Software Requirements Specification - Split Smart Software Team 2

Samuel Stenerson, Deniz Acikbas, Kyle McCarthy, Murad Tawfiq, Parashar Parikh

Table of Contents

1.0 Introduction

1.1 Project Scope

The proposed application, SplitSmart, aims to provide users with a comprehensive expense management solution similar to Splitwise. SplitSmart will allow users to create and manage user accounts, create and track expenses, manage groups, track balances, record payments, and generate reports. The app will cater to the needs of individuals, roommates, and groups of people who want to simplify expense tracking and settlement.

Major Inputs:

1. **User Accounts:** Each user will have their own account to log in and access their personal expense and balance information.

2. **Group Management:** Users can create and manage groups to more efficiently track shared expenses and balances within those groups. This feature will enable roommates, trip participants, or party organizers to collaboratively manage expenses.

3. **Expense Creation:** Users can create new expenses by providing essential information such as amount, date, description, shared manner (equally split or split by percentage), and an optional receipt image. Users can also specify other users involved in the expense, facilitating accurate balance adjustments.

4. **Approval Workflow**: The application will incorporate a system for reviewing and approving expenses before they are added to the system and used to adjust balances. The approval workflow may include multiple levels of approval or a single review process, depending on the needs of the group.

5. **Notification System:** SplitSmart will feature a notification system to inform users when other group members create new expenses. This real-time communication will enhance collaboration and keep all group members updated on financial activities.

6. **Balance Tracking:** The application will track the balance owed by each user to every other user, considering all expenses entered into the system. It will accurately calculate and display the net balance between users, simplifying debt settlement.

7. **Payment Tracking:** Users can record payments made to settle balances, ensuring that the balances reflect the actual financial transactions. This functionality will include a payment button with an amount, which, when clicked, will update the respective balances.

8. **Reporting:** SplitSmart will provide users with comprehensive reports and summaries of their expenses, balances, and payments. Additionally, users will have

the ability to generate custom reports by specifying date ranges and other parameters, allowing for personalized financial insights.

Processing Functionality:

1. User account management
2. Group Management
3. Expense creation and approval
4. Notification system
5. Balance tracking
6. Payment tracking
7. Reporting and summaries

The project scope for SplitSmart encompasses the development of a mobile application for expense management, replicating the core features of Splitwise. The app will enable users to create accounts, manage groups, create and approve expenses, track balances, record payments, and generate detailed reports. The application will focus on enhancing user experience, collaboration within groups, and providing accurate financial insights. By implementing these features, SplitSmart aims to streamline expense tracking and settlement processes, making it a valuable tool for individuals and groups alike.

1.2 Major Software Function

1. **User Account Management**
   - 1.1. User registration: Allow users to create new accounts with required information.
   - 1.2. User login: Enable users to log in securely using their credentials.
   - 1.3. Account authentication: Verify user credentials to ensure secure access to personal expense and balance data.

2. **Group Management**
   - 2.1. Create a group: Provide functionality to create new groups within the application.
   - 2.2. Add/remove group members: Allow group administrators to add or remove members from a group.
   - 2.3. Assign group administrators: Designate certain users as group administrators with additional privileges.
   - 2.4. Configure group settings: Enable group-specific settings such as group name, description, and privacy settings

   .

3. **Expense Creation and Approval**
   - 3.1. Create new expenses: Allow users to enter expense details such as amount, date, description, shared manner, and optional receipt image.
   - 3.2. Expense review and approval: Implement a workflow for reviewing and approving expenses before they are added to the system and used to adjust

balances.

4. **Notification System**
   - 4.1. Real-time notifications: Send notifications or alerts to users when other group members create new expenses.
   - 4.2. Instant updates: Ensure that users receive timely notifications on their devices to stay informed about financial activities within their groups.

5. **Balance Tracking**
   - 5.1. Calculate balances: Track the balance owed by each user to every other user based on entered expenses.
   - 5.2. Apply adjustments: Consider all expenses and accurately adjust balances to reflect the net owed amount.
   - 5.3. Real-time balance updates: Update balances in real-time as new expenses are added or payments are made.

6. **Payment Tracking**
   - 6.1. Record payments: Allow users to record payments made to settle balances.
   - 6.2. Balance adjustment: Automatically update balances based on recorded payments.
   - 6.3. User-friendly payment entry: Provide an intuitive interface for users to enter payment details easily.

7. **Reporting and Summaries**
   - 7.1. Generate reports: Enable users to generate reports and summaries of their expenses, balances, and payments.
   - 7.2. Customizable reports: Allow users to specify date ranges, group filters, or other parameters for personalized reports.
   - 7.3. Clear presentation: Display reports in a clear and understandable format for easy comprehension.

1.3 Performance/Behavior issues

To create an efficient, seamless, and positive user experience, this tool should adhere to the following performance and requirements criteria:

1. **Responsiveness:** The application should be responsive and provide quick responses to user interactions. Actions such as creating expenses, updating balances, or generating reports should be performed promptly, minimizing any noticeable delays.

2. **Reliability:** The software should be reliable and stable, minimizing crashes, errors,

or unexpected behavior. It should handle various scenarios, such as network interruptions or data inconsistencies, gracefully and provide appropriate error handling and recovery mechanisms.

3. **Scalability:** The application should be designed to handle a growing user base and increasing data volumes. It should be able to efficiently scale its performance as the number of users, groups, and expenses increases over time, ensuring a smooth experience for users regardless of the system load.

4. **Security**: User data, including account details, expenses, and balances, should be securely stored and transmitted. The application should implement robust security measures, including encryption, secure authentication, and access controls, to protect user privacy and prevent unauthorized access or data breaches.

5. **Usability:** The user interface should be intuitive, user-friendly, and visually appealing. The application should follow consistent design principles, use clear and concise language, and provide appropriate feedback to guide users through various actions and processes. Minimizing user input and providing default values or smart suggestions where possible can enhance usability.

6. **Optimized Performance:** The software should be optimized for performance to minimize load times and resource consumption. This includes efficient data retrieval and storage, caching strategies, and optimizing algorithms and queries to ensure smooth and fast operations, even with large datasets.

7. **Accessibility:** The application should be designed to be accessible to a wide range of users, including those with disabilities. It should follow accessibility guidelines and standards, provide appropriate contrast, support screen readers, and offer alternative input methods to accommodate different user needs.

8. **Consistency:** The software should maintain consistency throughout the user experience, ensuring that similar actions or features behave predictably across different screens or modules. This consistency applies to terminology, layout, navigation, and interaction patterns, providing a cohesive and familiar experience to users.

9. **Offline Capability:** While connectivity may not always be available, the application should offer offline capabilities, allowing users to perform certain tasks, such as creating expenses or viewing previous data, without an active internet connection. The app should sync data seamlessly once connectivity is restored.

10. **Feedback and Notifications:** The application should provide clear and informative feedback to users, confirming that actions have been successfully performed or alerting them to any issues or errors. Users should receive

notifications or updates in a timely manner, keeping them informed about new expenses, balance adjustments, or payment confirmations.

1.4 Management and Technical Constraints

1.4.1 Management Constraints

1. **Resource Limitations:** Depending on the size of the development team, the time frame, and budget allocated to this project, there may be limitations on the number of features that can be developed and the extent of their complexity.
2. **Time Constraints**: The schedule for project completion could impact the quality and number of features the software can offer. The project needs to be planned and managed to ensure timely delivery.
3. **Skills and Expertise:** The team's familiarity with the technologies required for this project might be a constraint.

1.4.2 Technical Constraints

1. **Data Security:** Ensuring the privacy and security of users' data is critical. Implementing robust security measures may add complexity and could limit some design choices.
2. **Platform Compatibility:** The software should be compatible across different platforms (iOS, Android, web browsers). This cross-platform compatibility could impose constraints on the technologies and tools used for development.
3. **Scalability**: As the number of users and transactions grow, the software should maintain performance. Building for scalability can impact the choice of technologies and the design of the database and architecture.
4. **Integration with Payment Systems:** While the software does not directly handle payments, future additions might involve integrating with payment systems. Therefore, the architecture must be flexible enough to allow such potential integrations.
5. **Offline Accessibility:** Depending on user needs, offline accessibility might be required. This could constrain the application design as data will need to be accessible and syncable when the app is back online.
6. **Network Performance:** Care must be taken to ensure the app works smoothly even under poor network conditions

2.0 Project Estimates

2.1 Historical Data Estimates

This project is used to improve the already existing web application splitwise. Currently there are no project estimates available for splitwise so based on the software functions a baseline estimate will be used.
It's important to note that every project is unique and there may be unforeseen circumstances that affect our estimates. Therefore, we will continually monitor and adjust our estimates as necessary throughout the project lifecycle.

2.2 Estimation Techniques Applied and Results

There are two estimation techniques that will be used to estimate the effort and cost of the software being created. The two methods are:

- Function Point Analysis (FPA)
- Lines of Code (LOC)-Based Estimation

2.2.1 FPA-based Estimation

A. Number of External Inputs (EI):

- ➔ Account Registration
- ➔ Update Account Information
- ➔ Create Group
- ➔ Edit Group
- ➔ Create Expense
- ➔ Edit Expense
- ➔ Record Payment

B. Number of External Outputs (EO)

- ➔ Notifications
- ➔ Expense Approval
- ➔ Balance Updates
- ➔ Payment Updates

C. Number of External Inquiries (EQ)

- ➔ Account Login
- ➔ View Account Details
- ➔ View Group Details
- ➔ View Expense Details
- ➔ View Balances
- ➔ View Payments

D. Number of Internal Logical Files (ILF)

- ➔ Account Information
- ➔ Group Information
- ➔ Expense Information
- ➔ Balance Information
- ➔ Payment Information

Number of External Interface Files (EIF)

- ➔ None identified for this project.

E. Function Points

Information Domain	**Value Count**	**Weight Factor**	**Total**
External Input (EI)	7	4	28
External Output (EO)	4	5	20
External Inquiries (EI)	6	4	24
Internal Inquiries (ILF	5	10	50
External Interface Files (EIF)	0	0	0
Total			122

F. **FP Value Adjustment Factors (0 (not needed) to 5 (absolute must))**

Backup required?	5
Data communication?	5
Distributed processes?	3
Optimal performance?	3
Heavily used operating system?	3
Online data security?	5
Complex inputs, queries, outputs?	4
Complex internal processing?	3

Reusable code?	2
Ease of use?	5
Total	38

Value Adjustment Factor [0.65 + 0.01 * Σ Fi] = [0.65 + 0.01 * 38] = **1.03**

FP = T * (0.65 + 0.01 x Σ Fi) = 122 * 1.03 = 125.66 ≈ **126**

Assuming the average productivity is 15 FP/pm (due to the complexity of the project), and the burdened labor rate is $8000 per month, we can calculate the cost per FP and the project estimates.

Cost per FP = $8000 / 15 = **$533.33 per FP**

Estimate for FPA-Based Estimation

Total Estimated Project Cost: $533.33 x 126 = **$67,200**.

Total Estimated Effort: 126 / 15 = **8.4 person-months**

Estimated Duration: 8.4 / 4 = **2.1 months**

2.2.2 LOC-Based Estimation

A. Weighting Factors:
   - External Inputs: 4
   - External Outputs: 5
   - External Inquiries: 4
   - Internal Logical Files: 10

LOC

Function	**FP**	**Weighting Factor**	**LOC/FP**	**Total LOC**
Account Registration	EI, ILF	4 + 10	50	700
Update Account Information	EI	4	50	200
Create Group	EI, ILF	4 + 10	50	700

Edit Group	EI	4	50	200
Create Expense	EI, ILF	4 + 10	50	700
Edit Expense	EI	4	50	200
Record Payment	EI, ILF	4 + 10	50	700
Notifications	EO	5	50	250
Expense Approval	EO	5	50	250
Balance Updates	EO	5	50	250
Payment Updates	EO	5	50	250
Account Login	EQ	4	50	200
View Account Details	EQ	4	50	200
View Group Details	EQ	4	50	200
View Expense Details	EQ	4	50	200
View Balances	EQ	4	50	200
View Payments	EQ	4	50	200
Total:				5,500

Average Productivity: **2000 LOC/pm**

Cost per LOC: $8000 / 2000 = **$4 per LOCA**

B. Estimate for LOC-Based Estimation

Total Estimated Project Cost: $4 x 5500 = **$22,000**

Total Estimated Effort: 5500 / 2000 = **2.75 person-months**

Estimated Duration: 2.75 / 4 = **0.69 months**

2.3 Reconciled estimate

The reconciled estimate is the final estimate for the project, which is derived by integrating and reconciling all the previous estimates. It is important to note that the reconciled estimate is not a simple average of the previous estimates. Instead, it takes into account various factors such as the assumptions, risks, constraints, uncertainties, and other relevant information that were used in deriving the previous estimates.

Based on the previous sections, we have estimated that the project will require approximately 5,500 lines of code to implement all the required features. Using a LOC-Based estimation model, we have estimated that the effort required for this project will be 5.58 person-months and the duration of the project will be 1.4 months.

To arrive at the reconciled estimate, we have considered the following factors:

Project scope: We have carefully reviewed the project scope and ensured that all the required features are included in the estimate. We have also factored in the possibility of additional features being added during the development process.

Technical complexity: We have taken into account the technical complexity of the project and the skill levels of the team members. We have also factored in the need for learning and experimentation during the development process.

Risks and uncertainties: We have identified the major risks and uncertainties associated with the project, such as the need for integrating with different 3D modeling tools and the availability of the AR library for the chosen platform. We have factored in the impact of these risks and uncertainties on the project estimates.

Resource availability: We have ensured that the team members have adequate time to work on the project, factoring in their availability and other commitments.

Based on our analysis, we believe that the reconciled estimate for the AR tool project is as follows:
Effort: 5.58 person-months
Duration: 1.4 months
Cost: $44,600
These estimates are based on our best understanding of the project at this time. As the project progresses, we will continue to monitor and update our estimates as necessary.

2.4 Project Resources

The project resources section in a software project plan outlines the resources needed for the project and how they will be allocated. This section includes a list of resources, their roles, and their responsibilities throughout the project's life cycle.

The following resources are necessary:

1. **Project Manager:** Responsible for managing the project schedule, budget, and resources. They will also be responsible for coordinating all project activities and ensuring that the project goals are met.

2. **Lead Developer:** Responsible for managing the development team and ensuring that the software is developed to meet the project requirements.

3. **Front-end Developers:** Responsible for developing the user interface and ensuring that it is intuitive and user-friendly.

4. **Back-end Developers:** Responsible for developing the backend architecture and ensuring that the application is scalable and can handle large amounts of data.

In addition to these core team members, the following resources may be necessary:

1. **Quality Assurance (QA) Analyst:** Responsible for testing the software and ensuring that it meets the project requirements.

2. **UX/UI Designer:** Responsible for designing the user experience and interface to ensure that it is visually appealing and user-friendly.

3. **Technical Writer:** Responsible for creating documentation and user manuals for the software.

4. Additional software developers or engineers as needed.

In addition to Human Resources, the following Technological Resources may be necessary:

1. **Development Tools:** This includes IDEs, project management tools, version control systems, and other software development tools.

2. **Hosting Platform:** A reliable server or cloud-based platform to host the application and database.

3. **Database Systems:** An appropriate database management system to store and manage user and transaction data.

4. **Software Libraries and Frameworks:** These will speed up the development process and ensure code quality.

5. **Security Tools:** Security software and services to protect user data and prevent unauthorized access.

These resources will be allocated based on the project schedule and budget, with each team member having defined responsibilities and timelines for their tasks. The Project Manager will be responsible for managing the project resources and ensuring that they are being used effectively to meet project goals and objectives.

3.0 Risk Management

This section states the general/potential risks associated with the project, including their possible consequences, what triggers the risk, and a contingency plan to mitigate impact. Proactively identifying these risks enables us to formulate strategies to avoid, minimize, or respond effectively to these risks, thus ensuring the project stays on track in terms of time, budget, and quality. The ongoing process of risk management is critical for the successful completion of the project.

3.1 Project Risks

1. Requirements change during development
    1.1. Consequence: Additional development time may be needed, affecting time, effort, and cost.
    1.2. Trigger: Changes in market conditions, shareholder, or user needs.
    1.3. Contingency: Establish a process to control changes, maintain communication with stakeholders, and allocate a contingency budget to develop required changes.

2. Technical Limitations or challenges
    2.1. Consequence: Development delays, increased costs, reduced functionality
    2.2. Trigger: Unforeseen technical issues, integration issues, software incompatibilities with specific devices or operating systems.
    2.3. Contingency: Conduct thorough research and prototyping, allocate resources for fallback options.

3. Insufficient resource availability
    3.1. Consequence: Development delays, cost increases, staff is overworked to make up for the lack of resources.
    3.2. Trigger: Budget constraints, unexpected staff issues.
    3.3. Contingency: Plan for resource requirements, maintain a pool of additional unused resources, and allocate budget for resource-related issues.

4. Poor software quality or performance
    4.1. Consequence: Users are not satisfied, increased cost due to bug fixing and reworks. Product reputation negatively affected.
    4.2. Trigger: Inadequate or incomplete testing, rushing development, or incomplete attention to quality assurance.
    4.3. Contingency: Implement thorough quality assurance processes, allocate additional time and resources to testing, and establish a plan to address defects.

5. Project schedule delays
    5.1. Consequence: Increased costs, potential market opportunities loss, and

stakeholder dissatisfaction.

5.2. Trigger: Unforeseen obstacles, resource unavailability, changes in scope.

5.3. Contingency: Project schedule is established realistically. Progress is monitored closely, and communication is maintained with stakeholders throughout the development process.

6. Inaccurate project estimates
   6.1. Consequence: Potential project failure due to insufficient budget or time allocation.
   6.2. Trigger: Incomplete requirements, estimation inaccurate.
   6.3. Contingency: Conduct thorough requirements analysis and allocate a budget and time for uncertainties.

7. Data Security Breaches
   7.1. Consequence: Legal repercussions, loss of user trust, damage to the product's reputation, and financial penalties.
   7.2. Trigger: Hacking attempts, poor security measures, insufficient staff training on data security practices.
   7.3. Contingency: Implement robust data security measures, conduct regular security audits, provide staff training on data security, and prepare a data breach response plan.

8. Ineffective Project Communication
   8.1. Consequence: Misunderstandings, delayed decision-making, decreased team morale, increased project risks.
   8.2. Trigger: Lack of regular communication, unclear communication channels, failure to involve all stakeholders in relevant discussions.
   8.3. Contingency: Develop a clear communication plan, use effective project management tools, and ensure regular meetings are held with all stakeholders.

3.2 Risk Table

Risk ID	Risk	Probability	Impact	RIS Pointer
PO-1	Requirements change during development	35%	2	RIS 1
PO-2	Technical Limitations or challenges	35%	2	RIS 2
PO-3	Insufficient resource	15%	2	RIS 3

	availability			
PO-4	Poor software quality or performance	15%	2	RIS 4
PO-5	Project schedule delays	45%	3	RIS 5
PO-6	Inaccurate project estimates	10%	1	RIS 6
PO-7	Data Security Breaches	5%	1	RIS 7
PO-8	Ineffective Project Communication	15%	3	RIS 8

Impact Scale: 4= Negligible, 3 = Marginal, 2= Critical, 1 = Catastrophic

3.3 Overview of Risk Mitigation, Monitoring, Management

RM3 approach for this project includes risk identification, analysis, mitigation, monitoring, and managing.

1. Risk Identification
   1.1. Team will regularly search for and identify potential risks through the project development process. Methods for identification include team brainstorming meetings, relevant historical data analysis, and consultations.

2. Risk Analysis
   2.1. After a risk has been identified, the risk will be analyzed to determine both probability and impact. This helps prioritize risks and to develop appropriate mitigation plans.

3. Risk Mitigation
   3.1. After risks have been identified and analyzed, the team can create a mitigation plan to reduce probability of occurrence and project impact. Strategies include contingency planning and preventive actions.

4. Risk Monitoring
   4.1. The team will continuously monitor and track risks that have been identified to ensure mitigation plans are appropriate. New risks may be identified, allowing them to be addressed promptly. Routine risk assessments conducted, with the risk table updated accordingly.

5. Risk Management
    5.1. The team will establish processes for risk management that involve risk reviewal meetings, risk reporting, and open communication with stakeholders regarding potential risks and their impact and probability. This ensures all team members and stakeholders are aware of the potential risks and their respective mitigation plans.

Risk Information Sheets:

Risk: Requirements Change During Development
Priority: 2
Risk Factor: Changes in stakeholder or user needs
Probability: 35%
Impact: Additional development time needed, affecting time, effort, and cost.
Monitoring Approach: Regular communication and consultation with stakeholders.
Contingency Plan: Establish a process to control changes and maintain a contingency budget for necessary changes.
Estimated Resources: 2-4 additional person-weeks for integrating changes and potential rework.

Risk: Technical Limitations or Challenges
Priority: 3
Risk Factor: Unforeseen technical issues.
Probability: 35%
Impact: Development delays, increased costs, reduced functionality.
Monitoring Approach: Regular technical reviews and prototyping.
Contingency Plan: Allocate resources for fallback options, investigate alternate technologies or methods.
Estimated Resources: 2-3 additional person-months depending on the complexity of the issue.

Risk: Inadequate Resource Availability
Priority: 2
Risk Factor: Budget constraints, unexpected team member issues.
Probability: 15%
Impact: Development delays
Monitoring Approach: Regular monitoring of project timelines and resource usage.
Contingency Plan: Plan for resource requirements, maintain a pool of additional unused resources, properly allocate resources.
Estimated Resources: 1-2 additional person-months to adjust workload

Risk: Poor Software Quality or Performance
Priority: 1
Risk Factor: Inadequate or incomplete testing, rushing development.
Probability: 15%

Impact: User dissatisfaction, increased cost due to reworks and bug-fixing.
Monitoring Approach: Regular quality assurance checks and user testing.
Contingency Plan: Implement thorough quality assurance processes, allocate additional resources to testing.
Estimated Resources: 3-4 additional person-weeks for thorough testing and bug fixes.

Risk: Project Schedule Delays
Priority: 1
Risk Factor: Unforeseen obstacles, resource unavailability, changes in scope.
Probability: 45%
Impact: Increased costs, potential loss of market opportunities.
Monitoring Approach: Regular monitoring of project timeline and milestones.
Contingency Plan: Realistic project scheduling, keeping stakeholders informed.
Estimated Resources: 2-3 additional person-months to manage delays

Risk: Inaccurate Project Estimates
Priority: 1
Risk Factor: Incomplete requirements, estimation inaccuracy.
Probability: 10%
Impact: Potential project failure due to insufficient budget or time allocation.
Monitoring Approach: Regular review and update of project estimates.
Contingency Plan: Conduct thorough requirements analysis, allocate a buffer for uncertainties.
Estimated Resources: 1-2 additional person-weeks to revise and update project plans.

Risk: Data Security Breaches
Priority: 1
Risk Factor: Hacking attempts, poor data security measures.
Probability: 5%
Impact: Legal repercussions, loss of user trust, reputational damage.
Monitoring Approach: Regular security audits, constant monitoring of systems for breaches.
Contingency Plan: Implement robust security measures, prepare a data breach response plan.
Estimated Resources: 3-4 additional person-weeks to manage and recover from a breach.

Risk: Ineffective Project Communication
Priority: 3
Risk Factor: Lack of regular communication, unclear communication channels.
Probability: 15%
Impact: Misunderstandings, delayed decision-making, decreased team morale.
Monitoring Approach: Regular team meetings and status updates.
Contingency Plan: Develop a clear communication plan, use effective project management tools.
Estimated Resources: Minimal, as regular and effective communication should be part of normal project operations.

4.0 Project Schedule

1. Planning
    a. Time and Resource Requirement Estimates
    b. Overview of features, plans for implementation
    c. Scheduling
2. Requirements Analysis
    a. Document Requirements
    b. Use cases
3. Design
    a. Analysis of plan
    b. Diagramming Process
4. Implementation
    a. Implement Features
    b. Test Features
5. Deployment
    a. Maintenance

4.1 Project Task Set

Planning

1. Identify project goals and objectives
2. Conduct stakeholder analysis
3. Develop preliminary scope
4. Establish project timelines/milestones
5. Define budget

Requirements Analysis

1. Gather user requirements
2. Document software requirements
3. Create use cases
4. Develop system specifications
5. Confirm and sign-off requirements with stakeholders

Design

1. Create high-level design of the software
2. Design user interface
3. Design database structure
4. Develop system architecture

Implementation

1. Coding modules

2. Develop user interface
3. Build database
4. Integrate system components
5. Conduct peer reviews
6. Develop test cases
7. Conduct unit testing
8. Conduct integration testing
9. Perform system testing
10. Fix identified bugs and issues

Deployment

1. Prepare deployment environment
2. Migrate system to production
3. Conduct post-deployment checks
4. User training
5. Monitor system performance
6. Provide user support and troubleshooting
7. Implement system updates
8. Regularly review and update system documentation

4.2 Functional Decomposition

Functional decomposition is a method of analysis that dissects a complex process to show its individual elements. For this project, the major functions can be decomposed as follows:

User Management

Function Name	**Description**
User Registration	Allows new users to create an account using their personal details
User Authentication	Validates user credentials and permits access to the system.
User Profile Management	Allows users to view and edit their profile details
User Deactivation	Allows users to deactivate their account

Group Management

Group Creation	Allows users to create a new group by specifying group details and adding members

Group Editing	Allows changes to group's details or member list
Group Deletion	Allows users to delete a group

Expense Management

Expense Creation	Allows users to add new expenses with relevant details
Expense Editing	Enables users to modify the details of an existing expense
Expense Deletion	Allows users to remove an expense from the system
Expense Approval	Implements an approval workflow for reviewing and approving expenses

Balance Tracking

Balance Calculation	Calculates the balance owed by each user based on the recorded expenses
Balance Update	Updates user balances when a new expense is added or an existing one is modified or deleted

Payment Tracking

Payment Record Creation	Allows users to create a record of payment made to settle a balance
Payment Record Editing	Allows users to modify the details of a recorded payment
Payment Record Deletion	Enables users to remove a payment record from the system

Reporting

Standard Reports	Generates predefined reports on user expenses, balances, and payments
Custom Reports	Allows users to generate reports on specified parameters

Report Export	Provides functionality to export reports in various formats

Notification System

Notification Creation	Creates notifications when relevant events occur
Notification Delivery	Delivers notifications to users through their chosen method
Notification Settings	Allows users to customize their notification preferences.

5.0 Staff Organization

5.1 Team Structure

The team structure will be organized in a small structure that is sturdy enough to have every role filled that is necessary. The project has 5 developers on the team, the roles are listed below with their responsibilities:

Project Manager:

- Define the scope and goals of the project.
- Create a project plan and timeline.
- Allocate resources and manage the project budget.
- Communicate with stakeholders and manage expectations.
- Monitor project progress and adjust plans as needed.
- Manage the project team.

Lead Developer:

- Define the software architecture and technology stack.
- Develop and implement software solutions.
- Ensure code quality and maintainability.
- Review and manage code changes made by other developers.
- Provide technical guidance and mentorship to other developers.
- Ensure compliance with software development best practices and standards.
- Collaborate with the project manager and stakeholders to ensure project success.

Front-End Developer(s):

- Develop and maintain the user interface of the application.
- Implement responsive design and accessibility features.
- Ensure cross-browser compatibility and consistent user experience.
- Optimize frontend performance and implement caching strategies.
- Collaborate with backend developers to ensure seamless integration of the frontend and backend systems.

Back-End Developer(s):

- Develop and maintain the backend server and API.
- Implement database design and management.
- Ensure scalability and security of the backend system.
- Optimize server performance and implement caching strategies.
- Collaborate with frontend developers to ensure seamless integration of the frontend and backend systems.

Beyond these roles, the team members agree to work collaboratively on documentation, research, and potential inter-team training on new or unknown technologies.

5.2 Management Reporting and Communication

Effective communication is critical for ensuring that the project is completed on time, within budget, and to the satisfaction of all stakeholders. The following strategies will be employed to facilitate communication and ensure that all parties are kept informed of project progress:

- **Weekly status meetings:** The project team will hold weekly status meetings to discuss progress, issues, and risks. These meetings will be attended by the project manager, the lead developer, and the other members of the development team.

- **Progress reports**: The project manager will provide weekly progress reports to the project sponsor, which will outline the status of the project, any issues that have arisen, and any risks that have been identified.

- **Change management:** Any changes to the project scope, schedule, or budget will be documented and communicated to all stakeholders. The project manager will be responsible for ensuring that any changes are approved by the project sponsor and that they do not adversely affect the project.

- **Issue tracking:** All issues that arise during the course of the project will be tracked and documented in an issue tracking system. The lead developer will be responsible for ensuring that issues are assigned, tracked, and resolved in a timely manner.

- **Risk management**: Risks that are identified during the course of the project will be documented and tracked in a risk register. The project manager will be responsible for ensuring that risks are managed and that appropriate risk mitigation strategies are developed.

- **Stakeholder communication:** The project manager will be responsible for communicating with all stakeholders, including the project sponsor, the development team, and end-users. Communication will be tailored to the needs of each stakeholder group and will include progress reports, issue tracking, and risk management updates.

- **Collaboration tools:** The development team will utilize collaboration tools, such as Microsoft Teams, WhatsApp and JIRA, to facilitate communication and collaboration among team members.

By implementing these strategies, we believe that we can ensure effective communication and collaboration among all stakeholders, and that we can complete the project on time, within budget, and to the satisfaction of all parties involved.

6.0 Tracking and Control Mechanisms

6.1 Quality Assurance and Control

The quality assurance and control process is critical to ensuring that the final product meets the requirements and specifications of the project plan. The quality assurance and control process includes the following activities:

- **Requirement review:** The project manager and lead developer will review the project requirements to ensure that they are clearly defined, measurable, and achievable. The front-end and back-end developers will be involved in the review process to ensure that all requirements are technically feasible.
- **Code review:** The lead developer will conduct a code review to ensure that the code is of high quality, follows best practices, and meets the project's technical requirements. The front-end and back-end developers will also participate in the code review process to ensure that the code is well-structured and maintainable.
- **Testing:** The front-end and back-end developers will be responsible for testing the code to ensure that it meets the project's functional and technical requirements. The lead developer will review and approve the testing plan and ensure that it is comprehensive and covers all aspects of the code.
- **Bug tracking and resolution:** The front-end and back-end developers will be responsible for tracking and resolving any bugs or issues that arise during development. The lead developer will oversee the bug tracking process and ensure that all bugs are resolved in a timely manner.
- **Deployment and maintenance:** The lead developer will oversee the deployment process to ensure that the code is properly deployed and configured for production. The front-end and back-end developers will also be involved in the deployment process to ensure that the code is stable and runs smoothly. Ongoing maintenance will be conducted by the front-end and back-end developers to ensure that the code remains up-to-date and stable.
- **Quality control:** The project manager will ensure that all quality control processes are followed, including regular reporting and tracking of quality metrics. The lead developer will be responsible for ensuring that the code meets the project's technical requirements and adheres to best practices. The front-end and back-end developers will ensure that the code is well-structured, maintainable, and meets the project's functional requirements.

6.2 Change Management and Control

Change management and control are critical aspects of software configuration management (SCM). The primary goal of change management and control is to ensure that changes made to software components, processes, and documentation are tracked, authorized, and implemented in a controlled and systematic manner. Change management and control activities include the following:

- **Change Request Management:** The project manager is responsible for managing the change request process. This includes capturing and logging all change requests, reviewing and analyzing them, and determining the impact of the proposed changes on the project schedule, budget, and quality.

- **Change Control Board (CCB):** The CCB is responsible for evaluating, prioritizing, and approving change requests. The CCB comprises the project manager, lead developer, front-end developer, and back-end developer. The CCB assesses the impact of change requests on the project scope, schedule, budget, and quality and approves or rejects them based on the assessment.

- **Configuration Identification:** Configuration identification involves identifying and labeling software components, processes, and documentation. Each component is given a unique identifier to facilitate tracking and tracing of changes made to it.

- **Version Control:** Version control is the process of managing changes to software components over time. The project manager is responsible for ensuring that version control procedures are in place and adhered to by all team members. Version control enables team members to keep track of changes made to software components, identify the version of a component, and revert to previous versions if necessary.

- **Configuration Control:** Configuration control ensures that only authorized changes are made to software components, processes, and documentation. The project manager is responsible for ensuring that configuration control procedures are in place and adhered to by all team members. Configuration control enables team members to identify unauthorized changes and revert to authorized versions if necessary.

- **Configuration Status Accounting:** Configuration status accounting involves tracking and reporting the status of software components, processes, and documentation. The project manager is responsible for maintaining configuration status accounting records, including the current version, status, and location of each software component.

Overall, change management and control are essential to ensure that the software project plan tool is developed in a controlled and systematic manner. By following a set of standardized procedures, the project team can minimize the risk of introducing errors or bugs into the software and ensure that any changes made are approved and tracked.