ICSI 518 Software Engineering Software Requirement Specification (SRS)

Team-3

Expense Nest

By

Venkata Veera Sai Yaswanth Ravuri

Sai Charan Papasani

Pravalika Yadlapalli

Siva Naga Mallika Gudditi

Praneeth Kumar Muvva

Karthik Konuru

Varun Jose Madanu

Table of contents:

1. The Purpose of the Product

a. Project Background

ExpenseNest is a financial management application aimed at helping families organize and manage their finances efficiently. The purpose of ExpenseNest is to provide an easy-to-use platform where families can collaborate, set financial goals, and track their progress. It allows families to manage their income, expenses, savings, and debts in a centralized and organized manner, helping them make informed financial decisions and stay on top of their financial health.

In modern families, managing household finances can be overwhelming due to numerous expenses, financial goals, and the need for tracking various aspects such as debts and savings. Without a structured financial plan, families may struggle to save effectively or prioritize spending. **ExpenseNest** was created to address this issue by offering an intuitive platform that fosters transparency and collaboration among family members. By leveraging technology, it helps families collectively achieve financial stability and success.

b. Goals:

The primary goals of ExpenseNest are:

- To enable families to **track and manage their income, expenses, and savings** with minimal effort.
- To provide tools that help users **set, track, and achieve financial goals** such as saving for education, home purchases, or vacations.
- To offer an **accessible and user-friendly interface** that caters to users with varying financial literacy.
- To promote **collaborative financial planning** where multiple family members can participate and contribute.

- To create features that help families **analyze their spending habits**, enabling them to make informed financial decisions.

2. Stakeholders:

a. Description of users

The primary users of ExpenseNest are families or households looking for an efficient and collaborative way to manage their finances. These users can include:

- **Parents** who handle household income, budgeting, and planning for family goals like vacations or children's education.
- **Young adults** who are learning financial management and contributing to family savings or paying off student loans.
- **Retired individuals** who need to monitor their spending while living on fixed incomes and planning for healthcare costs.
- **Families with shared expenses** who need to coordinate their budgeting efforts across multiple members.

b. Describe the Characteristics of the Users

- Diverse Financial Literacy: Users range from individuals with minimal financial knowledge to those well-versed in budgeting and saving strategies. The app must cater to both groups by being intuitive for beginners while offering advanced features for experts**.**
- Collaborative Focus: The application is designed for families with multiple participants. Users may include adults, teenagers, or elderly members who all contribute to a shared goal. Therefore, it needs to support collaborative features such as shared accounts or multiple-user access.
- **Goal-Oriented**: Users are focused on setting and achieving long-term and short-term financial goals, such as saving for a new home, funding education, or planning retirement.
- **Varied Income Levels**: Users may come from different financial backgrounds, including middle-income families, dual-income households, or single-parent families. The app must accommodate varying financial complexities.

- **Mobile and Web Savvy**: Many users are tech-savvy and expect a seamless experience across mobile and web platforms. They prefer quick access to information with features like push notifications and real-time updates on their financial goals.
- **Privacy-Focused**: Since the application deals with sensitive financial data, users expect robust security measures, including encryption, secure login methods (such as two-factor authentication), and data protection practices to keep their information safe.

3. Mandated Constraints

a. Solution Constraints

The project must adhere to the following solution constraints: This product will work in various internet browsers, including Google Chrome, Safari, Opera, and Firefox, among others. Moreover, it would have to work in both the phone and PC system.

The following technologies will be used in building the product:

- **React Js :**React.js is a popular JavaScript library developed by Facebook for building user interfaces, particularly single-page applications (SPAs).
- **Spring Boot:**Spring Boot is an open-source Java-based framework that simplifies the development of stand-alone, production-grade Spring applications. It provides a set of pre-configured templates and boilerplate code to accelerate the setup of new projects.
- **Django:**Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It follows the "batteries included" philosophy, providing a wide range of built-in features such as an ORM (Object-Relational Mapping), authentication, and an admin interface, which speeds up the development process.
- **Python:** Python is a versatile and powerful programming language widely used in artificial intelligence (AI) and machine learning (ML) applications. It boasts a rich ecosystem of libraries and frameworks, such as TensorFlow,

PyTorch, and scikit-learn, which facilitate data analysis, model building, and deployment.

- **SQL (Structured Query Language):** SQL is a standardized programming language used for managing and manipulating relational databases. It allows developers to perform a variety of operations, including querying data, inserting, updating, and deleting records, as well as managing database schemas.
- **Git and GitHub:** It is a Version control system and web-based platform for collaborating, code reviewing, issue tracking, and continuous integration and deployment.
- **Postman:** a tool used in application development to simplify the process of testing and debugging APIs.

b. Implementation Environment

- Product will be added to the current network of computers.
- For the project's dynamic web page creation and component reuse, ReactJS will be utilized while MongoDB will serve as the data storage system.
- Express JS will be used for backend development of the API's.
- Git and GitHub will be used for the version control of the project and for collaborative development of the project.
- For debugging, the API's created in the backend by using Express, Postman will be used.

c. Partner or Collaborative Applications

The product will not rely on any third-party or cooperative application.

d. Off-the-Shell-Software

The product will not utilize any off-the-shelf software.

e. Schedule :

Tasks	Duration Estimate	Estimate points/Effort (days)
Created environmental setup for the project		8
Set up backend environment		8
Created database schemas and tables for storing the Customers		14
Created database schemas and tables for storing the assets and expenses		16
Implementing the Income page		14
Design and Implement Expenses Page		8
Implementation of Assets and Financial Calculator		16
Creating the AI recommendations		8
Design the Budgeting Page		8
Design and Implement Income page		12

Implementation of financial goals		8
Design and Implement Financial Assistance page		16
End to end testing of Expense Module		8
End to end testing of Goals module		8
End to end testing of Assets module		12
Deployment of the project		8

4. Naming Conventions and Terminology

Term	Description
Environmental Setup	It is the process of setting up the environment for the project to kickstart.
Backend Setup	Components that are in the server side that are necessary for the application to function
Database	A collection of data that has been structured in such a way that it can be stored, retrieved, and managed efficiently.
User	A family member who uses the application to manage finances.
Registration	The process of creating a new user account within the application.

Login	The action of accessing the application with user credentials.
Income	Money received by the family from various sources, including salaries, bonuses, or investments.
Expense	Money spent on goods and services, categorized into fixed (rent, bills) and variable (entertainment, dining out).
Budget	A plan that outlines expected income and expenses over a specific period.
Goal	A financial target set by the user, such as saving for a vacation or paying off debt.
Income Page	To track the income of the user they can add it here
Expense Page	This is useful to track their and which will also give a detail about spending
Financial Goals	This can be useful to setup the financial goals for the user
Database Schema	It is a blueprint that is defined structure and organization data within the database.
End-to-End Testing	Testing of the entire application ensuring all the functionalities are working.
Deployment	It is the process of making the application available for the public by entering a URL in a browser

5. Assumptions:

Customer Awareness

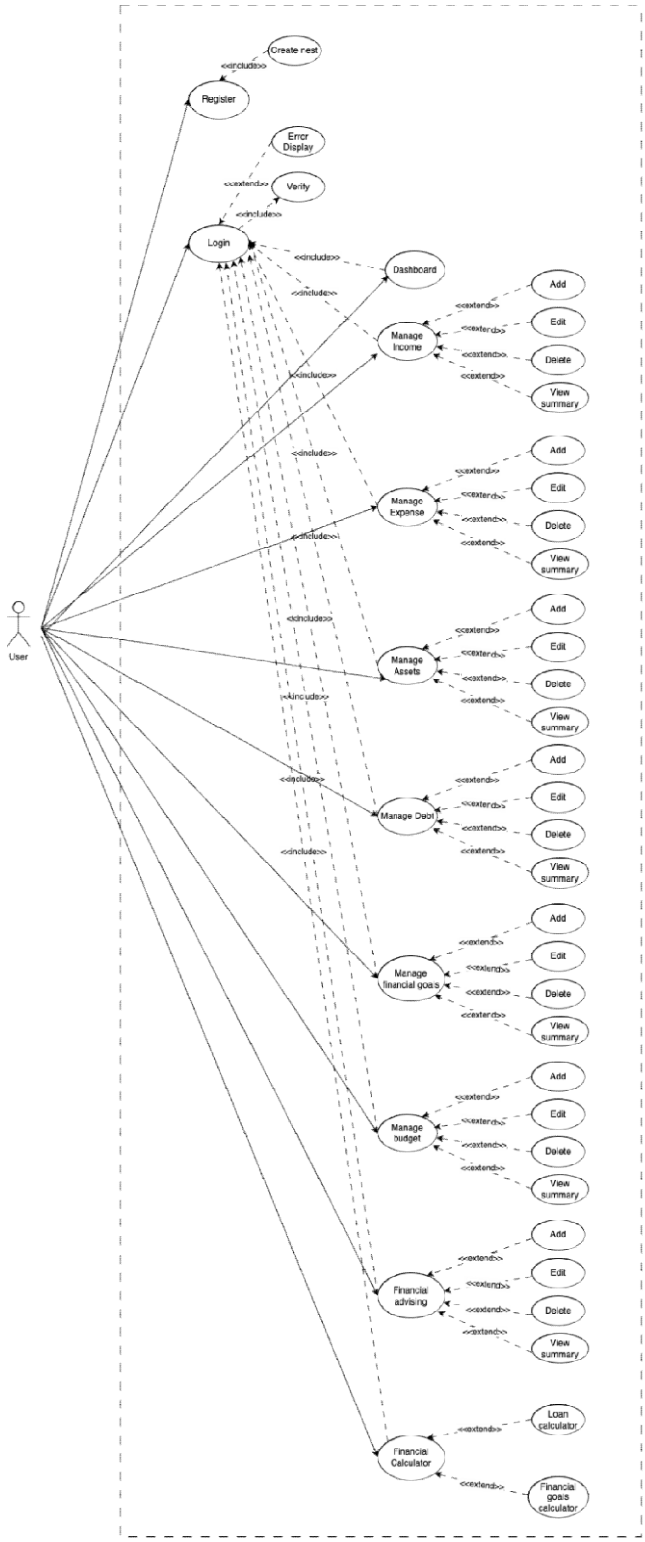
While navigating throughout the web application, necessary knowledge and understanding will be among users. Despite being user-friendly, as far as navigation is concerned, there is a requirementto have useful knowledge of web browsing as well as interactive interfaces on the part of the User.

Device Compatibility

In essence, every user is to have a personal computer or smartphone in which my developed web application may be run. The design will be responsive and cross-compatible with any other device being in use.

6.Scope

a. Use case diagram



b.Use case table:

Use Case ID	Use Case Name	Actor(s)	Input or Output
001	Register	User	All the users of this application must be able to complete the Registration successfully by giving all the valid details of their respective roles.
002	Create Nest	User	All users should be able to create a community for their family.
003	Login	User	I/P: All users should give their valid credentials as input to log into the system. O/P: The Login should be successful for the valid credentials.
004	Dashboard	User	O/P: Displays the overall financial summary and visual representations of financial status.
005	Manage Expense	User	All users should be able to manage expenses including Adding, Deleting, Editing expenses and Viewing summaries
006	Manage Income	User	All users should be able to manage Incomesincluding Adding, Deleting , Editing expenses and Viewing summaries

007	Manage Assets	User	All users should be able to manage Assets including Adding, Deleting , Editing Assets and Viewing summaries.
008	Manage Debts	User	All users should be able to manage Debts including Adding, Deleting, Editing Debts and Viewing summaries.
009	Manage Financial Goals	User	All users should be able to manage Financial Goals including Adding, Deleting, Editing Financial goals and Viewing summaries.
010	Manage Budget	User	All users should be able to manage Budgets including Adding, Deleting, Editing Budgets and Viewing summaries.
011	Financial Advising	User	User should be able to book appointments with the financial advisors.
012	Financial Calculator	User	I/P: Entering a principal amount and the interest type, rate and time period. O/P: Displays the loan amount including interest to be paid over a given period.

7. Functional Requirements

a. Complete set of the functional requirements:

User Authentication

Requirement #: 1 **Event/Use Case #:**

Description: Users can create an account, securely log in, and log out.

Rationale: User authentication is necessary to access personal data securely.

Fit Criterion: Users must register with an email and password. Account recovery via "forgot password" is also required.

- Users need an email for registration.
- Users access their dashboard after authentication.

Dependencies: Integration with a secure authentication provider.

Supporting Materials: Use – case Diagram

History: Created by Karthik

Nest (Family Group) Management

Requirement #: 2 **Event/Use Case #:** **002**

Description: Create and Manage Family "Nest"

Rationale: Users can create a family group ("Nest") and invite members.

Fit Criterion: Family members can view, track, and manage expenses together.

- User must be registered and logged in.
- Family group is created, and members are added to manage finances together.

Dependencies: Integration with user management and invitation system.

Supporting Materials: Use – case Diagram

History: Created by Karthik

Dashboard

Requirement #: 3 **Event/Use Case #:** **004**

Description: The system displays the summary of total assets, income, and expenses.

Rationale: The dashboard provides an overall financial status to users.

Fit Criterion: A user-friendly summary and interactive charts for spending and income trends.

- Users have added income and expenses.
- The dashboard reflects updated financial information.

Dependencies: Income and expense management, chart rendering tools.

Supporting Materials: Use – case Diagram

History: Created by Karthik

Asset Tracker

Requirement #: 4 **Event/Use Case #:** **007**

Description: Users can add and track changes in assets (e.g., property, stocks).

Rationale: Users need to monitor their assets and debts over time.

Fit Criterion: The system supports adding, tracking, and updating assets and debts.

- Users need to input asset values.
- Assets and debts are updated and displayed on the dashboard.

Dependencies: Database for storing assets and debt details.

Supporting Materials: Use – case Diagram

History: Created by Karthik

Income Management

Requirement #: 5 **Event/Use Case #:** **006**

Description: Users can add multiple income sources and set recurring incomes.

Rationale: It allows users to track various income streams (e.g., salary, freelance).

Fit Criterion: The system provides options for adding recurring income and displays income insights.

- User must input income details.
- Income is tracked, and variations are displayed over time.

Dependencies: Recurrence management and graphical display tools.

Supporting Materials: Use – case Diagram

History: Created by Karthik

Expense Management

Requirement #: 6 **Event/Use Case #:** **005**

Description: Users can log daily expenses and categorize them (e.g., food, rent).

Rationale: Tracking expenses is critical for managing budgets.

Fit Criterion: System supports manual and recurring expenses.

- Users need to input expenses.
- Expenses are saved and categorized.

Dependencies: Expense storage system, recurring management.

Supporting Materials: Use – case Diagram

History: Created by Karthik

Financial Goals

Requirement #: 7 **Event/Use Case #:**

Description: Users can set financial goals, track progress, and get reminders.

Rationale: Goal-setting helps users save and manage long-term finances.

Fit Criterion: The system supports setting deadlines and tracking goal progress visually.

- Users need to set specific financial goals.
- Progress is tracked over time.

Dependencies: Goal-setting module, progress tracking.

Supporting Materials: Use – case Diagram

History: Created by Varun Jose

Budget Management

Requirement #: 8 **Event/Use Case #:**

Description: Users can create budgets for categories and receive alerts when budgets are exceeded.

Rationale: Budgeting helps users control spending.

Fit Criterion: Users receive alerts and notifications when they exceed budgets.

- Users need to set budgets.
- Budgets are tracked, and alerts are triggered based on spending.

Dependencies: Budgeting system and alert notifications.

Supporting Materials: Use – case Diagram

History: Created by Varun Jose

Financial Calculators

Requirement #: 9 **Event/Use Case #:**

Description: The system provides loan and savings calculators to assist with financial planning.

Rationale: Users need assistance calculating monthly savings or loan repayments.

Fit Criterion: Accurate calculations based on user inputs (interest rate, loan duration).

- Users need to input loan/savings details.
- Calculations are displayed, and results are shared with users.

Dependencies: Calculator functions.

Supporting Materials: Use – case Diagram

History: Created by Varun Jose

Financial Advising

Requirement #: 10 **Event/Use Case #:**

Description: Users can schedule financial advising appointments.

Rationale: Provides professional financial guidance beyond AI recommendations.

Fit Criterion: Users can book appointments via the system.

- Users request financial advising.
- Appointments are scheduled.

Dependencies: Integration with appointment scheduling tools.

Supporting Materials: Use – case Diagram

History: Created by Varun Jose

AI Integration

Requirement #: 11 **Event/Use Case #:**

Description: AI algorithms automatically categorize expenses and provide recommendations.

Rationale: AI enhances user experience by automating routine tasks and providing insights.

Fit Criterion: The system automatically categorizes user inputs and improves suggestions over time.

- User data needs to be inputted.
  AI improves categorization and gives tailored suggestions.

Dependencies: AI models and user feedback systems.

Supporting Materials: Use – case Diagram

History: Created by Varun Jose

b. Traceability matrix (user-stories – requirements):

User Story	Description	Requirement
User Authentication	Users can register, log in, and securely log out of the system.	Users must be able to register, log in, and log out securely.
Create Nest	Users can create a family "Nest" to add family members, manage, and track expenses. The user creating the nest can act as an admin.	Users must be able to create a nest and assign admin roles.
Dashboard	Display an overview of total assets, income, and expenses, along with recent transactions and upcoming appointments.	Users must see a summary of financial status and recent activities.
Asset Tracker	Allow users to add, update, and track assets such as property, stocks, and savings, along with tracking debts.	Users must be able to log and monitor changes in assets and debts.
Income Page	Users can add and manage multiple income streams, with the option to set recurring income and view insights on variations over time.	Users must be able to add income sources and see detailed insights.
Add Expense	Allow users to log daily expenses in various categories (e.g., food, rent) and set recurring expenses.	Users must be able to input expenses and manage recurring ones.
Financial Goals	Users can set financial goals, track progress via visual bars, set deadlines, and receive AI recommendations based on expenses.	Users must be able to set and track financial goals with AI-based suggestions.
Financial/Loan Calculator	Provide tools to calculate loan payments based on interest rate and duration, as well as savings needed to meet financial goals.	Users must be able to calculate loan payments and savings plans.
Budgeting	Allow users to create monthly or annual budgets, track budget vs. actual spending, and receive alerts when close to exceeding the budget.	Users must be able to create budgets and get notifications for overspending.
Scheduling Appointment with Advisor	Users can schedule appointments with financial advisors through the system.	Users must be able to book financial advisor appointments.
Categorization Using AI	Automatically categorize expenses and income entries, with AI-based suggestions and improvements over time.	Users must have expenses categorized by AI and receive suggestions for uncategorized transactions.

8. Non-functional requirements:

Here are the non-functional requirements applicable to our **ExpenseNest** project. Each non-functional requirement is critical for ensuring that the system performs efficiently, securely, and reliably.

1. Product Requirements

I. Usability Requirements:

- **Efficiency of Use**: **90%** of users should be able to complete key tasks (e.g., adding an expense, generating a report) within **2 minutes** of interacting with the system, ensuring that frequent tasks are quick and intuitive to perform.
- **Ease of Remembering**: After a **two-week** gap in using the system, **80%** of users should be able to remember how to perform key actions (e.g., adding an expense, viewing transaction history) without needing external help or reviewing documentation.
- **Error Rate**: Users should experience an error rate of less than **1%** for key operations (e.g., adding or updating expenses, running reports) after using the product for **one month**. This ensures that the system is user-friendly and minimizes mistakes.
- **Overall Satisfaction**:An anonymous user satisfaction survey should show that **85%** of users are satisfied with the product after **three weeks** of use. The survey should evaluate ease of use, clarity of the interface, and the overall experience in managing family expenses.
- **Feedback and User Confidence**:**100%** of actions (e.g., adding expenses, generating reports, editing transactions) must provide immediate visual feedback (e.g., success messages, confirmation dialogs) within **1 second**, ensuring users feel confident that their actions are correctly executed.
- **Task Completion without Assistance**:**90%** of users should be able to navigate the system and complete key tasks (e.g., adding expenses, viewing reports, updating account settings) without any external assistance after a **30-minute** guided onboarding or tutorial session.

II. Efficiency Requirements:

- **Performance Requirements**
  - Any action taken by the user (such as adding a new expense, updating an entry, or viewing reports) should have a maximum response time of **2 seconds**.
  - If the ExpenseNest is connected to an online platform or cloud storage, it should **synchronize** new data entries (expenses, income updates) within **5 minutes** of making the change.

- **Pulling reports** (monthly summaries, category breakdowns) should be completed within **3 seconds**, ensuring quick and seamless access to detailed information.
- **Periodic expense calculations** (such as tracking budget usage, calculating averages) should be processed in the background without disrupting user interaction, completing within **10 seconds** or less for comprehensive calculations.
- If there is a mobile app, expenses added from a mobile device should appear on all connected devices (e.g., desktop, web, etc.) within **1 minute** of submission.

- **Space Requirements**
  - The system should not exceed **100 MB** of storage for each year of transaction data.
  - Database should be optimized to store historical data without performance degradation.

III. Reliability Requirements:

- **Successful Request Processing:99.9%** of all user requests (e.g., adding expenses, viewing reports, updating transactions) made between **8 AM and 10 PM local time** should be successfully processed by the server with an HTTP 200 OK success status response.
- **Transaction Processing Success**: At least **99.8%** of all financial transactions (e.g., adding, updating, deleting expenses) should complete successfully without any errors or data loss, ensuring a smooth and reliable user experience during financial operations.
- **Error-Free Report Generation**: **99.5%** of all expense reports and analytics generated by the system should be completed without failure or data inconsistency, ensuring accurate and reliable financial summaries over a continuous 24-hour period.
- **System Response Rate**:**99%** of all system responses should be returned within **3 seconds** during peak usage hours (8 AM to 10 PM local time), ensuring that the system is highly responsive under normal operating conditions.
- **Failure-Free Operation Period**: The ExpenseNestsystem should operate without failure for **at least 98% of the month**, providing near-continuous reliability and availability.

IV. Availability Requirements:

- **System Uptime**: The ExpenseNest should be available **99.9%** of the time throughout the year, ensuring reliable access for users. This translates to no more than **8.77 hours** of downtime annually.

- **Downtime Handling**: Any system maintenance or updates should be scheduled during off-peak hours, and users should be notified in advance. The system should aim for a maximum of **2 hours** of scheduled maintenance downtime per month.
- **Disaster Recovery**: A robust data backup and redundancy system should be in place to ensure that no data is lost during downtime. Daily backups should be stored in multiple locations to prevent loss in the event of a hardware failure.
- **Data Redundancy:** A robust data backup and redundancy system should be in place to ensure that no data is lost during downtime. Daily backups should be stored in multiple locations to prevent loss in the event of a hardware failure.

V. Portability Requirements:

The ExpenseNest should be accessible across different platforms, including desktop, mobile, and tablets. The technology that we are using, React.js,ensures cross-platform compatibility through responsive design.

VI. Accessibility Requirements:

- Provide a high-contrast mode for partially sighted users, ensuring that all text and essential features have sufficient contrast against the background (minimum contrast ratio of 4.5:1 per WCAG 2.0 guidelines).
- The interface should support text resizing without losing functionality or readability. Users should be able to zoom up to **200%** without needing to scroll horizontally.

VII. Look and Feel Requirements:

- **Attractiveness to a Family Audience**:
  - The ExpenseNest should have a **clean, user-friendly, and welcoming design** that appeals to family members of all ages, including parents, children, and elderly users.
  - A sample group of **representative family users** (including non-technical members) should start using the product within **5 minutes** of their first interaction, without needing instructions or guidance.
- **Intuitive and Consistent Interface**: The user interface should use a **consistent color scheme, typography, and layout** that conveys trust and ease of use. The color scheme should be calming and family-friendly, with a clear visual hierarchy for buttons, menus, and forms.
- **Mobile and Desktop Responsiveness**: The design should be **responsive**, ensuring that the website looks and functions well on both mobile devices and desktop screens, with clear and readable fonts and intuitive navigation on all platforms.
- **Clear and Inviting Feedback**: **Visual feedback** (e.g., success messages, error notifications, loading indicators) should be provided for all user actions, ensuring users feel confident that their actions have been processed.

Feedback should be unobtrusive but noticeable and fit within the overall design language.

VIII. Precision or Accuracy Requirements:

- **Monetary Amounts**:
  - **Precision:**Monetary amounts should be recorded and displayed in two decimal places (e.g., $100.50).
  - **Accuracy:** Ensure that recorded amounts are exact based on input, with minimal rounding errors, accurate to the cent.
- **Expense Date/Time**:
  - **Precision**: Track dates to the day, and if time is included, it could be precise to the minute or second if necessary (e.g., "2024-10-06 14:30").
  - **Accuracy**: Dates and times should correspond accurately to the actual times of expense occurrences.

2. Organizational Requirements

I. **Delivery Requirements:**The system should be delivered within the agreed project timeline and meet all project milestones, adhering to agile development methodologies.

II. **Implementation Requirements:**

- The system must use React.js for the frontend, Spring Boot for the backend, and MySQL for the database.
- It should be deployed in AWS cloud environment to ensure scalability.

III. **Standards Requirements:**

- The system should follow standard web development best practices, such as using HTTPS, OWASP security standards, and RESTful API design.
- The application should conform to General Data Protection Regulation (GDPR) for data privacy.

3. External Requirements:

I. **Interoperability Requirements:** The system should be designed to easily integrate with other financial systems (e.g., external APIs for bank transactions, payment gateways).

II. **Ethical Requirements:** Usersfinancial data should be handled ethically, ensuring privacy and data integrity, with transparent data policies in place.

III. **Privacy/Security Requirements:**

- **Data Encryption: SSL/TLS Encryption**: All data transmitted between the user's device and the**ExpenseNest's** server must be encrypted using **TLS 1.2 or higher** to protect sensitive information like expenses and login credentials.
- **Salted Hashing for Passwords**: User passwords must be stored using **salted hashing** to prevent them from being stored as plain text and protect against breaches.

- **Multi-Factor Authentication (MFA)**: Implement optional **MFA** for added security, requiring users to provide an additional authentication factor (e.g., a code sent to their phone) to access their account.
- **Database Security**: The database must be **password protected** and accessible only from **whitelisted IP addresses**, ensuring that only authorized systems can access the stored data.
- **Regular Security Audits**: The system should undergo **regular security audits** and vulnerability assessments to detect and fix any weaknesses.
- **Denial of Service (DoS) Protection**: The system must have protections in place to detect and mitigate **denial of service (DoS) attacks**, ensuring continuous service availability.

IV. Legislative Requirements:

- The system should comply with local financial regulations and standards (e.g., tax reporting laws).
- User data must be handled according to privacy laws (such as GDPR for European users or CCPA for California users).

V. Safety Requirements: Financial information and personal details should be securely backed up daily to prevent loss.

These non-functional requirements ensure that our project **ExpenseNest** will meet performance, security, scalability, and usability expectations, while also adhering to relevant organizational and legal standards.