Capstone Software Requirements Specification + Project Plan Description

Version 0
Group 13
COMPSCI 4ZP6
Dr. Mehdi Moradi
Oct 9th, 2025

Abdul-Hadi Siddiqui	sidda1@mcmaster.ca	400386070
Ankush Sarkar	sarkaa16@mcmaster.ca	400376621
Asad Muhammad	muhama23@mcmaster.ca	400368869
Eduardo Salvacion	salvacie@mcmaster.ca	400393061
Jayesh Anil	anilj@mcmaster.ca	400384263
Michael Zhang	zhanz479@mcmaster.ca	400387890
Viransh Shah	shahv47@mcmaster.ca	400394334

Table of Contents

Table of Contents 1
Document Information 3
Contribution History 3
Revision History 3
Glossary 3
1. Purpose of the Project 4
1.1 Background 4
1.2 Problem Statement 4
1.3 Project Objectives 4
1.4 Proposed Solution 5
2. Stakeholders 5
2.1 The Client 5
2.2 The Customer 5
3. Project Constraints 5
3.1 Off-the-Shelf Software 5
3.2 Environment Constraints 6
3.3 Schedule Constraints 6
3.4 Technical Constraints 6
3.5 Enterprise Constraints 6
4. Functional Requirements 6
4.1 Priority 0 (Minimum Viable Product) 6
4.2 Priority 1 (Core Feature Set) 7
4.3 Priority 2 (Non-critical Features) 8
4.4 Priority 3 (Future Features) 8
5. Data and Metrics 9
5.1 Datasets 9
5.1.1 SROIE and CORD public datasets of food receipts: 9
5.1.2 Google Places API: 9
5.2 Performance Metrics 10
6. Non-Functional Requirements 11
6.1 Look and Feel Requirements 11
6.2 Usability and Humanity Requirements 12
6.3 Performance and speed requirements 12
6.4 Operational and Environmental Requirements 12
6.5 Security and Privacy Requirements 12
6.6 Legal Requirements 12
7. Risks and Issues Predicted 12

7.1 Technical Risks..... 12
7.1.1 OCR Accuracy Issues..... 12
7.1.2 Voice Recognition in Noisy Environments..... 12
7.1.3 Cross-Platform Inconsistencies..... 13
7.1.4 Payment Integration Complexity..... 13
7.2 Business Risks..... 13
7.2.1 User Adoption..... 13
7.3 Legal Risks..... 13
7.3.1 Sensitive User Data..... 13
7.4 Timeline Risks..... 14
7.4.1 Tight Schedule..... 14
8. Communication Plan..... 14
8.1 Meetings..... 14
8.2 Documents..... 14
9. Team Member Roles..... 14
9.1 Requirement Responsibilities..... 14
9.2 Project Manager..... 14
10. Workflow Plan..... 15
10.1 Issue Management..... 15
10.2 Agile Methodology..... 15
10.3 Data Storage..... 15
10.4 Heavy Computation..... 15
10.5 Achieving Requirements and Performance Metrics..... 15
11. Proof of Concept..... 15
12. Technology..... 16
13. Project Scheduling..... 17

Document Information

Contribution History

Authors	Sections
Abdul-Hadi Siddiqui	5.1, 5.2, 10.4, 10.5, 13
Ankush Sarkar	6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 10.4, 10.5, 12, 13
Asad Muhammad	4.1, 4.2, 4.3, 4.4, 10.4, 10.5, 12, 13
Eduardo Salvacion	5.1, 5.2, 10.4, 10.5, 13
Jayesh Anil	3.1, 3.2, 3.3, 3.4, 3.5, 7.1, 7.2, 7.3, 7.4, 10.4, 10.5, 12, 13
Michael Zhang	1.1, 1.2, 1.3, 1.4, 2.1, 2.2, 3.1, 4.1, 4.2, 4.3, 4.4, 5.1, 5.2, 8.1, 8.2, 9.1, 9.2, 10.1, 10.2, 10.3, 10.4, 10.5, 11, 13
Viransh Shah	1.1, 2.1, 2.2, 3.1, 3.2, 3.3, 3.4, 3.5, 7.1, 7.2, 7.3, 7.4, 10.4, 10.5, 12, 13

Revision History

Version	Authors	Description	Date
0	All	Initial Document	10/10/2025

Glossary

Term	**Definition**
API	Application Programming Interface
CCPA	Consumer Privacy Protection Act
Expense	Refers to a “bill” or “transaction” that the user can split, and will be stored by the app
GDPR	General Data Protection Regulation
Host User	Referring to a user who is the one who paid for the split expense
ML	Machine Learning

MVP	Minimum Viable Product
OCR	Optical Character Recognition
P0-P3	Priority levels (0=MVP, 3=Future features)
Payee User	Referring to a user who is not the host user of a split expense
PM	Project Manager
PoC	Proof of Concept
UI	User Interface
UX	User Experience
WER	Word Error Rate

1. Purpose of the Project

1.1 Background

Coordinating group activities often involves taking into account aspects of many people's lives and preferences. Often, this is handled through a plethora of apps, while still having some people lacking information. The goal of this project is to combine common functionality that is beneficial to planning outings and streamline the process of planning, scheduling, and paying for outings.

1.2 Problem Statement

There are several applications that people use to plan events on the market, including generic social media. However, none of these are designed with planning to go out in mind, meaning there are kinks when using these apps to plan events.

1.3 Project Objectives

The objective of the project is to:

- Develop an app that allows users to conveniently plan group outings.
- Allow users to split group expenses for use cases like:
    - Restaurant bills
    - Large event budgeting
- Ensure users can split bills easily using receipt scanning and parsing for automatic itemization

- Allow users to use voice recognition to split expense items quickly
- Give users all the functionality they need to plan outings in one succinct app

1.4 Proposed Solution

The proposed solution will allow users to plan and schedule group outings on one app. It will also allow them to handle in-group payments and expenses that occur, while enabling users and non-users a quick method to send money to the user who paid for the whole expense. The system will also track the history of these transactions, allowing users to keep track of their money.

2. Stakeholders

2.1 The Client

- The client is the internal product team or organization funding and overseeing the development of this project, who aims to deliver a unified, user-friendly platform that improves the planning and payment experience for group outings
- Other clients include the teaching assistants and the professor for the capstone course, as they are invested in the development and success of the project.

2.2 The Customer

- Any group of people or organizing or participating in social activities such as dinners, karaoke, bowling, or escape rooms. They will use the app to streamline the process of planning and going to events with others.

3. Project Constraints

3.1 Off-the-Shelf Software

The following softwares are similar to some app features, but none offer the same solution:

- Banking Apps
    - Some banking apps allow users to split bills and payments
- PayPal
    - Paypal allows users to split bills
- Splitwise
    - Splitwise allows users to track and split bills
- Beli
    - Beli is an app that allows users to rank restaurants they've been to, and get information on restaurants

3.2 Environment Constraints

- Platform: iOS and Android
- Device Requirements:
    - Must work on 2019+ smartphones.
    - Enough disk space on device to install the app and store in-app data

3.3 Schedule Constraints

- The app is expected to have a functioning PoC by November 20th, 2025.
- The app is to be submitted for course evaluation on April 4th, 2025.

3.4 Technical Constraints

- OCR Processing: Offline On-device where possible for privacy and speed.
- Voice Recognition: Must work offline and in moderately noisy environments.

3.5 Enterprise Constraints

- Budget:
    - Use free-tier services where possible
    - Target external service costs <$125/month
- Regulatory & Privacy:
    - Must comply with GDPR and CCPA
    - Handles payment metadata (name, email, and amount sent to receiver)
    - System must support client modifications for regulatory compliance (e.g., GDPR, internal audit)
- Source Code License:
    - Proprietary license; source code is not openly distributed
    - Clients may be granted limited access under strict terms for compliance or security review
- Security:
    - System must support adaptation to meet organizational security standards (e.g., encryption, auth, network rules

4. Functional Requirements

4.1 Priority 0 (Minimum Viable Product)

- The host user should be able to manually add a list of expenses
- The host user should be able to scan a receipt and be given the parsed and itemized list of expenses

- The host user should be able to manually split the expenses among other people or users
- The host user should be able to vocally split the expenses among other payee users through the app (voice detection)
- The payee user should be redirected to their preferred payment method when a payment is requested
    - For people who need to make payments but do not have the app, the host user can input their email or phone number to send an payment notification
- User authentication
    - The user should be able to sign-up and login with their email, or associated Google/Apple account
    - The user should be able to link their e-transfer email or paypal/venmo details to seamlessly make payments when needed
    - The user should be able to access their recent expenses and expense history

4.2 Priority 1 (Core Feature Set)

- The app can be used offline
    - All expense splitting functionality should be available and work offline
        - The user can manually input a list of expenses
        - The user can scan a receipt and get a parsed list of expenses
        - The user can manually split the list of expenses
        - The user can split the bill through vocal recognition
    - Offline expense history will be merged when the user comes back online
        - Only if some user has been logged in prior
- The user can create events and add other users
    - Users will be added through their unique identifier
    - The user can also create stand-in users for people who do not have the app
        - The stand-in can have either a phone number or email for the purpose of sending a payment reminder
        - The user can also give the stand-in a location for recommendation purposes
    - Users of the app should be able to add dates and times they are available
        - Availability for stand-in users can be changed by other app users that are in the group
    - The user can see a list of other event members
    - The user can set a location for the event
    - The user can export the event date and time to their Google/Apple calendar (P2)

    - The user can send and receive messages from other event members (P3)
- The user can search through nearby locations
    - Locations can be filtered and sorted by based on an input location
    - Locations can be filtered and sorted by based on average of event members location
    - Locations can be filtered and sorted by based on dietary restrictions for restaurants (P2)
    - Locations can be filtered and sorted by cost (P2)
    - Locations can be filtered and sorted food category (P2)
    - Locations can be filtered and sorted by seating amount (P3)

4.3 Priority 2 (Non-critical Features)

- The user can access an in-app map that displays recommended locations of interest (restaurants, karaoke, etc.)
- Event members should be able to access a shared noteboard
    - The user can access the noteboard quickly from anywhere in the event interface
    - The noteboard can be modularly customized to include checklists and other features as necessary (P3)
- The user can wishlist dishes for the selected restaurant for planning purposes
    - The user can add a modular wishlist for dishes they want to order at the restaurant (P3)

4.4 Priority 3 (Future Features)

- The users can set their weekly/monthly budgets for events
    - Events have functionality to plan around each users budget
- The user can see location details for the event
    - Reviews for that location can be parsed and summarized
    - Location contact information can be provided to make reservations
    - Details on location waiting times and reservations can be displayed
    - Information on item price history of the location can be displayed
- The application can generate recaps for outings, including associated photos which users can add
- The user can access metrics for previous outings, such as spending habits and common locations
- The users can share their outing details with other users
- The users can plan optimal event transportation based on carpool routing
    - Users with cars can figure out who they need to pick up and in what order

5. Data and Metrics

5.1 Datasets

If we do proceed further in training our own AI mode, the following data will be used to train our AI model for our application:

5.1.1 SROIE and CORD public datasets of food receipts:

- How to get: We can gather from following links:
    - SROIE: https://github.com/zzzDavid/ICDAR-2019-SROIE
    - CORD: https://github.com/clovaai/cord
- Why they're needed:
    - We need these public datasets because to create a more accurate AI model for image and text recognition of the receipts users will upload. We need lots of data to train the model so that it learns to make more accurate decisions and predictions.
- What features it will be used for:
    - It will be used for the text recognition feature for receipts.

In total, we aim to get a total of over 500+ receipts consisting of: restaurant receipts, non restaurant entertainment receipts from activities where food is served like bowling, retail recipes, and other miscellaneous venues. To ensure better training given our goals/objectives, we plan to have:

- 50% of datasets be associated with restaurant receipts
- 20% associated with non restaurant entertainment receipts from activities where food is served like bowling
- 20% associated with retail receipts
- 10% associated with miscellaneous receipts

5.1.2 Google Places API:

- How to get:
    - We can get a list of locations from the following google API: https://developers.google.com/maps/documentation/places/web-service/overview
- Why they're needed:
    - Since our application has an algorithm that will recommend locations to users, we need this data to suggest locations based on a variety of factors.
- What features:
    - Used to implement AI algorithms used to recommend new restaurant locations to host.

5.2 Performance Metrics

Metric	Measures	Purpose	Goal
Receipt OCR – Accuracy (Total and Line Item Extraction)	How correctly the OCR system identifies amounts, item names, and totals from scanned receipts	Accurate total extraction is essential for fair bill splitting and user trust; errors directly affect payments	≥ 90% accuracy for total amount extraction (critical metric) ≥ 80% accuracy for line-item extraction (acceptable with manual editing fallback)
Voice Recognition and Word Error Rate (WER)	The percentage of incorrectly transcribed words in user voice input	Lower WER means the app can reliably understand user commands in noisy real-world environments (restaurants, bars)	WER ≤ 15%** (i.e., ≥ 85% correct word recognition) Ensures users can complete voice-based bill splits quickly and confidently
Voice Recognition – Precision and Recall (Intent Detection)	Precision: How many recognized commands or entities were correct Recall: How many relevant commands were successfully detected	Prevents incorrect bill assignments (high precision) and ensures no valid splits are missed (high recall)	Precision ≥ 85% Recall ≥ 80% Balances accuracy and coverage for reliable speech-based interaction
Processing Time / Latency	How long the app takes to process OCR scans, speech recognition, and core operations.	Users expect fast, real-time results during social outings, slow responses disrupt group flow	OCR processing ≤ 5 seconds on mid-range devices Other core operations ≤ 3 seconds Supports overall goal of completing bill splitting and

			settlement in under 2 minutes
System Reliability and Failure Rate	The percentage of scans or voice inputs that fail to process entirely.	A low failure rate indicates robustness and reduces frustration or repeated manual entry.	OCR failure rate ≤ 10% Voice command fallback mechanisms (e.g., manual input prompts) to handle remaining failures.
User Interaction Efficiency and Task Completion Time	The total time needed to finish one full bill-splitting workflow, from receipt scan to payment confirmation.	Reflects usability and overall app efficiency; shorter completion times increase user satisfaction and adoption.	Under 2 minutes per event (planning, splitting, and payment). Aligns with the app's core service goal of frictionless, real-time collaboration.
Content Recommendation – Precision and Recall at K	Precision at K: The proportion of relevant recommended items in the top K results Recall at K: The proportion of relevant items that are successfully recommended in the top K results.	We want enough relevant content to be recommended without pushing too much content without care for its relevance.	F1 Score (Harmonic mean of both) ≥ 0.65 This metric helps balance both metrics, since improving one metric likely will decrease the other. This ensures both are reasonable.

6. Non-Functional Requirements

6.1 Look and Feel Requirements

- Make sure the visual design of the app is consistent across all pages
- Support both light and dark mode
- The main actions a user performs (event creation, scan receipt, split bill etc) should be visually prominent

- The app looks clean
- The app feels intuitive to use

6.2 Usability and Humanity Requirements

- The UI should be intuitive so that first time users without any training are able to perform all the main actions.
- The UI will meet the Web Content Accessibility Guidelines (WCAG) 2.2.

6.3 Performance and speed requirements

- Voice recognition should be very accurate for quiet indoor environments, and have at least 70% accuracy for loud environments.
- Receipt parsing OCR should have a high accuracy.
- Any payment should have the maximum response time an e-transfer would usually take.

6.4 Operational and Environmental Requirements

- The user should be able to use our app on both android and ios.

6.5 Security and Privacy Requirements

- The user needs to be authenticated to perform any transaction.

6.6 Legal Requirements

- The app will comply with the apple app store and google play store policies.

7. Risks and Issues Predicted

7.1 Technical Risks

7.1.1 OCR Accuracy Issues

- Risk: Receipts vary wildly in format, OCR may fail frequently depending on image quality
- Mitigation:
    - Provide real-time feedback on image quality before scanning.
    - Check API confidence scores and give user suggestions to improve image quality.
    - Allow manual entry fallback.

7.1.2 Voice Recognition in Noisy Environments

- Risk: Voice splitting may not work in loud venues (bars, bowling alleys)
- Mitigation:

    - Use noise-resistant algorithms
    - Show clear error messages and suggestions to reduce noise if environment too noisy
    - Always provide manual alternatives
    - Test in real-world noisy conditions early

7.1.3 Cross-Platform Inconsistencies

- Risk: Bugs or platform-specific issues
- Mitigation:
    - Extensive testing on both iOS and Android.
    - Platform-specific code where necessary.
    - Beta test on a variety of devices.

7.1.4 Payment Integration Complexity

- Risk: E-transfer deep links may not work on all devices
- Mitigation:
    - Test on multiple devices and OS versions
    - Provide copy-to-clipboard link fallback
    - Clear instructions if deep link fails

7.2 Business Risks

7.2.1 User Adoption

- Risk: Users may not switch from existing solutions (Splitwise, Venmo, PayPal, calculators)
- Mitigation:
    - Focus on unique value props (OCR, voice, multi-activity support, etc.)
    - Beta test with college/university students (captive audience)
    - Viral loop (Social media): Invite friends to events

7.3 Legal Risks

7.3.1 Sensitive User Data

- Risk: Sharing name, email, and amount for e-transfers within groups going out together may risk exposing sensitive data or non-compliance if not properly controlled.
- Mitigation:
    - Restrict data sharing strictly to members of authorized groups only.
    - Obtain explicit user consent for sharing transfer details within groups.
    - Log and audit data sharing events to detect unauthorized access.

7.4 Timeline Risks

7.4.1 Tight Schedule

- Risk: 7 months may not be enough for all P0/P1 features
- Mitigation:
    - Prioritize and implement high-risk or complex features early to reduce uncertainty.
    - Develop complete, testable features across all layers (UI, backend, database) for early feedback and smoother integration.
    - Reserve dedicated time at the end of the project for thorough testing, bug fixes, and final polish before release.

8. Communication Plan

8.1 Meetings

Meetings will be conducted and planned through discord, at least once per sprint cycle. Meetings will consist of going over the sprint's issues on Linear, and any issues that need to be addressed brought up by any member. In each meeting, each member will report the status of their current task.

8.2 Documents

Documents will be shared on several platforms but categorized based on importance/frequency of use. Simple documents for one use can be shared on discord or between members individually, but important documents meant to be used often will be tracked on google drive and GitHub.

9. Team Member Roles

9.1 Requirement Responsibilities

Functional and non-functional requirements will be assigned each sprint. These will be delegated based on who has worked on related requirements, who is currently waiting for tasks, and prior experience.

9.2 Project Manager

Michael will be the project manager. He will handle coordinating meetings and other group activities. He will also assign issues during meetings, and monitor progress on tasks.

10. Workflow Plan

10.1 Issue Management

We will be using GitHub to track development. We will have a main branch that will be kept stable, with no current development. All members will develop on a branch off of the main branch (develop). Any time an issue is deemed complete, the developer will make a pull request that must be reviewed by two other members to merge into the main branch. Issues will be managed on Linear, which has GitHub integration. Anytime a pull request is approved and merged, linear will track it automatically.

10.2 Agile Methodology

We will be using the agile methodology. Our planning will be done in Linear. A sprint will be 1 week long, with minimum one meeting during that time period.

10.3 Data Storage

Data will be stored and tracked on GitHub, if we need data to train our own model. We will track what data is used, and training statistics here as well.

10.4 Heavy Computation

If necessary, training and compute heavy tasks will be done through mcmaster gpu resources when available. If unavailable for sufficient periods, we will be using google colab.

10.5 Achieving Requirements and Performance Metrics

To ensure each requirement is achieved, we will be tracking each requirement with a linear issue. Using agile methodology, we will often review the status and progress of tasks and be able to respond accordingly. If deadlines for requirements are not being met, we will try to discover and address the root cause. We will also be following a figma design, to ensure that our non-functional requirements are met, without risk of consistency changes.

To achieve the performance metrics and to ensure previous achievements are not undone by any updates, we will be using unit testing, regression testing, and smoke testing. These will ensure that complete features will not break after unrelated updates, and testing will allow us to measure the performance of any models we train. This will allow us to measure the success of our models.

11. Proof of Concept

During the proof of concept demonstration, we aim to have the skeleton of the app complete, along with basic user authentication. The app should have manual list

creation and bill splitting complete, but without bill history being saved. We aim to have some method of voice recognition and receipt parsing through images, but without measuring their performance.

12. Technology

Design:

- Figma

Frontend:

- Language: Typescript
- Framework: Expo/React Native (cross-platform)
- State Management: Zustand, TanStack Query

Backend:

- Database: Cloud Firestore
- Auth: Firebase Auth
- Analytics: Firebase Analytics

ML Libraries:

- Google ML Kit

We may need to use GPUs if we are going to be training our AI model for OCR and voice to text model on McMaster available GPUs (preferrable) or GPUs on Google Colab Pro environment.

13. Project Scheduling