SUBTRACKER

INITIAL REQUIREMENTS VERSION 0.2

April 28, 2025

REQUIREMENTS

1. Project Background and Description

Target Release: 0.2

Epic: Core Functionality

Status: Draft

Document Owner: [Your Name]

Designer: [Your Name]

Tech Lead: [Your Name]

2. Project Scope

SubTracker is a personal finance and subscription management application. It provides users with tools to track their financial accounts, manage deposits and withdrawals, and organize recurring autopayments like subscriptions or bills. Users will be able to visualize upcoming financial obligations via a calendar interface.

The system is designed for individual users to manage their own finances locally, with the future goal of expanding to cloud and mobile platforms.

3. High-Level Requirements

SubTracker will include the following:

- An interface to manage financial accounts and starting balances.
- An interface to input deposits and withdrawals.
- An interface to schedule and manage recurring autopayments (subscriptions).
- A calendar view to visualize upcoming payments and events.

- A backend system to store user financial data securely.
- Basic reporting or viewing tools to summarize financial activity.
- Future support for optional mobile expansion and biometric login integration.

4. Entities

Entities include all affected stakeholders, systems, and services.

SubTracker will affect the following entities:

- **SubTracker System**: The ASP.NET Core MVC web application frontend.
- **SubTracker Backend**: The Entity Framework Core layer managing database operations.
- **User**: The individual managing their personal finances via the system.
- **SQL Server Database**: The storage of user account, transaction, and autopayment data.

5. Success Metrics

The following metrics can be used to judge the project's success:

Goal	Metric
Users can create accounts with starting balances	Accounts are created and displayed
Users can add deposits and withdrawals	Transactions are recorded and displayed
Users can schedule recurring autopayments	Complete
Calendar view is functional and displays events	Calendar auto-populates with data
System is mobile-friendly (stretch goal)	Key functionality reachable on mobile devices
System handles missing database connections safely	Graceful error handling without app crash

6. Considerations

Requirement	User Story	Status
User can create a financial account	As a user, I want to create accounts so I can track different balances	In Progress
User can add deposits or withdrawals	As a user, I want to record deposits and withdrawals to track my balance accurately	In Progress
User can schedule recurring autopayments	As a user, I want to automate future payments to stay organized	Complete
User can view upcoming payments on a calendar	As a user, I want to see my upcoming obligations visually by date	Planned
System can handle SSL issues on local SQL Server	As a developer, I want the system to bypass untrusted SSL certificates for local connections	Complete

7. Specific Exclusions from Project Scope

The following functionality is planned for SubTracker but is currently outside of the scope of the project:

- Bank API integration for automatic deposit and withdrawal updates.
- Multi-user account management and cloud syncing.
- Biometric login (e.g., Face ID) implementation.
- Mobile app version (to be planned after core web version).

8. Dependencies and Technologies

The following technologies and dependencies are required:

- ASP.NET Core MVC
- Entity Framework Core
- SQL Server Developer or Express Edition (local instance)
- Visual Studio 2022+
- Microsoft.EntityFrameworkCore.SqlServer NuGet package
- SQL Server Management Studio (SSMS) for database access
- FullCalendar.io JavaScript library (for calendar view, future phase)

9. Assumptions

SubTracker is designed under the following assumptions:

- The system will run locally on the user's machine initially.
- Users will manually input their financial data.
- Windows Authentication will be used to connect to the local SQL Server instance.
- SSL certificates for local SQL Server connections may not be trusted, and the system will allow TrustServerCertificate=True.
- Future enhancements will focus on mobility, bank integration, and better user experience.

10. Designs and Diagrams

(To be created during project development)

Entity Relationship Diagram (ERD), UI wireframes, and database migration scripts will be documented in future versions.