# 

# 

# 

# 

# CSC 431 FundsBook Software Requirements Specification (SRS)

**Team 16**

|  |  |
| --- | --- |
| Luis Diaz | Developer |
| Julio Ojalvo | Developer |
| Temuulen Ganbold | Developer |

# Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Change Comments |
| 1 | 2/11/20 | Luis Diaz,  Temuulen Ganbold,  Julio Ojalvo | * Added functional and non-functional system requirements |
| 2 | 3/9/21 | Luis Diaz,  Temuulen Ganbold,  Julio Ojalvo | * Corrections and additions made to functional and non-functional system requirements. * Added Use Case Models * Added System constraints and evolutionary requirements * Added Table of Tables * Added Table of Figures |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[1.](#_3dy6vkm) System Requirements 6

[1.1](#_1t3h5sf) Functional Requirements 6

[1.2](#_2s8eyo1) Non-Functional Requirements 11

[2.](#_3rdcrjn) System Constraints 12

[2.1](#_26in1rg) Tool Constraints 12

[2.2](#_35nkun2) Language Constraints 13

[2.3](#_44sinio) Platform Constraints 13

[2.4](#_z337ya) Hardware Constraints 13

[2.5](#_1y810tw) Network Constraints 14

[2.6](#_2xcytpi) Deployment Constraints 14

[2.7](#_3whwml4) Transition & Support Constraints 14

[2.8](#_qsh70q) Budget & Schedule Constraints 14

[2.9](#_1pxezwc) Miscellaneous Constraints 15

[3.](#_2p2csry) Requirements Modeling 16

[4.](#_3o7alnk) Evolutionary Requirements 20

[4.1](#_32hioqz) Non-Functional Requirements 20

# Table of Tables

1. System Requirements 6

1.1.1 Profile Creation 6

1.1.2 Profile Editing 6

1.1.3 Personal Budget 7

1.1.4 Auxiliary Budgets 7

1.1.5 Followers 8

1.1.6 Friends 8

1.1.7 Close Friends 9

1.1.8 Searching 9

1.1.9 Achievements 10

1.1.10 Notifications 10

1.2.1 Information Encryption 11

1.2.2 Privacy 11

1.2.3 Bank Account 11

1.2.4 Balance 11

1.2.5 Database Management System (DBMS) 12

1.2.6 Suspicious Activity Detector 12

2. System Constraints 12

2.1.1 Environment 12

2.2.1 Java 13

2.2.2 Swift 13

2.3.1 Android 13

2.3.2 IOS 13

2.4.1 Equipment 13

2.5.1 Connectivity 14

2.6.1 Publishing 14

2.7.1 Transitionary Requirements 14

2.7.2 Continued Maintenance 14

2.8.1 Time Management 15

2.9.1 Communication 15

4. Evolutionary Requirements 20

4.1.1 Data logging

# Table of Figures

3. Requirements Modeling

[3.1.1 Profile login 1](#_147n2zr)6

3.1.2 Profile editing 16

3.1.3 Budget management 17

3.1.4 Auxiliary budget 17

3.1.4 Socialization 18

### 1. System Requirements

#### 1.1 Functional Requirements

*< List all functional requirements in the following example format >*

##### 1.1.1 Profile creation

|  |  |
| --- | --- |
| Title | Creating a Profile |
| Description | Users will create their personal profile that others will be able to see and where they can look through their budget plans and badges |
| Priority | 0 |
| Precondition(s) | The app must be downloaded from the app store |
| Basic Flow | * Upon entering the app, the user is prompted with a screen to choose what language they want to see the app in * After this, the user is prompted with a page to either sign in to an existing one or create a profile * If they choose to sign in, they will be prompted with an email or username and password box and will be entered into their profile if the credentials match a profile on record * If the user has forgotten their password, an option will be given to email them a link to reset it * If the user is new, they will create an account and choose an email and username that hasn’t been taken as well as a password with at least 8 characters, a lowercase and uppercase letter, and one special character * On the last step of creating the profile, users will receive a confirmation email and when the link in the email is clicked, they will be allowed into the app |
| Postconditions(s) | The user can now edit their profile and use the major functionalities of the app |
| Use Case Diagram | 3.1.1 |

##### 1.1.2 Profile editing

|  |  |
| --- | --- |
| Title | Editing a Profile |
| Description | Users can edit their profile by adding budget plans and badges to be displayed for others as well as a profile picture and banner behind the profile picture |
| Priority | 0 |
| Precondition(s) | A profile must be created and logged into |
| Basic Flow | * Once the user creates a profile, they will be presented with a blank one that they can edit * A blank profile picture when clicked prompts the user to add one of their own, same as the banner behind it * A button will prompt the user to click on it and create their first budget plan * A button will allow the user to look through all their badges and add up to 7 to their page for others to see |
| Postconditions(s) | Once the user makes an edit, it will be updated so everyone else can see their changes when they go into that person’s profile |
| Use Case Diagram | 3.1.2 |

##### 1.1.3 Personal Budget

|  |  |
| --- | --- |
| Title | Creating a personal budget |
| Description | Ability to create a personal budget plan and put money into certain savings accounts based on the plan created by the user. |
| Priority | 0 |
| Precondition(s) | The UI of the app must be functional and the user must have a profile |
| Basic Flow | * User clicks a button to enter into a screen for editing budgets * User has a box labeled “category” to type in the name of one of the aspects of the plan e.g. car payments, cell phone bill, credit cards * Under each category, the user can input what percent of the total budget is going towards it * User can choose what color that category will be represented in * A pie chart will be shown to the side to demonstrate the totals of each category in the color chosen * If the user inputs a percent that brings the total over 100%, a warning is given and the number is removed * Once, the user is done and clicks the finish button they are given a notification if their budget plan does not reach 100% total but they are allowed to continue and that remaining money will be left untouched |
| Postconditions(s) | Once the main budget plan is made, auxiliary budget plans are now available to be made either for the use of other users or for personal use at another point in time |
| Use Case Diagram | 3.1.3 |

##### 1.1.4 Auxiliary Budgets

|  |  |
| --- | --- |
| Title | Creating Auxiliary Budget Plans |
| Description | Ability to create secondary budget plans set for different priorities for personal use or for others to follow |
| Priority | 2 |
| Precondition(s) | User must have a profile and have a main budget plan already created |
| Basic Flow | * Once the main budget is completed, another button will be available to enter into the same process to create a budget * This time, the budget plan will have the option of having a unique name to identify it with * Tags can also be added to the plan, similar to hashtags on traditional social media, that helps users more easily find that plan based on similar search terms * Users will have the option to switch an auxiliary budget plan to their main one at any given moment |
| Postconditions(s) | User is able to have multiple budget plans with different set priorities and be able to invest into whichever they please as well as share those budget plans with others |
| Use Case Diagram | 3.1.4 |

##### 1.1.5 Followers

|  |  |
| --- | --- |
| Title | Ability to Follow and use Others’ Budget Plans |
| Description | In the given case that successful millionaires/users decide to interact with this app and be some sort of guidance for other users, users should be able to follow those influencers and financial advisor to be able to use as a reference to meet their budgeting plans |
| Priority | 4 |
| Precondition(s) | Must have a profile to follow from |
| Basic Flow | * Users can click on other’s profiles to see their plans as long as the profile is public * If the profile is private, an option to request to follow them is given and a request is sent to that profile owner where they can accept or deny it * If the profile is public, an option to follow is given which when clicked will go through without any further processing |
| Postconditions(s) | User now receives notifications when the profile they followed makes a change if they choose to be notified of it in the settings |
| Use Case Diagram | 3.1.5 |

##### 1.1.6 Friends

|  |  |
| --- | --- |
| Title | Ability to Add Others as Friends |
| Description | Users should be able to interact with other users, having the option to add other users as “friends” and be able to see how they are doing with their budget plans and in a way give each other motivation to reach those goals. |
| Priority | 4 |
| Precondition(s) | Two people must follow each other in order to become friends |
| Basic Flow | * Once two users have chose to follow each other, they will become friends |
| Postconditions(s) | Once friends, users can privately message each other and choose who will be on their close friends list |
| Use Case Diagram | 3.1.5 |

##### 1.1.7 Close Friends

|  |  |
| --- | --- |
| Title | Ability to designate close friends |
| Description | Users can choose who on their friends list is a close friend |
| Priority | 4 |
| Precondition(s) | User must be friends with someone in order to have the choice of adding them to their own close friends list |
| Basic Flow | * When a user looks at their own profile, there will be a tab to see your close friends * Initially it will say “No close friends yet” and prompt them to click on a button that will take them to a list of their friends * From their they can click on all the friends they would like to add to their close friends list and allow for more detailed customization   + Allow notifications only from close friends   + Allow only close friends to see these budget plans   + Allow only close friends to see all my badges   + Send this message out to only close friends |
| Postconditions(s) | A user that has been added to another’s friends list can see more detailed information when they check their profile |
| Use Case Diagram | 3.1.5 |

##### 1.1.8 Searching

|  |  |
| --- | --- |
| Title | Ability to search for users or budget plans |
| Description | Users will have a search bar they can use to find either a certain type of budget plan or another user’s profile |
| Priority | 1 |
| Precondition(s) | Must have a profile |
| Basic Flow | * One of the main tabs on the app will bring the users to the search bar * On the search bar, users can type a keyword to find a certain type of budget plan e.g. long term investments, saving for an expensive car in a short amount of time, paying off high interest loans or credit cards (tags on budget plans will be used to bring the user to a budget plan most similar to their search) * Users can also search for other users based on their profile name |
| Postconditions(s) | Once a user is found, they can be followed and once a plan is found it can be saved as an auxiliary plan that can be switched over to the main budget plan at any moment |
| Use Case Diagram | 3.1.1 |

##### 1.1.9 Achievements

|  |  |
| --- | --- |
| Title | Achievement system |
| Description | An achievement system that will reward users for making budget plans, using tags in them, and following their own budget plans for long periods of time |
| Priority | 3 |
| Precondition(s) | A main budget plan must be made and a profile to display the achievements from |
| Basic Flow | * Users will receive a small badge on their profile if they fulfill certain achievements such as   + Following a budget plan for one week, one month, one year, and every subsequent year   + Making 5 different budget plans   + Adding 10, 20, and 50 different tags among all the plans made * Once a user sets a budget plan as their main one, a counter will start to keep track of how long since it’s been the main one. That counter will be used to determine when to give the badges out * A counter for total number of budget plans made and tags used will be used for those badges * Total number of followers and profiles followed will be counted as well for their own badges |
| Postconditions(s) | Other users can now see that badge on their profile if the user who earned it chose to display it |
| Use Case Diagram | 3.1.2 |

##### 1.1.10 Notifications

|  |  |
| --- | --- |
| Title | Send notifications to the users |
| Description | User will select what notifications they want to receive if any and they will be sent one whenever what they chose to see occurs |
| Priority | 1 |
| Precondition(s) | Notifications must be enabled when the profile is created and the settings for what else the user wants to be notified about must be adjusted |
| Basic Flow | * Once the app is downloaded and a profile is created, the user will be prompted to allow the app to send them notifications * If they say yes, they will only be notified if a friend or close friend sends them a private message, if someone they follow creates a new budget plan, and when someone follows them or accepts their follow request * They will have notification settings where they can choose what notifications to receive   + Only show notifications from close friends   + Show notifications if a followed profile makes a change to a budget plan   + Only show notifications on activity of friends/close friends, etc. |
| Postconditions(s) | Users will be sent the notifications they chose in the settings |
| Use Case Diagram | 3.1.1 |

#### 1.2 Non-Functional Requirements

*< List all non-functional requirements in the following example format >*

##### 1.2.1 Information Encryption

|  |  |
| --- | --- |
| Title | Information Encryption |
| Description | Since the app deals with sensitive information, passwords and any bank information will be heavily encrypted using hashing functions to make it as close to impossible as we can to being hacked |
| Priority | 0 |
| Applicable FR(s) | Profile creation, Personal budget |

##### 1.2.2 Privacy

|  |  |
| --- | --- |
| Title | Privacy Settings |
| Description | Private profiles will have no available information on them if the user wants to be very careful of others knowing what they’re doing |
| Priority | 0 |
| Applicable FR(s) | Profile creation, personal budget, friends/close friends, achievements |

##### 1.2.3 Bank Account

|  |  |
| --- | --- |
| Title | Linking to Bank Account |
| Description | the application should allow for securely linking a users bank account without any security threats or breaches to prevent hacking and leaks if sensitive information (such as social security number) |
| Priority | 0 |
| Applicable FR(s) | Profile creation, personal budget |

##### 1.2.4 Balance

|  |  |
| --- | --- |
| Title | Money Balance |
| Description | The application should be able to keep track of how much money the user has put into their budget and how much they have spent |
| Priority | 3 |
| Applicable FR(s) | Personal budget |

##### 1.2.5 DBMS

|  |  |
| --- | --- |
| Title | Database System |
| Description | The app should be able to accurately store a users information using hash tables for easy encryption and safety from hackers and to make accessing the information seamless and quick for the user |
| Priority | 2 |
| Applicable FR(s) | Personal budget |

##### 1.2.6 Suspicious activity detector

|  |  |
| --- | --- |
| Title | Detect suspicious activity |
| Description | Programmed to detect unauthorized entry into the account or entry from an unknown location and notify the user. If the user doesn’t assert that they were responsible for setting it off, the account is locked to avoid unauthorized entrance |
| Priority | 2 |
| Applicable FR(s) | Profile creation, notifications |

### 

### 2. System Constraints

#### 2.1 Tool Constraints

*< List all tool constraints in the following example format >*

##### 2.1.1 Environment

|  |  |
| --- | --- |
| Title | Workspace |
| Description | Quiet office/workspace for all members to work on the project |
| Priority | 0 |

##### 

|  |  |
| --- | --- |
| Title | Visual Studio IDE |
| Description | We will work on the code using the Visual Studio IDE and share iterations on it using GitHub |
| Priority | 0 |

#### 2.2 Language Constraints

*< List all language constraints in the following example format >*

##### 2.2.1 Java

|  |  |
| --- | --- |
| Title | Java |
| Description | The primary language used for programming the android version of the app will be Java |
| Priority | 0 |

##### 2.2.2 Swift

|  |  |
| --- | --- |
| Title | Swift |
| Description | The primary language used for programming t he IOS version of the app will be Swift |
| Priority | 0 |

#### 2.3 Platform Constraints

*< List all platform constraints in the following example format >*

##### 2.3.1 Android

|  |  |
| --- | --- |
| Title | Android |
| Description | One version of the app will be programmed and designed to function on an android operating system |
| Priority | 0 |

##### 2.3.2 IOS

|  |  |
| --- | --- |
| Title | IOS |
| Description | The other version of the app will be made to work on an Iphone or IOS operating system |
| Priority | 0 |

#### 

#### 2.4 Hardware Constraints

*< List all hardware constraints in the following example format >*

##### 2.4.1 Equipment

|  |  |
| --- | --- |
| Title | Cross Platform testing |
| Description | Lack of distinct devices for cross platform testing. Devices include: iOS phones, iOS tablets, android phones and android tablets. |
| Priority | 3 |

#### 2.5 Network Constraints

*< List all network constraints in the following example format >*

##### 2.5.1 Connectivity

|  |  |
| --- | --- |
| Title | Connectivity among platforms |
| Description | Although the app will be available to both IOS and Android devices, all must be connected to the same network and function with each other seamlessly |
| Priority | 0 |

#### 2.6 Deployment Constraints

*< List all deployment constraints in the following example format >*

##### 2.6.1 Publishing

|  |  |
| --- | --- |
| Title | Publishing an app |
| Description | Number of tedious steps to deploy an app on two different app stores. |
| Priority | 0 |

#### 2.7 Transition & Support Constraints

*< List all transition & support constraints in the following example format >*

##### 2.7.1 Transitionary Requirements

|  |  |
| --- | --- |
| Title | Transitionary Requirements |
| Description | Once the user makes an account and edits their profile, our service must download and package all the data in a convenient way for the user |
| Priority | 0 |

##### 2.7.2 Continued Maintenance

|  |  |
| --- | --- |
| Title | Continued Maintenance |
| Description | This is a term project, ending all maintenance one the grading period is over or a new team can come in and continue maintenance |
| Priority | 5 |

#### 2.8 Budget & Schedule Constraints

*< List all budget & schedule constraints in the following example format >*

##### 2.8.1 Time Management

|  |  |
| --- | --- |
| Title | Spring 2021 Semester |
| Description | Due to class schedule, we only have until May 1st to complete this project |
| Priority | 0 |

#### 2.9 Miscellaneous Constraints

*< List all miscellaneous constraints in the following example format >*

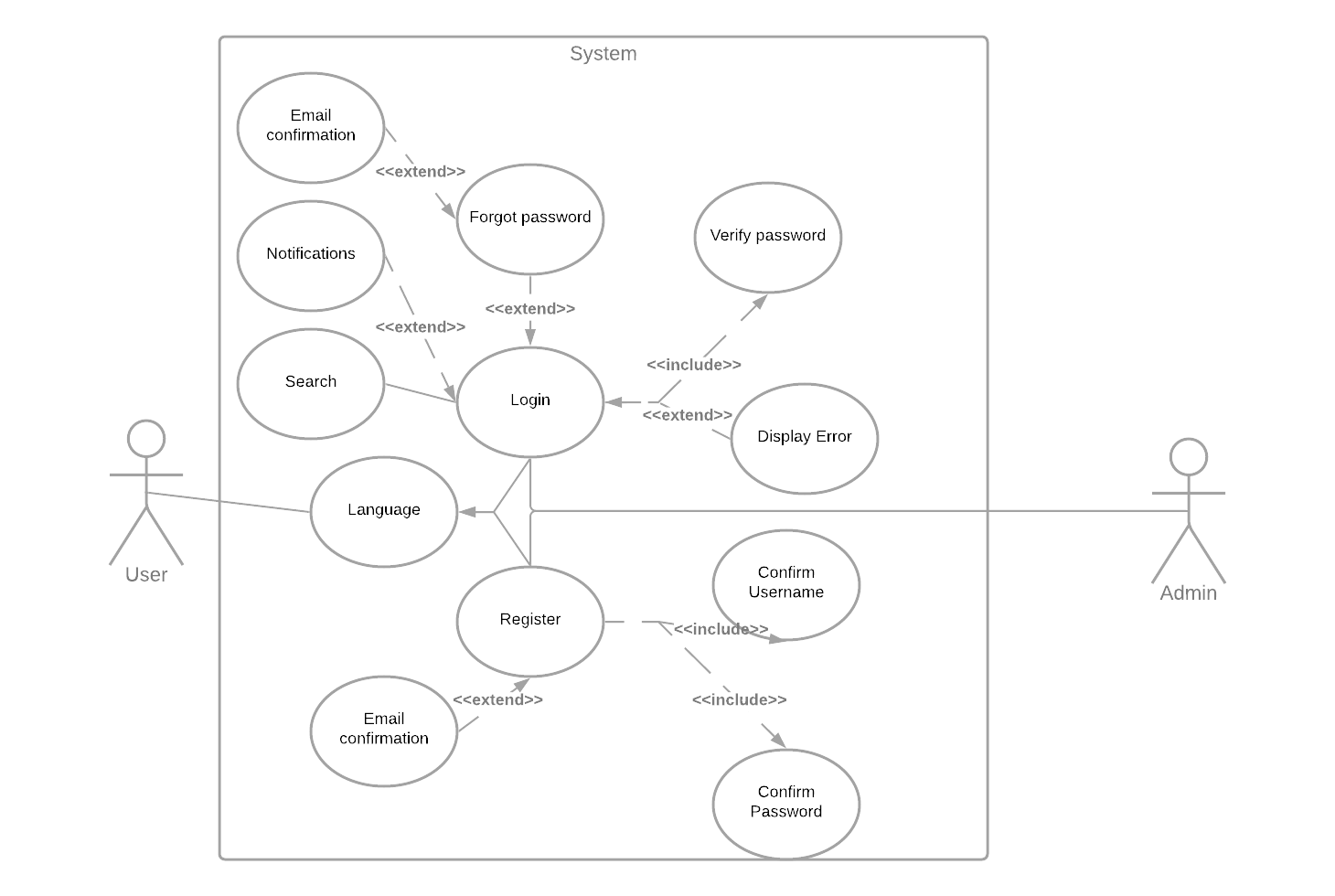
##### 2.9.1 Communication

|  |  |
| --- | --- |
| Title | Communication between members |
| Description | Time Zone difference between members and difficulty of communication. |
| Priority | 0 |

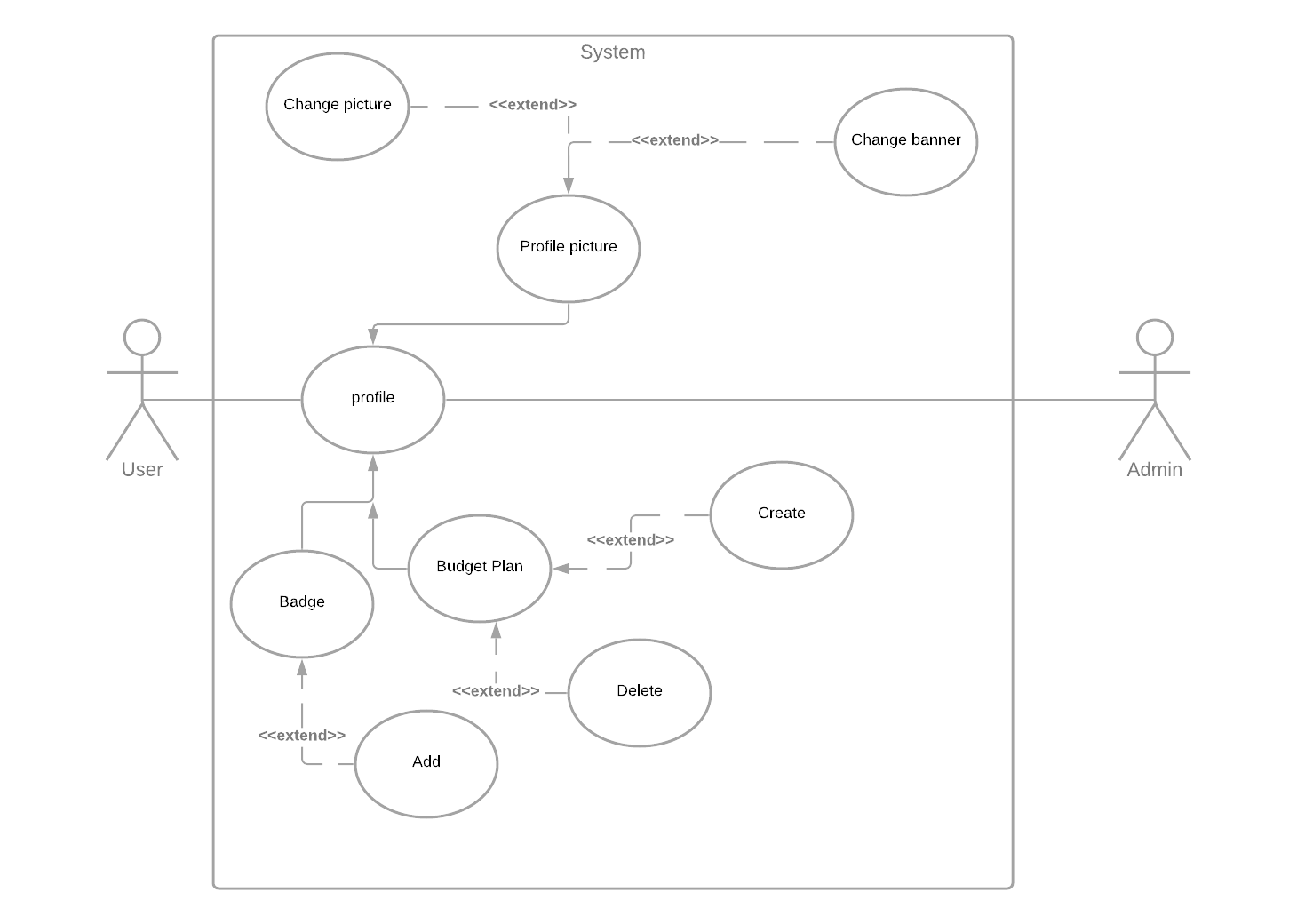
### 3. Requirements Modeling

*< List all Use-case diagrams for the functional requirements in the following format>*

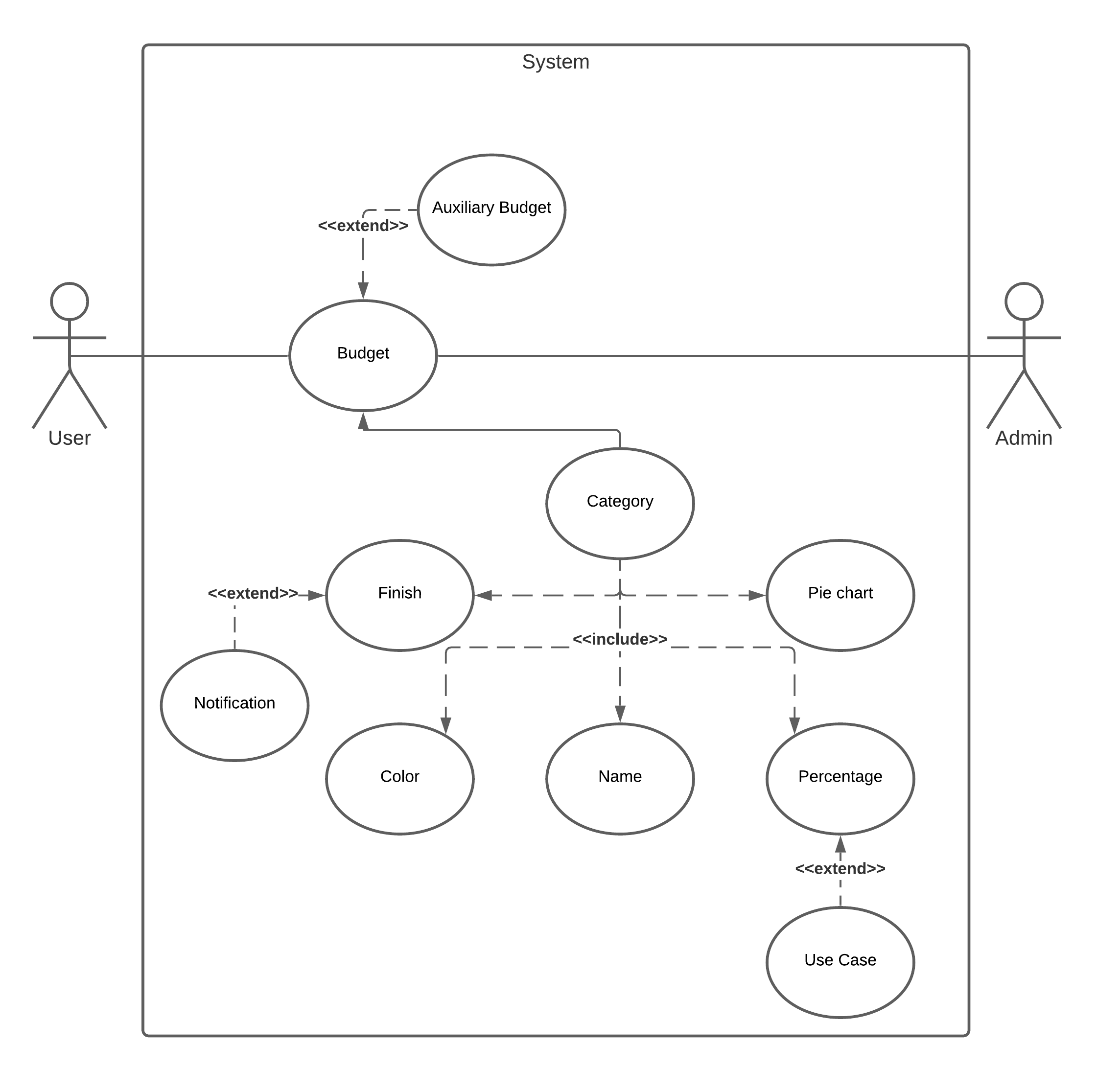
##### 3.1.1 Profile Login



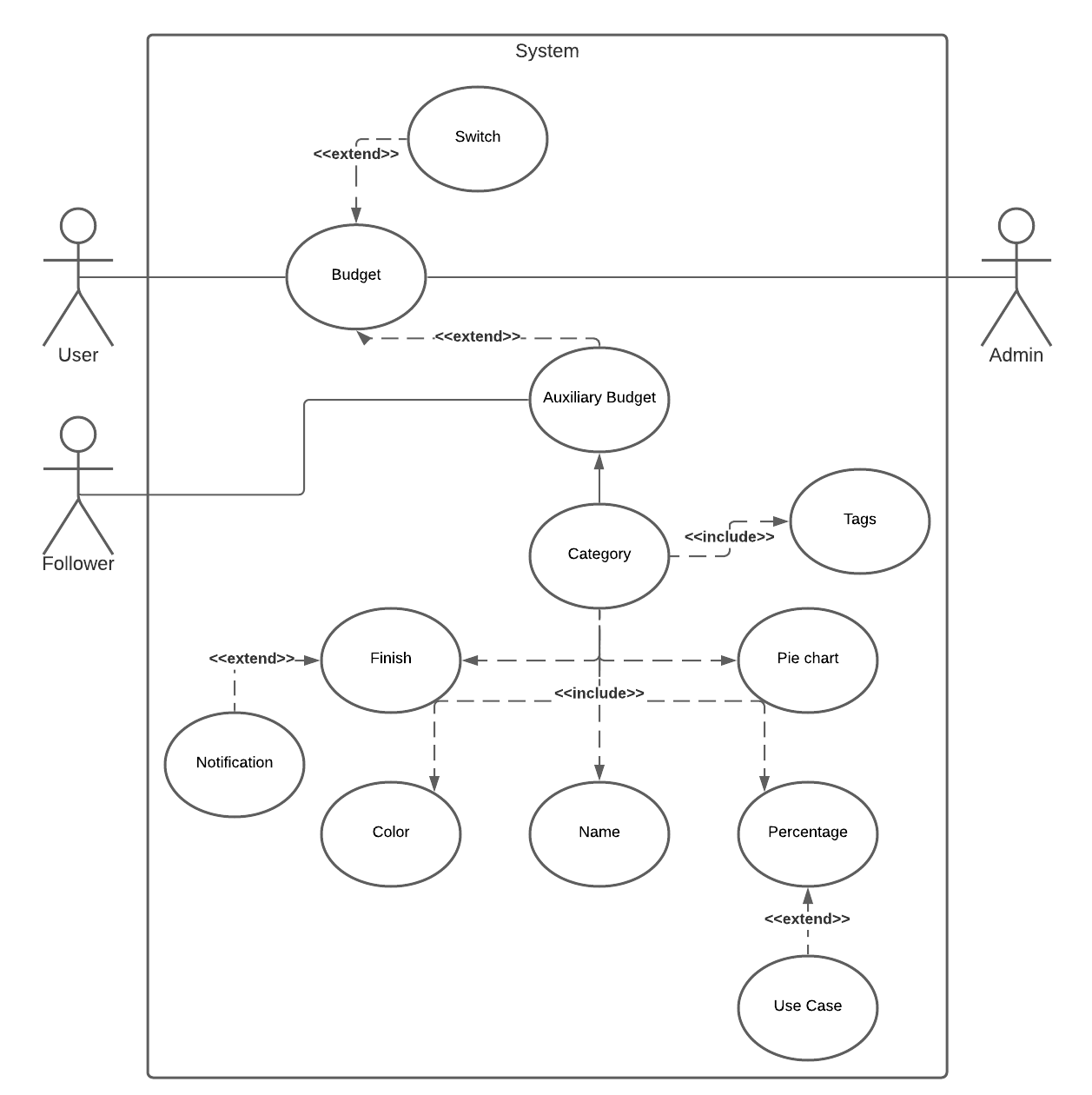
##### 3.1.2 Profile Editing



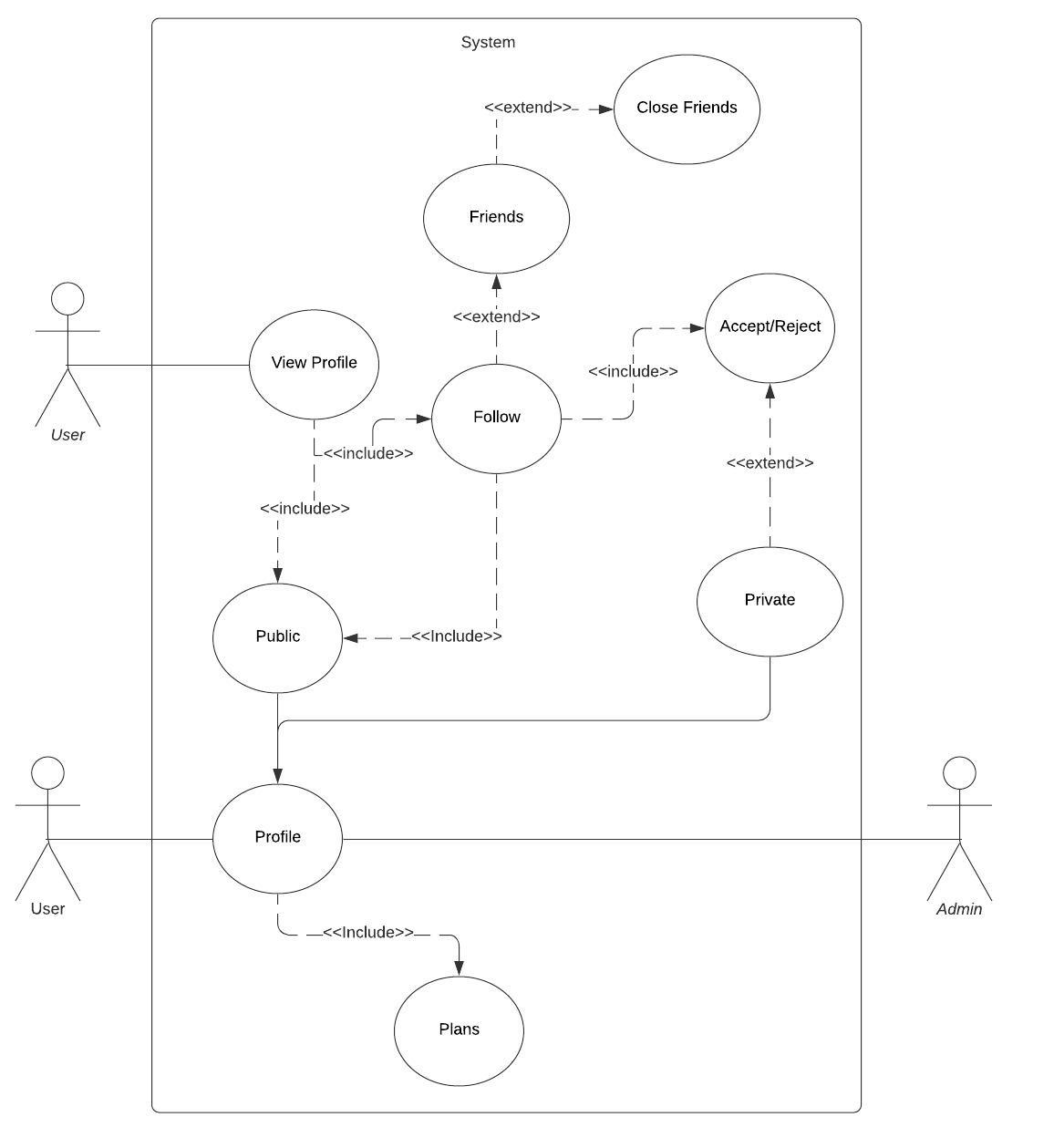
##### 3.1.3 Budget Management



##### 3.1.4 Auxiliary Budget



##### 3.1.5 Socialization



### 4. Evolutionary Requirements

#### 4.1 Non-Functional Requirements

*< List all non-functional requirements in the following example format >*

##### 4.1.1 Data logging

|  |  |
| --- | --- |
| Title | Data logging |
| Description | Logging all the data stored by the app and any changes or previous versions in order to update as well as look through previous versions in case of a bug to figure out where the issue started |
| Priority | 1 |
| Applicable FR(s) | N/A |