

ExpenseFlow Project Documentation

Knight Foundation School of Computing and Information Sciences (KFSCIS) —
Florida International University

Instructor: Masoud Sadjadi | Version: 2025 Fall | License: MIT

1. Executive Summary

The problems that we sought to fix were ways to store financial data reliably, without the clutter and mess of doing it manually. We wanted the method we used to be simple and efficient, while being easy to read and use. People who would benefit from our web applications include students and the public who need help balancing their finances. This would include individuals who have problems identifying overspending patterns with disorganized spending records. Through proper use, this could even help with taxes and general expenses for more balanced spending habits throughout the month or lower financial stress due to difficulty saving. This would also target those who keep physical reminders of budgeting, such as tracking it in a notebook, which can lead to data loss and has no real management system outside of the user's memory.

Our solution to this problem is to use an application or website to keep track of daily expenses. For our main goals, we wanted the financial tool to be simple, beginner-friendly, but visually appealing and fast/responsive. Our result ended up being a web application that looks great, has an easy-to-understand GUI, and is incredibly responsive. Built using HTML primarily, ExpenseFlow is our solution for easy financial management. The web application looks modern and sleek, with no clutter to confuse the user, and it has quality of life built into itself with many features. Key results include highlighting various financial inputs, easy adding and removal of expenses, and an easy-to-use filtering system.

2. Problem & Requirements

Users / Primary Stakeholders

- Students and members of the public who want a simple and safe way to record and track daily expenses.
- Individuals who currently use physical methods of tracking (notebooks receipts) with the potential for data loss and what a lightweight web alternative.

Requirements

- Clean and easy to read GUI
- Easy to innovate coding with room for improvements

- Clear use of the application without explanation through use of colors or graphics
- Responsive UI
- Add a new expense with: date, description, category, and amount
- List expenses in a table or card view with ability to delete individual items.
- Persist data between sessions (client-side persistence)
- Fast page load and instant client-side filtering
- No data loss during normal usage (within the constraints of client-side storage)
- User data stored locally (not uploaded to external servers) unless user opts in to sync

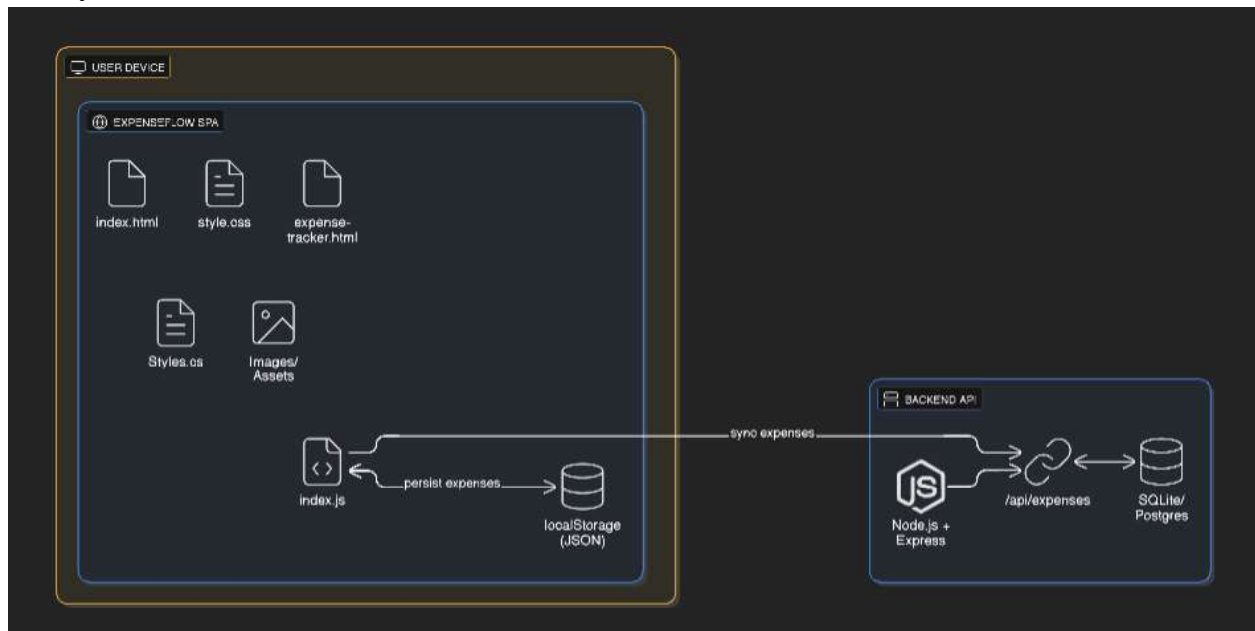
Success Criteria

- Typical user can add an expense and see updated totals within 10 seconds of first use.
- Users can identify the top two spending categories within a month.
- No major UI bugs reported in basic flows (add/delete/filter) in initial testing.

Risks

- No cross-browser or multidevice sync.
- No reliable backup if user error occurs.

3. System Overview & Architecture



Key Technologies, frameworks & services

- Frontend: HTML5, CSS3, JavaScript (vanilla)
- Styling: Custom CSS files (`css/style.css`, `css/Styles.css`)
- Persistence: Browser `localStorage` (JSON)
- Optional backend (recommended upgrade): Node.js + Express with a relational DB (SQLite/Postgres)

4. Implementation Notes

Software Used: HTML, CSS, JavaScript (used plotly for graphs)

Style.css:

used for styling website design and layout

expense-tracker.html:

Main site that will be used to showoff the expense tracking

index.js:

Scripts that help in tracking the expenses

- function addExpense: main function to add expenses to the site and updating the overall balances aswell as a deleteExpense that deletes them and updates

- update functions that update each different aspect of the site, like categories, chart, and total expenses (ex: updateExpensesList, updateCategoryList, updateTotal, updateChart)

click event to make the site filter with different functions by the month that was inputted and also update the totals of the site by what as inputted for that month

(ex:applyFilterBtn.addEventListener -> renderFilteredTransactions(filtered),
updateCategoryListFiltered(filtered), updateChartFiltered(filtered))

A clear filter click event to clear the filters and return back to default of viewing expenses-Used

Plotly graphs to help visualize the spending with color coding each different category

5. Testing & Evaluation

Reliability

Description

test3

Category

Shopping

Date

11/04/2025

Amount

123.542

Type

Expense

Add

Please enter a valid value. The two nearest valid values are 123.54 and 123.55.

Filter

--Select a month--

Filter

Clear Filter

Recent Transactions

test2

2025-12-03

Salary

+\$2321.00

Delete

test1

2025-12-03

Education

-\$1780.00

Delete

\$2469.54

Total Expenses

\$3111.20

Total Income

+\$641.66

Balance

Add Transaction

Description

Enter description

Category

--Select category--

Date

mm/dd/2025

Amount

0.00

Type

Expense

Add

Filter

--Select a month--

Filter

Clear Filter

Recent Transactions

test6

2025-02-12

Investment

-\$566.00

Delete

test5

2025-02-11

Salary

+\$554.80

Delete

test4

2025-11-19

Investment

+\$235.40

Delete

test3

2025-11-04

Shopping

-\$123.54

Delete

test2

2025-12-03

Salary

+\$2321.00

Delete

test1

Education

-\$1780.00

Delete

Categories Overview

Salary

+\$2875.80

Education

-\$1780.00

Investment

-\$330.60

Shopping

-\$123.54

Monthly Overview

Amount (\$)

2000

1000

0

-1000

-2000

Jan

Feb

Mar

Apr

May

Jun

Jul

Aug

Sep

Oct

Nov

Dec

Salary

Investment

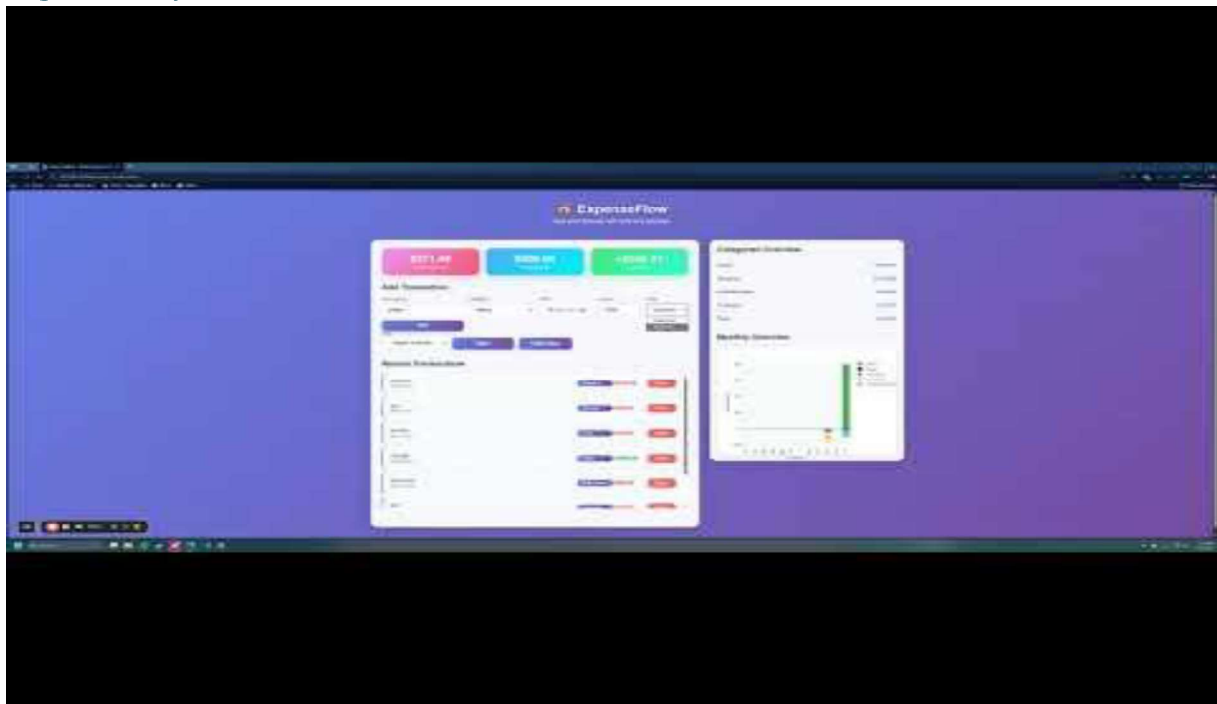
Shopping

Education

Investment



<https://www.youtube.com/embed/OtTnIcJ2miE?si=EUGMU-GidlalVIdD>



Live Demo for testing purposes.

6. Security, Privacy, Accessibility & Compliance

Security and Privacy

Currently data is stored locally in the browser, by design the app does not transmit the users data to remote servers.

Risks

- As data is saved locally it could be deleted due to user error, via clearing of the data, this could be mitigated with CSV exporting and importing backup functionality.
- Use HTTPS everywhere, JWT or session-based auth, server-side input validation, rate limiting.
- Encrypt backups at rest if storing sensitive data.
- Follow OWASP Top 10 protections (CSRF tokens, input validation, output encoding).

Privacy and Compliance

- Keep default behavior local-only.
- Provide clear privacy notice: no data leaves the browser unless user chooses to export or connect to third-party sync.
- If implementing analytics, make it optional and anonymized.
- Avoid collecting personally identifiable information unless necessary; if collecting, ensure consent and secure handling.
- Provide export and delete options so users can manage their data.

7. Deployment & Operations

- **Static deployment:** Copy CAPSTONE/ folder to a static host:
 - GitHub Pages (push to gh-pages or use repo root).
 - Netlify (drag-and-drop deploy or connect repo).
 - S3 static website with CloudFront.
- No server operations required for the current version.

Operational notes

- **Backup / Recovery:** Encourage users to export CSV periodically.
- **Monitoring:** If hosted publicly, monitor availability via standard uptime tools.

8. Limitations & Future Work

Limitations

- No multi-device sync.

- Potential formatting differences for currency across locales

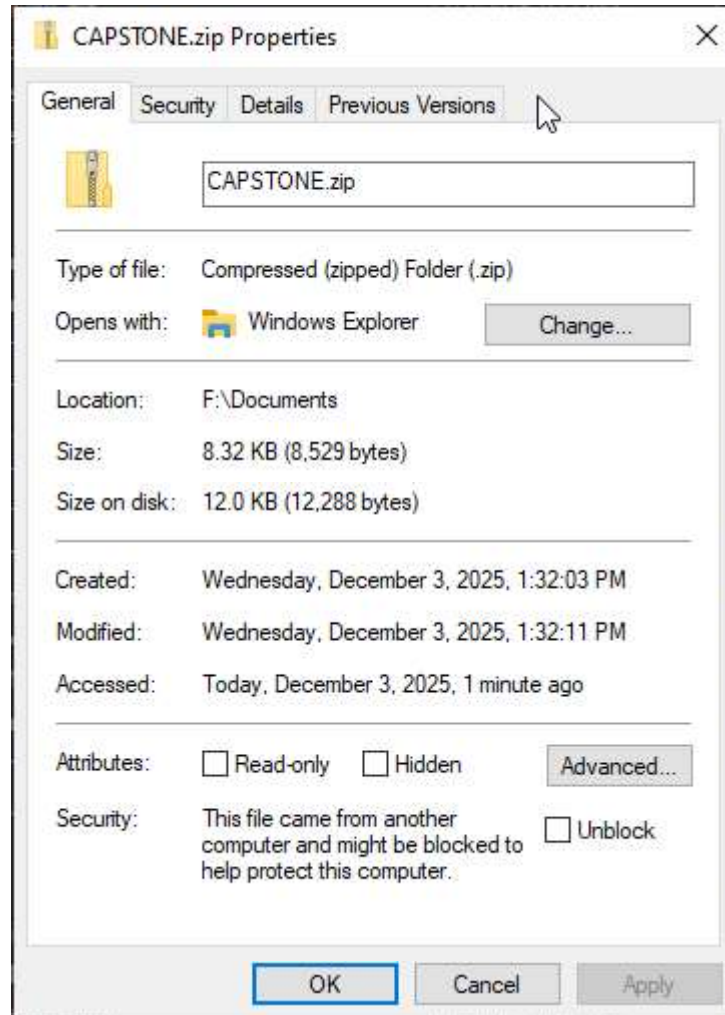
Future Work

- User login and cloud-based storage
- Pie charts, trend lines, and advanced analytics
- Export to PDF or Excel
- Mobile app version
- Fix readability on mobile version
- Cross sync between mobile and desktop version
- Monthly budgeting goals and reminder
- More filtering options (by category, spent/earned a certain amount)

Appendices (Evidence & How-To)

1. Installation Guide

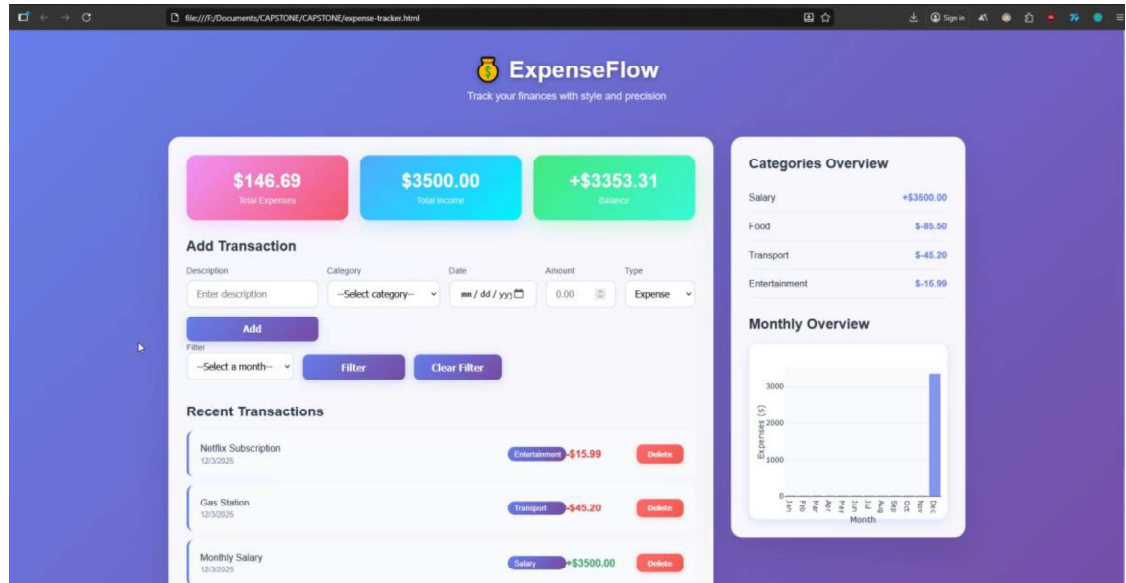
a. Download the web application



b. Extract the web application

Documents > CAPSTONE.zip > CAPSTONE						
Name	Type	Compressed size	Password ...	Size	Ratio	Date modified
css	File folder					9/24/2025 1:36 PM
Images	File folder					9/17/2025 2:48 PM
JavaScript	File folder					9/29/2025 12:10 PM
expense-tracker.html	Firefox HTML Document	2 KB	No	6 KB	77%	10/28/2025 2:21 PM
index.html	Firefox HTML Document	1 KB	No	3 KB	72%	10/1/2025 12:46 PM

c. Run expense-tracker.html



2. User Manual

- The web application should work without issue so the main steps are learning what each button does.

Add Transaction Form:

Description: Enter description

Category: --Select category--

Date: mm / dd / 2025

Amount: 0.00

Type: Expense

Filter:

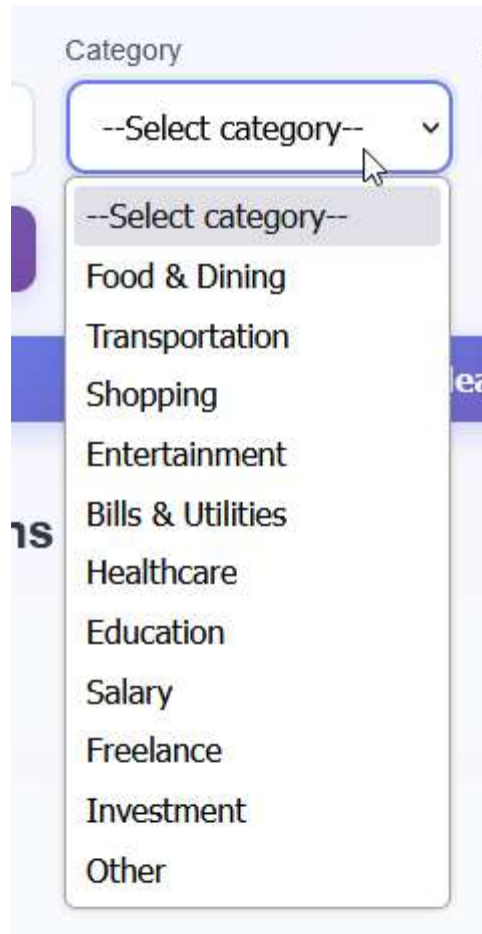
--Select a month--

Buttons: Filter, Clear Filter

Recent Transactions:

- Description is what the expense or income will be named as, with categories being prefilled for common expenses such as

Transportation, Entertainment and Food Etc;



Category

--Select category--

--Select category--

Food & Dining

Transportation

Shopping

Entertainment

Bills & Utilities

Healthcare

Education

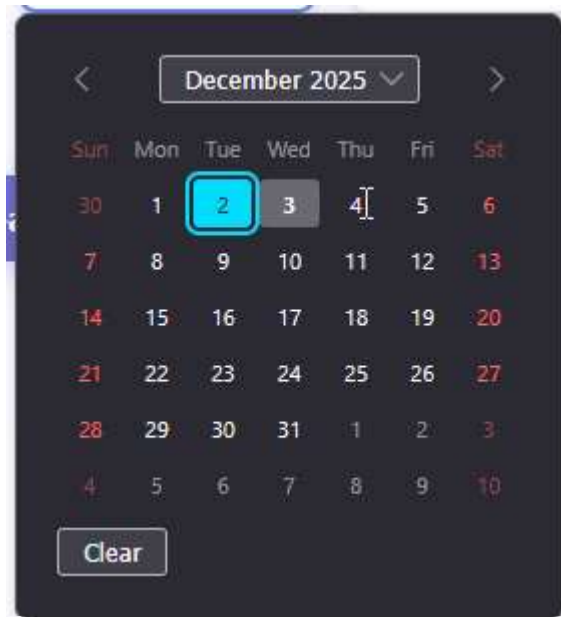
Salary

Freelance

Investment

Other

- c. Data is what the day the expense occurred, with safeguards that prevent a future date as to not confuse the system.



< December 2025 >

Sun Mon Tue Wed Thu Fri Sat

30 1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30 31 1 2 3

4 5 6 7 8 9 10

Clear

- d. After adding the expense there exist filters for easy sorting and you can even export the graph for easy safekeeping.



3. Poster

- a. https://drive.google.com/file/d/1l7zNIH7tdNipNJxc1_pXKG5VYoGXivfw/view?usp=sharing

4. Scrum Documentation

- a. https://drive.google.com/file/d/1l7zNIH7tdNipNJxc1_pXKG5VYoGXivfw/view?usp=sharing