




Smart Ledger

Team #1

By: Blake Childress, Andrei Cozma, Hunter Price, Tyler
Beichler, Emanuel Chavez, Jacob Leonard, Lillian Coar



Problem Definition

- People need a quick, easy, and free way to split the cost and keep track of expenses between people or groups of people.
- Current solutions are either not free, not user-friendly, or lack in features.
- The primary function of Smart Ledger is to split purchases into groups.
- Possible additions include
 - Receipt scanning
 - Direct integration with companies like Venmo and PayPal
 - Analytics for businesses

Background Information

- Current expense tracking solutions are either not free, not user-friendly, or lack in features.
 - No Paying from within the app (Splitwise)
 - Essential features behind paywall (Evenfy)
 - Not on all platforms, limiting who you can add (Splittr)
 - Limited to certain types of transactions (Tab)
- Venmo and CashApp
 - Have to manually calculate split
 - Keep track of purchases with another method

Requirements Specification

Customer Requirements	Description
1. Functionality	Core features such as dashboards that accurately track spendings, group support, and splitting expenses
2. Usability	Clear visual hierarchy for optimal user experience
3. Accessibility	Work seamlessly on major platforms and devices, support multiple currencies
4. Privacy	Securing user information and monetary transactions
5. Receipts	Scan or manually enter receipt information

Requirements Specification

Engineering Requirements	Description
1. Cloud Service	Cloud service that best supports frontend and backend needs while also being cost-efficient
2. Database	Store user and group information regarding spendings
3. Security	Implement exception handling and other security protocols
4. Artificial Intelligence Tools	Reports and analytics
5. Optical Character Recognition	Scan receipts

* All requirements are design variables (DV)

Requirements Specification

Features:

- Free to download and use
- Convenient expense splitting
- Support multiple currencies
- Scan receipts or enter manually
- Support creating and joining multiple groups
- Automatically allocate cost of purchase between members of the group
- Personal & group dashboard with reports with analytics & insights about spending habits
- Payment deadlines & notification reminders
- Easily accessible on all major platforms
- Simple and user-oriented user interface
- Accessible, intuitive, informative

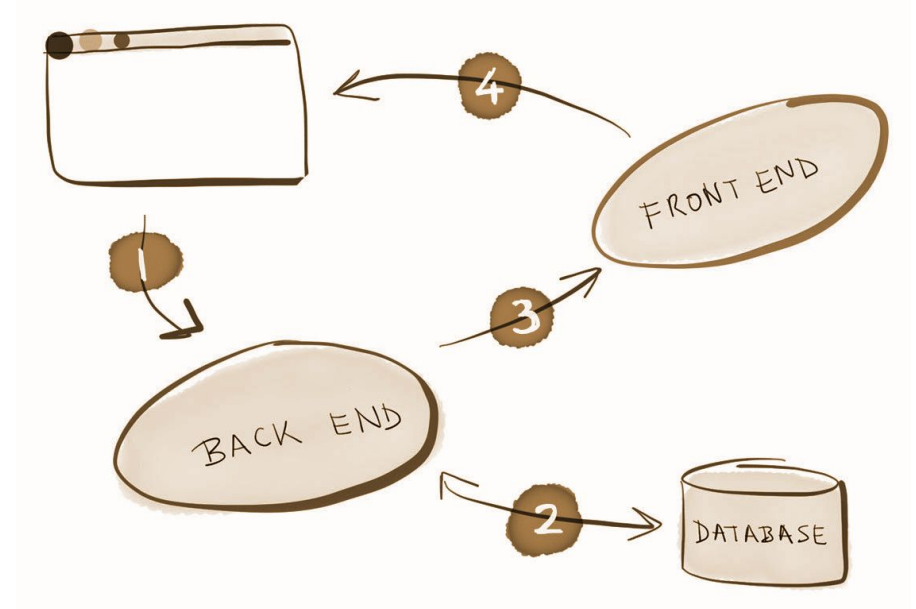
Technical Approach

Front End

- **Features**
 - Mobile friendly
 - Frameworks
 - Libraries

Back End

- **Use a database to store information**
 - Relational?
 - Non-relational?
- **Features:**
 - Secure
 - Reliable
 - Resilient



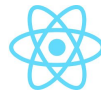
Design Decision Identification

- Front end
 - Functional & accessible design
- Back end
 - Fast response time & easy to maintain
- Database
 - Consistent data retrieval
- Hosting
 - Available 24/7 without breaking the bank
- All choices factor into scalability and security

Front End

Criteria:

- Fast & Easy Deployment
 - Learning Curve
 - Support
- Maintainable Development
- Performance
- Familiarity



Frameworks:

- Angular
- React
- Vue

Front End Weighted Decision Matrix

Framework	Learning Curve	Familiarity	Support	Performance	Structure	Score
Angular	1	2	2	1	3	9
React	2	3	3	2	2	12
Vue	3	1	1	3	1	9

Back End

Cloud Services:

- Amazon Web Services (AWS)
- Google Cloud Platform (GCP)
- Microsoft Azure

Criteria:

- Familiarity
- Documentation
- Pricing
- Relevance



Databases:

- MongoDB
- CouchDB
- Couchbase Server

Criteria:

- Familiarity
- CAP Principles
 - Consistency (C)
 - Availability (A)
 - Partition tolerance (P)

Back End Cont.

Criteria:

- Familiarity
- Performance
- Ease Of Use
- Flexibility
- Relevance

Backend:

- PHP
- Node.js express
- Flask



Back End Weighted Decision Matrix

Cloud Service	Familiarity	Documentation	Pricing	Job Relevance	Score
AWS	3	3	2	3	11
GCP	2	2	3	2	9
Azure	1	1	1	1	4

Database	Familiarity	Consistency	Availability	Partition Tolerance	Score
MongoDB	3	3	1	3	10
CouchDB	2	1	3	1	6
Couchbase Server	1	2	2	2	8

scale 1-3

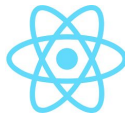
Back End Weighted Decision Matrix Cont.

Backend	Familiarity	Performance	Ease Of Use	Flexibility	Relevance	Score
PHP	1	1	1	1	1	5
Node.js w/ Express	2	3	3	2	3	13
Flask	3	2	2	3	2	12

Design Selection

- Scaling, flexibility, and familiarity led us to choose AWS over GCP and Azure
- React.js is widely used, has the most support, and higher performance
- MongoDB picked for consistency due to handling money
- Node.js with Express *and/or* Flask

Component	Tools/Libraries
Frontend	React.js
Backend	NodeJS
Backend DB	MongoDB
Hosting Service	Amazon Web Services (AWS)



Deliverables

- **Working Prototype of WebApp**
 - Front-End
 - Back-End w/ API endpoints & DB
- **Additional Proof-of-Concept Showcase**
 - Artificial Intelligence (AI) Tools
 - Classification & Prediction
 - Optical Character Recognition (OCR)
 - Scanning receipts
- **Design Guidelines**
 - Wireframes & renders
- **Documentation**
 - Deployment instructions
 - Scripts

Pages/Components
Login
Registration
Home
Groups
Transactions
Transaction Review
Account Information
Settings

Project Management

- Discord
 - Fast and easy communication
 - Voice chats, text chats, screen sharing
 - Weekly meetings
- Atlassian Jira
 - Issue & Project Tracking Software
 - Roadmap, Back-log, Boards
- Atlassian Confluence
 - Documentation & Reports
 - Useful templates

Projects / Smart Ledger

Project Definition & Planning

The screenshot shows a Jira board titled "Project Definition & Planning" under the "Smart Ledger" project. The board is divided into three columns: "TO DO 1 ISSUE", "IN PROGRESS", and "DONE 4 ISSUES".

- TO DO 1 ISSUE:** Contains one issue titled "Survey For Shared Transactions Spending Usage" with a sub-task "INITIAL PROJECT DEFINITION & P..." and a due date of "01 OCT".
- IN PROGRESS:** Currently empty.
- DONE 4 ISSUES:** Contains four completed issues:
 - "Identify Goals, Objectives, and Strategies" with sub-task "INITIAL PROJECT DEFINITION & P..." and due date "01 OCT".
 - "Identify Requirements, Deliverables, and Success Criteria" with sub-task "INITIAL PROJECT DEFINITION & P..." and due date "01 OCT".
 - "Defining Software Stack, Dev Environment, and Work-Flow" with sub-task "INITIAL PROJECT DEFINITION & P..." and due date "08 OCT".
 - "Identify Competitors & Weaknesses to Solve" with sub-task "INITIAL PROJECT DEFINITION & P..." and due date "15 OCT".



Jira



Confluence



Discord



GitHub

Project Roadmap - Spring 2022



Sprint 1 | 3 Weeks (Jan)

1. Planning, Research, and Discovery
2. Research & Briefs for Front-end+Back-end Architectures
3. Overall Design Planning and Wireframe

Sprint 2 | 4 Weeks (Feb)

1. Homepage/Front Page Design & Development
2. Internal Pages Design & Development
3. Content Creation, Features, and SEO
4. Overall Development and Coding

Sprint 3 | 4 Weeks (Mar)

1. Beta Testing, Feedback, and Improvements
2. Implementing Fixes & Improvements from Beta Testing
3. Final Improvements & Documentation
4. Launch & Working Demo

Sprint 4 | 3 Weeks (April)

1. Working on Final Reports & Documents
2. Finalizing Reports & Presentations
3. Wrapping up

Budget Proposal

Item	Estimated Cost
AWS Hosting Services - Free Tier	\$0
MongoDB	\$0
Jira - Student	\$0
Discord	\$0
GitHub Student Developer Pack	\$0
Domain	\$15/Year
Total	\$15

Questions?