**Documentation of SplitRight**

Shlok Manandhar

Westcliff University

PRG 402: Mobile Programming Applications

Prof. Khanal

Mar 24, 2025

1. **Define goal and project purpose.**

* Problem statement: Shared expenses in groups (with friends, roommates, colleagues) usually is not easy to manage. It is difficult to keep track of who owes what and disputes could happen easily.
* Team Audience: Whether it is with groups of friends, roommates, families, colleagues, or event organizers, there is no better way than a simple and efficient way to split expenses.
* Value proposition: SpliteRight enhances the group expense tracking, splitting, and settling across a vertical segmentation of invoices which greatly diminishes friction of managing finances from a group perspective.
* Competitive differentiation: Whereas other expense tracking apps are generic, SpliteRight makes it easy by allowing users to create a group, then invite members to split an expense.

1. **Specify the Application Type.**

* App type: Cross-platform mobile application.
* Category: Productivity/Finance.
* Feature: This has real time updates, requires real time projections for expenses and cloud-based storage of data.

1. **Mention supported devices.**

* Devices: Android OS and iOS.
* Screen Compatibility: Compatibly in any mobile size, and any tablets size.
* OS compatibility: Support latest iOS and Android versions with at least 2 previous versions in backward compatibility.

1. **Define Mobile Platforms**

* Target Platforms: Android & iOS users.
* Framework: React Native.
* Constraints: The UI might be optimized on the basis of platform.

1. **Provide a Service/ Product Description.**

Functionality Overview:

* Users are able to create groups and to send invitations to multiple users.
* Track shared expenses within groups.
* When you manually switch expenses to people, you auto split the expenses with split info showing who owes who.
* Send reminders and manage settlements.

Business Objectives Alignment: It helps in better financial management, reduces the disputes, and brings the transparency in the group expenses.

User Workflows;

* Create an account and log in.
* Create a group and invite members.
* Add expenses and assign payees.
* Track and settle balances.
* Receive notifications for settlements.

Scalability Plan:

* Start small, with small user groups
* In the future, add more feature like bill scanning, and other new features.

1. **Create a Feature list:**

Core Features:

* Creating Groups & Invitations: Allow users to create groups and invite members via link.
* The Features: Users can log expenses.
* This task is automated splitting: either equal size splits or custom size splits.
* Settlement Balances: See balances and solve debts.
* Notifications & Reminders: Send out notifications and reminders to the users of pending payments.
* User Authentication: Secure login with email/phone authentication.

Future Enhancements: Bill Scanning: Use OCR to auto-fill expenses.

1. **Define Functional Requirements**

User Interactions & Work:

* Signup/Login: OAuth-based authentication
* Group Management: Create, delete, join, leave groups.
* Expense Handling: Add, delete, edit and settle expenses.
* Notifications: Receive notification if a user, wants to remind the group for pending payments.

API Endpoints

* /auth/signup – User registration.
* /auth/login – User authentication.
* /groups/create – Create a new group.
* /groups/join – Accept invitations.
* /expenses/add – Add an expense.
* /expenses/split – Calculate owed amounts.
* /payments/settle – Mark debts as paid.

Error Handling: using try, catch and block methods to be used

1. **Provide Technical Specifications**

* Backend Technology: Node.js with Express.js.
* Database: MongoDB or MySQL
* Third party Integrations:
* Firebase Authentication for secure login.
* Twilio or SendGrid for notifications.
* Stripe/PayPal for future payment integrations.
* Programing Stacks:
* Frontend: React Native with JavaScript and TypeScript
* Backend: Nodejs and Express js.
* Deployment: Cloud base (AWS)
* CI/CD strategy: GitHub to be used.

1. **Outline Required Services**

API and backend service

* User authentication.
* Group management.
* Expense tracking and calculations.

Maintenance & Support:

* Bug fixes, performance optimizations.
* Feature updates based on user feedback.

CI/CD Implementation

* Automated testing pipeline.
* Deployment via cloud services.

1. **Add Additional Information**

Timelines & Milestones:

* Week 1-2: Research & UI/UX Design.
* Week 3-5: Backend & Database Setup.
* Week 6-8: Frontend Development.
* Week 9-10: Testing & Debugging.
* Week 11: Deployment & User Testing.

Estimated Budget:

* Development tools & hosting: Rs 70,000/month.
* API services & maintenance: Rs 40,000/month.

Constraints & Risks:

* Ensuring seamless cross-platform performance.
* Secure financial transactions and user data.
* Handling large user bases efficiently.

Regulatory Considerations:

* GDPR compliance for user data protection.
* Secure storage of financial transactions.
* Terms of Service and Privacy Policy adherence.