The Islamia University of Bahawalpur

Department of Software Engineering

Faculty of Computing

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

for

FINANCEMATE

Version 1.0.0

By

Shoaib Ahmad

F21BSEEN1E02039

7TH-E1

Session Fall 2021 – 2025

Supervisor

Sir Umair Zafar Khan

Bachelor of Science in Software Engineering

Revision History

Name	Date	Reason for changes	Version
KhataApp	10-11-2024	Unprofessional	1.0.0

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Dr.	Supervisor, SENG- 4301	<date>

Application Evaluation History

Comments (by committee) ***include the ones given at scope time both in doc and presentation**	**Action Taken**

Supervised by

< Sir Umair Zafar Khan>

Signature_____

Table of Contents

1. Introduction

This Software Requirements Specification (SRS) outlines the comprehensive functional and non-functional requirements for the development of **Financemate**. The document serves as a structured guideline to ensure a clear understanding among stakeholders—including developers, testers, and end-users—of the system's objectives, functionalities, constraints, and intended outcomes.

Financemate is designed to provide an intuitive, efficient, and user-friendly platform to manage financial activities. The system aims to enhance financial record-keeping, tracking, and reporting through a feature-rich application that ensures security, scalability, and ease of use. By precisely documenting requirements, this SRS mitigates ambiguity and lays the foundation for robust system design, implementation, and validation.

This SRS document is essential for guiding the development process, ensuring adherence to the defined requirements, and supporting the system's lifecycle stages, including deployment, maintenance, and updates. It reflects the collective vision and expectations for **Financemate**, ensuring its alignment with user needs and technological feasibility.

1.1 Purpose

The purpose of this document is to define the requirements for the software project. It outlines the specific functionality, constraints, and assumptions for the development of the system, ensuring alignment with the goals and expectations of stakeholders.

1.2 Scope

The software product, **Financemate**, is designed as a comprehensive financial management solution for individuals and small businesses. It will enable users to track their income, expenses, and savings, offering insights and analytics to help with budgeting and financial planning.

The system will:

- Facilitate the recording and categorization of financial transactions.
- Generate detailed reports and summaries of financial data.
- Provide alerts and reminders for bill payments and savings goals.
- Offer a user-friendly interface for both web and mobile platforms.

The system will not:

- Handle large-scale enterprise-level financial operations.
- Provide real-time integration with banking systems without additional customization.
- Offer investment advice or stock trading capabilities.

Application and Benefits:

The Financemate application aims to simplify personal and small-scale financial management. Key benefits include:

- Enhanced financial awareness through detailed tracking and reporting.
- Improved financial decision-making with personalized insights.
- Reduction in financial mismanagement by providing timely reminders and alerts.

1.3 Definitions, Acronyms, and Abbreviations.

This section provides the definitions of terms, acronyms, and abbreviations that are essential for the interpretation of the Software Requirements Specification (SRS) for the **Financemate** project. These terms are used throughout the document to ensure clarity and consistency.

Definitions

The following terms are defined to ensure a clear understanding of the requirements and specifications outlined in this document:

- **Financemate**: The name of the personal financial management system being developed, which helps users track spending, manage budgets, and make informed financial decisions.
- **User**: An individual or entity who interacts with the system to monitor and manage their personal finances.
- **Account Management**: Tools within the system for managing user financial accounts, including integration with banking systems.
- **Transaction Categorization**: The process of grouping financial transactions into predefined categories (e.g., groceries, utilities, entertainment) for easier tracking and analysis.
- **Real-Time Analytics**: The ability of the system to process and display financial data and insights in real-time as users interact with the system.
- **Personalized Insights**: Custom financial recommendations and reports based on user data, helping them make informed decisions.
- **Machine Learning (ML)**: A type of artificial intelligence used in the system to analyze user behavior and predict financial trends or categories for transactions.
- **Localization**: The adaptation of the system to different languages, regional financial formats, and cultural norms.
- **Data Security**: The protection of user data from unauthorized access, including the use of encryption and multi-factor authentication.

Acronyms

The following acronyms are used in this document:

- **SRS**: Software Requirements Specification
- **API**: Application Programming Interface
- **UI**: User Interface
- **ML**: Machine Learning
- **REST**: Representational State Transfer

- **MFA**: Multi-Factor Authentication
- **DBMS**: Database Management System
- **CSV**: Comma-Separated Values
- **JSON**: JavaScript Object Notation
- **AES**: Advanced Encryption Standard
- **POSTGRESQL**: PostgreSQL, a relational database management system used in the project.

Abbreviations

The following abbreviations are used throughout the document for brevity:

- **Req**: Requirement
- **Dev**: Development
- **API**: Application Programming Interface
- **UX**: User Experience
- **DB**: Database
- **RESTful**: A web service architecture using HTTP methods and status codes.
- **CSV**: Comma-Separated Values (a format for exporting financial data).
- **MFA**: Multi-Factor Authentication (security measure for user accounts).
- **TLS**: Transport Layer Security (protocol for encrypting communication).

1.4 References

The following documents and standards are referenced in the **SRS** for **Financemate**:

1. **IEEE Standard for Software Requirements Specifications (IEEE 830-1998)**
    - **Title**: IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
    - **Date**: 1998
    - **Publishing Organization**: IEEE (Institute of Electrical and Electronics Engineers)
    - **Source**: Available for purchase on IEEE Xplore: https://ieeexplore.ieee.org
2. **RESTful Web Services (Fielding, R.)**
    - **Title**: Architectural Styles and the Design of Network-based Software Architectures
    - **Author**: Roy Fielding
    - **Date**: 2000
    - **Publishing Organization**: University of California, Irvine
    - **Source**: Available at https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
3. **PostgreSQL Documentation**
    - **Title**: PostgreSQL Documentation (for version 13)
    - **Date**: 2020
    - **Publishing Organization**: PostgreSQL Global Development Group
    - **Source**: https://www.postgresql.org/docs/
4. **OAuth 2.0 Authorization Framework (RFC 6749)**
    - **Title**: The OAuth 2.0 Authorization Framework
    - **RFC Number**: 6749
    - **Date**: October 2012

   - **Publishing Organization**: Internet Engineering Task Force (IETF)
   - **Source**: Available at https://tools.ietf.org/html/rfc6749
5. **Advanced Encryption Standard (AES) - Federal Information Processing Standards (FIPS) 197**
   - **Title**: FIPS PUB 197, Advanced Encryption Standard (AES)
   - **Date**: November 2001
   - **Publishing Organization**: National Institute of Standards and Technology (NIST)
   - **Source**: Available at https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
6. **ISO/IEC 9126-1:2001 - Software Engineering: Product Quality**
   - **Title**: ISO/IEC 9126-1:2001, Software Engineering – Product Quality – Part 1: Quality Model
   - **Date**: 2001
   - **Publishing Organization**: International Organization for Standardization (ISO)
   - **Source**: Available for purchase on the ISO website: https://www.iso.org
7. **Machine Learning Yearning by Andrew Ng**
   - **Title**: Machine Learning Yearning
   - **Author**: Andrew Ng
   - **Date**: 2018
   - **Publishing Organization**: Self-published by Andrew Ng
   - **Source**: Available for free download at https://www.mlyearning.org/
8. **Flutter Documentation**
   - **Title**: Flutter Docs
   - **Date**: Ongoing (Latest version)
   - **Publishing Organization**: Google
   - **Source**: https://flutter.dev/docs

1.5 Overview

This **Software Requirements Specification (SRS)** outlines the functional and non-functional requirements for **Financemate**, a personal financial management application designed to help individuals and small businesses track their income, expenses, savings, and budgeting. The goal of this document is to ensure clarity for all stakeholders involved in the development, deployment, and maintenance of the system. It serves as a comprehensive guide to understanding what **Financemate** must achieve and how the system will be developed to meet user needs.

This document provides detailed specifications and guidelines for developers, testers, and users involved in the project. **Financemate** will be designed to provide a seamless experience on both web and mobile platforms, offering users an intuitive, efficient, and secure environment for managing their finances.

Structure and Organization of the SRS

The **SRS** is organized as follows to address the needs of different stakeholders and cover all aspects of the system's development:

- ## Introduction
  This section provides an overview of **Financemate**, including the scope, purpose, definitions, and abbreviations used throughout the document. It sets the foundation for the document and helps stakeholders understand the context and objectives of the project.
    - **Purpose**: The purpose of **Financemate** is to simplify financial management for individuals and small businesses by providing tools for income tracking, expense categorization, and budgeting.
    - **Scope**: It covers features such as expense tracking, budget management, reports, alerts, and reminders.
    - **Definitions and Acronyms**: Key terms like "Transaction Categorization", "Real-Time Analytics", "API", and "MFA" are defined for clarity.
- ## System Overview
  This section gives a high-level view of the system's objectives and key features:
    - **System Purpose**: **Financemate** will enable users to manage their personal and business finances effectively.
    - **Key Features**:
        - Expense tracking and categorization.
        - Budget management and savings goal tracking.
        - Personalized insights using machine learning.
        - Alerts and reminders for bill payments and financial milestones.
        - Detailed financial reporting and analytics.
    - **Target Audience**: Primarily individuals and small businesses looking to manage their finances with minimal complexity.
- ## Functional Requirements
  This section outlines the core functionalities of the system, divided into different components:
    - **User Account Management**:
        - Users must be able to create and manage their accounts.
        - Integration with third-party authentication systems like Google or Facebook.
        - Multi-factor authentication (MFA) to ensure account security.
    - **Transaction Management**:
        - Users can add, edit, and delete financial transactions (income, expenses, savings).
        - Transactions must be categorized (e.g., groceries, entertainment, utilities).
        - Support for recurring transactions (e.g., monthly bills).
    - **Budgeting and Reporting**:
        - Users can set monthly and yearly budgets for categories (e.g., groceries, entertainment).
        - Detailed financial reports should be available (monthly expense report, savings report).
        - Graphical representations of income vs. expenditure.
    - **Reminders and Alerts**:
        - Notifications for bill payments, budget limits, and savings goal progress.
        - Alerts can be delivered via email, SMS, or push notification.
    - **Personalized Financial Insights**:

        - The system will use machine learning to suggest ways to optimize spending based on user behavior.
        - Reports will include trends, future predictions, and expense-saving recommendations.

- ## Non-Functional Requirements
  This section details the system's performance, security, and usability requirements:
    - **Performance**:
        - The system should respond to user actions within 2 seconds.
        - The system should be able to handle up to 500 concurrent users.
    - **Security**:
        - User data should be encrypted using AES (Advanced Encryption Standard).
        - Regular security audits and vulnerability assessments should be performed.
        - User accounts must be protected using multi-factor authentication (MFA).
    - **Scalability**:
        - The system should be able to scale to handle more users and larger datasets as the user base grows.
    - **Usability**:
        - The system should have a responsive design to work seamlessly on both web and mobile platforms.
        - The UI should be intuitive and easy to navigate for non-technical users.
- ## System Architecture and Design
  This section describes the high-level system architecture and design considerations:
    - **System Architecture**:
        - **Frontend**: The frontend will be built using React for the web and Flutter for mobile applications (iOS and Android).
        - **Backend**: A RESTful API using Node.js, with a PostgreSQL database for data storage.
        - **Machine Learning**: ML models will run on a separate microservice for personalized insights and analytics.
        - **Authentication**: Use of OAuth for third-party integrations and MFA for secure login.
    - **Design Constraints**:
        - The system will be designed to comply with GDPR data privacy regulations.
        - It should support localization for different currencies, time zones, and languages.
- ## External Interfaces
  Financemate will interact with several external systems and APIs:
    - **Banking Integration**: Integration with third-party APIs (such as Plaid or Yodlee) for automatic import of bank transactions.
    - **Data Import/Export**: Users can import/export their financial data in CSV or JSON formats.
    - **Email and SMS Services**: Integration with email and SMS APIs for sending alerts and notifications.
- **Verification and Validation**
  This section outlines the strategies and test cases for verifying and validating **Financemate**:

    - **Functional Testing**: Verify that all system functionalities work as intended, including transaction categorization, budget tracking, and reporting.
    - **Performance Testing**: Test the system's response time under various load conditions.
    - **Security Testing**: Conduct penetration testing to identify potential vulnerabilities.
- **Appendices**
  The appendices will contain:
    - **UI Mockups**: Visual representation of the system's user interface.
    - **Data Models**: A representation of the database schema and relationships.
    - **API Documentation**: Detailed API endpoints and their specifications.

2. The Overall Description

The **Financemate** system is being developed to address the common needs of individuals and small businesses in managing their finances. Several factors affect the product and its requirements, including market demands, user behaviour, technological advancements, and legal considerations. The following provides an overview of these factors, laying the groundwork for the specific requirements described in Section 3.

Target Audience and Use Cases

Financemate is designed with simplicity and accessibility in mind. It caters primarily to individuals and small businesses who need an easy-to-use tool to keep track of their financial activities. These users may not have extensive financial knowledge but still need powerful features like budgeting, expense tracking, and report generation.

- **Individuals**: Personal users are looking for a way to manage day-to-day finances, track spending, save for goals, and ensure they stay within their budget.
- **Small Businesses**: Small business owners need to track income, expenses, and taxes, generate financial reports, and maintain accurate records for their business operations.

This broad target audience informs the system's design, ensuring that it offers both depth for more advanced users and simplicity for those who may not be well-versed in financial management.

User Needs and Expectations

The key needs of users that influence the requirements include:

- **Ease of Use**: Users expect an intuitive, easy-to-navigate interface that minimizes complexity while still offering powerful features. The goal is to ensure that anyone, regardless of their technical expertise, can use **Financemate** without needing a steep learning curve.
- **Real-Time Access**: As personal finance management often involves ongoing monitoring, users require the ability to update their financial data in real time, view their current balances, and receive instant alerts for bills or budget limits.
- **Automation**: Users expect features that reduce manual effort, such as automatic categorization of transactions, recurring payment reminders, and financial insights

based on their historical data. This allows them to spend less time tracking expenses and more time making financial decisions.
- **Security and Privacy**: Security is critical, as users trust **Financemate** with sensitive financial data. The system must ensure that data is encrypted, protected by strong authentication methods, and never compromised. Multi-factor authentication (MFA) and AES encryption are examples of how security considerations have been integrated into the system's design.

Market and Industry Trends

Several industry trends are influencing the features and capabilities of **Financemate**:

- **Cloud Integration**: Cloud-based financial tools are becoming increasingly popular as they allow users to access their financial data from anywhere, on any device. This allows **Financemate** to offer both web and mobile platforms, ensuring accessibility across a variety of devices.
- **Personalization and Machine Learning**: Users increasingly expect personalized financial recommendations based on their spending habits. With the use of machine learning (ML), **Financemate** will offer insights into user behaviour, suggesting ways to save money, optimize spending, and achieve financial goals.
- **Data Portability**: As users want to have control over their data, **Financemate** will support importing and exporting financial information in popular formats like CSV and JSON. This ensures that users can migrate their data from other platforms or share it with accountants or financial advisors.
- **Mobile Usage**: A growing number of users access financial tools via mobile devices. This trend has led to the decision to build a mobile application using **Flutter**, ensuring a smooth experience on both Android and iOS platforms.

Regulatory and Compliance Factors

The financial industry is heavily regulated, and **Financemate** must adhere to various legal and security requirements, including:

- **GDPR (General Data Protection Regulation)**: The system must comply with data privacy regulations such as GDPR for users in the EU. This means offering users control over their data and ensuring that all sensitive information is handled with care.
- **Data Encryption and Secure Transactions**: As financial data is highly sensitive, **Financemate** will use advanced encryption methods, such as AES, to protect user data at rest and in transit. All financial transactions must be conducted securely, with TLS ensuring that communication is encrypted.
- **Tax Compliance**: For small business owners, **Financemate** will help organize income and expenses in a way that facilitates tax filing. The system will allow users to categorize transactions that are relevant for tax reporting, such as business expenses, deductible items, and sales tax.

Technical and Design Considerations

From a technical perspective, several factors influence the design of **Financemate**:

- **Scalability**: As the user base grows, **Financemate** must be able to scale to accommodate a larger number of users and their data. The architecture must support this growth, ensuring that performance is maintained even as the system becomes more complex.
- **Responsiveness and Cross-Platform Functionality**: The system must function smoothly across a variety of devices and platforms. The web platform will be responsive, adapting to different screen sizes, while the mobile application will provide native functionality on both Android and iOS devices.
- **Integration with External Services**: **Financemate** will integrate with third-party financial services, such as APIs for banking transaction imports (e.g., **Plaid**) and email/SMS services for alerts. This enables automatic transaction tracking and simplifies the user experience.

Potential Limitations and Exclusions

Although **Financemate** will offer a rich set of features, it will not cater to the needs of large-scale enterprises or handle complex financial tasks such as investment management or stock trading. It is not intended for use by large organizations with intricate financial structures. Instead, the system focuses on personal and small business finances, streamlining the essential tasks that everyday users encounter.

2.1 Product Perspective

Financemate is designed as a comprehensive financial management solution primarily targeted at individuals and small businesses. While it is a self-contained product, it is important to place it within the broader context of similar financial tools available in the market. This section explores how **Financemate** compares to other products and systems, outlining its relationships with external systems, components, and interfaces, and providing insight into its design and unique features.

Comparison with Similar Products

There are several established products in the market that offer personal finance management, budgeting, and financial tracking solutions. A few notable examples include:

- **Mint**: A popular personal finance tool that offers budget tracking, bill payment reminders, and credit score monitoring. It integrates with users' bank accounts to automatically track expenses and categorize transactions. **Mint** is known for its intuitive UI and free service model, although it has limited customization options for advanced users.
- **YNAB (You Need A Budget)**: Focused on budget management, YNAB emphasizes proactive budgeting by allocating money to specific categories. It offers detailed financial reports and helps users plan for long-term goals, such as saving and debt repayment. However, YNAB requires manual entry of transactions, making it less automated than some other tools.
- **QuickBooks**: Primarily targeted at small businesses, QuickBooks provides comprehensive financial management tools including income and expense tracking, invoicing, payroll, and tax reporting. It offers powerful features for accounting professionals but can be overwhelming for individuals or smaller businesses due to its complex interface and broad functionality.

Financemate offers a unique blend of features, focusing on simplicity, automation, and personalization, which makes it distinct from competitors like **Mint** or **YNAB**. Key differentiators include:

- **Personalized Insights**: Leveraging machine learning, **Financemate** will offer tailored financial advice and predictions based on user behavior, something that most competitors don't emphasize as strongly.
- **Real-Time Analytics**: Unlike many systems that require manual updates, **Financemate** will provide real-time financial data and alerts, allowing users to track their finances seamlessly across devices.
- **Scalability**: While products like **Mint** are geared mainly toward personal use, **Financemate** is designed to scale for small businesses as well. It offers features suited to both individuals and businesses, providing flexibility based on user needs.

Integration with External Systems

While **Financemate** is a self-contained application, it will interface with several external systems to enhance functionality:

- **Banking APIs (e.g., Plaid)**: To enable automatic import of transaction data, **Financemate** will integrate with banking systems via APIs such as **Plaid**. This allows the system to automatically categorize transactions and track expenses without requiring manual entry.
- **Email/SMS Notification Systems**: For sending reminders about bill payments or financial goals, **Financemate** will utilize external email or SMS services to notify users in a timely manner.
- **Cloud-Based Data Storage (e.g., AWS, Google Cloud)**: Financial data will be securely stored in the cloud, ensuring that users can access their information from any device and ensuring scalability as the user base grows.
- **Payment Gateways**: For managing bill payments, **Financemate** will integrate with payment processors, enabling users to pay bills directly through the platform if desired.

System Architecture Context

While the specific design details are not covered in this document, **Financemate** can be described as a component within a larger financial ecosystem. The system will interact with several external services, such as banking APIs for real-time data access and cloud services for storage and scalability.

- **User Interface (UI)**: Both web and mobile applications where users interact with **Financemate**.
- **Business Logic Layer**: This layer includes features such as transaction categorization, real-time analytics, and personalized insights.
- **External Services**: The external systems **Financemate** interacts with, such as banking APIs for transaction data, email/SMS services for alerts, and cloud storage for data hosting.
- **Data Layer**: The database layer, which houses transaction records and user data, ensuring data integrity and security.

Related Research and Innovations

In the context of the research-oriented nature of **Financemate**, there are several advancements in machine learning and AI-driven financial tools that inform its development:

- **Behavioral Finance and Machine Learning**: Several studies in behavioral finance suggest that using machine learning algorithms can improve financial decision-making by providing tailored advice. **Financemate** will leverage these insights to offer personalized financial recommendations based on user behavior patterns, aligning with these research findings.
- **Predictive Analytics for Personal Finance**: Research on predictive analytics in finance shows that analyzing user spending patterns can predict future financial behavior, offering an opportunity for proactive financial management. **Financemate** will use these insights to provide actionable predictions for users' financial goals and budgeting.

2.1.1 Operations

The operations required by the "Financemate" application are categorized into normal and special operations. These include user interaction modes, data processing requirements, and backup/recovery operations. Below is a detailed description of each aspect:

1. Modes of Operation in the User Environment

- **Interactive Operation:**
  - Users interact with the system through web and mobile platforms, accessing features such as budget tracking, transaction categorization, and financial insights.
  - Operations are user-driven and require an active internet connection for real-time data synchronization.
- **Unattended Operation:**
  - The system performs periodic background tasks such as syncing transaction data with banking APIs, generating financial insights, and updating machine learning models for better predictions.
  - Scheduled operations include daily data backup and analytics generation during off-peak hours to ensure uninterrupted user interaction.

2. **Periods of Interactive and Unattended Operations**

- **Interactive Periods:**
  - The system remains active 24/7, allowing users to access their financial data and perform operations at any time.
  - Peak usage hours are anticipated to be between 6 PM and 11 PM, as users are more likely to engage in financial management activities after work hours.
- **Unattended Periods:**
  - Data synchronization and system updates are scheduled during midnight hours (12 AM to 4 AM) to minimize disruptions to user activity.

    - Automated tasks such as machine learning model retraining and log cleanup are executed weekly during these hours.

3. Data Processing Support Functions

- The system supports the following data processing operations:
    - **Transaction Categorization:**
        - Uses machine learning algorithms to automatically categorize transactions.
    - **Budget Analysis:**
        - Generates insights into spending patterns, allowing users to track and manage budgets effectively.
    - **Custom Reporting:**
        - Enables users to generate detailed reports on financial activities in various formats such as PDF, CSV, and Excel.
    - **API Integration:**
        - Ensures seamless retrieval and processing of financial data from external banking systems via Plaid API or similar services.

4. Backup and Recovery Operations

- **Data Backup:**
    - Regular backups are performed daily during unattended periods to ensure data safety and integrity.
    - Backup copies are stored securely in cloud storage using encryption protocols (AES-256).
- **Recovery Operations:**
    - Implements a robust disaster recovery mechanism to restore user data in case of system failures or data corruption.
    - The recovery process involves validating the backup, restoring data to the last known stable state, and notifying the user about the restoration process.
- **Data Redundancy:**
    - Maintains redundancy through multi-region backup storage to minimize downtime during critical failures.
- **User-Triggered Data Export:**
    - Allows users to manually export their financial data at any time, ensuring personal backup options for users.

By addressing both normal and special operations, "Financemate" ensures seamless functionality, robust data handling, and a secure user experience across diverse scenarios.

2.1.2 Site Adaptation Requirements

The "Financemate" application requires specific site-related configurations and adaptations to ensure smooth installation and operation. These requirements address both hardware and software needs, as well as any necessary preparation at the user’s environment.

1. Data and Initialization Sequences

- **Data Setup:**

    - Users must provide their initial financial data, including past transactions, account details, and budget preferences, for system initialization.
    - The system supports importing data via standard formats (CSV, Excel) or direct API connections to banking platforms.
- **API Configuration:**
    - Users are required to set up and authenticate API connections to their respective banking or financial institutions for real-time data retrieval.
- **Database Initialization:**
    - A cloud-hosted database instance is pre-configured, but users can optionally integrate their own database server, which must meet the following specifications:
        - **DBMS:** MySQL 8.0+, PostgreSQL 13+, or equivalent
        - **Storage:** Minimum of 10 GB of free space
        - **Connectivity:** Secure access with encryption (TLS 1.2+)

2. Site or Mission-Related Features

- **Hardware Requirements:**
    - A stable internet connection is mandatory to ensure real-time data synchronization and API operations.
    - For organizations hosting their own server instance, the following hardware specifications are required:
        - **Processor:** Quad-core CPU (2.5 GHz or higher)
        - **Memory:** 16 GB RAM minimum
        - **Storage:** SSD storage with a minimum read/write speed of 500 MB/s
        - **Backup Power Supply:** A UPS (Uninterruptible Power Supply) with at least 2 hours of backup is recommended to prevent downtime during power outages.
- **Software Requirements:**
    - The system is compatible with Windows (10 or later), macOS (Big Sur or later), and Linux (Ubuntu 20.04+).
    - For mobile platforms, iOS 14+ and Android 10+ are supported.
    - Required third-party software includes:
        - **Node.js 16+** for backend services
        - **Docker** for containerized deployments (if applicable)
        - **Browser Requirements:** Chrome, Firefox, or Safari (latest versions) for web access.

3. Modifications to Customer Work Area

- **Environmental Setup:**
    - No significant modifications to the user environment are required for most cases since "Financemate" is primarily a cloud-based application.
    - Organizations opting for on-premises deployments may need to ensure the following:
        - A secure server room with adequate cooling systems (minimum of 5,000 BTU air conditioning).
        - Firewall configuration to allow required ports (e.g., 443 for HTTPS, 3306 for database access).
- **Pre-Installation Tasks:**

- If integrating with an existing database, users must create and provide empty data tables based on the schema provided during deployment.
- Populate these tables with necessary configuration data (e.g., user roles, default categories for transactions) as part of the initialization process.
- Ensure administrative access is granted to relevant IT personnel for installation and maintenance.

4. Additional Requirements

- **Data Privacy and Security Compliance:**
  - Users must ensure compliance with relevant financial data regulations (e.g., GDPR, CCPA) by providing any required certifications or validations during setup.
- **User Training:**
  - Organizations may need to provide basic training sessions to familiarize users with application workflows and features.

By addressing these site-specific requirements, "Financemate" ensures a seamless setup and operational experience, minimizing potential disruptions.

2.2 Product Functions

The **Financemate** application is designed to simplify financial management for individuals and small businesses. The major functions are categorized for clarity and described in terms easily understandable to the customer. Below is an overview of these functions and their relationships.

1. Budget Management

- **Create and Manage Budgets:**
  - Users can create customized budgets based on specific time frames (monthly, quarterly, annual).
  - Categorization of income and expenses into predefined or custom categories.
- **Real-Time Tracking:**
  - Automatic tracking of spending against the allocated budget in real-time.
  - Notifications for overspending or nearing limits in specific categories.
- **Reports and Analytics:**
  - Graphical and tabular representations of budget utilization.
  - Insights into spending patterns and saving opportunities.

2. Expense and Income Tracking

- **Manual and Automated Entry:**
  - Users can manually log transactions or enable automatic syncing with bank accounts.
- **Categorization:**
  - Automatic categorization of transactions using AI/ML algorithms or manual adjustments.
- **Recurring Transactions:**

    - Support for tracking and managing recurring payments like subscriptions or salaries.
- **Transaction History:**
    - Comprehensive history with filtering and searching capabilities.

3. Goal Setting and Financial Planning

- **Savings Goals:**
    - Users can set financial goals (e.g., saving for a vacation or emergency fund).
    - Progress tracking against the goal based on automated savings or user inputs.
- **Debt Management:**
    - Tools for planning and tracking debt repayments.
    - Suggestions for optimizing repayments to minimize interest.

4. Real-Time Synchronization

- **Bank Integrations:**
    - Secure integration with major financial institutions for real-time data synchronization.
    - Support for multi-currency transactions and international accounts.
- **Device Synchronization:**
    - Seamless synchronization across multiple devices (web, mobile, and desktop).

5. User Management and Security

- **Multi-User Support:**
    - Role-based access for multiple users (e.g., family members or business partners).
- **Two-Factor Authentication (2FA):**
    - Enhanced security through optional 2FA for sensitive operations.
- **Data Encryption:**
    - End-to-end encryption for all financial data stored or transmitted.

6. Reporting and Insights

- **Comprehensive Financial Reports:**
    - Generate detailed financial reports for specific time frames or accounts.
    - Export options in multiple formats (PDF, CSV, Excel).
- **AI-Powered Insights:**
    - Predictive analytics for potential overspending or savings opportunities.
    - Personalized financial tips and advice.

7. Integration and Extensibility

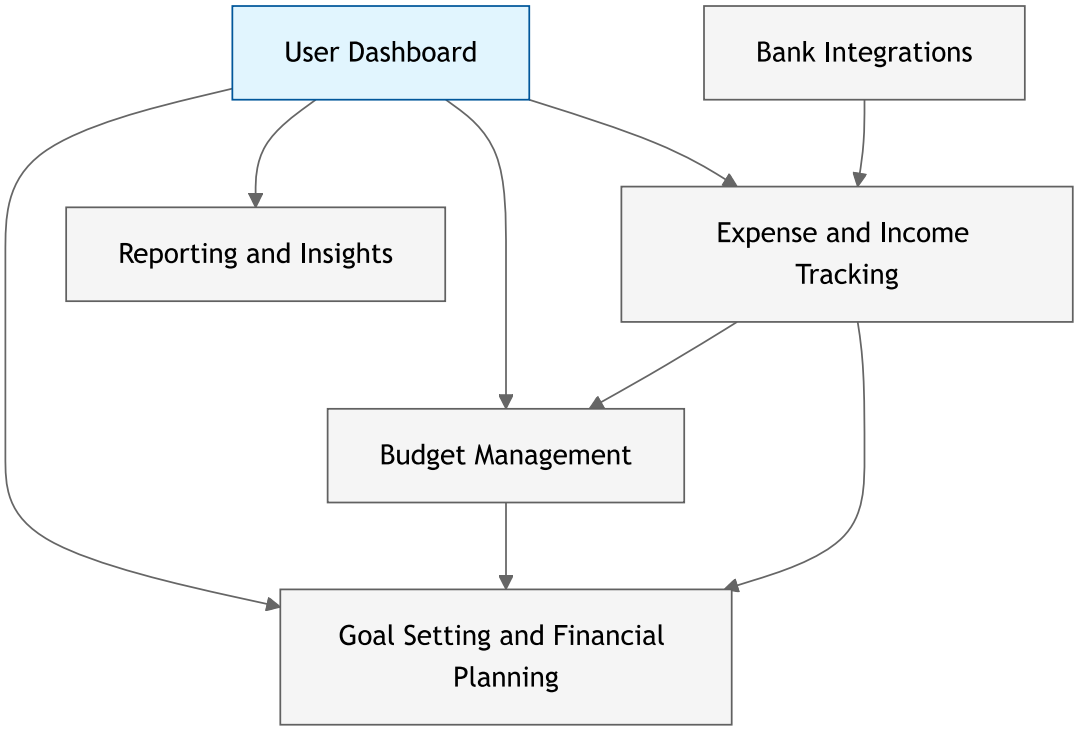
- **Third-Party Tools:**
    - Integration with accounting software like QuickBooks or Xero for businesses.
- **APIs for Developers:**
    - Open API access for developers to extend functionality or integrate with other tools.

8. Notifications and Alerts

- **Real-Time Alerts:**
  - Notifications for upcoming payments, low balances, or irregular spending.
- **Customizable Reminders:**
  - Alerts for important financial tasks such as filing taxes or budget reviews.

Logical Relationship Diagram

Below is a graphical representation of the major functions and their interdependencies:



2.3 User Characteristics

User Characteristics

The **Financemate** application is designed to serve a diverse user base, encompassing both individual users and small business owners. The characteristics of the intended users are as follows:

1. General Characteristics

- **Educational Background:**
  - The majority of users are expected to have a basic understanding of financial concepts but may not necessarily possess advanced education in finance or accounting.

    - Some users, particularly small business owners, may have practical experience managing budgets but may lack formal financial training.
- **Technical Expertise:**
    - The user base includes both tech-savvy individuals and those with minimal technical expertise. As such, the application must prioritize ease of use and provide guidance where necessary.
    - Users may have experience with basic tools like spreadsheets but may not be familiar with advanced financial software or analytics.

2. Experience with Financial Management

- **Individual Users:**
    - May include working professionals, students, or retirees managing personal or household finances.
    - Likely to have experience with basic budgeting and expense tracking but may require additional support for goal setting or advanced insights.
- **Small Business Owners:**
    - Typically handle day-to-day financial tasks, including expense tracking, revenue recording, and basic reporting.
    - May require additional tools for integrating with accounting software or handling multi-user environments.

3. Behavioral Characteristics

- **Frequency of Use:**
    - Users are expected to interact with the system frequently but for short durations (e.g., daily budget updates, weekly reviews).
    - Small business users may use the system during regular working hours, whereas individual users may interact during evenings or weekends.
- **Goals and Expectations:**
    - Users seek simplicity, speed, and accuracy in financial management tasks.
    - Expect real-time feedback and actionable insights to make informed financial decisions.

4. Potential Design Impact

- **UI/UX Design:**
    - Interfaces must be intuitive and accessible, using clear language and visuals to assist users with varying levels of expertise.
    - A tiered design approach (basic and advanced options) may be beneficial to cater to both beginners and experienced users.
- **Guidance and Support:**
    - The system should include contextual help, tutorials, and an FAQ section to support users with limited experience in financial management.
- **Accessibility:**
    - Design must consider accessibility standards, ensuring inclusivity for users with disabilities or limited access to high-end devices.

By understanding these user characteristics, the design of **Financemate** will aim to create a user-friendly, inclusive, and adaptable financial management platform that meets the needs of its diverse user base.

2.4 General Constraints

The following constraints define non-functional requirements and limitations that influence the development of the "Financemate" system:

1. **Regulatory Policies:**
   - The system must comply with financial data privacy and security regulations such as GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act).
   - Integration with banking systems must adhere to Open Banking standards and API compliance requirements.
2. **Hardware Limitations:**
   - The application will be designed to run efficiently on devices with at least 2 GB of RAM and dual-core processors.
   - Mobile devices must meet the minimum specifications: Android 9.0 (Pie) or iOS 14.0 and above.
3. **Interface to Other Applications:**
   - The system integrates with external financial systems using APIs such as Plaid for banking data retrieval.
   - Compatibility with data import/export formats like CSV and Excel will ensure smooth integration with third-party applications.
4. **Parallel Operation:**
   - The system must support simultaneous user sessions across multiple devices with real-time data synchronization.
5. **Audit Functions:**
   - Transaction logs must be maintained for all user actions to support transparency and debugging.
   - The application will include an activity log feature to track login attempts, data updates, and other critical user operations.
6. **Control Functions:**
   - Users will have role-based access control to ensure proper segregation of functionality (e.g., admin vs. standard user roles).
7. **Higher-Order Language Requirements:**
   - The system will primarily use Dart for development, ensuring compatibility with the Flutter framework.
8. **Signal Handshake Protocols:**
   - Secure API communication will use standard protocols such as HTTPS, OAuth 2.0 for authentication, and JSON-based RESTful API interaction.
9. **Reliability Requirements:**
   - The system must maintain 99.9% uptime in production environments to ensure reliable access to financial data.
   - Automated failover and recovery mechanisms must be implemented for cloud-based database services.
10. **Criticality of the Application:**

- As a financial management tool, the application is critical for users' financial planning. Therefore, real-time accuracy of data and robust security are paramount.

11. **Safety and Security Considerations:**
    - User data must be encrypted using AES-256 during both storage and transmission.
    - Multi-factor authentication (MFA) is required to enhance user account security.
    - The system must regularly update its security protocols to address emerging vulnerabilities and threats.

2.5 Assumptions and Dependencies

This section outlines the factors that may influence the requirements of the "Financemate" system. These factors are not direct design constraints but could necessitate changes to the Software Requirements Specification (SRS) if altered.

1. **Operating System Availability:**
    - The system assumes the availability of Android and iOS operating systems on the intended mobile devices. Changes in the support lifecycle or compatibility with these platforms could affect the requirements.
2. **Third-Party API Dependencies:**
    - The application depends on APIs such as Plaid for financial data aggregation and Stripe for payment processing. Any modifications, deprecations, or service outages of these APIs will require updates to the SRS.
3. **Cloud Services:**
    - The application assumes the availability and reliability of cloud platforms such as AWS or Google Cloud for hosting backend services and databases. Disruption in these services could affect system operations.
4. **Development Tools and Frameworks:**
    - The system assumes the continued support of the Flutter framework and Dart programming language. Major changes or discontinuation of these tools would require adjustments in the development process and potentially the system architecture.
5. **User Device Capabilities:**
    - The application is designed with the assumption that user devices will meet minimum hardware and software specifications. A significant shift in device trends (e.g., lower specs or alternative operating systems) may necessitate redefinition of requirements.
6. **Data Connectivity:**
    - The system assumes users have access to a stable internet connection to perform most functions. Changes in connectivity norms (e.g., offline-first user needs) could alter system design and requirements.
7. **Regulatory Landscape:**
    - Assumes current financial and data security regulations (e.g., GDPR, CCPA) remain applicable. New or amended regulations could require adjustments in how user data is collected, processed, and stored.
8. **User Base Expectations:**

- The system assumes users are familiar with digital financial tools and mobile app usage. A shift toward a user base requiring extensive accessibility features or simplified interfaces could influence design priorities.

9. **Hardware Ecosystem Evolution:**
   - The system assumes no significant shifts in mobile device hardware ecosystems, such as the widespread adoption of foldable screens or new interaction paradigms (e.g., AR-based interfaces).
10. **Data Sources and Formats:**
    - The system assumes consistent access to structured financial data formats (e.g., CSV, JSON) from external systems. Changes in data source availability or format standardization could affect integration requirements.
11. **Market Trends:**
    - The design assumes that financial management apps remain relevant and in demand. A shift in market trends or the emergence of disruptive technologies could necessitate re-evaluation of system objectives and features.
12. **Security and Authentication Standards:**
    - The system relies on established security protocols like AES-256, HTTPS, and OAuth 2.0. Updates or breakthroughs in cybersecurity (e.g., quantum encryption) could impact the system's security framework.

3. Specific Requirements

This section details the specific requirements for the project, ensuring they are correct, traceable, unambiguous, verifiable, prioritized, complete, consistent, and uniquely identifiable. The requirements are organized for accessibility and clarity without imposing constraints that belong to the design phase.

Functional Requirements

1. **User Authentication**
   - **3.1.1:** Users must be able to register using an email, phone number, or social media account.
   - **3.1.2:** The system must provide multi-factor authentication for secure login.
2. **Dashboard and Reporting**
   - **3.2.1:** The system must display an interactive dashboard summarizing key performance metrics.
   - **3.2.2:** Users must be able to generate downloadable reports in PDF and Excel formats.
3. **Data Processing**
   - **3.3.1:** The system must process input data within 1 second per 1,000 entries.
   - **3.3.2:** It must detect and notify users of errors in data input in real time.
4. **Third-Party Integration**
   - **3.4.1:** APIs must enable integration with [specific tools/services].
   - **3.4.2:** All interactions with external systems must be logged for auditing.
5. **Notifications and Alerts**
   - **3.5.1:** Email and SMS notifications must be sent for predefined events.
   - **3.5.2:** The system must support user-configured thresholds for alerts.

Non-Functional Requirements

1. **Performance**
    - **3.6.1:** The system must handle up to 10,000 concurrent users with response times <2 seconds.
    - **3.6.2:** Data synchronization with third-party services must occur within 5 minutes.
2. **Security**
    - **3.7.1:** Sensitive data must be encrypted in transit (TLS 1.3) and at rest (AES-256).
    - **3.7.2:** The system must comply with GDPR and ISO 27001 standards.
3. **Reliability**
    - **3.8.1:** The system must maintain 99.9% uptime, excluding planned maintenance.
4. **Scalability**
    - **3.9.1:** The system must support scaling horizontally to accommodate increased loads.
5. **Usability**
    - **3.10.1:** The interface must meet WCAG 2.1 AA accessibility standards.
6. **Maintainability**
    - **3.11.1:** Code documentation must meet [specific style guide] standards, with a minimum coverage of 80%.

Traceability Matrix

Each requirement is mapped to its corresponding customer requirement:

- **3.1.1** ↔ CR-1
- **3.2.1** ↔ CR-4
- **3.7.1** ↔ CR-8

This structured approach ensures that requirements serve as a clear foundation for the software's design, development, and testing, aligning with the overall project goals.

3.1 External Interface Requirements

This section specifies the external interfaces the system must support for interaction with users, hardware, and other software systems.

3.1.1 System Interfaces

• Database System Interface

- **Functionality**: The software interacts with a relational database (e.g., MySQL or PostgreSQL) to store and retrieve user data, project configurations, and logs.
- **Interface Description**: Communication is achieved via SQL queries using a pre-defined schema. The application uses an ORM (e.g., Hibernate or SQLAlchemy) for seamless integration.

• User Authentication System

- **Functionality**: Integrates with a third-party authentication provider (e.g., OAuth2 with Google or Firebase) to handle user login and registration.
- **Interface Description**: REST API endpoints provided by the authentication system are used for token generation and validation.

- **Payment Gateway** (if applicable)

  - **Functionality**: Facilitates online payments for premium features or services.
  - **Interface Description**: Uses the payment provider's SDK or REST API (e.g., Stripe or PayPal) to process transactions securely.

- **External API Integration** (e.g., Weather Data, Maps API)

  - **Functionality**: Retrieves real-time data from external APIs to enrich the user experience.
  - **Interface Description**: Communicates via RESTful APIs with JSON payloads over HTTPS, following the provider's API documentation.

- **Hardware Interfaces** (if hardware is involved)

  - **Functionality**: Interfaces with devices such as sensors or IoT hardware for data collection and processing.
  - **Interface Description**: Communication occurs via standard protocols like Bluetooth, UART, or MQTT, depending on the hardware.

3.1.2 Interfaces

- **Logical Characteristics of User Interfaces**

  - The software will provide a **Graphical User Interface (GUI)** to interact with end-users.
  - The GUI will include intuitive navigation menus, form inputs, interactive dashboards, and visual feedback to guide user actions effectively.
  - For advanced users, a **Command Line Interface (CLI)** will be available to enable scriptable interactions or automation of tasks.

- **Optimization for User Interaction**

  - The GUI will be designed using **responsive design principles** to ensure usability across devices (desktop, tablet, and mobile).
  - Consistent design patterns will be followed (e.g., Material Design or Apple's Human Interface Guidelines) to reduce the learning curve for users.
  - **Accessibility Features**:
    - Support for screen readers and high-contrast modes to comply with **ADA** requirements.
    - Keyboard shortcuts and focus indicators for seamless navigation without a mouse.
    - Adjustable text size and color schemes to accommodate users with visual impairments.

- Multilingual support will be considered for diverse user demographics.

• Special Interface Requirements

- **Error Prevention and Feedback**: The interface will provide real-time validation, clear error messages, and undo options for critical actions.
- **User Customization**: Users will be able to customize the interface, such as rearranging widgets or toggling dark mode.
- **Onboarding and Help**: Step-by-step tutorials, tooltips, and a dedicated help section will guide first-time users through the system.

3.1.3 Hardware Interfaces

The system has no hardware interface requirements.

This project is purely a software solution that does not interact with or control any external hardware devices. All functionalities and processes are confined to the software environment, relying solely on standard computing infrastructure (e.g., desktops, laptops, or mobile devices) and their built-in peripherals (e.g., keyboard, mouse, touchscreen).

If future iterations require hardware interactions, such as connecting to external devices or IoT hardware, detailed specifications and protocols will be provided.

3.1.4 Software Interfaces

• Required Software Products

- **Name**: MySQL
  - **Mnemonic**: MYSQL
  - **Specification Number**: N/A
  - **Version Number**: 8.0.33
  - **Source**: MySQL Official Website
- **Name**: Google Maps API
  - **Mnemonic**: GMA
  - **Specification Number**: N/A
  - **Version Number**: Latest supported version
  - **Source**: Google Cloud Platform
- **Name**: Flutter SDK
  - **Mnemonic**: FSDK
  - **Specification Number**: N/A
  - **Version Number**: 3.10.6
  - **Source**: Flutter Official Website

• Interface Details

- **MySQL**
  - **Purpose**: Serves as the primary database for storing and retrieving application data, such as user information, transactions, and other backend content.
  - **Message Content and Format**:

        - **Content**: SQL queries for Create, Read, Update, and Delete (CRUD) operations.
        - **Format**: Structured Query Language (SQL) commands sent through a backend server using an HTTP API or direct database connection.
- **Google Maps API**
    - **Purpose**: Enables map-based functionalities such as location visualization, route planning, and geolocation services.
    - **Message Content and Format**:
        - **Content**: API requests to fetch geolocation data, maps, and markers.
        - **Format**: RESTful API requests using HTTP with JSON responses.
- **Flutter SDK**
    - **Purpose**: The core development framework for building the mobile application, supporting platform-independent UI development and deployment.
    - **Message Content and Format**:
        - **Content**: Integration with platform-specific rendering engines and plugins.
        - **Format**: Pre-defined Dart packages and APIs for cross-platform compatibility.

Microsoft SQL Server 7

• Required Software Products

- **Name**: MySQL
    - **Mnemonic**: MYSQL
    - **Specification Number**: N/A
    - **Version Number**: 8.0.33 (or as specified by the customer)
    - **Source**: MySQL Official Website

• Interface Details

- **MySQL**
    - **Purpose**: Serves as the database component for the application, storing all required data including user profiles, transactions, and application-specific information. Communication with the database is facilitated through structured SQL queries.
    - **Message Content and Format**:
        - **Content**: CRUD (Create, Read, Update, Delete) operations defined in SQL syntax.
        - **Format**: Communication via ODBC (Open Database Connectivity) or a direct database connector compatible with the selected backend framework (e.g., MySQL Connector for Python or Java).
    - **Setup Details**:
        - The system must provide database schema definitions, including table structures, relationships, and stored procedures, for the database administrator (DBA) to review and set up.
    - **Constraints**:

- If MySQL is a customer-specified requirement, the system cannot use alternative database technologies. If not specified, it may be treated as a design choice.

3.1.5 Communications Interfaces

1. **Local Network Protocols**
   - **Protocol**: HTTP/HTTPS
     - **Purpose**: Communication between the Flutter application and the backend server for data exchange.
     - **Details**:
       - Standard RESTful APIs will be used for client-server communication.
       - HTTPS is mandated for secure transmission of sensitive data.
       - Follows standard HTTP status codes and JSON response format for error handling and data exchange.
2. **Authentication Protocols**
   - **Protocol**: OAuth 2.0
     - **Purpose**: Secure authentication for user access and token-based session management.
     - **Details**:
       - Access tokens are passed in the Authorization header using the Bearer scheme.
       - Refresh tokens are implemented for long-lived sessions.
3. **Push Notification Protocol**
   - **Protocol**: Firebase Cloud Messaging (FCM)
     - **Purpose**: Push notifications to communicate updates and alerts to users.
     - **Details**:
       - Device tokens will be registered and managed via FCM SDK for Flutter.
       - Notifications will include both data messages and display notifications, based on the user's action.
4. **Database Communication**
   - **Protocol**: MySQL Connector Protocol
     - **Purpose**: Establish a secure connection between the backend and the MySQL database.
     - **Details**:
       - Communication occurs over standard database ports (3306 for MySQL).
       - Connections will be encrypted to ensure data security.
5. **Third-Party API Communication**
   - **Protocol**: REST over HTTPS
     - **Purpose**: Interaction with external APIs for services like payment gateways or external data providers.
     - **Details**:
       - All interactions will comply with the third-party API documentation and standards.
       - Error handling and rate-limiting measures will be implemented to manage external API limits.

3.2 Functional Requirements

1. **User Authentication and Authorization**
    - **Feature**:
        - The system shall allow users to register, log in, and log out.
        - The system shall validate user credentials using OAuth 2.0.
    - **Use Case**:
        - As a user, I want to log in securely so that I can access my personalized features.
        - As an admin, I want to manage user roles to control access levels.
2. **Dashboard and Analytics**
    - **Feature**:
        - The system shall provide a user dashboard displaying personalized data and analytics.
        - The system shall update data in real-time through backend API calls.
    - **Use Case**:
        - As a user, I want to view my performance metrics in a graphical format to track my progress.
3. **Data Management**
    - **Feature**:
        - The system shall allow CRUD (Create, Read, Update, Delete) operations on specific datasets.
        - Changes to data shall be reflected immediately across the application.
    - **Use Case**:
        - As a user, I want to add or edit records to keep my data up-to-date.
4. **Notifications**
    - **Feature**:
        - The system shall send push notifications to alert users of critical updates or deadlines.
        - Users shall have the ability to enable or disable notifications in their settings.
    - **Use Case**:
        - As a user, I want to receive alerts about important events so that I do not miss deadlines.
5. **Search Functionality**
    - **Feature**:
        - The system shall implement a search bar to query specific records or items.
        - Search results shall be displayed with relevant filters and sorting options.
    - **Use Case**:
        - As a user, I want to quickly find information using keywords for faster access.
6. **Payment Integration (if applicable)**
    - **Feature**:
        - The system shall integrate with a payment gateway (e.g., Stripe or PayPal) for secure transactions.
        - Payment receipts shall be sent to users via email.
    - **Use Case**:

        - As a user, I want to make payments securely and receive confirmation of transactions.
7. **Reporting**
    - **Feature**:
        - The system shall generate downloadable reports based on user activity and data.
        - Reports shall be available in multiple formats such as PDF and Excel.
    - **Use Case**:
        - As an admin, I want to generate activity reports to analyze trends.
8. **Accessibility**
    - **Feature**:
        - The system shall comply with WCAG (Web Content Accessibility Guidelines) to ensure usability for differently-abled users.
    - **Use Case**:
        - As a differently-abled user, I want to navigate the system easily using assistive technologies.

3.2.1 User Authentication and Authorization

Introduction

This feature enables users to securely log in, log out, and manage their account credentials. It ensures only authorized users access the system's resources.

Inputs

- User credentials (username/email and password).
- OAuth token for third-party authentication (if applicable).

Processing

- Validate user credentials against the database.
- Generate and store a session token upon successful authentication.
- Verify user roles and permissions for access control.

Outputs

- Success: Logged-in user redirected to the dashboard.
- Failure: Error message indicating incorrect credentials or unauthorized access.

Error Handling

- Incorrect credentials: Display "Invalid username or password" message.
- Token expiration: Prompt the user to log in again.
- Unauthorized access: Display "Access denied" message.

3.2.2 Dashboard and Analytics

Introduction

This feature provides users with a personalized dashboard showcasing data visualizations and performance metrics in real time.

Inputs

- User ID or session token to fetch personalized data.
- Filter and sort options selected by the user.

Processing

- Query the database for user-specific data.
- Process data to generate visualizations (e.g., charts or tables).
- Update data dynamically via backend API calls.

Outputs

- Display data visualizations and metrics on the user dashboard.
- Provide options to export data or reports.

Error Handling

- Data retrieval failure: Display "Unable to fetch data" message.
- Visualization issues: Display placeholder content with "Data unavailable" message.

3.2.3 Data Management

Introduction

This feature allows users to manage their data by performing CRUD operations on specific records.

Inputs

- Data entries provided by the user (e.g., text fields, dropdowns).
- IDs for records to be updated or deleted.

Processing

- Validate user inputs for consistency and completeness.
- Interact with the database to create, read, update, or delete records.
- Reflect changes across the application in real time.

Outputs

- Success: Confirmation messages for successful CRUD operations.
- Failure: Error messages for invalid inputs or failed operations.

Error Handling

- Invalid input: Highlight errors and prompt the user to correct them.
- Database connection failure: Display "Unable to process request" message.

...

3.3 Use Cases

This section outlines the key use cases of the system, describing how users interact with the system to achieve their goals.

3.3.1 Use Case #1: User Login

Actors: End-user, System
Preconditions:

- The user must have valid login credentials (username/email and password).
- The system is online and operational.

Steps:

1. The user navigates to the login screen.
2. The user enters their username/email and password.
3. The user clicks the "Login" button.
4. The system validates the credentials against the database.
5. If the credentials are correct, the system grants access and redirects the user to the dashboard.

Postconditions:

- The user is logged into the system and can access authorized features.

Exceptions:

- If the credentials are incorrect, the system displays an "Invalid username or password" message.
- If the system is offline, a "Service unavailable" message is displayed.

3.3.2 Use Case #2: Data Upload

Actors: End-user, System
Preconditions:

- The user must be logged into the system.
- The system must have storage space available.

Steps:

1. The user navigates to the "Upload Data" section.
2. The user selects a file to upload (e.g., CSV, Excel).
3. The user clicks the "Upload" button.
4. The system validates the file format and processes the data.
5. If successful, the system stores the data and displays a confirmation message.

Postconditions:

- The uploaded data is stored in the database and visible to the user.

Exceptions:

- If the file format is invalid, the system displays an "Unsupported file format" error.
- If the upload fails, the system displays a "File upload failed" message.

3.4 Classes / Objects

This section describes the major classes and objects in the system, including their attributes, functions, and references to relevant functional requirements and use cases.

3.4.1 User Class

Attributes

- userId: A unique identifier for the user (integer).
- username: The name of the user (string).
- email: The email address of the user (string).
- password: Encrypted user password (string).
- role: The role of the user, e.g., admin or regular user (string).

Functions

- login(): Validates user credentials and establishes a session.
- logout(): Terminates the user session.
- updateProfile(): Allows the user to modify their profile information.
- getRole(): Retrieves the user's role for access control.

References:

- Functional Requirement: User Login (3.2.1)
- Use Case: User Login (3.3.1)

3.4.2 <FileUpload Class>

Attributes

- fileId: A unique identifier for the uploaded file (integer).
- fileName: The name of the uploaded file (string).
- fileType: The type of file (string, e.g., CSV, Excel).
- fileSize: The size of the file in bytes (integer).
- uploadedBy: The user who uploaded the file (User object reference).
- uploadDate: The timestamp of when the file was uploaded (DateTime).

Functions

- validateFileFormat(): Checks if the file format is supported.
- processFile(): Reads and processes the file content.
- storeFile(): Saves the file to the database or storage.
- getFileDetails(): Retrieves metadata about the uploaded file.

References:

- Functional Requirement: Data Upload (3.2.2)
- Use Case: Data Upload (3.3.2)

3.5 Non-Functional Requirements

This section outlines the system-wide attributes and constraints that must be achieved for successful implementation.

3.5.1 Performance

- The system must process 95% of transactions within 2 seconds.
- File upload and processing should not exceed 5 seconds for files under 10MB.
- The user interface should respond to interactions within 300 milliseconds.

3.5.2 Reliability

- The system should achieve a Mean Time Between Failures (MTBF) of at least 30 days.
- Error rates in data uploads must not exceed 0.01%.
- Recovery from failures should occur automatically within 5 minutes.

3.5.3 Availability

- The system must be available 99.9% of the time, with downtime not exceeding 1 hour per month.
- Scheduled maintenance should occur outside of business hours and be communicated 48 hours in advance.

3.5.4 Security

- User data must be encrypted using AES-256 encryption both in transit and at rest.
- Multi-factor authentication (MFA) must be required for all administrative users.
- The system must comply with GDPR and other applicable data privacy regulations.
- Unauthorized access attempts must be logged and reported within 10 seconds.

3.5.5 Maintainability

- The system should allow for modular updates without affecting unrelated components.
- Code must adhere to standardized coding practices to ensure readability and maintainability.
- Bug fixes should be implemented within 5 working days of identification.

3.5.6 Portability

- The application must run seamlessly on Windows, macOS, and Linux operating systems.

- The system should support major browsers, including Chrome, Firefox, and Edge, with no performance degradation.
- Migration to a different database platform should require minimal changes and must be completed within 2 weeks.

3.6 Inverse Requirements

This section outlines actions or features that the system must explicitly *avoid* to ensure usability, performance, and compliance with project goals.

Avoid Data Duplication

- The system must not store redundant copies of user data unless explicitly required for backups or caching.

No Overloading of User Interface

- The user interface must not include unnecessary elements that could confuse or overwhelm users.

Restrict Unauthorized Access

- The system must not allow any unverified or unauthorized user to access sensitive data or perform administrative functions.

Prevent Resource Overuse

- The application must not overuse device resources such as CPU, memory, or network bandwidth, which could degrade the performance of the user's device.

Avoid Vendor Lock-In

- The system must not be designed to depend on proprietary technologies that prevent future scalability or portability to other platforms.

No Hardcoding of Configuration Data

- Critical configuration values (e.g., database credentials, API keys) must not be hardcoded in the application code.

Restrict Unsupported File Formats

- The system must not accept or process unsupported file formats, which could introduce errors or security vulnerabilities.

Avoid Unnecessary Downtime

- The system must not enter a maintenance or offline state without prior notice to users unless for critical emergency repairs.

3.7 Logical Database Requirements

This section outlines the logical structure and requirements for the data managed by the

system.

Types of Information Used by Various Functions

- **User Information: Includes username, password (hashed), email, and contact details, used for authentication and communication.**
- **Content Data: Information related to user-generated content, such as posts, images, or uploaded files, stored along with metadata like timestamps and tags.**
- **Transaction Logs: Tracks user activities, errors, and other system events for auditing and debugging.**

Frequency of Use

- **User Authentication Data:** Accessed frequently during login or session validation.
- **Content Data:** Accessed frequently for browsing and less frequently for updates.
- **Transaction Logs:** Accessed infrequently, mainly for audits or during error investigations.

Accessing Capabilities

- **Admin Access:** Full read/write access to all data for system management purposes.
- **User Access:** Restricted read/write access to their own data and permitted shared content.
- **Audit Access:** Read-only access to logs for monitoring and compliance purposes.

Data Entities and Their Relationships

- **User:** Central entity related to Content (one-to-many) and Logs (one-to-many).
- **Content:** Related to Tags (many-to-many) for categorization.
- **Tags:** Independent entity for organizing and filtering content.

Integrity Constraints

- **Primary Keys:** Unique identifiers for each entity (e.g., UserID, ContentID, TagID).
- **Foreign Keys:** Relationships between entities (e.g., UserID in Content table).
- **Validation Rules:** Enforce proper email formats, strong passwords, and non-nullable

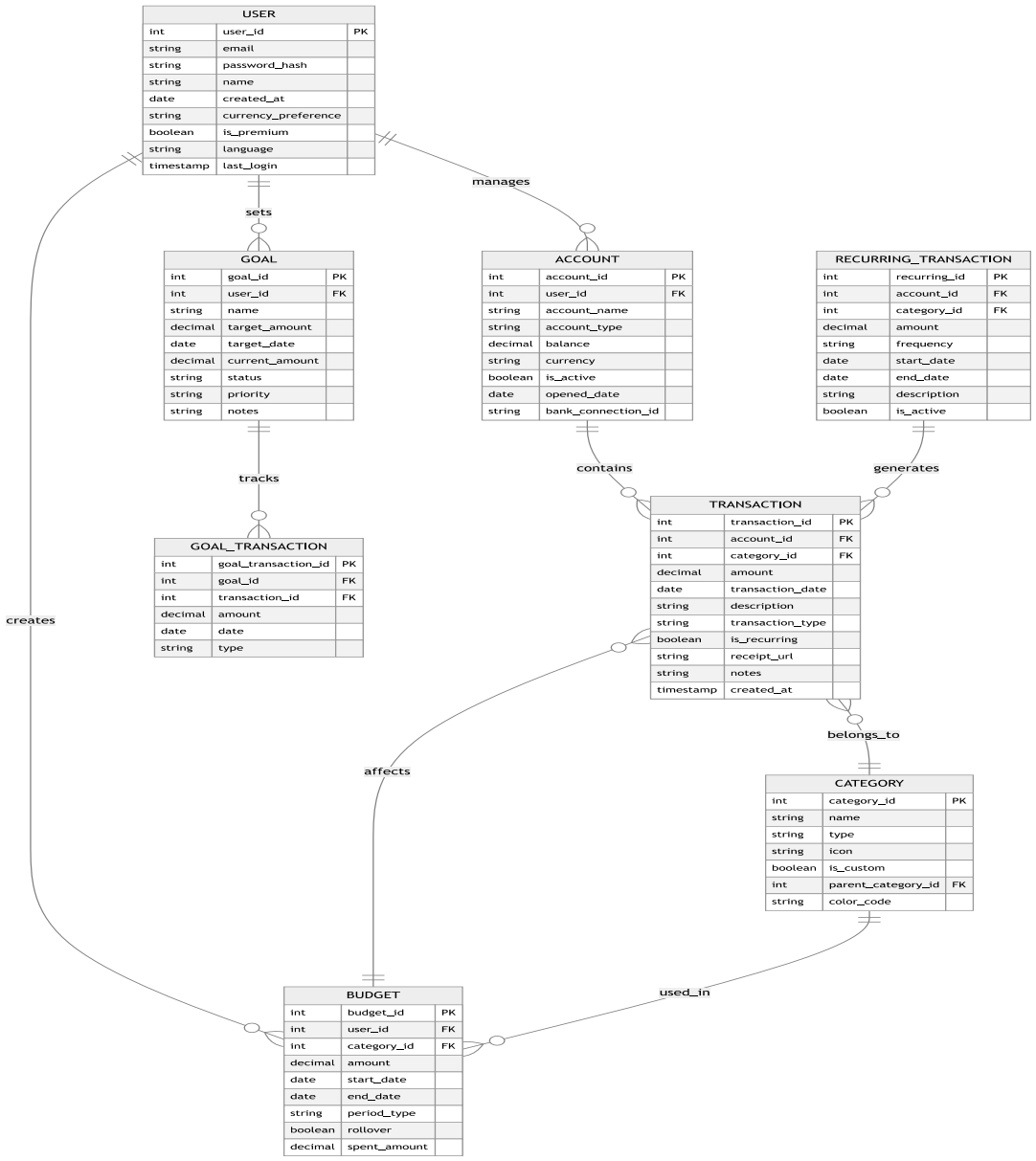
fields where required.

Data Retention Requirements

- **User Data**: Retained until account deletion, with an additional 30 days for potential recovery.
- **Content Data**: Retained indefinitely unless explicitly deleted by the user.
- **Logs**: Retained for a maximum of 1 year unless required longer for compliance.

ER Diagram

An Entity-Relationship (ER) diagram can visually represent the above relationships and constraints for better understand**.**



3.8 Design Constraints

The system design is subject to the following constraints due to external standards, hardware limitations, and project-specific requirements:

1. **Hardware Limitations**
    - The application must run efficiently on devices with a minimum of 2 GB RAM and dual-core processors.
    - The system should ensure compatibility with both Android and iOS platforms, leveraging Flutter's cross-platform capabilities.
2. **Regulatory Compliance**
    - The system must adhere to GDPR for data protection and privacy if used within the European Union.
    - If financial transactions are involved, the application must comply with PCI-DSS (Payment Card Industry Data Security Standard).
3. **Software Environment**
    - The backend must be compatible with MySQL 8.0 or higher, ensuring database features like JSON storage and advanced indexing are supported.
    - APIs must adhere to RESTful standards for communication between client and server components.
4. **Performance Constraints**
    - The application should maintain responsiveness, with no operation exceeding a latency of 1 second under normal load conditions.
5. **UI/UX Guidelines**
    - The design must follow Material Design guidelines for consistency across platforms and ease of use.
6. **Integration Requirements**
    - The application must seamlessly integrate with third-party authentication providers such as Google and Facebook for user login.
7. **Security Constraints**
    - All data transmitted between the client and server must be encrypted using

SSL/TLS protocols.

- Sensitive data, such as passwords, must be hashed before storage using a secure algorithm like bcrypt or Argon2.

3.8.1 Standards Compliance

The system must comply with the following standards and regulations to ensure consistency, transparency, and accountability:

1. **Report Format**

    - All generated reports must be exported in PDF format, with clear headings, timestamps, and unique identifiers for traceability.

    - Financial reports must conform to International Financial Reporting Standards (IFRS) for global applicability.

2. **Data Naming**

    - Database entities must follow a consistent naming convention: singular nouns in lowercase, separated by underscores (e.g., user_profile, transaction_record).

    - Variables and attributes in the codebase must adhere to camelCase for readability and maintainability.

3. **Accounting Procedures**

    - All monetary transactions must log both the debit and credit entries, ensuring double-entry accounting practices are adhered to.

    - Financial data must include precise timestamps and user identifiers to track

activities.

4. **Audit Tracing**

   - The system must maintain an audit log for all critical operations, including
   - user logins, data modifications, and transactions.
   - **Audit logs must include:**
     - Before and after states of the data.
     - Timestamps of changes.
     - User identifiers for traceability.
   - **Audit logs should be immutable and encrypted for security.**

These compliance measures ensure the system adheres to necessary standards while maintaining transparency, security, and usability.

4. Analysis Models

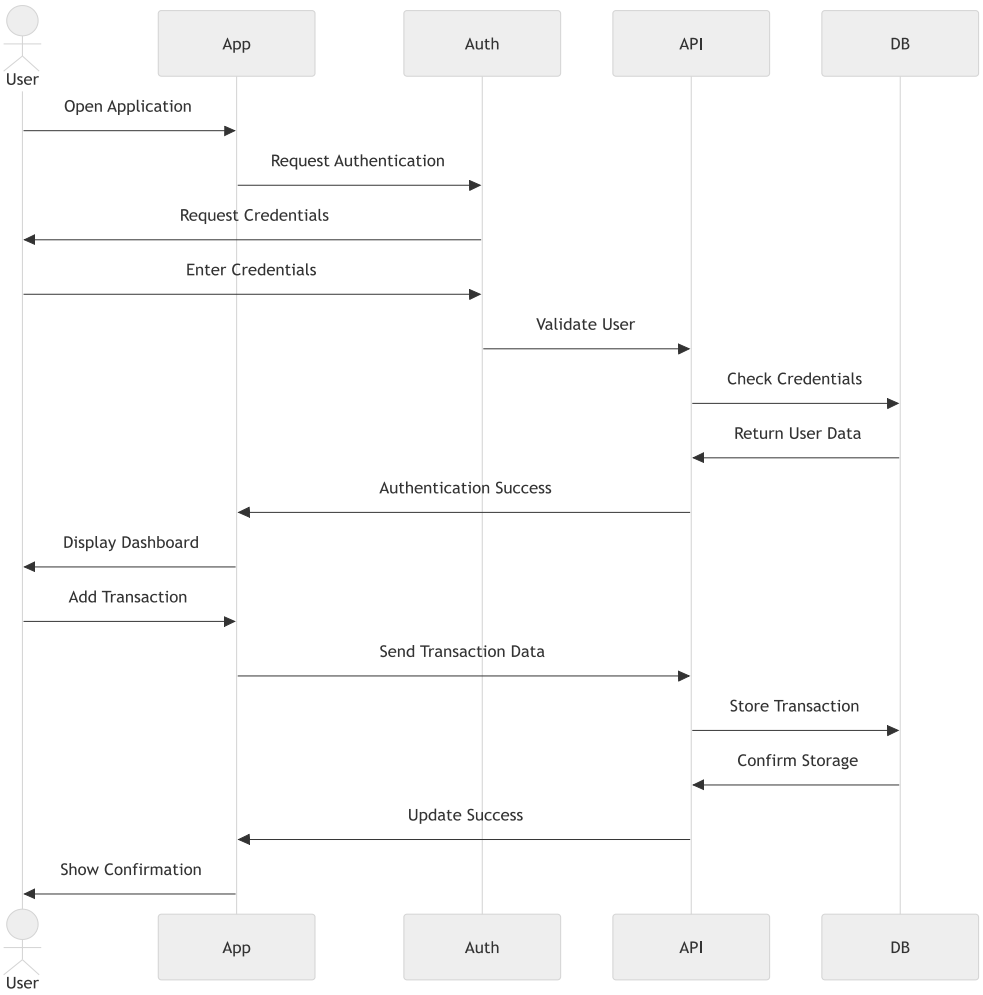
This section outlines the analysis models used to derive and validate the specific requirements stated in the SRS. Each model is introduced with a narrative description and is traceable to relevant requirements for clarity and alignment.

4.1 Sequence Diagrams

Introduction:
Sequence diagrams illustrate the interactions between users and system components over time, ensuring that the system's functional requirements are accurately reflected in its behaviour.

Narrative Description:

- **User Authentication**:
  - Describes the steps for user login, including credential validation and access control.
  - Related Requirements: 3.2.1.1 (Login), 3.5.4 (Security).
- **Transaction Management**:
  - Shows how transactions are initiated, validated, processed, and confirmed to the user.
  - Related Requirements: 3.2.2.1 (Transaction Handling), 3.5.1 (Performance).

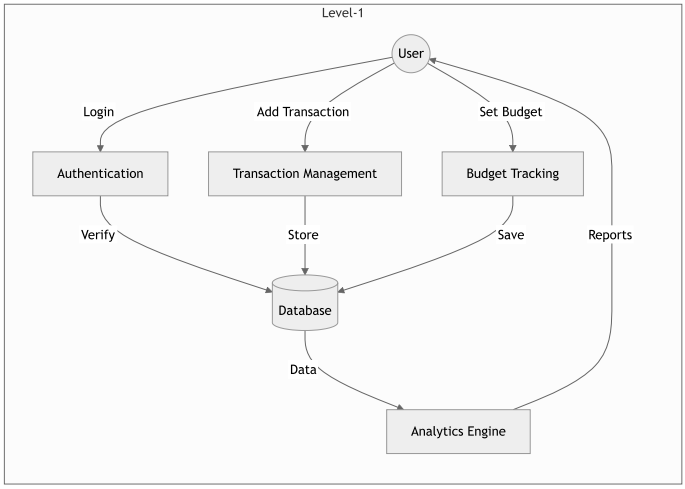


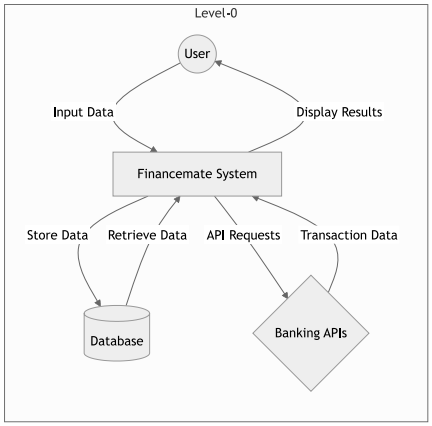
4.2 Data Flow Diagrams (DFD)

Introduction:
Data Flow Diagrams provide a graphical representation of the flow of information through the system, highlighting key processes and their data interactions.

Narrative Description:

- **Level 0 - Context Diagram**:
  - Depicts external entities such as users and external APIs interacting with primary system processes like authentication and reporting.
  - Related Requirements: 3.2.1 (User Management), 3.3.1 (Use Cases).
- **Level 1 - Process Decomposition**:
  - Breaks down processes such as data validation, storage, and retrieval.
  - Related Requirements: 3.2.1, 3.2.3 (Data Management).



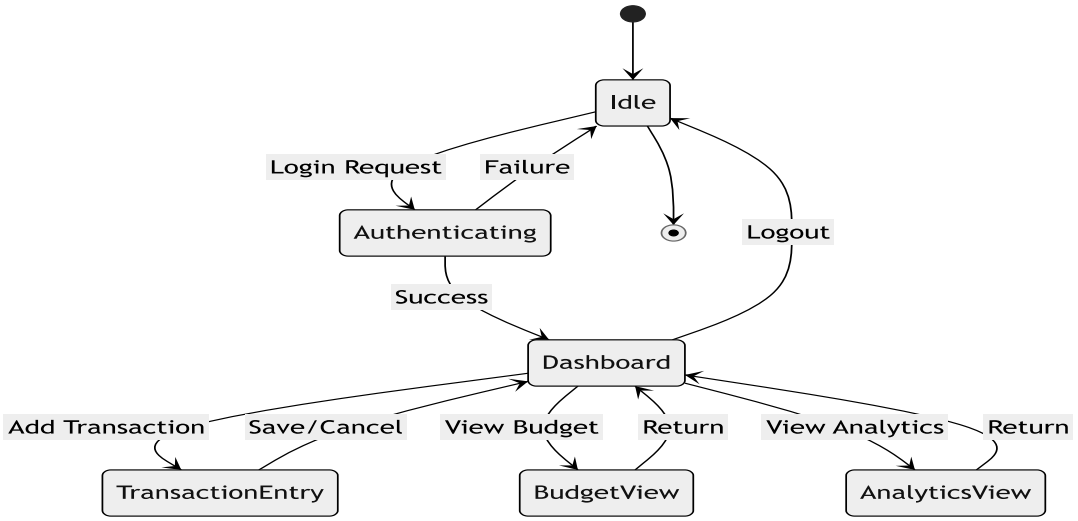


4.3 State-Transition Diagrams (STD)

Introduction:
State-transition diagrams model the states of key components in the system and their transitions triggered by events or conditions.

Narrative Description:

- **User Account Lifecycle**:
  - **States**: Logged Out, Active, Suspended.
  - **Transitions**: Logging in, locking due to failed attempts, and admin-driven suspension.
  - **Related Requirements**: 3.2.1.3, 3.5.4.
- **Transaction Lifecycle**:
  - **States**: Idle, Processing, Success, Failure.
  - **Transitions**: Initiating a transaction, validating data, and handling success or failure responses.
  - **Related Requirements**: 3.2.2, 3.5.1.



5. Supporting Information

This section provides additional references, glossaries, and other details that support the requirements and design process for the system.

References

Include any documents, standards, or external resources referenced during the creation of this SRS.

- **Flutter Documentation**: Official guide for Flutter framework development (https://flutter.dev/docs).
- **Dart Programming Language**: Language reference and tutorials (https://dart.dev).
- **MySQL Documentation**: Reference for database setup, queries, and optimization (https://dev.mysql.com/doc).
- **IEEE SRS Standard (830-1998)**: For structuring software requirement specifications.

Glossary

Defines key terms and acronyms used throughout the SRS:

- **SRS**: Software Requirements Specification.
- **API**: Application Programming Interface.
- **GUI**: Graphical User Interface.
- **CRUD**: Create, Read, Update, Delete operations on the database.
- **UML**: Unified Modeling Language.

Appendices

Include diagrams, tables, and other resources that complement the main sections of the SRS:

- **ER Diagrams**: Representing database schema.
- **DFD Levels**: Context and process decomposition diagrams.
- **Sequence Diagram Snapshots**: Highlighting key functional flows.

Appendix A – Background Research on:

Flutter Framework

Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It uses the Dart programming language and offers a rich set of pre-designed widgets for fast and flexible UI development.

MySQL Database

MySQL is a widely-used open-source relational database management system (RDBMS). It supports SQL queries, transaction management, and scalability, making it suitable for applications requiring robust data management.

Integration of APIs in Mobile Applications

API integration involves connecting applications with external systems to access and exchange data. RESTful APIs are commonly used in Flutter to enable seamless communication with backend servers and external services.

User Experience (UX) Design Principles

Focuses on creating intuitive, accessible, and engaging interfaces. Key principles include simplicity, responsiveness, and usability to enhance the overall user satisfaction.

Appendix B – Data Dictionary

Field Name	Type	Description	Constraints
user_id	**Integer** Unique	identifier for a user	Primary key, Not null
username	Varchar(50)	Name chosen by the user	Unique, Not null
email	Varchar(100)	Email address for communication and login	Unique, Valid email format
password	Varchar(256)	Encrypted user password	Not null
created_at	Timestamp	Record creation timestamp	Default: current timestamp
updated_at	Timestamp	Last modification timestamp	Default: current timestamp