COMP 354
Requirements for the project 354TheStars

Group 5

September 30, 2019

Table 1: Group

Name	ID Number	Email
Morteza Ahmadi	40038235	morinob93@gmail.com
Mohd Tanvir	40014010	mohatanvir@hotmail.com
Arunraj Adlee	40059206	arunraj.adlee@hotmail.com
Dina Sadirmekova	26321755	dina.sadirmekova@gmail.com
Saima Syed	40044790	saima.syedb@gmail.com
Mehdi Skouri Saidi	40057700	mehdi879@hotmail.com
Trevor Lall	40044047	trevorlall95@gmail.com
Stefan John Bosco	40057206	johnboscostefan@gmail.com
Timothy Rodriguez	40075447	timmy_258@hotmail.com
Lyonel Zamora	27385986	yonelz516@gmail.com
Radhep Sabapathipillai	40033092	Radhep.Saba@gmail.com
Miguel Jimenez	40022302	migueleduardo298@hotmail.com

Table 2: Revision history

Version	Date	Changes
1.0	29th September 2018	completed requirements

Contents

List of Figures

List of Tables

1 Document Purpose

The purpose of this document is to define requirements for the web application 354TheStars. This document may thus be used to orient the development of the application. It seeks to understand the requirements of the problem, formulate the necessary functions and properties needed to answer this problem and its requirements, and then test these functions against the requirements. Hence, it may be used by our users to specify the problem and its requirements, by the developers to understand what functions their system must implement, and what to test their system against. The primary audience for this document are the stakeholders of the project and the development team of the system, as well the project testers for fine-tuning their testing strategy.

2 Project description

2.1 System Context

Our system will mainly involve the user interacting with a web application interface. The user shall create an account to be able to buy or sell products, however user does not require to have an account for browsing the website. The user information will be stored in a remote database server. Users will not be able to create an account or login by using any social media platforms. It will discussed in more details in future version of this documentation.

Figure 1: System Context

2.2 Project Scope

354TheStars is planning to provide a worldwide platform, where various size of companies as well as individuals can sell or buy products. All of the available products shall be eligible for trading.

2.3 Business Goals

- **Compete with existing solutions**

  We are not the sole developers of e-commerce applications by far. As such, if we want to attract users to our platform, we must be able to compete on the same playing field as these current platforms. It is because of that reason, we must, at the very least, provide a secure, easy to use, and performant platform.

- **Reduce service fee** We are aiming to empower businesses, in order to do so it is necessary to provide a great service and competitive service fee.

2.4 Personas

Admin

Our main actor is the user. All use cases are triggered by the user, as their requests that directly cause the bank system or our system to take action. All identification needed to access the bank system will be provided by the user. Using this information, the system will be able to answer queries made by the user as described in the use cases.

Buyer

Our sole other actor is the bank(s) system. Each bank system provides an API for accessing its bank account data. Our system will pull this data directly from the bank systems (who act as secondary actors), using identification as given by the user for the bank's authentication system. In other words, our system will merely act as a middleman between the user demanding information in a specific format, and the banks holding that information in an inconvenient format.

Seller

Our sole other actor is the bank(s) system. Each bank system provides an API for accessing its bank account data. Our system will pull this data directly from the bank systems (who act as secondary actors), using identification as given by the user for the bank's authentication system. In other words, our system will merely act as a middleman between the user demanding information in a specific format, and the banks holding that information in an inconvenient format.

Paypal

Our sole other actor is the bank(s) system. Each bank system provides an API for accessing its bank account data. Our system will pull this data directly from the bank systems (who act as secondary actors), using identification as given by the user for the bank's authentication system. In other words, our system will merely act as a middleman between the user demanding information in a specific format, and the banks holding that information in an inconvenient format.

Guest User

Our sole other actor is the bank(s) system. Each bank system provides an API for accessing its bank account data. Our system will pull this data directly from the bank systems (who act as secondary actors), using identification as given by the user for the bank's authentication system. In other words, our system will merely act as a middleman between the user demanding information in a specific format, and the banks holding that information in an inconvenient format.

2.5 Functional Requirements

This section will cover the functional requirements associated with the 354TheStars web application. These are the software capabilities that must be present in order for the user to carry out the services provided by the app or to execute the use cases.

- **User account creation:** The system allows the creation of user accounts, with information (user-name, email, password, first name, and last name) provided by the user.
- **Sell:** The system shall grant a user access only to a properly authenticated user. If a user does not enter credentials, the system shall notify the user of the failure to authenticate.
- **Buy:** The system shall require proper authentication to modify and/or delete a user account.
- **Search:** The system permits a logged-in user to add, manipulate, or remove bank accounts that are connected to the user account.
- **Admin panel:** The system permits a logged-in user to access the details of transactions associated with all bank accounts that were added to the user account. The system obtains these transactions through the banking system's API. The system allows different display formats, such as 'sorted by date' or 'sorted by type' or 'sorted by bank account.
- **Purchase process:** The system permits the categorisation of transactions, a property by which the transactions may be sorted.

3 User-Stories

User-Story 1 (Admin)
As an admin:

- I should be able to create, manage and delete users account
- I should have the options to view personal information of the users including:
  - Username
  - Email
  - Address
  - Phone Number
- I should be able to view and manage every single product that is posted in the website.
- The application should provide quality user experience when accessing and viewing account information of a user

User-Story 2 (Buyer)
As a buyer:

- I should be able to sign up using:
  - Username
  - Email
  - Address
  - Phone Number
  - Password
- I should be to edit my personal information.
- I should be able to see a product's detail including:
  - Image
  - Description
  - Price
  - Reviews
- I should have the ability to search for a product by:
  - Name
  - Category
- I should be able to filter and sort the search results based on:
  - Price

  - Location
  - Reviews
- I should be able to pay with my Paypal account.
- I should get a confirmation email showing a successful purchase.
- I should be able to add up to three shipping addresses.
- I should be able to add a review to a product after fifteen days of purchasing the product.
- I should be able to see my orders history.

User-Story 3 (Seller)
As a seller:

- I should be able to sign up using:
  - Email
  - Username
  - Address
  - Phone number
  - Password
- I should be able to edit my personal info including:
  - Email
  - Address
  - Phone number
  - Password
- I should be able to add a product to sell.
- I should be able to edit a product's info including:
  - Price
  - Image
  - Description
  - Stock's count
- I should have the same ability as a buyer.

User-Story 4 (Guest User)
As a guest user:

- I should be able to browse through products.
- I should be able to search for a product by using:
  - Name
  - Category
- I should be able to filter and sort the search results based on:
  - Price
  - Location
  - Reviews
- I should be able to see a product's detail including:
  - Image
  - Description
  - Price
  - Reviews

4 Glossary of Domain Concepts

Table 3: Glossary of Domain Concepts

Expression	Definition
User	The person that is using the application and the main provider of requests to the system.
User Account	A data object containing user information. It also contains the various bank accounts that a user may have linked to the system.
Bank Account	A data object containing transactions linked with a specific bank account in a bank institution. One user account may have more than one bank accounts.
Transaction	Any kind of money exchange associated with a bank account.
Transfer	A type of transaction that occurs between two parties.
Deposit	A type of transaction where the owner puts money in his own bank account.
Withdrawal	A type of transaction where the owner of the bank account removes money from his balance.
Database	A local or online container which holds data in an organised, efficient manner.
Server	a computer that is accessible on a network, on which a database and/or system may be hosted. The bank institutions' databases will be hosted on here.
Object-Oriented Programming	A programming paradigm which separates entities into objects, and uses the concept of inheritance of properties, polymorphism of objects, encapsulation of objects. We use this paradigm for its maintainability and structural benefits.
MVC - Model-View-Controller Architecture	An architectural pattern which strictly separates components into the model (manages the data and logic), the view (output of the model), and the controller (handling input and passing it to the model or view).
Interface	A component of a system by which other entities (be it humans or other systems) may engage in an exchange of data with the system in question.
API - Application Programming Interface	A protocol or set of functions which serve as a method of communication to a software system. It is a type of interface, and the one by which our system will communicate with the banking institutions' databases.
DAO - Data access object	An object that provides an abstract interface to some type of database or other persistence mechanism.

5 Use Cases

5.1 Overview

Use cases 1 through 4 deal with the user manipulating his user and bank accounts. Use cases 5 through 8 deal with the user viewing the data in different formats. Use case 9 deals with the user manipulating the transactions categories.

NB: Note that the alternate flows are described in the same order as the exception. That is, for example, alternate flow 2 specifies the flow of the application if exception 2 is met.

Below is figure 2 which represents our use case diagram. Following this diagram, we list our uses cases

Figure 2: Use Case Diagram

Table 4: Use Case 1 - Create User Account

Action	**Create User Account**
Case ID	01
Summary	User gives information about a new user account to the system, which validates it then creates the account.
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants fast and easy account creation and a comprehensible display of the account requirements. 2. Company: Wants user interests to be fulfilled, validation of input, communication with the local account database as well as failure tolerance when dealing with database manipulation.
Pre-Conditions	User has launched the application.
Success Guarantee	Account successfully saved in local account database, with account details as specified by the user.
Main Success Flow	1. User clicks on the sign-up menu, enters a username, first name, last name, email, phone number, and address and password. 2. System validates user account details, then creates new account. 3. System moves to login screen.
Exceptions	1. Another account with the same username already exists. 2. User specified is not created if the business rules for account creation are not met.
Alternate Flows	1. The System will notify the user the his username is already used by an existing user account, and return to the account creation menu. 2. The System will notify the user of unmet account requirement, then return to the account creation menu.
Priority	High
Traces to Test Cases	Functional test case 1

Table 5: Use Case 2 - Delete User Account

Action	**Delete User Account**
Case ID	02
Summary	User deletes his user account, removing all bank accounts and information associated with that account.
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants easy navigation and secure account deletion, no risk of accidental deletion. 2. Company: Wants user interests to be fulfilled and clean deletion of account and associated data in the database.
Pre-Conditions	User has logged in the user account he wants to delete.
Success Guarantee	user account is successfully deleted, all associated bank information is deleted, and user is returned to the home menu.
Main Success Flow	1. User enters the user account details view, then selects 'delete account'. 2. System brings up a confirmation menu to ensure that this selection was not accidentally entered. 3. User confirms his choice. 4. System successfully deletes the user account as well as all associated bank account and transactions in the local database, then brings the user back to the login/create user menu.
Exceptions	1. User enters incorrect password and is denied ability to delete user account.
Alternate Flows	1. Incorrect password error message will be displayed, the user account will not be deleted, and the user will be returned to the user account details view.
Priority	Medium
Traces to Test Cases	Functional test case 2

Table 6: Use Case 3 - Add Bank Account to a User Account

Action	**Add Bank Account to a User Account**
Case ID	03
Summary	User gives information about a new bank account, system sends it to the bank for verification then creates necessary entries in the local database once the bank approves the information.
Scope	money and budget management application
Level	user-goal
Actors	**User**, Bank
Stakeholders and Interests	1. User: Wants fast and easy account creation, clear and comprehensible display of bank account requirements. 2. Company: Wants user and bank interests to be fulfilled, wants to prevent erroneous input, wants fast communication with the local account database as well as the bank, wants fault tolerance in case of database conflicts, issues with the bank, or other possible remote database problems. 3. Bank: Wants to satisfy its own customer base, wants correctly formatted account information given to its API, inexpensive and non-redundant communication of bank account data to third-party applications.
Pre-Conditions	User has logged in a user account and is in the account list menu.
Success Guarantee	Bank account successfully saved in local account database, with information corresponding the data validated by the bank.
Main Success Flow	1. User enters his/her bank account ID in the text field for bank account addition. 2. System validates input, and sends it to the bank for verification and connection. 3. Bank validates bank account number, and then responds with the bank account information. 4. System records the valid bank account details securely in its database, and the account list is updated with this new account.
Exceptions	1. An invalid account ID was provided. 2. Account has already been added to the user account. 3. Account was not validated by the bank (could not find account, invalid ID on their side of the validation, etc.)
Alternate Flows	1. The account ID is not sent to the bank servers for validation, instead an error will be displayed requesting a valid bank account ID then the system returns to the account list menu. 2. the account ID is validated but found to have been already associated with this user account; this error is displayed by the system, which then returns to the account list menu. 3. the accound ID is validated and queried to the bank, which

Table 7: Use Case 4 - Remove Bank Account from a User Account

Action	**Remove Bank Account from a User Account**
Case ID	04
Summary	User removes a bank account associated with his user account.
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants easy navigation and secure account deletion. 2. Company: Wants user interests to be fulfilled, wants to ensure clean deletion from the database.
Pre-Conditions	User has logged in the user account whose active association with a bank account is the one the user wants to delete.
Success Guarantee	Bank account is successfully removed from that user account, all associated bank information is deleted from the local database, and user is returned to the account home menu.
Main Success Flow	1. User selects the account he wants to remove, then selects *remove selected account*. 2. System successfully deletes the bank account and its associated transactions in the local database, then updates the account list to display this change.
Exceptions	1. The user clicks *remove selected account* without having selected an account.
Alternate Flows	1. No account is deleted and the system ignores the call, staying in the accounts list menu.
Priority	High
Traces to Test Cases	Functional test case 4

Table 8: Use Case 5 - View Transactions for Specific Bank Account

Action	**View Transactions for Specific Bank Account**
Case ID	05
Summary	User selects specific bank account and views transactions associated with selected bank account
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants quick and convenient viewing of transactions associated with a specific bank account. 2. Company: Wants user interests to be fulfilled.
Pre-Conditions	User has created and logged into a user account, and has added at least one bank account to his/her myMoney account.
Success Guarantee	User can view transactions of specific bank account.
Main Success Flow	1. User selects specific bank account from list of all accounts. 2. System enters the detailed view for that bank account, displaying all transactions for that account.
Exceptions	1. User has no transactions for selected bank account.
Alternate Flows	System displays the detailed view of the bank account nonetheless, but the list of transaction is empty.
Priority	High
Traces to Test Cases	Functional test case 5

Table 9: Use Case 6 - View All Transactions from all Bank Accounts

Action	**View All Transactions from all Bank Accounts**
Case ID	06
Summary	User can view all transactions that have been made from all bank accounts
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants easy and convenient viewing of all transactions among all bank accounts. 2. Company: Wants user interests to be fulfilled.
Pre-Conditions	User has created and logged into a user account.
Success Guarantee	User is able to conveniently view all transactions from all institutions in one display.
Main Success Flow	1. User selects View All Transactions option. 2. System shows all transactions across all accounts in one unified list.
Exceptions	1. User has not added any account. 2. User has no transactions to record.
Alternate Flows	1. System displays an empty list of transactions. 2. System displays an empty list of transactions.
Priority	High
Traces to Test Cases	Functional test case 6

Table 10: Use Case 7 - Update User Account

Action	**Update User Account**
Case ID	07
Summary	User can make changes and update personal information
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants to user account attributes to keep up-to-date records on information such as address and email as well as changing user account password. 2. Company: Wants user interests to be fulfilled and ensure validation of changed user account attributes.
Pre-Conditions	User has created and logged in a user account.
Success Guarantee	User can easily update changes in personal information to guarantee accurate personal records
Main Success Flow	1. User selects *Your profile*. 2. System shows current user account information. 3. User selects the piece of information that requires updating. 4. User enters new information. 5. System updates and saves the new information.
Exceptions	1. Passwords do not match, user information is not updated 2. First name, last name, or password is left empty, user information is not updated. 3. email is an invalid email address.
Alternate Flows	1. Error message is displayed prompting user that his passwords do not match and user account is not updated. 2. Error message is displayed prompting user that he is missing required information and user account is not updated. 3. Error message is displayed prompting user that his email address is invalid nd user account is not updated.
Priority	Medium
Traces to Test Cases	Functional test case 7

Table 11: Use Case 8 - Sort Transactions by Any Attribute

Action	**Sort Transactions by Any Attribute**
Case ID	08
Summary	User can sort transactions by any attribute (date, transaction amount, category, and transaction type.)
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants to view transactions sorted by any one attribute. 2. Company: Wants user interests to be fulfilled.
Pre-Conditions	User has created and logged in a user account.
Success Guarantee	User can sort transactions by selecting one attribute. The default sorting is ascending order by alphabet or value of number. Double-clicking will sort in descending order.
Main Success Flow	1. User enters a transaction list view by choosing to view all bank account transactions or a detailed bank account view. 2. System shows a transaction list. 3. User selects to view sorted transaction by clicking any one attribute. 4. System sorts transactions by the selected attribute in ascending order.
Exceptions	1. User has no transactions in his account.
Alternate Flows	1. The system will display an empty transaction list.
Priority	High
Traces to Test Cases	Functional test case 8

Table 12: Use Case 9 - Categorize Transaction

Action	**Categorize Transaction**
Case ID	09
Summary	User can select a category to represent a transaction.
Description	This use case describes how a user can categorize their transactions with specific labels in order to sort their spending and better manage where most of their money is being spent
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants to label each transaction as a specific type of spending, such as rent, bills, leisure, etc., to better manage and track spending habits. 2. Company: Wants user interests to be fulfilled. Wants to facilitate user's ability to budget and manage money through simple categorization of spending.
Pre-Conditions	User has created and logged in a user account, added at least one bank account, and has made at least one type of transaction.
Success Guarantee	User can quickly and efficiently categorize each transaction.
Main Success Flow	1. User enters a transaction list view by choosing to view all bank account transactions or a detailed bank account view. 2. User selects the desired transaction to categorize. 3. User selects the categorize option. 4. System displays the categories in a drop down menu. 5. User selects preferred category to represent the current transaction, or types in his own category. 6. System changes the category attribute on the chosen transaction to the category that the user selected or wrote in.
Exceptions	1. the user writes in a category name of exceeding length (exceeding as defined by the business rules.)
Alternate Flows	1. The system rejects the call and leaves the category unchanged.
Priority	Medium
Traces to Test Cases	Functional test case 9

Table 13: Use Case 10 - Filter Transactions by Date Range

Action	**Filter Transactions by Date Range**
Case ID	10
Summary	User can view all transactions between two selected dates.
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants to view transactions between two selected dates, to optimize spending habits for select durations. 2. Company: Wants user interests to be fulfilled.
Pre-Conditions	User has created and logged in a user account.
Success Guarantee	User can view all transactions from select bank accounts, or all bank accounts between two specified dates.
Main Success Flow	1. User decides to view all bank accounts or selects a specific bank account. 2. System shows transactions. 3. User selects desired dates to view transactions between the two dates. 4. System shows all transactions between the two specified dates.
Exceptions	1. User selects impossible range of dates.
Alternate Flows	1. no transactions are displayed.
Priority	Low
Traces to Test Cases	10

Table 14: Use Case 11 - Filter Transactions by Category

Action	**Filter Transactions by Category**
Case ID	11
Summary	User can choose a category to filter by, which will result in all transactions which are not in that category to be removed from the view
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants to view only the transactions in a specific category. 2. Company: Wants user interests to be fulfilled.
Pre-Conditions	User has created and logged in an account.
Success Guarantee	Only the transactions in the specified category will be displayed.
Main Success Flow	1. User decides to view all bank accounts or selects a specific bank account. 2. System shows transactions. 3. User clicks in the text field for searching by category, and writes in the category he wishes to filer by. 4. System updates the view to show only the transactions with the specified category.
Exceptions	1. User enters a non-existent category (a category by which no transaction is categorised).
Alternate Flows	1. No transactions are displayed.
Priority	High
Traces to Test Cases	Functional test case 11

Table 15: Use Case 12 - Export Statement as a CSV file

Action	**Export Statement as a CSV file**
Case ID	12
Summary	Export of a CSV statement consisting of all the transactions currently displayed in the application window.
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants to save a CSV file of his transactions somewhere on his machine, for easy external use of his transaction history. 2. Company: Wants user interests to be fulfilled.
Pre-Conditions	The user has created a user account, logged in, connected at least one bank account and has reached a transactions view either through *view all transactions* view, either through *account details* view.
Success Guarantee	A CSV file has been successfully generated from the transactions currently displayed, and was saved where the user asked for it to be saved.
Main Success Flow	1. User chooses to view all bank account transactions or a detailed bank account view. 2. System shows transactions. 3. User clicks on the button to export to CSV. 4. System displays a menu asking where the user wants this statement to be saved. 5. User selects the directory wherein he wishes to export the statement. 6. System generates the statement and saves it in the desired folder of the user's choosing. It may be blank if the user has no transaction displayed in his view.
Exceptions	1. User enters a non-existent path. 2. User enters an unauthorised path. 3. User enters a valid path, but a file with the same name as the statement to be generated already exists.
Alternate Flows	1. The save button is greyed out, preventing the user from selecting that path and thus requiring a valid path to be entered to move on. 2. The save button is greyed out, preventing the user from selecting that path and thus requiring a valid path to be entered to move on. 3. A confirmation box should ask the user if he wants to overwrite a file if a file already exists in the specified location.
Priority	Low
Traces to Test Cases	Functional test case 12

Table 16: Use Case 13 - Email Statement

Action	**Email Statement**
Case ID	13
Summary	User wants to have a CSV statement consisting of all the transactions currently in the application window sent to his email.
Scope	money and budget management application
Level	user-goal
Actors	**User**
Stakeholders and Interests	1. User: Wants to receive a CSV statement of all his transactions, for easy exportation of his transaction history. 2. Company: Wants user interests to be fulfilled.
Pre-Conditions	A user created a user accountl, logged in, connected at least one bank account and has reached a transactions view either through *view all transactions* view, either through *account details* view.
Success Guarantee	A CSV file has been successfully generated from the transactions currently displayed, and was saved where the user asked for it to be saved.
Main Success Flow	1. User chooses to view all bank account transactions or a detailed bank account view. 2. System shows transactions. 3. User clicks on the button to email CSV. 4. System generates the statement and sends it to the user's email.
Exceptions	1. User hasn't set his email in his user account details.
Alternate Flows	1. the System notifies the user that his email address is empty and doesn't send the CSV file.
Priority	Low
Traces to Test Cases	Functional test case 13

6 Reference

- User information: As our user and use-cases was based on feedback provided by our developers, our references lie mainly within our own team.
- Craig Larman - Applying UML and Patterns
- Greg Butler's course COMP 354 content
- MIT Curricular Information System Software Requirements Document
- Carnegie Mellon Business Goals
- Use-Case: Oracle