Software Requirements Specification Document

Nick Azzouz, Christian Balan, Leonardo Diaz, Xavier Evans, Zane Richards

Table of Contents

Table of Contents **1**
1. Introduction **2**
1.1. Purpose of Document 2
1.2. Project Summary 2
1.3. Background 2
1.4. Project Scope 3
1.5. System Purpose 3
1.5.1. Users 3
1.5.2. Location 4
1.5.3. Responsibilities 4
1.5.4. Need 4
2. Functional Requirements **5**
2.1. High Priority 5
2.2. Medium Priority 5
2.3. Low Priority 5
3. Non-Functional Requirements **6**
3.1. Reliability 6
3.2. Usability 6
3.3. Performance 6
3.4. Security 6
3.5. Supportability 6
3.6. Online user Documentation and Help 6
3.7. Interfaces 7
3.8. Scalability 7
4. Context Model **7**
4.1 Goal Statement 7
4.2 Context Diagram 7
4.3 System Externals 8
5. Use Case Model **9**
5.1 System Use Case Diagram 9
5.2 Use Case Descriptions 9
6. Class Model **14**

1. Introduction

1.1. Purpose of Document

This document is the software requirement specification document for the SmartSplit program. This program will allow users to track and share expenses between each other and allow for simple and easy deliberation between groups when tracking large payments. This document will provide an in depth overview of the functional and non-functional requirements, as well as context, use case, and class models. It will serve as a guide to direct the design and implementation of the final system.

1.2. Project Summary

Project Name: SmartSplit
Documentation: Cristian Balan
System Designer: Leonardo Diaz
Programmer: Zane Richards
Testing: Xavier Evans
Training: Nick Azzouz

1.3. Background

Smart Split is an innovative app that will enhance bill splitting between groups or partners. Humans naturally make mistakes and it is hard to remember every bill you split with a group or a person. Our app will help document the purchases made between you and someone else, as well as calculate the bill so that you won't have to do the math. This app will be available on the web and in mobile applications such as IOS and Android.

We are a group of students determined to leave an impact on the world. We guarantee the app will be beneficial and it will be a part of everyone's day to day purchases. Currently we will only be working on the website but mobile applications will definitely be on our list after we publish our web software. We aim to create future proof code that can be updated easily in the future. As a wise man once said, "Any fool can write code that a computer can understand but intelligent programmers write code that humans can understand."

1.4. Project Scope

The scope of SmartSplit is a web-based system that allows tracking and management of group-contributed transactions. The product is intended to be delivered directly to the end-users. Currency management, product advertising, funding accounts and not in the scope of this project.

The project will incorporate a database for member data, group data and transaction history. A web domain and hosting service will be purchased Website security certification, moving of funds between banks, receipt validation are not part of this project.

1.5. System Purpose

1.5.1. Users

The users that will most likely benefit from this program are as follows.

Group Travelers:
People who travel in groups can often find themselves sharing expenses when visiting new places. Whether it’s lodging, food, souvenirs, or other expenses it is not uncommon for people to need to split costs. This system will allow users to have a seamless and easy experience doing so.

Roommates:
Roommates often need to split rent costs as well as manage other expenses like food, internet, etc. The SmartSplit system would allow tracking these payments simple and clear.

Landlord:
Landlords and their tenants could create a group in the SmartSplit application to help keep track of rent. This makes the landlord's job much easier as all of their payments are tracked in one place. If the landlord is responsible for multiple properties, they can create separate groups and have a very organized system within the application.

Everyday users:
Lastly, everyday users could find use in the application when creating small expenses throughout the day, whether it's getting

food, going on a day trip, or anything else, there is a place for the SmartSplit application.

1.5.2. Location

The system will be available on both web and mobile platforms. As long as the user has an internet connection and a device they can access the program from any location.

1.5.3. Responsibilities

The primary responsibilities of the new system:

- Provide users with a reliable online payment tracking tool that can be used for payments and payment requests with involved parties.
- Record group transactions
- Alert users of pending settlements and new group transactions
- Provide monthly statements to users consisting of paid and unpaid balances for each group
- Allow users to add other users to their groups
- Allow users to name their group
- Allow users to write the detail/reason for a transaction/payment request
- Allow users to confirm a cash payment toward a balance
- Allow users to add/delete/update payment methods

1.5.4. Need

This application clearly has a need as described above by showcasing the various users. It allows for a simple and easy method for users to split costs from wherever they are in the world.

2. Functional Requirements

2.1. High Priority

1. The system shall allow users to create an expense for themselves or as a group. This will improve user experience by giving individuals within a group the ability to pay and request expenses as needed.

2. The system shall provide balance tracking with accurate payment transaction data. This will improve user experience by displaying the most up-to-date balances within a group which any member can view.

3. The system shall allow a user to decide which payment method they use and which payment methods they would like to save to their account. This will improve the amount of options the user has and will make paying for transactions more convenient.

4. The system shall allow users to create an account locked behind a password to allow for secure transactions.

2.2. Medium Priority

1. The system shall allow users to create usernames to identify themselves. This will help to know who is who and who is paying what.

2. The system shall allow users to insert a profile picture in their bio page. This is just optional whether the user wants to upload their profile picture or not.

3. The system shall allow the user to customize their bio page. The user can add an “about me” section in their page. They also add a cover photo and picture.

2.3. Low Priority

1. The system shall translate web pages into the languages of the countries where the purchases are made. This will improve customer service and reduce the number of support calls from foreign customers.

2. The system shall allow the user to make the fonts in the website larger. This will help people who find it hard to read small letters, especially on a computer screen.

3. The system shall allow users to input their birthday so that the system can wish them a happy birthday on their special day. This will make users feel like valuable consumers to our product.

3. Non-Functional Requirements

3.1. Reliability

- The system shall be completely operational as long as there is no high traffic on the website.

- The system will schedule maintenance updates at night when everyone is most likely sleeping.
- Customer service will be there to help resolve any issues the users may have.

3.2. Usability

- The system will split bills between two or more parties. The user will have the option to input their purchase on the web.
- The system will maintain a clean UI to allow for a simple user experience.

3.3. Performance

- The system should be able to handle simultaneous connections among users ranging from 2 to 100,000.
- The web application will update to reflect the information in the database in a timely manner. Not more than 30 seconds.
- On demand to-date reports generated by the system should take no more than 30 seconds to be available for download.

3.4. Security

- The system's user-facing services such as the user page shall only be accessible via password for each respective user account.
- Transactional information shall only be sent to the involved parties via email or download from the web-app.

3.5. Supportability

- The system should be able to accommodate new payment methods as they arise and are approved.
- The system's web-app must be operable on browsers such as: Google Chrome, FireFox, and Safari.

3.6. Online user Documentation and Help

- The web-app shall provide an optional tutorial for first-time users who have created an account.
    - The webpages and system services will have limited functionality until this tutorial is complete.
- The tutorial will be condensed into a step-by-step startup guide that users may navigate to from any page's header or side bar.

3.7. Interfaces

The system must interface with:

- The SQLite database for group and transactional information
- The financial records and transactions database (bank transactions)

- The language translation tool

3.8. Scalability

- The servers should provide adequate resources for the amount of user accounts and simultaneous usage.
  - No less than 30% of headroom for memory and storage resources.

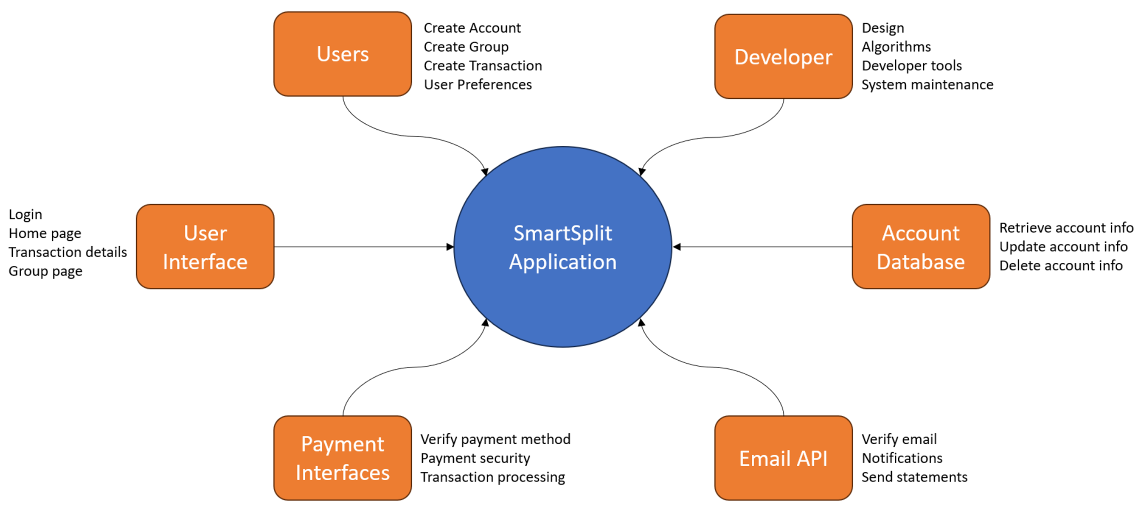
4. Context Model

4.1 Goal Statement

The goal is to allow users to easily split and track their transactions to cover the costs of whatever good or service they purchase by:

- Allowing wide support for different payment methods and bank accounts.
- Having a fast and reliable connection from the server side.
- **Quickly** and **accurately** reflecting changes on the user-facing side (web-page).
- Generating periodic statements as well as on-demand reports for a specified range.

4.2 Context Diagram



[Link to high resolution image]

4.3 System Externals

Users:

- Any user of the system, who will have access to the main functionalities such as creating an account, transaction, group or any more of the uses the program offers.

Developer:

- Developers include any member of the software development team, who may need special access to the systems within the application to mitigate bugs or issues that arise.

Account Database:

- The account database is the back-end database that will manage all the users within the SmartSplit system. It will allow for the functionalities to retrieve, update, or delete any user's information upon the user's request.

Email API:

- The email API is what the system uses to allow for notifications to be sent to users as well as validate their email address during account creation.

Payment Interfaces:

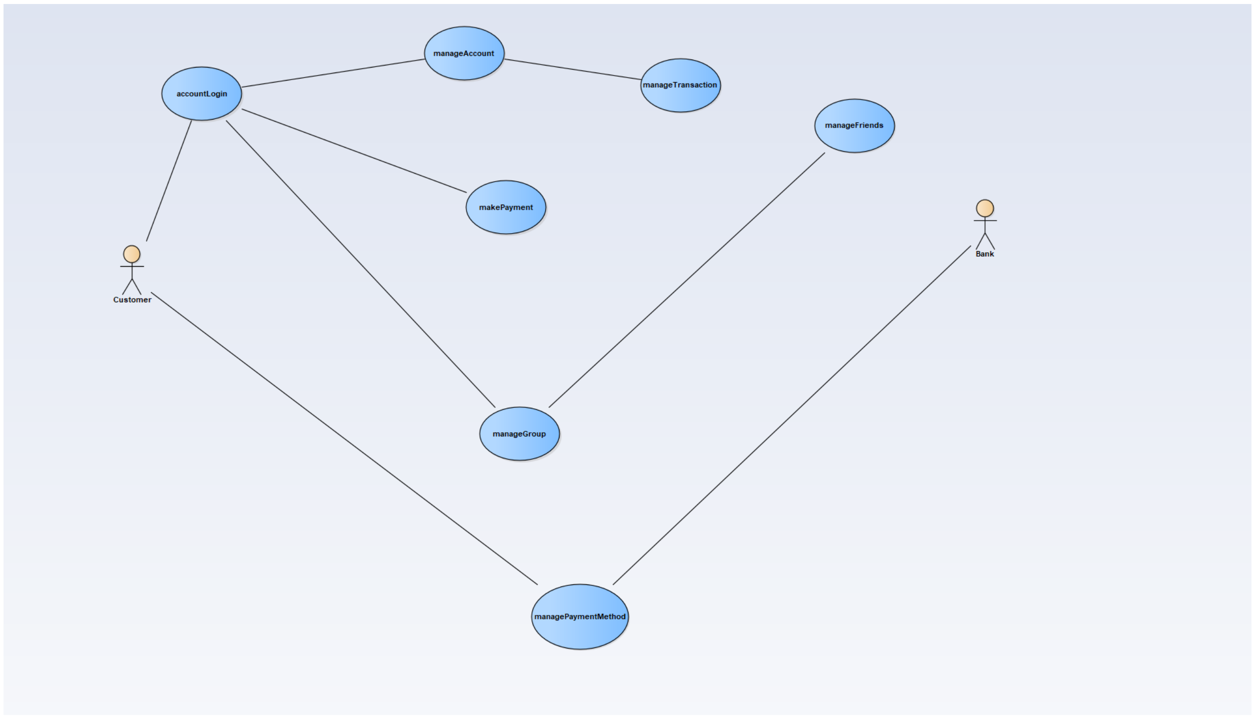
- This interface manages how users will be able to make payments towards transactions. It is how users connect their payment method and it also provides security to said payments

User Interface:

- The UI that the user interacts with when using the system. Versions for both mobile and web to allow for easy use anywhere.

5. Use Case Model

5.1 System Use Case Diagram



Higher Quality: https://imgur.com/a/piBWBpt

5.2 Use Case Descriptions

Use Case Name:	**Create Account**
Participating actor:	Customer
Entry condition:	Valid email
Exit condition:	Customer has account
Event flow:	1. Customer selects create account option 2. Customer enters a valid email 3. Customer enters a sufficient password 4. Customer completes recaptcha 5. Customer receives confirmation via email

Use Case Name:	**Account Login**
Participating actor:	Customer
Entry condition:	The customer is registered.
Exit condition:	The customer can now obtain data and perform functions
Event flow:	1. The use case starts when a user indicates that he wants to login. 2. The system requests the username and password. 3. The user enters his/her username and password. 4. The system verifies the username and password against all registered users. 5. The system starts a login session and displays the home page.

Use Case Name:	**Manage account**
Participating actor:	Customer
Entry condition:	The customer is registered.
Exit condition:	The customer can now obtain data and perform functions.
Event flow:	1. The use case starts when a user indicates that he wants to login. 2. The system requests the username and password. 3. The user enters his/her username and password. 4. The system verifies the username and password against all registered users. 5. The system starts a login session and displays the home page.

Use Case Name:	**Manage payment method**
Participating actor:	Customer/ Bank
Entry condition:	The customer is logged in and has valid payment information.

Exit condition:	The customer’s account will now have the desired payment methods set to their account.
Event flow:	1. The use case starts when a customer indicates that he/she wants to add, modify, or delete payment method(s). 2. The system displays all current payment methods associated with the customer’s account. 3. The customer adds/updates/deletes payment method(s). 4. The system stores any change to the customer’s payment method(s).

Use Case Name:	**Manage transaction**
Participating actor:	Customer
Entry condition:	Transaction has been created between 2 or more customers.
Exit condition:	Any changes to the transaction that the customer makes is applied and saved.
Event flow:	1. Customer clicks the manage transaction option 2. Customer makes changes to transaction (Cost, splitting method, etc.) 3. Customer saves changes made to transaction.

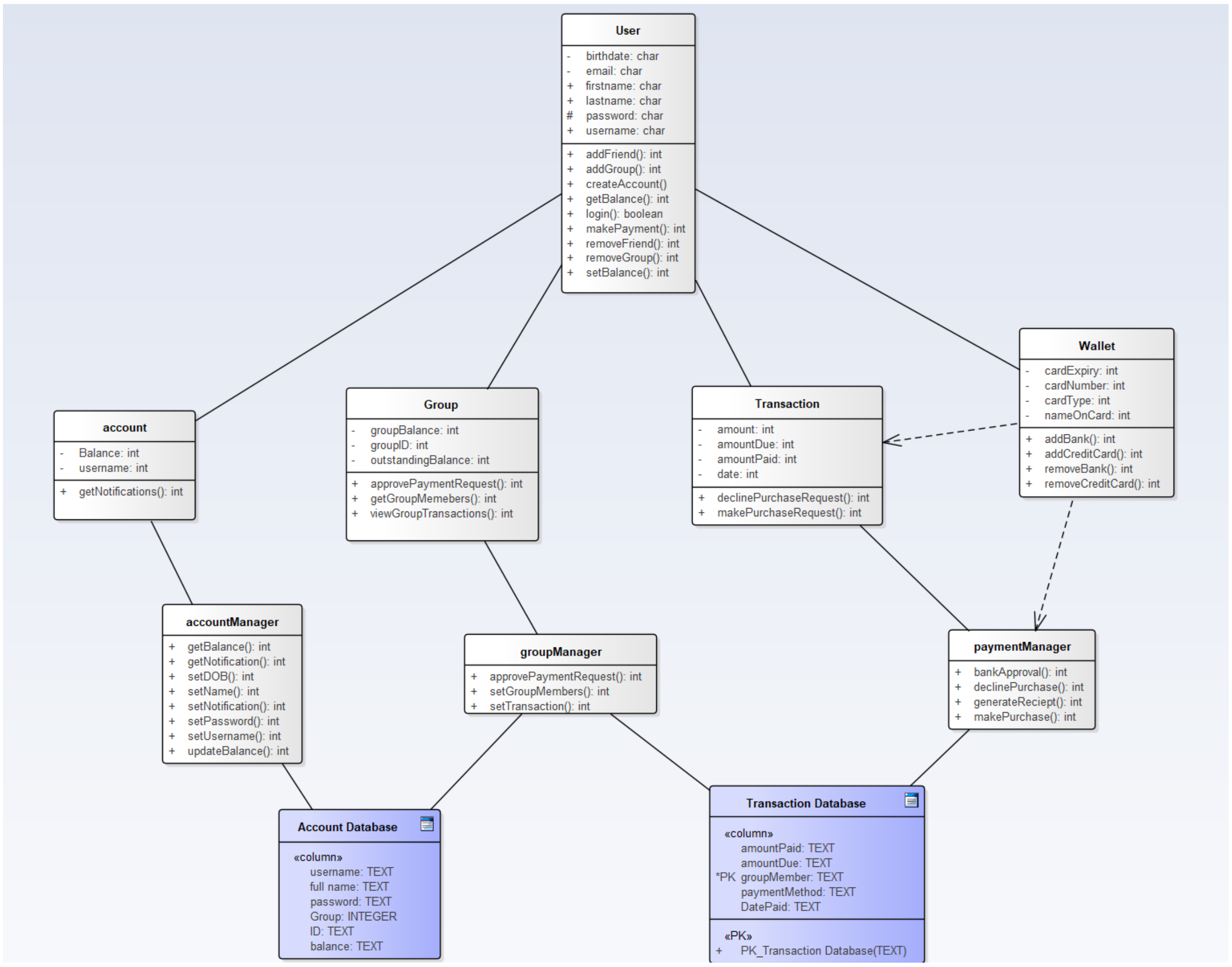
Use Case Name:	**Make payment**
Participating actor:	Customer/ Bank
Entry condition:	Transaction has been created between 2 or more customers.
Exit condition:	The customer’s portion of a transaction is payed via their defined payment method
Event flow:	1. Customer selects transaction 2. Customer confirms the amount owed and pays 3. Money is taken via customers defined payment method 4. Transaction updates that the customer has paid

	their share.

Use Case Name:	**Manage group**
Participating actor:	Customer/ Users
Entry condition:	Create a group with friends
Exit condition:	Group successfully gets created
Event flow:	1. Select users from the friends list 2. Create a group 3. Create a name for the group 4. Group successfully gets created

Use Case Name:	**Manage friends**
Participating actor:	Customer/User
Entry condition:	Add a friend or delete a friend
Exit condition:	Friends list is updated
Event flow:	1. User will look up a friend by their Username 2. User will have the option to add a friend or delete a friend 3. Friend’s list will be updated

6. Class Model



High quality link: https://imgur.com/a/w2sKPxb