SubClear

Team 8

Minh Bui, Bryan Lam, Braylon Spann and Kevin David

Software Requirements Specification Document

Version: (67)
(12/01/2025)

Date:

Table of Contents

1 Purpose

Many people today subscribe to multiple digital services such as streaming platforms, cloud storage, software subscriptions, and memberships. Over time, it becomes hard to remember exactly what they are paying for, when each service renews, and how much money is leaving their account every month. Subscriptions are often set to auto-renew, so users may keep paying for services they rarely use or completely forgot about. This leads to wasted money and makes personal budgeting more difficult.

The purpose of the SubClear system is to provide a simple way for users to keep track of their recurring subscriptions in one place. By recording each subscription's name, monthly amount, billing cycle, due date, and category, the system gives users a clear overview of their total subscription spending. SubClear aims to solve the problems of forgotten subscriptions and hidden recurring charges by making all subscriptions visible on a single dashboard and showing summary totals.

With SubClear, users can quickly see:

- How many subscriptions they currently have
- How much they are spending per month
- When the next payments are due

This helps users make better financial decisions, cancel services they no longer need, and stay in control of their recurring expenses.

2 Scope

Our system is called SubClear. In the rest of this document we will use "SubClear" to refer to the whole web plugin application.

The main goal of SubClear is to help users keep track of all their recurring subscriptions in one place. The system will let a user:

- Create an account and log in
- Add subscriptions with details like name, monthly amount, due date, and category

- See a dashboard that shows all their subscriptions in a list
- See simple summaries like total number of subscriptions and total monthly cost
- Edit or delete subscriptions when things change

SubClear is meant for people who have multiple digital subscriptions (Netflix, Spotify, cloud storage, gym memberships, etc.) and want an easier way to see what they're paying for. Instead of checking different apps, emails, or bank statements, they can open SubClear and see everything on one screen.

The main benefits of the software are:

- Users can quickly see how much money is going out every month for subscriptions
- It becomes easier to spot services they don't use anymore and cancel them
- It supports basic budgeting by making recurring costs more visible

We are not trying to replace a full banking app or budgeting tool. SubClear focuses only on recurring subscriptions and giving a clear view of them.

3 User characteristics

SubClear is aimed at everyday people who pay for multiple subscriptions and want a simple way to see them all in one place. Typical users might be college students, working adults, or anyone who pays monthly for streaming, software, or memberships.

3.1 Key users

User role responsibilities – what they do with the product

- Main job of the user is to keep their own subscriptions up to date in the system.
- They register an account, log in, add new services when they subscribe to something, and delete or edit subscriptions when things change.
- They use the dashboard to quickly check how many subscriptions they have and how much they're spending each month.

Subject matter experience (subscriptions / personal finance)

- Most users will have basic knowledge of subscriptions and bills.
- We assume they know what a “monthly fee” or “due date” is, but they are not financial experts.
- In terms of domain level, they are closer to novice / everyday user than “master” of budgeting.

Technological experience

- Users are expected to be comfortable with basic web apps: filling out forms, clicking buttons, logging in/out.
- They might not be programmers or power users, so the UI needs to be simple and clear.
- Overall tech level: novice to intermediate.

Other user characteristics

- Physical / intellectual abilities: We assume a general audience with normal vision and motor skills, but we try to keep good contrast and large buttons so the app is easier to use for most people.
- Attitude toward technology: Neutral to positive. They are fine using a website if it helps them save money, but they don’t want anything complicated.
- Education / language: At least basic reading ability in English. Text should be short and direct (simple labels like “Service Name,” “Monthly Amount,” “Due Date”).
- Age group: Mainly young adults and adults (around 18–50), but the design does not depend on a specific age.
- Gender: No gender specific assumptions. SubClear is intended to be usable by anyone.

3.2 Secondary users

User role responsibilities / what they do with the product

- A secondary user could be a system administrator or support person who helps set up the system, reset accounts, or troubleshoot issues.
- In a class/project setting, instructors or TAs might also be secondary users who run the system to grade or demo it.

Subject matter experience

- Admin/support users usually have a better understanding of technical details and may know more about how the system stores data or is deployed.
- Their domain knowledge about subscriptions is at least journeyman level (comfortable with financial/recurring payments).

Technological experience

- Secondary users are expected to be more technical than normal users.
- They are comfortable installing software, editing configuration files, and checking logs or database entries if needed.
- Tech level: intermediate to advanced.

Other user characteristics

- Physical / intellectual abilities: Same as general users; we don’t assume anything special here.
- Attitude toward technology: Generally positive; they use tools like SubClear as part of their work or teaching.
- Education / language: Likely college-level or above, able to read technical instructions in English.

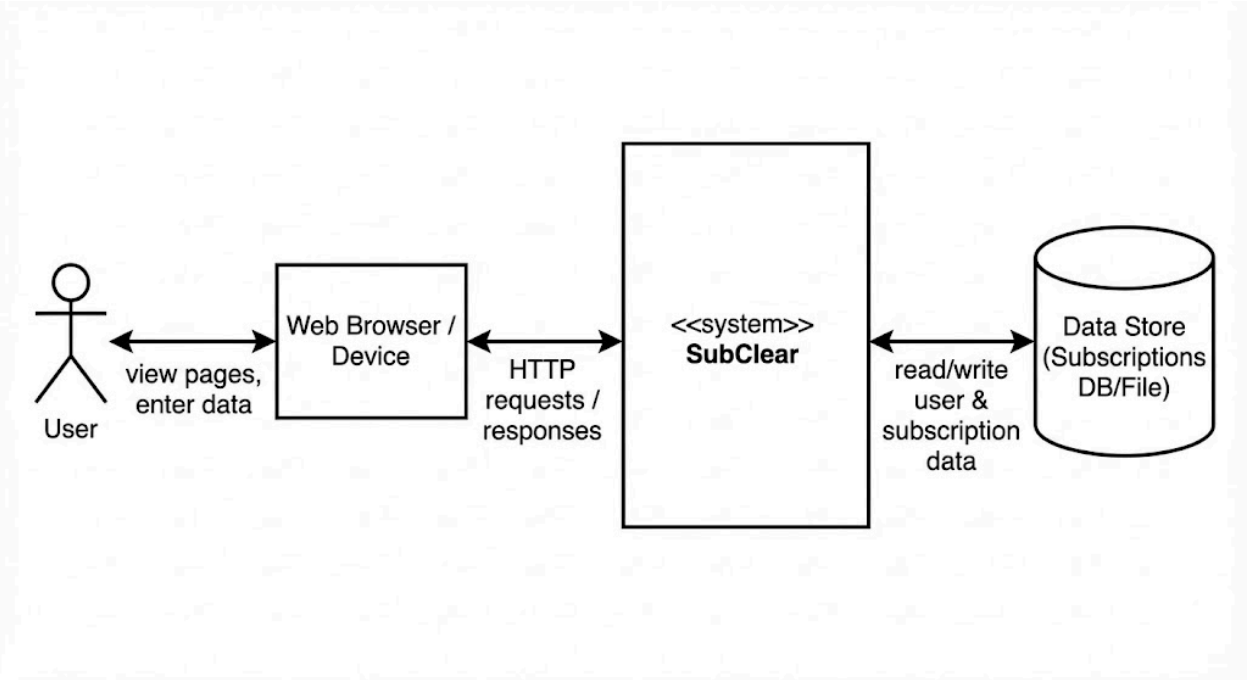
- Age group / gender: No specific assumptions; the system should work the same for all.

4 Product perspective

4.1 System Context

SubClear is a small, standalone web application. It is not part of a larger enterprise system and does not directly integrate with bank APIs, payment processors, or other external services in the current version. The main interaction is between an end user and the SubClear web server through a web browser.

The user opens SubClear in a modern browser on their laptop or desktop, logs in, and then views or updates their subscription data. The browser sends requests to the SubClear backend, which processes the request and reads or writes data in the system's data store. The data store keeps user accounts and subscription information so it is available the next time the user logs in.



4.2 User interfaces

The following requirements describe how SubClear's user interface must behave.

1. **UI-1 – Web based interface**
   It is required that the system interface is presented as a web application that runs in a modern desktop browser (e.g., Chrome, Edge).

2. **UI-2 – Consistent layout and theme**
   It is required that all pages (Login, Register, Dashboard, Add/Edit Subscription) use a consistent color theme and layout, with a centered main card containing the page title and content.

3. **UI-3 – Form input requirements**
   It is required that all data entry is done through labeled form fields. Required fields (such as username, password, service name, monthly amount, and due date) must be clearly indicated to the user.

4. **UI-4 – Error messages**
   It is required that, when user input is invalid (for example, missing required fields or wrong login credentials), the system displays a clear, short error message near the

affected form or at the top of the card. Raw technical error messages must not be shown.

5. **UI-5 – Navigation between pages**
   It is required that users can navigate between Login and Register pages using visible links, and that after successful login the system automatically navigates to the Dashboard page. A Logout control must be available on the Dashboard.

6. **UI-6 – Dashboard display**
   It is required that the Dashboard page displays:

   - A summary section showing total subscriptions, active subscriptions, and monthly total; and

   - A list of individual subscriptions, each with name, amount, due date, category, and controls to edit or delete the subscription.

7. **UI-7 – Add/Edit Subscription interaction**
   It is required that the Add/Edit Subscription page provides fields for service name, monthly amount, due date, category, and an optional description, along with **Save/Add** and **Cancel** buttons.

8. **UI-8 – Feedback on actions**
   It is required that after key actions (adding, editing, or deleting a subscription, or logging in/out), the interface provides visible feedback—for example, by updating the dashboard list and summary values, or redirecting the user to the appropriate page.

9. **UI-9 – Basic accessibility considerations**
   It is required that the user interface uses readable font sizes and sufficient contrast between text and background so that typical users can read the content without difficulty.

4.3 Software interfaces

SubClear is a small web application and only needs to interact with a few other software components.

1. **SW-IF-1 – Web browser interface**

- The system shall provide an HTML/HTTP interface that can be accessed from a modern web browser (such as Google Chrome or Microsoft Edge).

- All user interactions (login, registration, dashboard, add/edit subscription) shall be available through standard HTTP requests and HTML pages.

- No browser plug-ins or extensions shall be required.

2. **SW-IF-2 – Application server / runtime**

   - The SubClear backend shall run on a Java application server or runtime that supports Java and Maven (for example, a Spring Boot based server).

   - The system shall expose its functionality over HTTP so that the browser can communicate with it.

3. **SW-IF-3 – Data management interface**

   - The system shall store user accounts and subscription data in a persistent data store (for example, a relational database or equivalent).

   - The application shall communicate with the data store using standard database connectivity (such as JDBC) so that the underlying database product can be changed without affecting the rest of the system.

   - The exact database product (MySQL, PostgreSQL, or an embedded DB) is a design choice and is not fixed by this requirement.

4. **SW-IF-4 – External systems**

   - In the current version, SubClear is not required to integrate with any external payment systems, banking APIs, or third party authentication providers. Any such integrations in the future would require new interface specifications.

4.4 Deployment requirements

SubClear is a small web application and does not need any special hardware. The following environment is required for installation and operation:

- The backend must run on a computer or server with:
    - A modern operating system (Windows, macOS, or Linux).
    - Java JDK (version 17 or later).
    - Maven installed to build and run the project.
    - Enough memory and disk space for a small web app and its data (a few GB is more than enough for this project).
- If a real database is used, the database server must be installed and running before SubClear is started, and the connection settings (URL, username, password) must be configured in the application.
- Clients access SubClear through a web browser, so:
    - Each user needs a device (laptop or desktop) with a modern web browser such as Chrome or Edge.
    - The device must be able to reach the server over a local network or the internet (depending on where the server is hosted).
- No special network equipment, extra cooling, or dedicated hardware is required for this course project. A single machine can host both the server and database for development and testing.

5 Assumptions and Dependencies

1. **A1 – Users have internet access and a modern browser.**
   We assume users can access SubClear through a modern web browser (Chrome, Edge, etc.) and have a stable connection to the server.

2. **A2 – Java and required runtime tools are installed on the server.**
   We assume the deployment machine has a compatible Java JDK and Maven (or an

equivalent runtime) available so the backend can run.

3. **A3 – A suitable data store is available.**
   We assume there is a data store (relational database or embedded DB) that SubClear can connect to for storing user accounts and subscriptions. If the data store type changes, connection settings and some requirements may need to be updated.

4. **A4 – Single-tenant, small-scale usage.**
   We assume the system will be used by a relatively small number of users at a time (class project / small personal use). High traffic or enterprise level scaling is out of scope.

5. **A5 – Users manage their own subscription data.**
   We assume users will manually enter, update, and delete their subscription information. The system does not automatically pull data from banks or third party services.

6. **A6 – No strict regulatory or compliance requirements.**
   We assume the system is used for personal budgeting and a course project, not for a financial institution that must meet strict legal or compliance rules. If that assumption changed, additional security and data-handling requirements would be needed.

6 Specific requirements

This section describes the functional and non functional requirements for SubClear.

6.1 System Functional Requirements

6.1.1 Register account

User story (high level)

As a new user, I want to create an account so that I can log in and save my subscriptions.

Detailed scenario

1. The user opens the Register page.

2. The system shall display input fields for username, email, full name, and password.

3. The user fills in all required fields and clicks Create Account.

4. The system shall validate that required fields are not empty and that the password meets basic rules (minimum length).

5. If the input is valid, the system shall create a new user record and store it in the data store.

6. The system shall confirm account creation (for example, by redirecting to the login page or logging the user in).

7. If the input is not valid, the system shall show a clear error message and shall not create the account.

6.1.2 Log in / log out

User story (high level)

> As a registered user, I want to log in and log out so I can access my dashboard safely.

Detailed scenario – login

1. The user opens the Login page.

2. The system shall provide fields for username and password and a Login button.

3. The user enters their credentials and clicks Login.

4. The system shall check the credentials against stored user data.

5. If they match, the system shall create a session for that user and redirect to the Dashboard page.

6. If they do not match, the system shall display an error message and remain on the Login page.

Detailed scenario – logout

1. While on the Dashboard, the user clicks the Logout button.
2. The system shall end the user session.
3. The system shall redirect the user back to the Login page.

The system shall prevent access to dashboard functions when there is no active session.

6.1.3 View dashboard and subscription summary

User story (high level)

> As a logged in user, I want to see all my subscriptions and summary totals so that I understand how much I am spending each month.

Detailed scenario

1. After successful login, the system shall show the Dashboard page.
2. The system shall load all subscriptions that belong to the current user from the data store.
3. For each subscription, the system shall display at least: service name, monthly amount, due date, and category.
4. The system shall calculate and display:
    - total number of subscriptions,
    - number of active subscriptions,
    - total monthly cost (sum of monthly amounts for active subscriptions).
5. If the user has no subscriptions, the system shall show an empty-state message (“No subscriptions yet”) and a link or button to add a new subscription.

6.1.4 Manage subscriptions (add / edit / delete / status)

User story (high level)

As a user, I want to add, edit, delete, and mark subscriptions as active or inactive so that my list stays accurate.

Detailed scenario – add subscription

1. From the Dashboard, the user clicks Add New Subscription.
2. The system shall display an Add Subscription form with fields: service name, monthly amount, due date, category, and optional description.
3. The user fills in the form and clicks Add Subscription.
4. The system shall validate that required fields are present and that the monthly amount is numeric and non-negative.
5. If valid, the system shall create a new subscription linked to the current user, store it, and return to the Dashboard.
6. The Dashboard shall update the list and summary totals.

Detailed scenario – edit subscription

1. On the Dashboard, the user chooses a subscription and clicks Edit.
2. The system shall display an Edit form pre filled with the subscription's current values.
3. The user changes one or more fields and clicks Save.
4. The system shall validate the new values.
5. If valid, the system shall update the subscription record and refresh the Dashboard list and totals.

Detailed scenario – delete subscription

1. On the Dashboard, the user clicks Delete for a subscription.

2. The system shall ask for confirmation (or clearly indicate that the item will be removed).

3. After confirmation, the system shall remove the subscription record from the data store and update the Dashboard list and totals.

Detailed scenario – active / inactive status

1. On the Dashboard, each subscription shall have a control (a toggle) to mark it Active or Inactive.

2. When the user changes this status, the system shall update the stored subscription.

3. The system shall recalculate the number of active subscriptions and the monthly total using only active ones.

6.2 Logical Database Requirements

The SubClear system needs to store user accounts and their subscriptions in a persistent data store.
The logical data model contains two main entities:

- User – stores information about each account (userId, username, email, passwordHash). userId is the primary key.

- Subscription – stores information about each subscription (subscriptionId, userId, name, monthlyAmount, billingCycle, nextPaymentDate, category, status). subscriptionId is the primary key. userId is a foreign key that links the subscription to its owner.

Each User can own zero or many Subscriptions (1 to 0..*). The userId foreign key in Subscription must reference a valid User, and monthlyAmount must be non-negative.

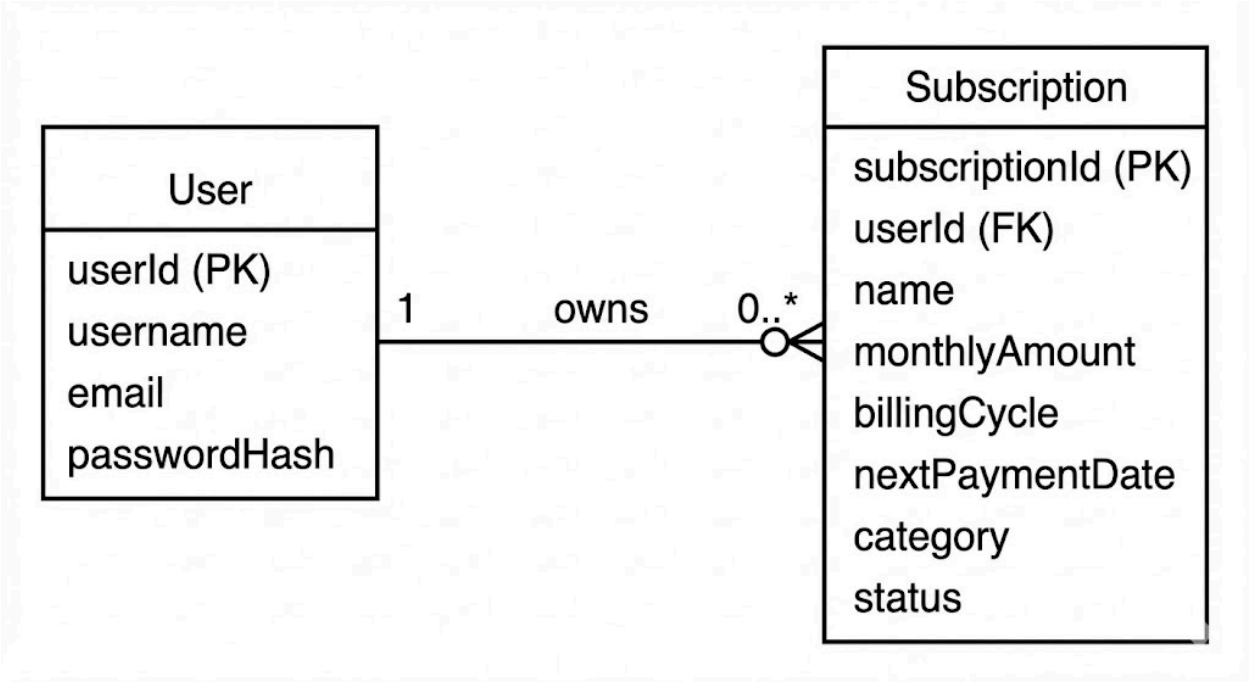


Figure X – Logical data model for SubClear

6.3 Software System Attributes

6.3.1 Usability

US-1 The system shall present a consistent layout and theme across Login, Register, Dashboard, and Add/Edit Subscription pages.

US-2 Labels for all input fields shall use clear, everyday language (“Service Name”, “Monthly Amount”, “Due Date”).

US-3 A first time user shall be able to register, log in, and add at least one subscription without reading any external documentation. (This can be tested with a short usability test.)

6.3.2 Performance

PF-1 For a typical user with up to 50 subscriptions, the Dashboard page shall load and display all subscriptions within 3 seconds on a normal development machine.

PF-2 Adding, editing, or deleting a subscription shall update the dashboard list and summary totals within 2 seconds after the user submits the form.

6.3.3 Reliability/Dependability

RL-1 The system shall handle invalid user input (missing required fields, non-numeric amount) without crashing; instead it shall display validation messages.

RL-2 In case the data store is temporarily unavailable, the system shall display an error message and shall not corrupt existing data.

RL-3 User and subscription data stored in the data store shall persist across server restarts.

6.3.4 Security

SC-1 The system shall authenticate users using a username (or email) and password before granting access to any subscription data.

SC-2 The system shall store passwords only in hashed form, not as plain text.

SC-3 The system shall ensure that a user can only view and modify subscriptions that belong to their own account.

SC-4 The system shall invalidate the user session on logout so that the dashboard cannot be accessed without logging in again.

6.3.5 Maintainability

MT-1 The system shall separate presentation (HTML/CSS), application logic, and data access into different modules or layers to make changes easier.

MT-2 Configuration values that may vary between environments (database connection string) shall be stored in a configuration file rather than hard-coded.

MT-3 The code base shall be organized so that new features such as additional subscription fields or new categories can be added without changing existing behavior for other users.