automatic categorization of transactions
  + Credit score monitoring
  + Personalized budgeting tips
* YNAB (You Need a Budget)
  + Detailed budgeting tools
  + Real-time synchronization across devices
  + Educational workshops and guides
* PocketGuard
  + Automatic expense categorization
  + Bill tracking
  + Insights into recurring payments
* Goodbudget
  + Syncs across multiple devices
  + Joint budgeting
  + In-depth reports
* Personal Capital
  + Comprehensive financial tracking
  + Retirement planning tools
  + Investment management
* Mvelopes
  + Real-time budgeting
  + Debt reduction planning
  + Financial coaching
* Wally
  + Expense tracking by photo
  + Bill reminders
  + Multi-currency support
* Emma (UK)
  + Subscription management
  + Budgeting tools
  + Transaction analysis
* Money Dashboard (UK)
  + Spending analysis
  + Budgeting
  + Financial goal setting
* Snoop (UK)
  + Money-saving tips
  + Transaction analysis
  + Bill tracking
* Cleo (UK)
  + AI-powered financial insights
  + Savings challenges
  + Daily budgeting reminders
* Plum (UK)
  + Automated savings
  + Investment options
  + Bill switching service

# Features of this Application

## Basic Features

1. Basic Income and Expense Tracking:

- Users can input and categorize expenses and income. - Done

- Date, category, amount, and description fields. -Done

2. Real-Time Balance Overview:

- Instant calculation of balance, total income, and total expenses. -Done

- Clear visual indicators like charts or progress bars.

3. Budgeting:

- Users can set monthly/weekly budgets for different categories.

- Notify users when they are approaching their budget limits.

4. Automated Reports:

- Simple reports (bar charts, pie charts) showing expense categories and trends.

- Downloadable in PDF or CSV format.

5. User Authentication (Registration/Login):

- Secure user registration with email/password.

- User login, password recovery, and profile management.

6. Search and Filter Transactions:

- Allow users to search and filter expenses based on dates, categories, or amounts.

7. Responsive Design:

- A clean and mobile-friendly interface that works seamlessly across devices. -Done

## Upcoming Features

1. Recurring Transactions:

- Automate recurring expenses (e.g., rent, subscriptions) or income.

- Notify users of upcoming recurring transactions.

2. Expense Categories Customization:

- Users can create, rename, and delete custom categories for tracking.

3. Data Backup and Restore:

- Option to backup data to cloud services or export to a file.

- Import transactions or settings from previous backups.

4. Currency Conversion:

- Allow users to set a preferred currency and convert amounts for international users.

- Integration with exchange rate APIs for real-time currency conversion.

5. Advanced Reporting and Insights:

- Compare spending over time (e.g., month-to-month, year-to-year).

- Identify top spending categories and trends.

6. Multi-Language Support:

- Offer multi-language support for broader user adoption.

## Upcoming Features (Advanced)

1. Machine Learning for Predictive Analytics:

- Use machine learning algorithms to predict future expenses and income trends.

- Insights like "You may exceed your budget by X next month."

2. AI-Powered Expense Categorization:

- Automatically categorize transactions based on user patterns.

- Suggest budget adjustments based on spending behavior.

3. Collaborative Expense Sharing:

- Enable users to share budgets or financial goals with others (e.g., family or roommates).

- Real-time updates and notifications for shared expenses.

4. Multi-Account Management:

- Users can link multiple bank accounts or track separate budgets (personal and business).

- Aggregation of transactions from different accounts.

5. Custom Financial Goals & Tracking:

- Set specific goals (e.g., saving for a trip) and track progress over time.

- Notify users of milestones or goal completion.

6. Security Features:

- Two-factor authentication (2FA).

- Data encryption at rest and in transit (secure HTTPS and encrypted databases).

- Automatic logout and session management for enhanced security.

7. Offline Mode:

- Ability to use the app without an internet connection.

- Sync data once the connection is restored.

8. Integration with Banking APIs:

- Fetch real-time bank transactions via APIs (like Plaid).

- Automatic categorization and syncing of bank transactions.

9. Personalized Financial Recommendations:

- Provide tailored financial advice based on user spending habits.

- Suggest saving opportunities or better budgeting strategies.

10. Expense Split and Group Payments:

- Users can split expenses with others and track who has paid their share.

- Sync with contacts to easily manage shared transactions.

## Scalability & Performance Optimizations

1. Cloud-Hosted Architecture:

- Host on scalable cloud platforms (like AWS, Google Cloud).

- Microservices or serverless architecture for different components.

2. CDN Integration for Fast Load Times:

- Use Content Delivery Networks (CDN) to ensure fast page load times globally.

3. CI/CD Pipeline:

- Continuous Integration and Continuous Deployment for smooth, automated updates.

4. Unit and Integration Testing:

- Extensive automated tests to ensure app reliability across devices and use cases.

5. Logging and Monitoring:

- Integrated error reporting (like Sentry) and usage monitoring (Google Analytics) for proactive bug fixes and feature improvements.

By covering this wide range of features, from basic to advanced, you’ll be able to showcase your project as a highly professional and well-rounded application suitable for both end users and businesses.

# Software methodology

# Agile Methodology

# User Stories

# Product Backlog

# Architecture and Design

## System Architecture

## Data Models

## User Interface Design

# Development Process

## Coding Standards

## Version Control

## CI/CD Pipeline

# Testing

## Test Plan

## Integration Testing

## User Acceptance Testing

# User Documentation

# Developer Documentation

# Deployment

## Deployment Plan

## Deployment Strategy

## Post-Deployment Support

# Maintenance

# Project Retrospective

## Challenges

## Process Improvement

# Glossary

# References