Financial Tracker Web Application

Milestone 2.0 - Part D

IST 303 Software Development- Fall 2025

Team-Hoenn: Cesar · Shah · Ucheoma . Shawn . Sameer

Application Concept

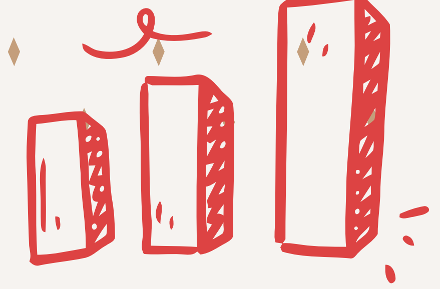
The Financial Tracker Web Application is a Flask-based web application that helps users track income, expenses, budgets, and savings over time.

It enables users to:

1. Record income and expense transactions
2. Categorize spending for better insight
3. Create and monitor budgets
4. Generate financial reports with securely stored data
5. secure data storage

Milestone 2.0: Key Features & Accomplishments

In Milestone 2.0, we delivered the core functionality of the Financial Tracker Web Application:



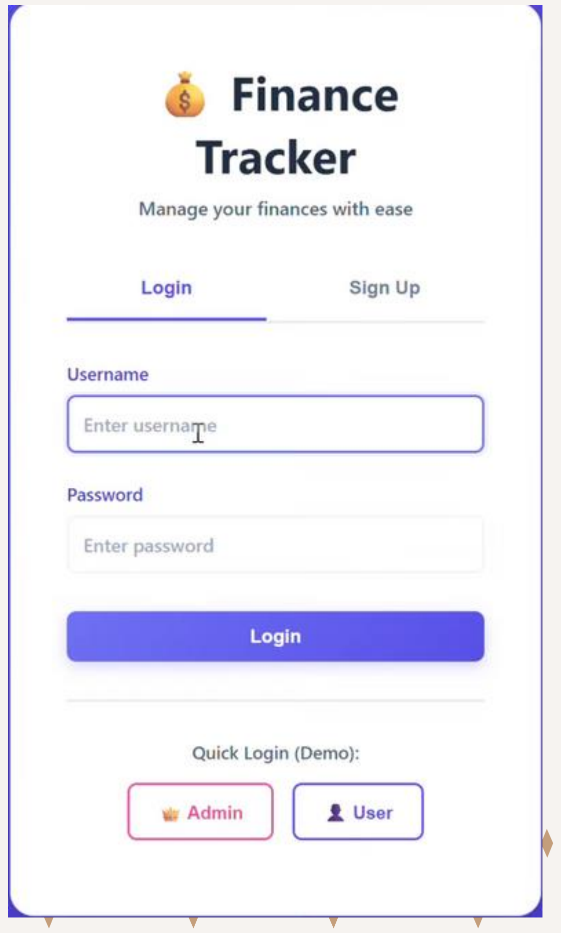
- User authentication and secure login system
- Income and expense tracking with categorization
- Budget creation and monitoring
- Financial reports and visualizations
- Summary views and charts help users understand trends in spending and savings

- Each user has a private account and protected data.
- Users can log transactions and tag them as income or different expense types
- Users define monthly budgets and compare them against actual spending.

Live Application Demo

What We'll Demonstrate

User login and navigation to the dashboard	Adding income and expense transactions	Viewing categorized transaction lists	Seeing budget vs actual summaries and financial reports


💰 Finance Tracker

Manage your finances with ease

Login | Sign Up

Username: Enter username

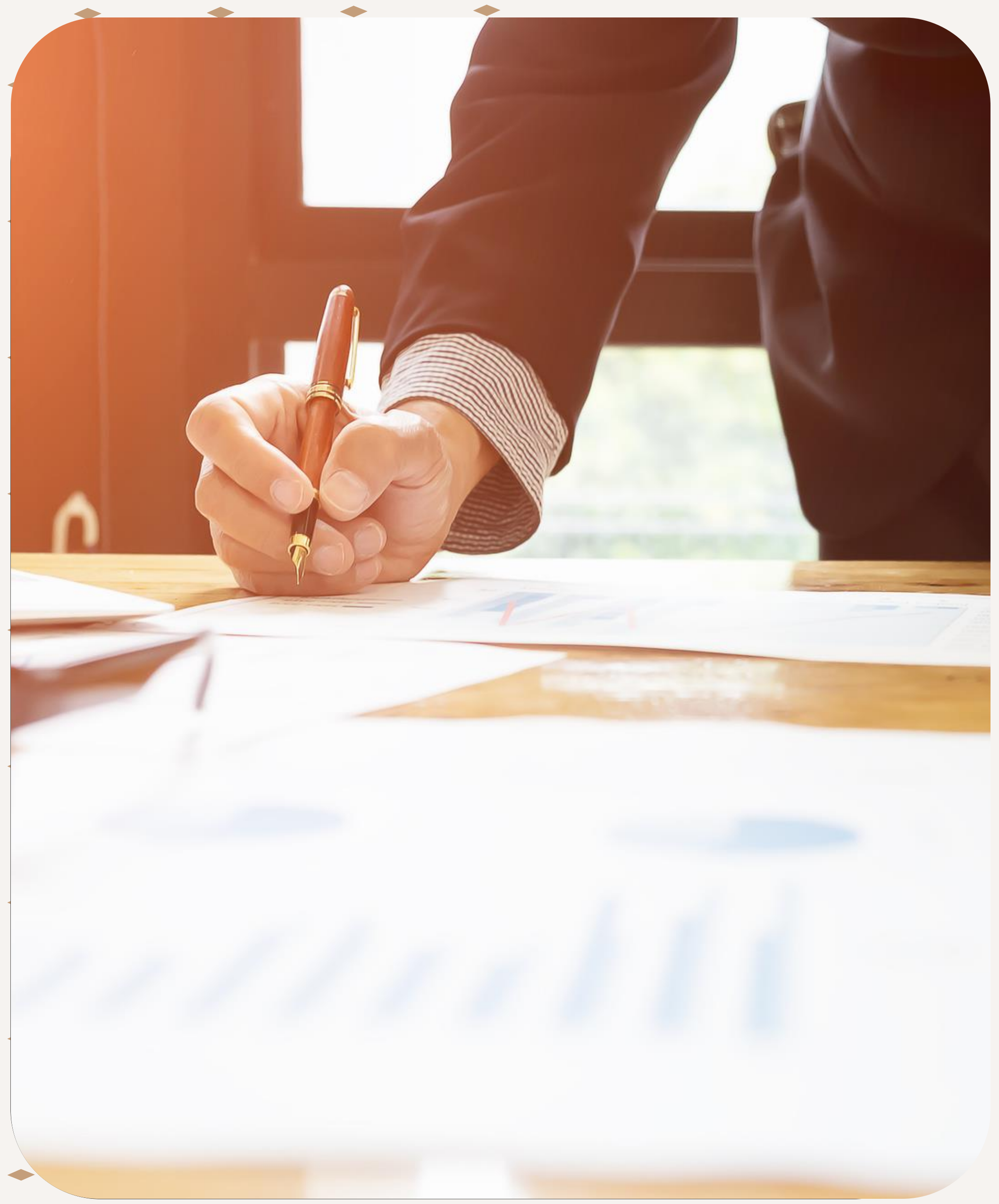
Password: Enter password

Login

Quick Login (Demo):

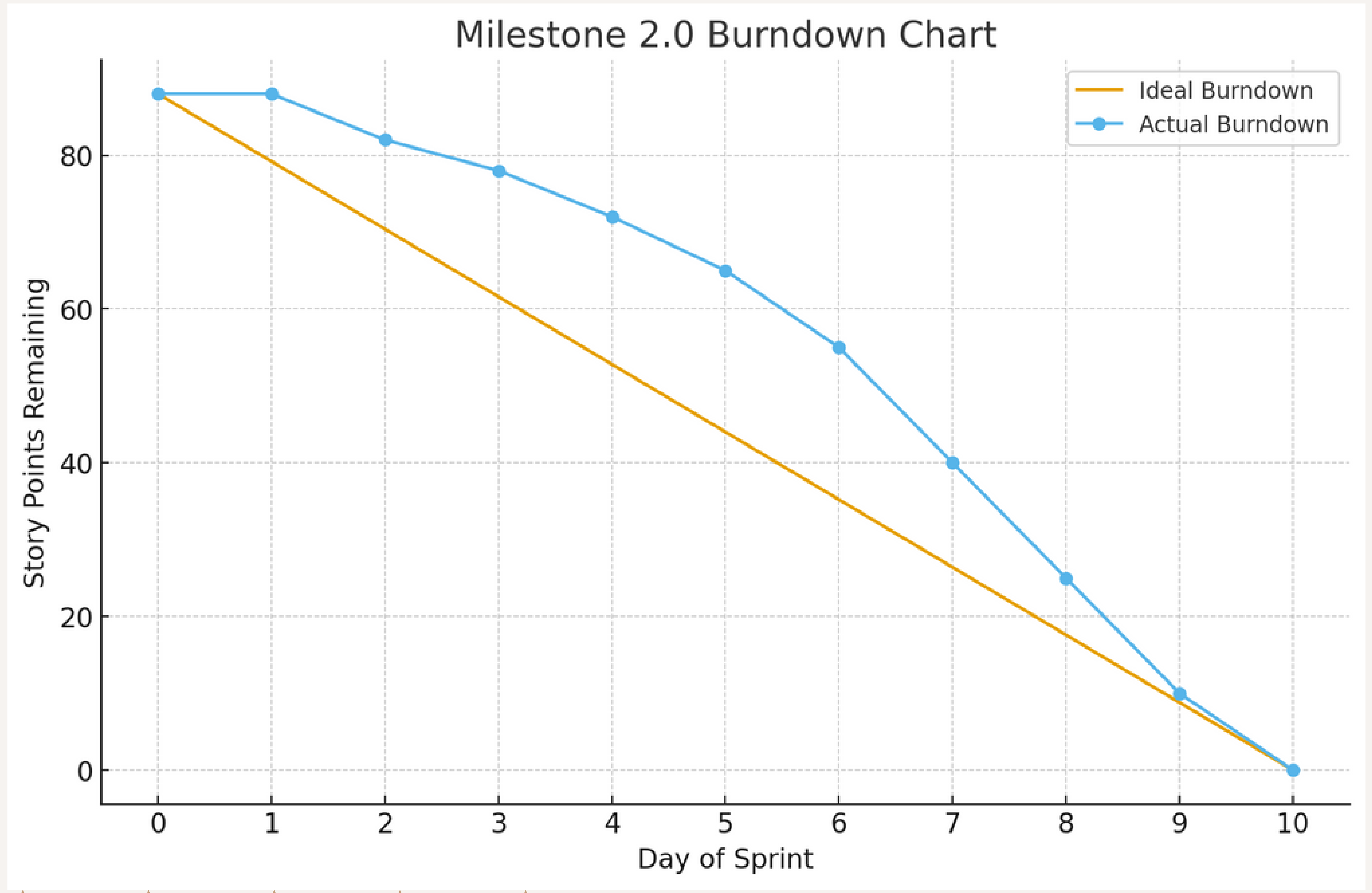
Admin | User

Agile Process & Sprint Methodology



- ✓ 2-week sprints with regular (daily or frequent) stand-ups
- ✓ User stories prioritized by business value
- ✓ Focused on core flows first (auth, tracking, budgets, reports)
- ✓ Sprint reviews and retrospectives
- ✓ Demo at end of sprint, plus feedback and improvement discussions
- ✓ Continuous integration with automated testing
- ✓ Code changes integrated via GitHub, tested before merge

Burndown Chart & Project Velocity



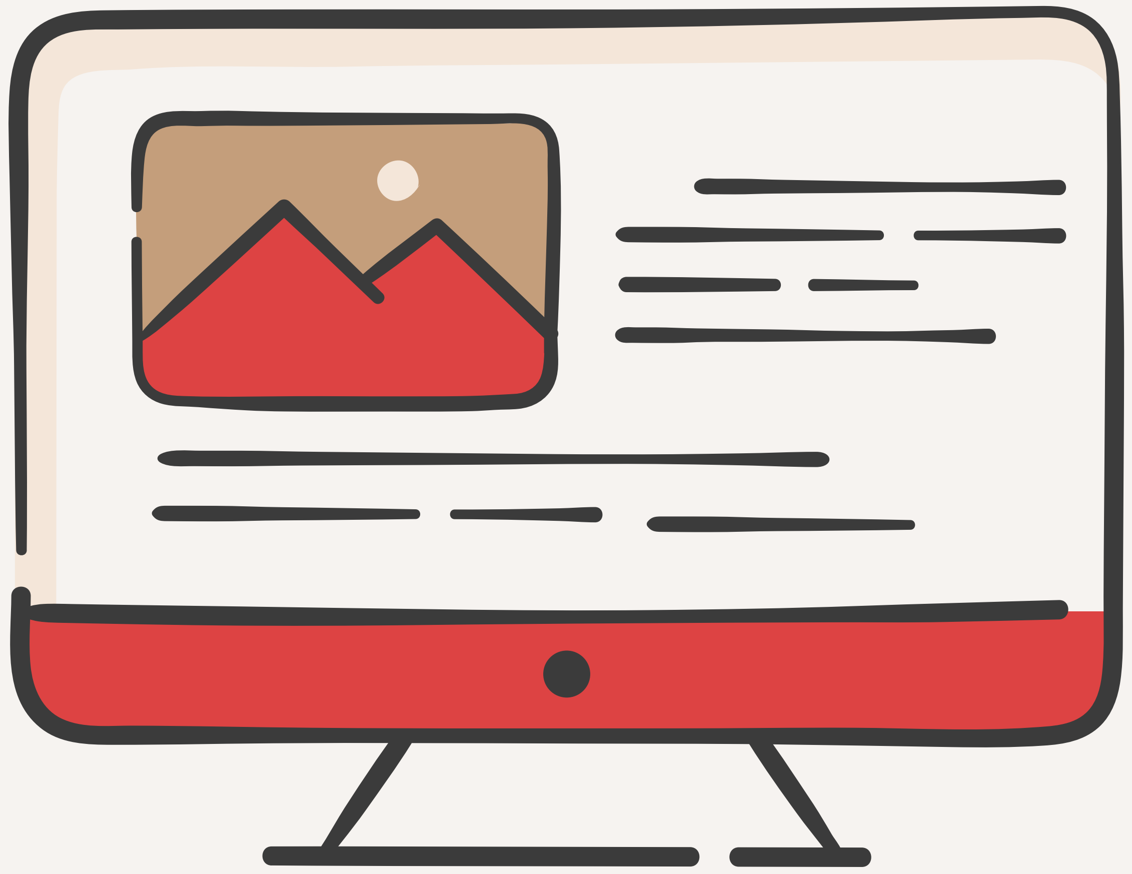
Initial velocity: 15 story points per sprint

Final velocity: 22 story points per sprint

Total story points completed: 88 points

The burndown chart shows how our remaining work decreased across the sprints, and how our team's velocity improved as the project progressed.

Testing Strategy & Code Coverage



Testing Approach:

Unit tests using the pytest framework

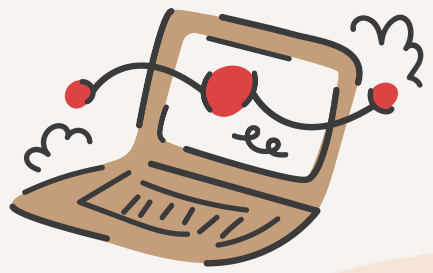
Tested core business logic and validation

Integration tests for key API endpoints and routes

Ensured end-to-end flows (login, add transaction, view reports) worked correctly

Tests run automatically on new commits or pull requests

Coverage Results:



92% test coverage achieved

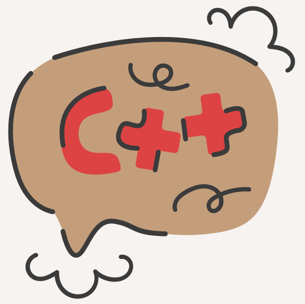
Most critical modules (auth, transactions, reports) are well covered

Remaining gaps focus on rare edge cases and error handling paths

Technical Stack & Architecture

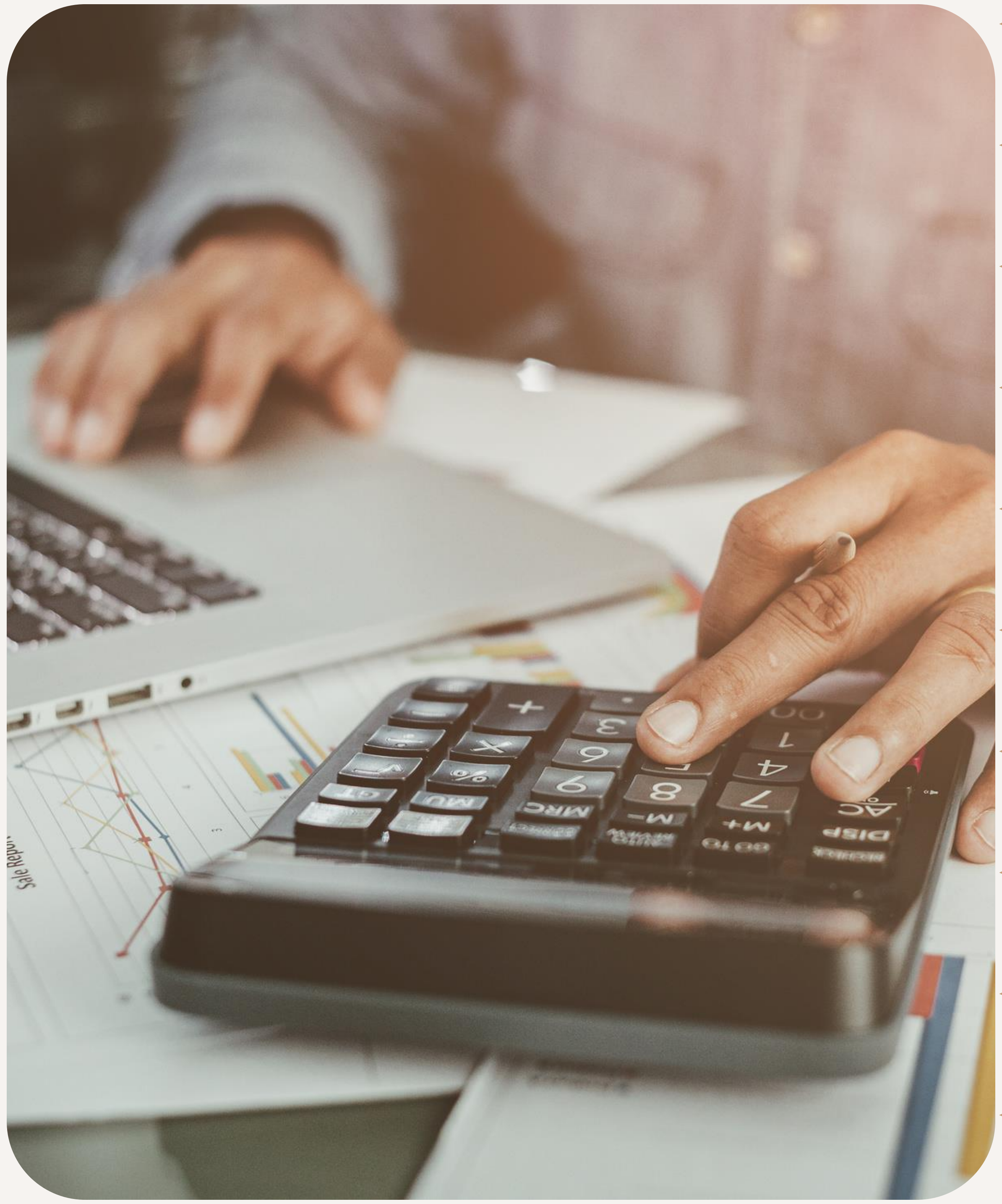
Body split into Backend / Frontend:

Backend
Python 3.12
Flask framework
SQLAlchemy ORM
PostgreSQL database



Frontend
Jinja2 templates for server-side rendering
HTML / CSS

The application follows a typical Flask architecture: routes → views/templates → database models, with SQLAlchemy handling persistence.



Milestone 3.0 Planned Features

For Milestone 3.0, we plan to

- Add user profiles and more robust role-based permissions
- Enhance dashboards with richer charts and visualizations
- Implement recurring transactions (rent, subscriptions, etc.)
- Add CSV export for income/expense data
- Increase automated test coverage from ≈ 92% → closer to 100% in critical modules

Long-term Roadmap & Enhancements

- Long-term, we envision the Financial Tracker evolving to:

  Provide smart alerts for overspending or missed savings targets

  Offer optional bank integration to import real transactions

  Be fully mobile-friendly with a responsive design

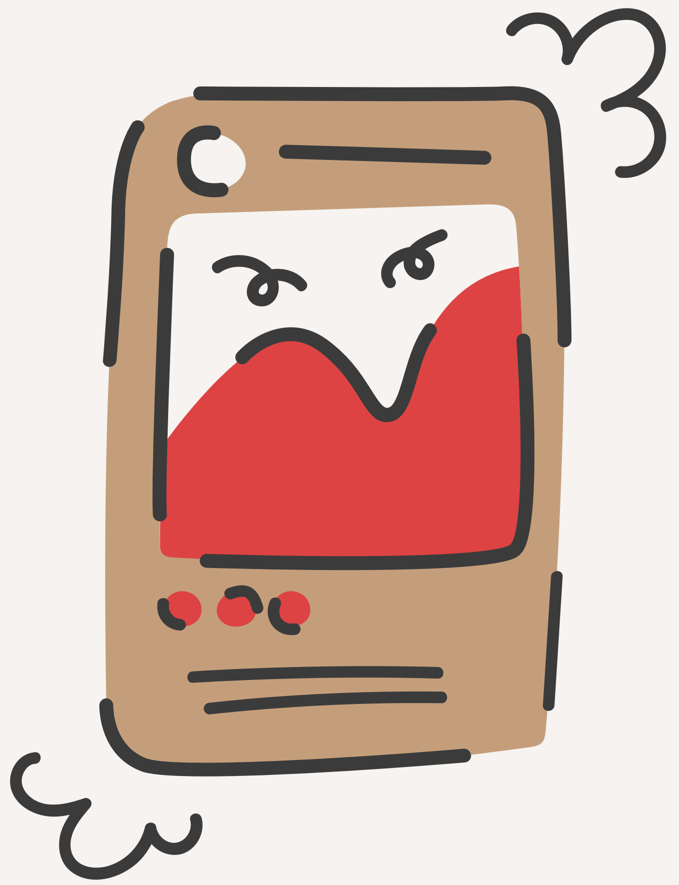
  Continuously improve code quality through refactoring + more tests

Key Lessons Learned

- ✓ Team Collaboration: Built trust through consistent delivery.
- ✓ Members worked independently while staying confident in progress.
- ✓ Hybrid Methodology: Mixed Scrum structure with agile flexibility.
- ✓ Used sprints, Planning Poker, backlog, retrospectives, testing and code reviews.
- ✓ Regular Meetings: Kept team aligned and provided time to sync code.
- ✓ Peer feedback helped us maintain consistent style, spot edge cases, and share knowledge.

Software Development Takeaways & Conclusion

- ✓ Theory vs. Practice: Implementing was harder than understanding in class.
- ✓ Early and frequent testing, along with coverage measurement, prevented late-stage bugs.
- ✓ Agile visibility (task board, burndown, stand-ups) helped the team adapt to reality instead of rigidly following an initial plan. Tools Require Learning: Git critical but needs dedicated time to master
- ✓ The Finance Tracker app development process provided invaluable agile, testing, and collaborative experience. Success required BOTH technical coding skills AND mastery of collaborative processes. Theory-to-practice gap larger than expected, prepared us for real work.



Thank You