This software development plan outlines the strategy for developing a React Native-based iOS application called "TABS" designed to enhance dining experiences by easily splitting bills. This application is aimed at enhancing group purchases by simplifying the process of dividing the bill. Utilizing advanced technologies such as React Native for cross-platform mobile development, Tesseract.js/Google Vision api for receipt scanning and information extraction, and Firebase for user authentication and data management, this application addresses the often cumbersome task of bill splitting by automating the division of items on a receipt among diners.

The development of this application involves several pieces: initial requirement analysis, design of the user interface and experience, integration of external APIs (Tesseract and Firebase), development of the core functionality (receipt scanning, item parsing, and assignment), testing for reliability and usability, and final deployment to the iOS platform. Key milestones include completion of the functional requirements document, technical design document, initial prototype, MVP, and feature updates.

4.1.1 Project Deliverables

- Functional Requirements Document (01/2024): Detailed specification of application features and user interactions. This document will serve as the blueprint for development, outlining how the application will function from the user's perspective.
- Technical Design Document (01/2024): Architectural overview of the application, including the selection of the React Native framework, use of Google Vision API for text recognition, and Firebase for user management and data storage. This will detail the technical infrastructure and external services integration.
- Initial Prototype (early 03/2024): A working prototype demonstrating the core functionality of receipt scanning and item parsing, offering a tangible glimpse into the application's potential.
- MVP (late 03/2024): A near-complete version of the application available for selected users for testing. This version will incorporate user feedback for improvements.
- Final Application (04/2024): The completed application, fully tested and ready for launch on the iOS App Store. This includes all documented code, user manuals, and deployment guides with features updates we are able to accomplish.

4.2 Project Resources

4.2.1 Hardware Resources

- Development Computers: Majority of us will be using Windows and one person Mac.
- iOS Testing Devices: iOS devices including iPhones and iPads to ensure compatibility and performance across different screen sizes and systems.
- Server (Optional for Firebase): Firebase cloud services will be utilized, eliminating the need for a separate server for backend operations.

4.2.2 Software Resources

- React Native: The cross-platform framework selected for mobile application development.
- Google Vision API: For implementing receipt scanning and text recognition functionalities.
- Firebase: For managing user authentication, data storage, and potentially hosting the application.
- Xcode: The development environment used for iOS app development, testing, and deployment.
- Expo: Windows development environment for iOS.
- Git/GitHub: For version control and collaborative development among potential contributors.
- Jira/Trello: For project management, task assignment, and progress tracking.

4.3 Project Organization/Human Resources

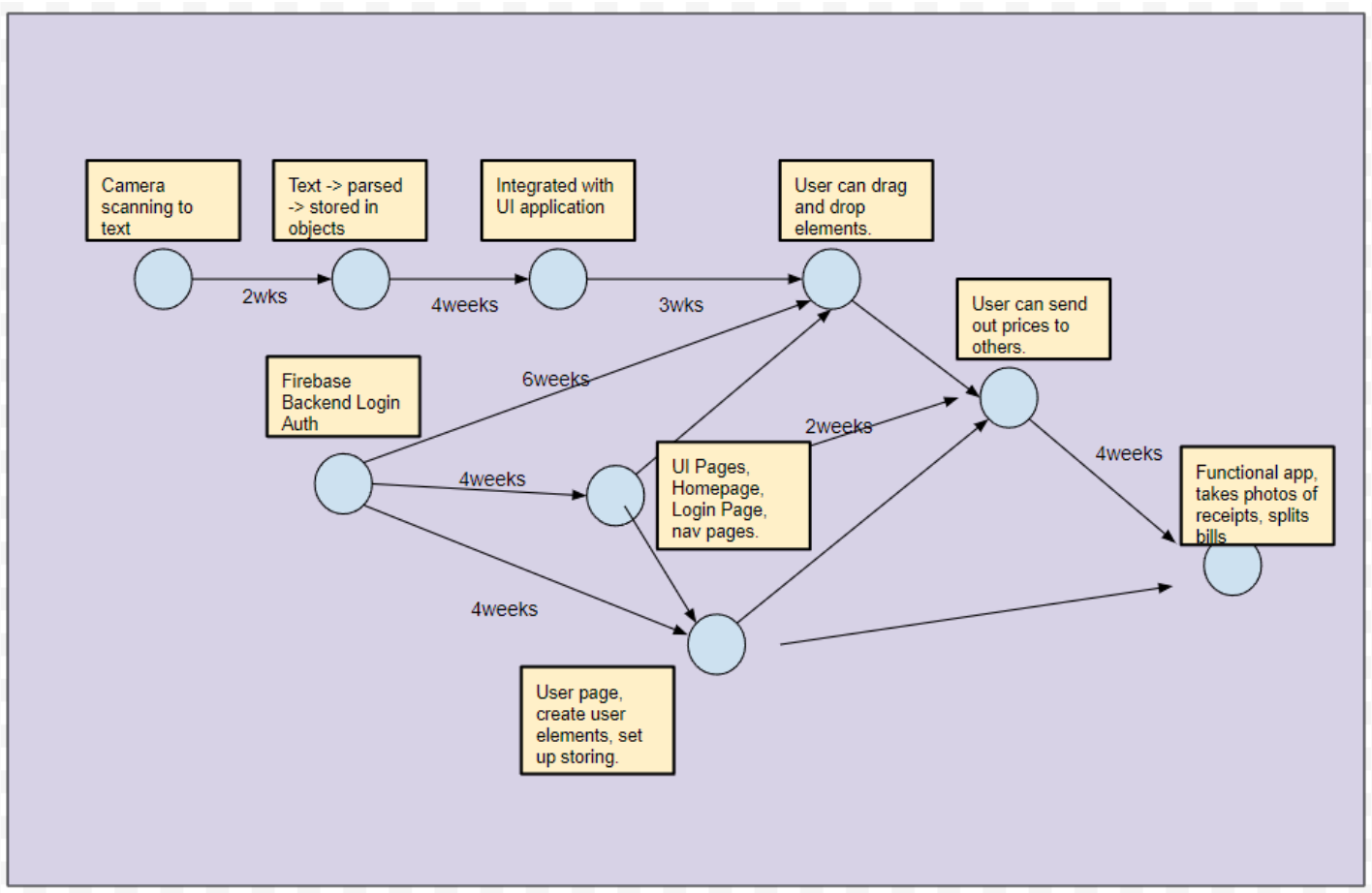
Given the nature of this project, the development process will be segmented into major functions as follows:

- Requirements Analysis and Design: Focusing on defining the functional requirements and crafting the user experience and interface design.
- API Integration: Integrating Google Vision API for receipt scanning and Firebase for user management.
- Core Functionality Development: Implementing the primary features of receipt scanning, item parsing, and assignment among users.
- Testing and Quality Assurance: Ensuring the application's reliability and usability through testing.
- Deployment and Maintenance: Preparing the application for launch.

4.4 Schedule

This is described by the Gantt chart below, but primarily we have two people working on the backend and two on the front end. The front end needs to be completed at 8 weeks, then both teams will integrate their work and the bulk of the work will be getting the app fleshed out.

4.4.1 PERT Chart



4.4.2 Task/Resource Table

Task	Assigned To	Hardware Resource	Software Resource
Requirements Analysis	Connor	Windows PC	Google Docs, Jira, EXPO
UI/UX Design	Evan	MacBook Pro	Adobe XD, Figma
API Integration (Tesseract)	Owen	Windows PC	Visual Studio Code, Postman, Google Vision API, EXPO

API Integration (Firebase)	Mitchell	Windows PC	Visual Studio Code, Firebase SDK, EXPO
Frontend Development (React Native)	Connor, Evan	Evan: MacBook Pro, Connor: Windows PC	Evan: React Native, Xcode, Connor: React Native, EXPO
Backend Development (Firebase Functions)	Owen, Mitchell	Windows PC	Firebase Console, Node.js, EXPO
Testing (Unit Tests)	Connor	Evan: iPhone X, iPhone 12, iPad Pro, Connor: Android Devices	Jest, React Native Testing Library, EXPO
Testing (Integration Tests)	Evan	iPhone X, iPhone 12, iPad Pro	Postman, Firebase Test Lab
Documentation	Mitchell	Windows PC	Google Docs, Markdown Editors, EXPO
Deployment	Owen	Windows PC	EXPO, Firebase Hosting
Post-Launch Support	All	Evan: MacBook Pro, iPhone X, iPhone 12, iPad Pro, Others: Windows PC, Android Devices	Slack, Firebase Console, EXPO