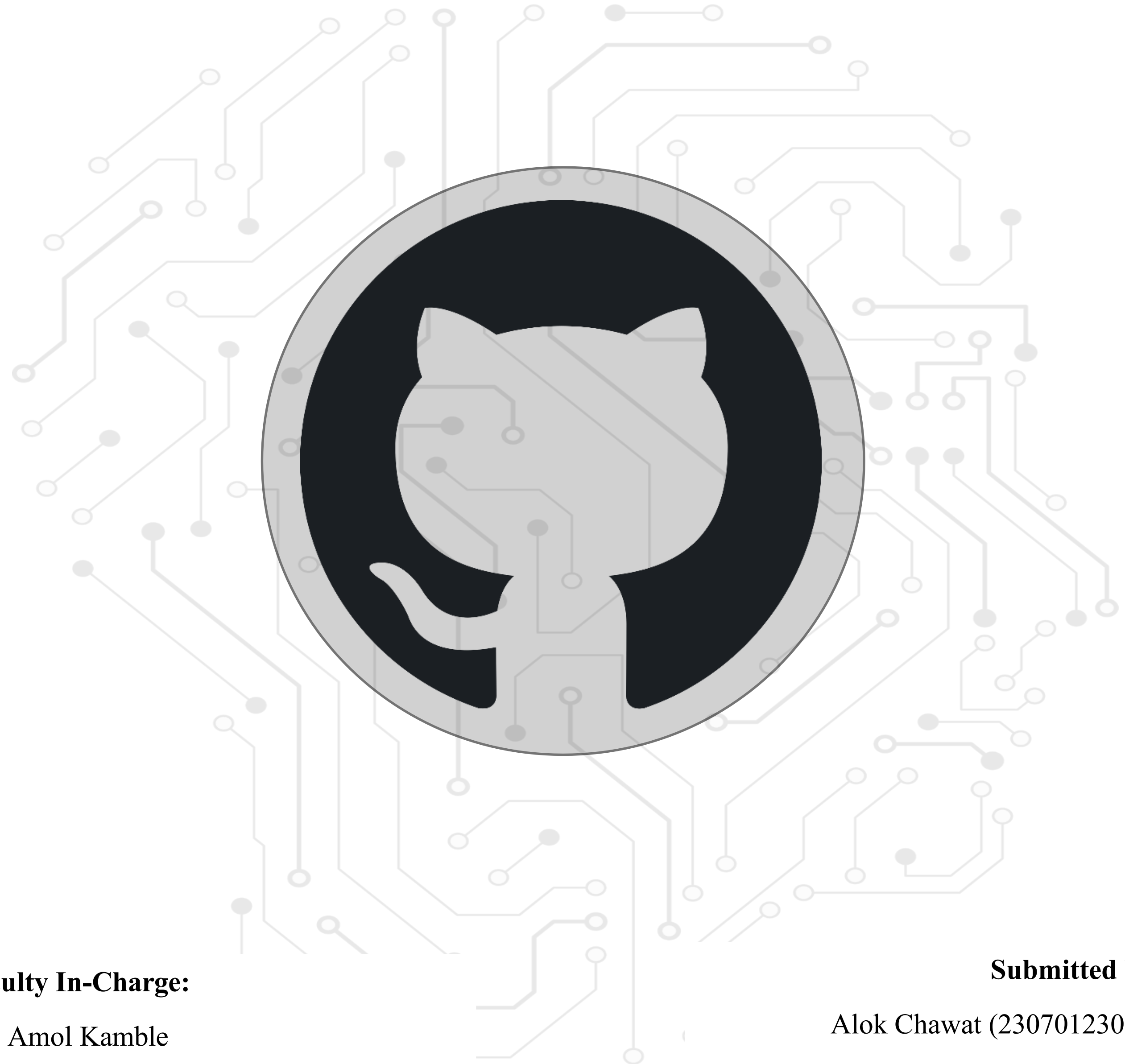
SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Constituent of Symbiosis International (Deemed University), Pune

Open-Source Technologies

Continuous Assessment-03



Faculty In-Charge:

Dr. Amol Kamble

Submitted by:

Alok Chawat (23070123016)

Aman Prajapati (23070123017)

Siddhant Bhosale
(23070123038)

Chakravarthula Prabhat
(23070123040)

Chirag Tekwani
(23070123042)

Introduction:

The Expense Tracker project was developed as part of the Open-Source Technologies course to apply open-source development workflows using GitHub.
The goal was to build a functional personal finance tracking web application that enables users to manage income and expenses efficiently. It integrates frontend, backend, and database elements while emphasizing collaborative software development practices such as version control, issue tracking, and project management on GitHub.

Project Overview:

Project Title: Expense Tracker
Objective: To design and implement a web-based application that allows users to record, categorize, and visualize financial transactions, supporting better financial decision-making through real-time insights.
Key Goals:

- Create and manage an open-source project on GitHub.
- Implement CRUD operations using Flask and SQLite.
- Develop a responsive frontend using HTML, CSS, and JavaScript.
- Apply collaborative debugging, issue creation, and enhancement practices.

Requirements:

1. Linux OS
2. Bash shell
3. Utilities: ping, curl, ip, iwconfig, nslookup
4. Optional (for detailed DNS checks): dig (dnsutils package)
5. GitHub

Features Implemented:

Feature	**Description**
Add / Edit / Delete Transactions	Perform CRUD operations on income or expenses.
Categorize Income & Expenses	Group transactions under custom categories for detailed insights.
Real-Time Summary	Display total income, expense, and remaining balance dynamically.
Pagination	Show 5 records per page for smoother navigation.
Category-wise Chart	Visualize expense distribution using Chart.js.
Input Validation	Ensure correct and sanitized user input.
SQLite Integration	Maintain persistent data storage with an embedded database.

Tech Stack:

Component	Technology Used
Frontend	HTML, CSS, Vanilla JavaScript
Backend	Python (Flask Framework)
Database	SQLite
Visualization	Chart.js
API Communication	Fetch API + Flask CORS
Version Control	Git & GitHub

Project Structure:

```
Expense_Tracker/
│
├── app.py            # Flask backend with REST APIs
├── index.html        # Frontend user interface
├── finance.db        # SQLite database file (auto-generated)
├── README.md             # Project documentation
└── requirements.txt    # Python dependencies
```

GitHub Repository and Issue Management:

The project was maintained using open-source best practices:

- Branch management for feature updates and fixes
- Issue tracking for bugs and enhancements
- Collaborative documentation of problems and resolutions
-

#	Issue Title	Type	Status	Reported By
1	Title tag missing closing bracket causes browser tab name issue	Bug	Closed	Alok Chawat
2	Pagination offset miscalculation in /list route	Bug	Closed	Alok Chawat
3	Summary route returns zero income due to typo	Bug	Closed	Aman Prajapati

#	Issue Title	Type	Status	Reported By
4	IndexError in dict_from_row() due to wrong index access	Bug	Closed	Chakravarthula Prabhat
5	Amount casting error in /add route	Bug	Closed	Chirag Tekwani
6	Chart re-renders overlap due to missing destruction of previous instance	Bug	Closed	Siddhant Bhosale
7	Incorrect Database Mentioned in README	Documentation	Closed	Aditya Sthawarmath
8	Typo in “Flask CROS” instead of “Flask CORS”	Documentation	Closed	Aditya Sthawarmath

All issues were collaboratively tracked and resolved on GitHub.

Innovation / Novelty / Contribution

The Expense Tracker project introduces several innovative aspects that go beyond a simple CRUD web app. Key highlights include:

- Full-Stack Integration: Seamlessly connects the frontend (HTML/CSS/JS) with a Flask backend and an SQLite database for persistent storage.
- Dynamic Visualization: Uses Chart.js to generate real-time visual analytics that update dynamically based on transaction data.
- Pagination and Filtering: Implements client-side pagination for smooth navigation through transaction records.
- Collaborative Development: Built entirely through GitHub, emphasizing issue management, version control, and peer collaboration.
- Error Handling and Validation: Includes extensive input validation to prevent errors, ensuring secure and consistent data storage.
- Scalable Design: The structure allows easy transition to cloud databases or frameworks like Django or React for future scalability.

This project combines teamwork, open-source methodologies, and web development fundamentals to deliver a functional, extensible, and maintainable application.

Student Individual Contributions:

Name	PRN	Role / Contribution
Alok Chawat	23070123016	Led the frontend development and handled UI design, pagination, and bug fixes related to title tag and route issues.
Aman Prajapati	23070123017	Developed backend API routes for summary calculation and database interaction; resolved issues in income aggregation.
Siddhant Bhosale	23070123038	Worked on chart rendering, dynamic data updates using Chart.js, and resolved overlapping chart issues.
Chakravarthula Prabhat	23070123040	Debugged Python backend functions, particularly dict indexing errors; tested data flow between frontend and backend.
Chirag Tekwani	23070123042	Focused on form validation, typecasting, and error handling for user input in the add and edit routes.

Challenges Faced:

Challenges Faced (Elaborated)

1. **Frontend-Backend Integration Issues:**
   Synchronizing Flask backend responses with JavaScript Fetch API calls initially caused asynchronous data update delays. The team resolved this by refining route handling and implementing proper JSON response structures.
2. **Database Schema Management:**
   Designing a normalized database to efficiently handle transactions and categories was challenging. Early schema mismatches caused data redundancy, which was corrected by redesigning the database and implementing consistent foreign keys.
3. **Input Validation and Error Handling:**
   Preventing invalid or empty data entries required custom validation on both client and server sides. This ensured robust data integrity and user experience.
4. **Chart Rendering Overlaps:**
   Chart.js re-rendered overlapping graphs during updates. Siddhant implemented a solution by destroying the previous chart instance before drawing a new one, ensuring visual clarity.
5. **Merge Conflicts and Version Control:**
   With multiple contributors pushing commits, Git merge conflicts became frequent. Proper branch naming conventions and regular pull requests helped maintain version consistency.
6. **Pagination Logic Errors:**
   Off-by-one pagination logic caused records to skip or repeat. Debugging and implementing offset recalculations resolved this.
7. **Typographical and Documentation Errors:**

8. Minor documentation inconsistencies (e.g., “Flask CROS” vs “Flask CORS”) were identified and corrected to maintain technical accuracy.
9. **Cross-Browser Compatibility:**
   Ensuring the interface behaved consistently across browsers like Chrome and Firefox required CSS tweaks and testing.
10. **Team Coordination:**
    Managing parallel feature branches, resolving dependencies, and integrating code demanded disciplined GitHub collaboration and communication.

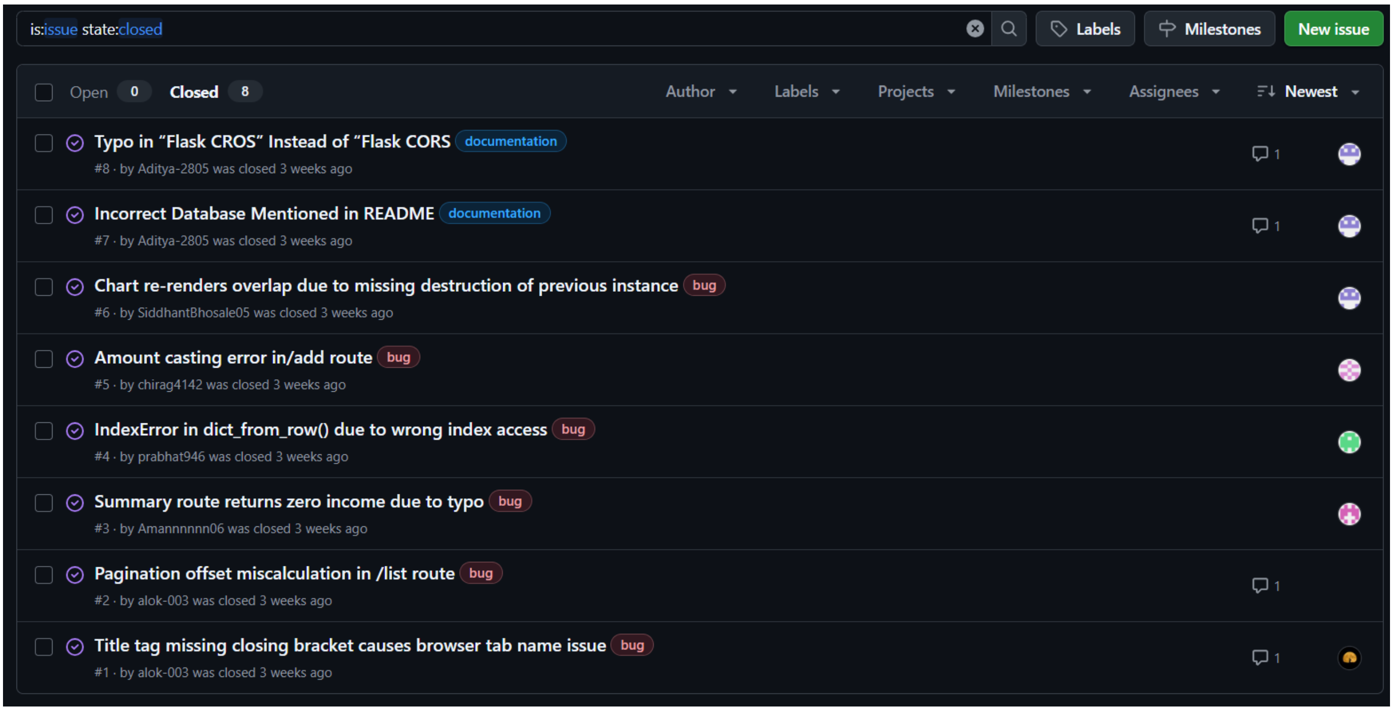
Learnings and Outcomes:

The team gained practical experience in:

- Developing a full-stack web application using open-source tools.
- Using GitHub for team collaboration, version control, and issue management.
- Designing REST APIs and establishing frontend-backend communication.
- Collaborative problem-solving and effective project management.

This project enhanced the team’s technical proficiency and teamwork capabilities in an open-source environment.

Screenshot:



□ Open 0 Closed 8

□ Typo in “Flask CROS” Instead of “Flask CORS documentation
#8 · by Aditya-2805 was closed 3 weeks ago · 1

□ Incorrect Database Mentioned in README documentation
#7 · by Aditya-2805 was closed 3 weeks ago · 1

□ Chart re-renders overlap due to missing destruction of previous instance bug
#6 · by SiddhantBhosale05 was closed 3 weeks ago

□ Amount casting error in/add route bug
#5 · by chirag4142 was closed 3 weeks ago

□ IndexError in dict_from_row() due to wrong index access bug
#4 · by prabhat946 was closed 3 weeks ago

□ Summary route returns zero income due to typo bug
#3 · by Amannnnnn06 was closed 3 weeks ago

□ Pagination offset miscalculation in /list route bug
#2 · by alok-003 was closed 3 weeks ago · 1

□ Title tag missing closing bracket causes browser tab name issue bug
#1 · by alok-003 was closed 3 weeks ago · 1

GitHub Repository Link:
https://github.com/alok-003/Expense_Tracker

How to Run the Project

Steps to Execute:

1. Clone the Repository
   Open your terminal and run:
   ```
   git clone [https://github.com/alok-003/Expense_Tracker]
   ```
2. Navigate to the Project Directory
   ```
   cd Expense_Tracker
   ```
3. Create a Virtual Environment (optional but recommended)
   ```
   python -m venv venv
   source venv/bin/activate      # For Linux/Mac
   venv\Scripts\activate         # For Windows
   ```
4. Install Required Dependencies
   ```
   pip install -r requirements.txt
   ```
5. Run the Application
   ```
   python app.py
   ```
6. Access the Web App
   Open your browser and go to:
   http://127.0.0.1:5000/
7. Usage
   - Add, edit, and delete transactions.
   - Categorize income and expenses.
   - View real-time summaries and visual charts.
   - Explore data pagination for easier navigation.

Conclusion:

The Expense Tracker successfully delivers an intuitive and interactive personal finance management tool. It effectively demonstrates open-source development methodologies and highlights the benefits of teamwork, code transparency, and iterative improvement.