AI Expense Tracker

UCS 503 Software Engineering Project Report

Mid-Semester Evaluation

Submitted by:

(102303862) SAUMIL MAKKAR

(102303860) DHEERAJ KUMAR BHASKAR

(102303964) DIVYANSH SHARMA

BE Third Year, COE

Group No: 3C62

Submitted to:

Computer Science and Engineering Department
TIET, Patiala

TABLE OF CONTENTS

1 Project Selection Phase

1.1 Software Bid

Group : 3C62 Dated:02-09-2025

Team Name: CODEFUSION
Team ID (will be assigned by Instructor):

Please enter the names of your Preferred Team Members:

1. You are required to form **a three to four person** teams
2. Choose your team members wisely. You will not be allowed to change teams.

Name	Roll No	Project Experience	Programming Language used	Signature
SAUMIL MAKKAR	102303862	Worked on 1.Movie Recommendation System 2 Document Classifier	React.js, Python,Tail wind,Mongo DB	Saumil Makkar
DHEERAJ KUMAR BHASKAR	102303860	Worked on 1.Document Classifier 2.Budget Tracker	Python, React.js,Express.js	Dheeraj
DIVYANSH SHARMA	102303964	Worked on 1. Friend Recommendation System 2. Health Tracker	Python,Node.js, SQL	Divyansh

Programming Language / Environment Experience

List the languages you are most comfortable developing in, **as a team**, in your order of preference. Many of the projects involve Java or C/C++ programming.

1. React.js
2. Node.js+Express.js
3. MongoDB
4. Vision API

Choices of Projects:

Please select **4 projects** your team would like to work on, by order of preference: *[Write at-least one paragraph for each choice (motivation, reason for choice, feasibility analysis, etc.)*

Choice	Project Name	Unique Selling Point
First Choice	**AI Expense Tracker**	**AI Expense Tracker is an advanced web-based financial assistant designed to automate personal and business expense management. It integrates AI-powered receipt scanning, natural language voice commands, and predictive analytics to provide seamless expense tracking. The system ensures secure authentication using JWT, performs automated recurring transactions via Cron Jobs, and utilizes MongoDB aggregation for real-time financial insights. By eliminating manual entry and offering visual spending trends, it helps users make informed financial decisions and achieve sustainable financial growth.**
Second Choice	**AI Health Monitor**	**AI Health Monitor** is an intelligent wellness companion that tracks, analyzes, and predicts an individual's health metrics in real-time. Using data from smart wearables or manual inputs, it leverages **machine learning algorithms** to detect anomalies, provide personalized health recommendations, and predict potential risks like stress or fatigue. The system includes a **daily health dashboard**, **smart alerts**, and **AI chat-based consultations**, promoting proactive healthcare and early intervention. Its unique selling point lies in its predictive approach, focusing on prevention rather than treatment.
Third Choice	**TravelMate**	**TravelMate** is an **AI-powered smart travel companion** that redefines how people plan and experience travel. From **personalized itinerary generation** and **budget optimization** to **real-time route updates** and **emergency assistance**, it ensures a smooth and adaptive journey. The platform integrates **AI chat assistance**, **cloud-based data storage**, and **geo-tracking features** for safe and connected travel. Its unified interface for booking, tracking, and journaling

Choice	Project Name	Unique Selling Point
		offers a highly customized and intelligent travel experience for explorers, digital nomads, and frequent travelers alike.
Fourth Choice	**PetCare Manager**	**PetCare Manager** is a smart digital assistant designed for pet owners to simplify and enhance pet care. It automates **health tracking**, **diet planning**, **appointment scheduling**, and **vaccination reminders**. Using **AI-based behavior insights** and **GPS integration**, it provides real-time activity monitoring and connects users with nearby vets or pet service providers. Its unique selling point lies in its combination of **AI analytics** and **community-driven pet wellness**, ensuring pets stay healthy and happy while reducing the burden of manual management for owners.

Additional Remarks/ Inputs

Please tell us about any other factors that we should take into consideration (e.g., if you really would like to work on a project for some particularly convincing reason).
We are strongly motivated to work on the AI Expense Tracker project for several compelling reasons:

5. Technological Alignment: Our chosen stack (React.js, Node.js, Express.js, MongoDB, JWT) perfectly matches the project's requirements, allowing our team to leverage its strongest skills in JavaScript/full-stack development. This minimizes the learning curve and maximizes the likelihood of a successful, high-quality delivery.
6. Addressing a Practical Need: The core problem of tedious manual expense tracking is a universal frustration. Our innovative solution, combining AI Receipt OCR and the newly added Voice Assistant for hands-free data entry, directly solves this practical, real-world pain point.
7. Feasibility and Experience: The team has experience with both JavaScript/JS and database queries (SQL/MongoDB), which are crucial for the frontend UI, backend API, and complex analytical pipelines. We are confident in our ability to integrate external APIs (Vision API) and implement automation tools (Cron Jobs). This project is challenging yet entirely feasible for our skill set.
8. Cutting-Edge Feature: The integration of the Voice Assistant makes this project unique and showcases a commitment to developing cutting-edge, accessible technology, aligning with modern software trends.

1.2 Project Overview

1. SUMMARY

This project introduces the **AI Expense Tracker**, a smart, full-stack web application designed to automate and simplify personal and business finance management end-to-end. The system integrates advanced **AI/OCR technology** for automatic receipt scanning and a **Voice Assistant** for hands-free transaction entry. It provides secure JWT authentication, automates recurring expenses, and delivers actionable financial insights through MongoDB-powered analytics and monthly emailed reports, offering a unified, intelligent, and time-saving financial solution.

2. PROBLEM STATEMENT

Manual financial tracking is often time-consuming, prone to error, and tedious. Users face challenges in:

1. **Data Entry:** Manually typing transaction details from receipts is slow and stressful.
2. **Organization:** Keeping track of recurring bills and generating timely monthly summaries requires continuous effort.
3. **Insight:** Basic spreadsheets lack the advanced analytical tools needed to visualize spending patterns, budget adherence, and income/expense trends. These issues collectively lead to inaccurate financial records and unnecessary stress, hindering long-term money management success.

3. PROPOSED SOLUTION

The AI Expense Tracker offers a single, intelligent web platform that automates the financial workflow. The core solution components are:

1. **AI Receipt Scanning:** Uses a **Cloud-based Vision API** to extract transaction details automatically from uploaded receipts.
2. **Voice Assistant:** Allows users to add expenses instantly via voice command, eliminating typing.
3. **Automation:** Utilizes **Cron Jobs** to schedule recurring transactions and generate/email comprehensive monthly reports.
4. **Advanced Analytics:** Employs **MongoDB aggregate pipelines** to power dynamic pie charts and line charts, providing deep visual insights into spending behaviour.

4. UNIQUE SELLING POINT

Unlike existing fragmented solutions, the AI Expense Tracker uniquely integrates **AI-driven automation** into one platform. Its primary unique selling points are:

- **Dual Automation Input:** The combination of **AI Receipt Scanning (OCR)** and **Voice Assistant** provides maximum flexibility and eliminates the need for manual data entry entirely.
- **Deep, Actionable Analytics:** It moves beyond simple lists by using powerful MongoDB aggregation to provide users with meaningful, visual data to make informed financial decisions.
- **Zero-Effort Reporting:** Automated monthly report generation delivered via email ensures users stay on top of their finances without lifting a finger.

5. OBJECTIVE

The primary objective of the AI Expense Tracker is to **simplify and automate financial management** for individuals and businesses. The platform aims to:

1. Maximize data entry efficiency by achieving over **90% automation** using AI/Voice input.
2. Deliver secure access via **JWT authentication**.
3. Provide clear, real-time awareness of financial status through dynamic analytical dashboards.
4. Ensure transactional data integrity and reliable automation of recurring tasks and monthly reports.

6. METHODOLOGY

The methodology for transaction processing begins with user input via either a **receipt upload (OCR)** or a **voice command (Voice Assistant)**.

- **Data Capture:** The input is captured and processed by the corresponding API (Vision API or Voice API).
- **Transaction Management:** The extracted data is sent to the **Node.js/Express.js** backend, where it is automatically categorized and stored in **MongoDB**.
- **Analytics:** Users access their data via the **React.js** frontend, where it triggers complex **MongoDB aggregate pipelines** to generate visual charts.
- **Automation: Cron Jobs** run on a schedule to check for recurring entries and to trigger the monthly report generation and email service.

7. OUTCOMES

The expected outcomes of the AI Expense Tracker project include the development of a functional **full-stack web application**. The app will showcase:

1. Seamless, intelligent **AI/Voice-based data input**.
2. A robust backend built with **Node.js/Express.js** handling secure **JWT authentication**.
3. A highly interactive frontend developed with **React.js**, displaying advanced analytics (pie and line charts) with flexible filtering options.
4. A fully operational automation layer for recurring expenses and automated monthly reporting.

8. PRODUCT PERSPECTIVE

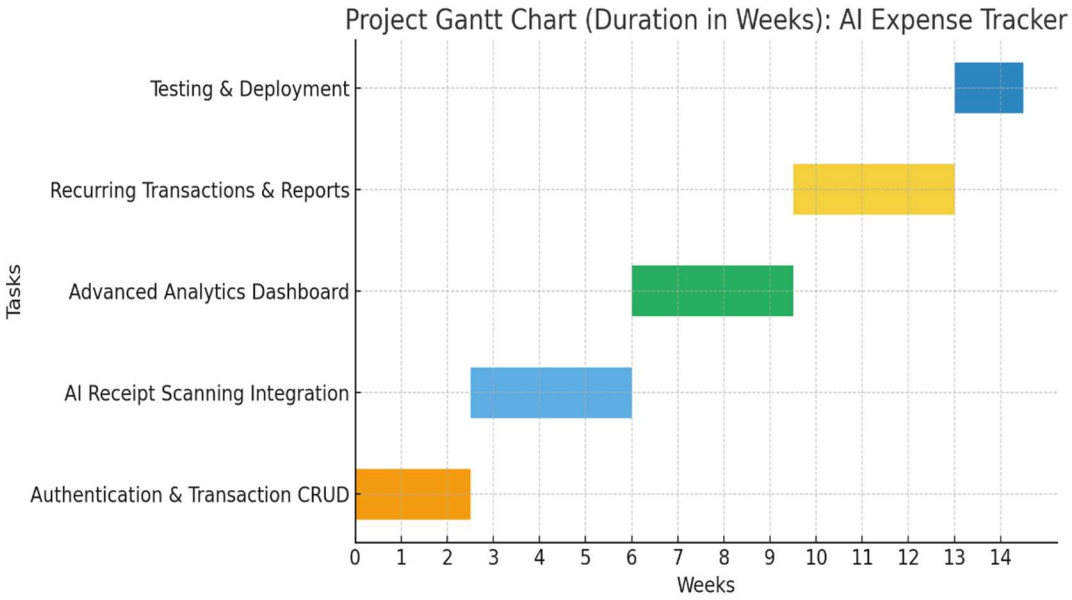
The frontend of the AI Expense Tracker will be developed using **React.js**, ensuring a modern, interactive, and responsive web interface. The backend will be built using **Node.js/Express.js** to handle APIs, secure **JWT authentication**, and data management. The system will integrate external services such as the **Cloud-based Vision API** (for OCR) and an email service (for reports). In terms of design, the product will prioritize simplicity and intuitiveness with visually clear dashboards and an easy-to-use transaction view that reduces the learning curve for users.

9. SCOPE OF APPLICATION

The AI Expense Tracker has a broad scope of application, targeting:

1. **Individuals/Students:** For personal budget management and achieving saving goals.
2. **Small Business Owners/Freelancers:** For simplified tracking of business expenses, tax preparation, and income management.
3. **Financial Advisors:** Can use the platform's detailed reports and analytics to better advise their clients. The system is essential for anyone seeking to eliminate the manual effort involved in expense tracking and gain valuable insight into their financial health.

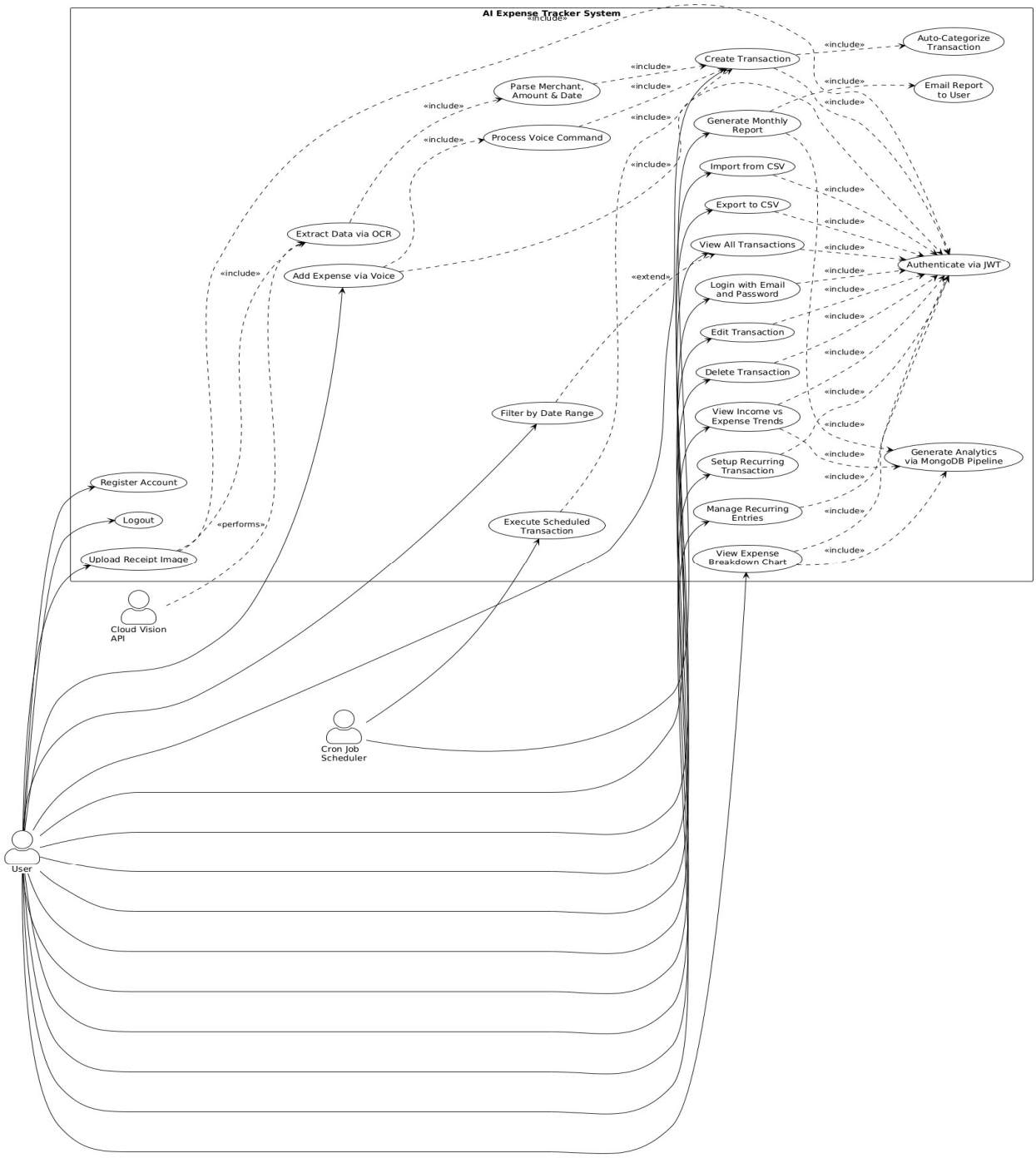
10. GANTT CHART



2. ANALYSIS PHASE

2.1 USE CASES

2.1.1 USE CASE DIAGRAM


AI Expense Tracker System
Register Account
Logout
Upload Receipt Image
Add Expense via Voice
Extract Data via OCR
Parse Merchant, Amount & Date
Process Voice Command
Create Transaction
Auto-Categorize Transaction
Email Report to User
Generate Monthly Report
Import from CSV
Export to CSV
View All Transactions
Login with Email and Password
Edit Transaction
Delete Transaction
Filter by Date Range
View Income vs Expense Trends
Setup Recurring Transaction
Manage Recurring Entries
Execute Scheduled Transaction
View Expense Breakdown Chart
Authenticate via JWT
Generate Analytics via MongoDB Pipeline
User
Cloud Vision API
Cron Job Scheduler
«include»
«extend»
«performs»

2.1.2 Use Case Templates

1. Use Case Title	User Authentication (Login/Register)
2. Abbreviated Title	Login/Register
3. Use Case Id	1
4. Actors	User, System
5. Description	This use case allows a user to securely log in or register to access the AI Expense Tracker. The system validates credentials using JWT and loads the user's financial dashboard upon successful authentication.
5.1. Pre Conditions:	User has the web application loaded.
5.2. Task Sequence	1. User navigates to the Login/Register screen. 2. User enters login/registration credentials (email and password). 3. System validates credentials via the **JWT-based** authentication service. 4. If successful, the user's personalized dashboard is displayed. 5. If authentication fails, an error message is shown with a retry option.
5.3. Post Conditions:	• User successfully logged in and redirected to the dashboard. • User profile and JWT session are loaded.
6. Modification History:	Date - [Current Date]

1. Use Case Title	Automatic Transaction Entry via Receipt Scanning
2. Abbreviated Title	AI Scan
3. Use Case Id	2
4. Actors	User, System, Cloud-based Vision API
5. Description	This use case enables a user to upload a receipt image, allowing the system to automatically extract key transaction details (merchant, amount, date) using the **Cloud-based Vision API**.
5.1. Pre Conditions:	User must be logged in. User has a clear image of a receipt ready to upload.
5.2. Task Sequence	1. User navigates to the "Add Transaction" screen and selects "Upload Receipt." 2. User uploads the receipt image file. 3. System securely sends the image to the **Cloud-based Vision API** (OCR) for analysis. 4. The Vision API returns the extracted transaction data. 5. System populates a new transaction form with the extracted details and automatically suggests a category. 6. User reviews the details and selects "Save" to confirm the transaction.
5.3. Post Conditions:	• A new transaction with extracted details and automatic categorization is recorded in the MongoDB database.
6. Modification History:	Date - [Current Date]

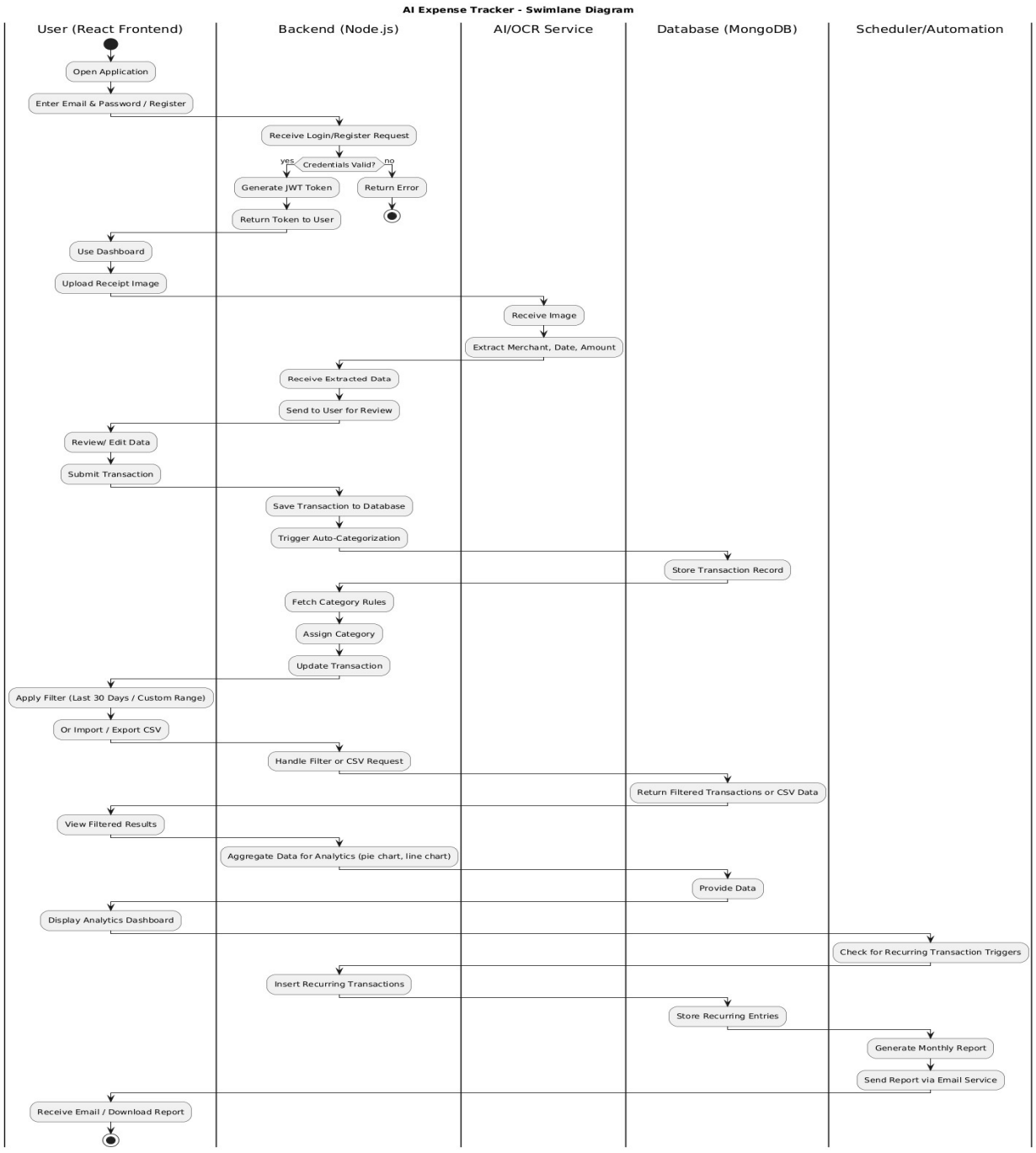
1. Use Case Title	Transaction Entry via Voice Assistant
2. Abbreviated Title	Voice Add
3. Use Case Id	3
4. Actors	User, System, Voice Recognition API
5. Description	This use case allows the user to quickly record an expense or income transaction by speaking the details to the system using the **Voice Assistant** feature.
5.1. Pre Conditions:	User must be logged in. User has a microphone access enabled in their browser.
5.2. Task Sequence	1. User activates the Voice Assistant feature (e.g., clicks a microphone icon). 2. User speaks the transaction details (e.g., "Spent thirty dollars at the grocery store yesterday"). 3. System uses the Voice Recognition API to transcribe and parse the intent (amount, merchant, date, category). 4. System displays a confirmation prompt with the parsed details for user review. 5. User confirms the transaction. 6. System saves the new transaction to the database.
5.3. Post Conditions:	• A new, confirmed transaction is recorded in the database.
6. Modification History:	Date - [Current Date]

1. Use Case Title	View Financial Analytics and Reports
2. Abbreviated Title	Analytics Dashboard
3. Use Case Id	4
4. Actors	User, System, MongoDB
5. Description	This use case allows the user to access actionable financial insights through dynamic expense breakdown pie charts and income/expense line charts, powered by **MongoDB aggregate pipelines**.
5.1. Pre Conditions:	User must be logged in. User must have transactions recorded in the database.
5.2. Task Sequence	1. User navigates to the "Analytics" dashboard. 2. User applies a filter (e.g., selects a **custom date range** or "Last 30 Days"). 3. System executes the necessary **MongoDB aggregate pipelines** on the transaction data. 4. System renders the Expense Breakdown Pie Chart and the Income/Expense Line Chart based on the aggregated results. 5. User interacts with the charts to gain deeper insight.
5.3. Post Conditions:	• User gains visual insight into their spending trends and category performance for the selected period.
6. Modification History:	Date - [Current Date]

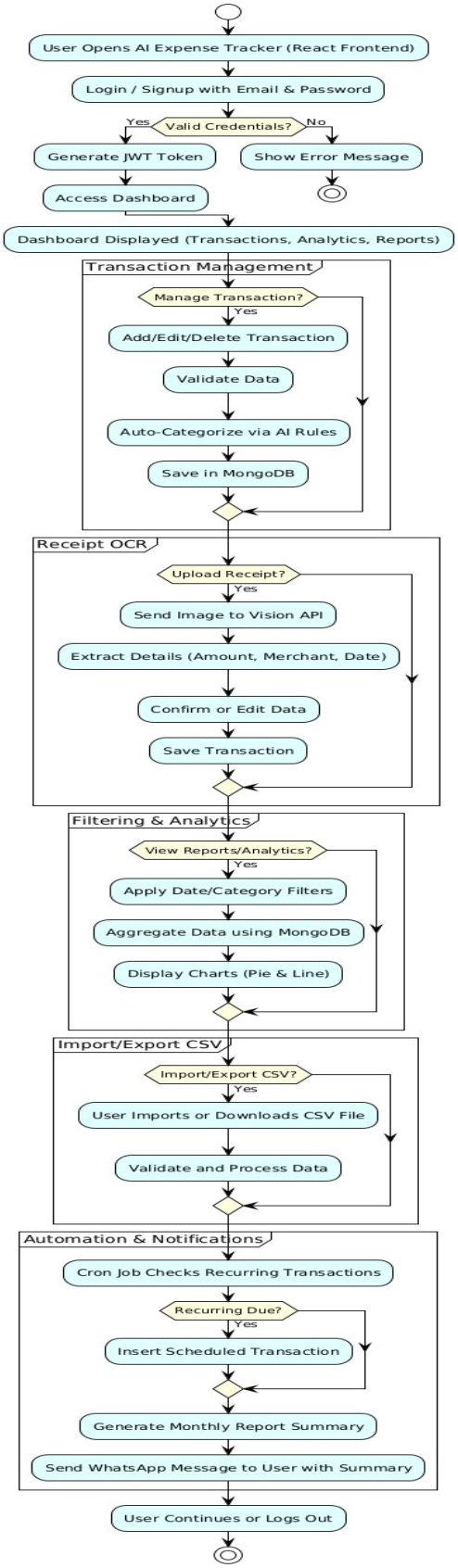
1. Use Case Title	Automate Monthly Financial Reporting
2. Abbreviated Title	Auto Report
3. Use Case Id	5
4. Actors	System, Cron Jobs, Email Service
5. Description	This use case describes the automated process by which the system generates a comprehensive financial summary report and emails it to the user at the end of each month. The process is scheduled and executed using **Cron Jobs**.
5.1. Pre Conditions:	The system's **Cron Job** must be configured and active. User must have a valid email address on file.
5.2. Task Sequence	1. The **Cron Job** triggers the report generation process on the scheduled date (e.g., the first day of the month for the prior month's data). 2. System retrieves all relevant transaction data from MongoDB. 3. System compiles the data into the auto-generated financial summary report. 4. System uses the integrated Email Service to send the summary to the user's registered email address. 5. System logs the report generation status.
5.3. Post Conditions:	• The user receives a comprehensive monthly financial report via email. • Report generation status is logged by the system.
6. Modification History:	Date - [Current Date]

1. Use Case Title	Manage Recurring Transactions
2. Abbreviated Title	Recurring Entry
3. Use Case Id	6
4. Actors	User, System, Cron Jobs, MongoDB
5. Description	This use case allows the user to define a transaction as recurring (e.g., monthly rent or subscription), which the system then automatically adds to the database on schedule using **Cron Jobs**.
5.1. Pre Conditions:	User must be logged in.
5.2. Task Sequence (User Action)	1. User selects a transaction and designates it as "Recurring." 2. User specifies the frequency (e.g., Monthly, Weekly) and start date. 3. System saves the recurring entry template in the database.
5.2. Task Sequence (System Automation)	4. The **Cron Job** runs daily to check for pending recurring transactions. 5. If a due date matches, the System creates a new, non-recurring transaction entry based on the template. 6. System updates the status or next due date of the recurring template.
5.3. Post Conditions:	• A recurring transaction template is successfully stored. • The system automatically generates new transaction records on the specified schedule.
6. Modification History:	Date - [Current Date]

2.2 SWIMLANE DIAGRAM AND ACTIVITY DIAGRAM

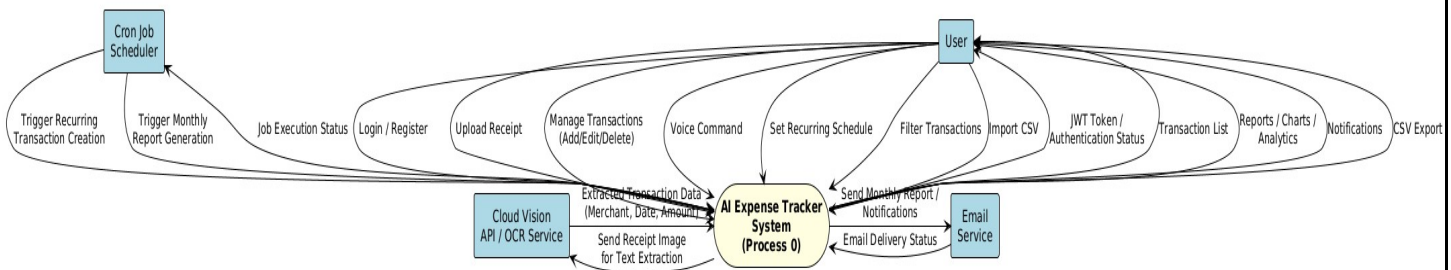


AI Expense Tracker – Simplified Activity Diagram (WhatsApp Notification)

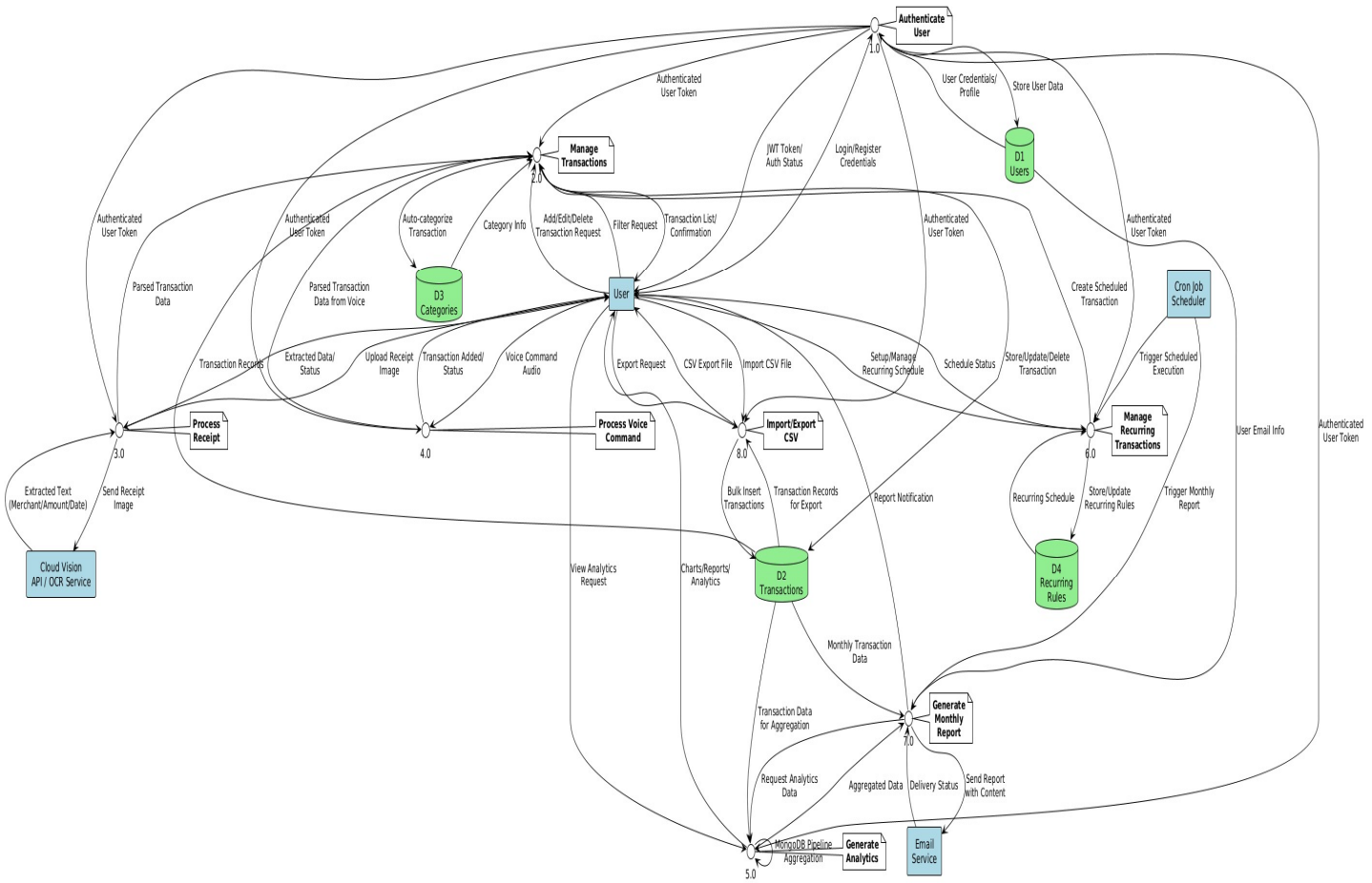


2.3 DATA FLOW DIAGRAMS (DFDs)

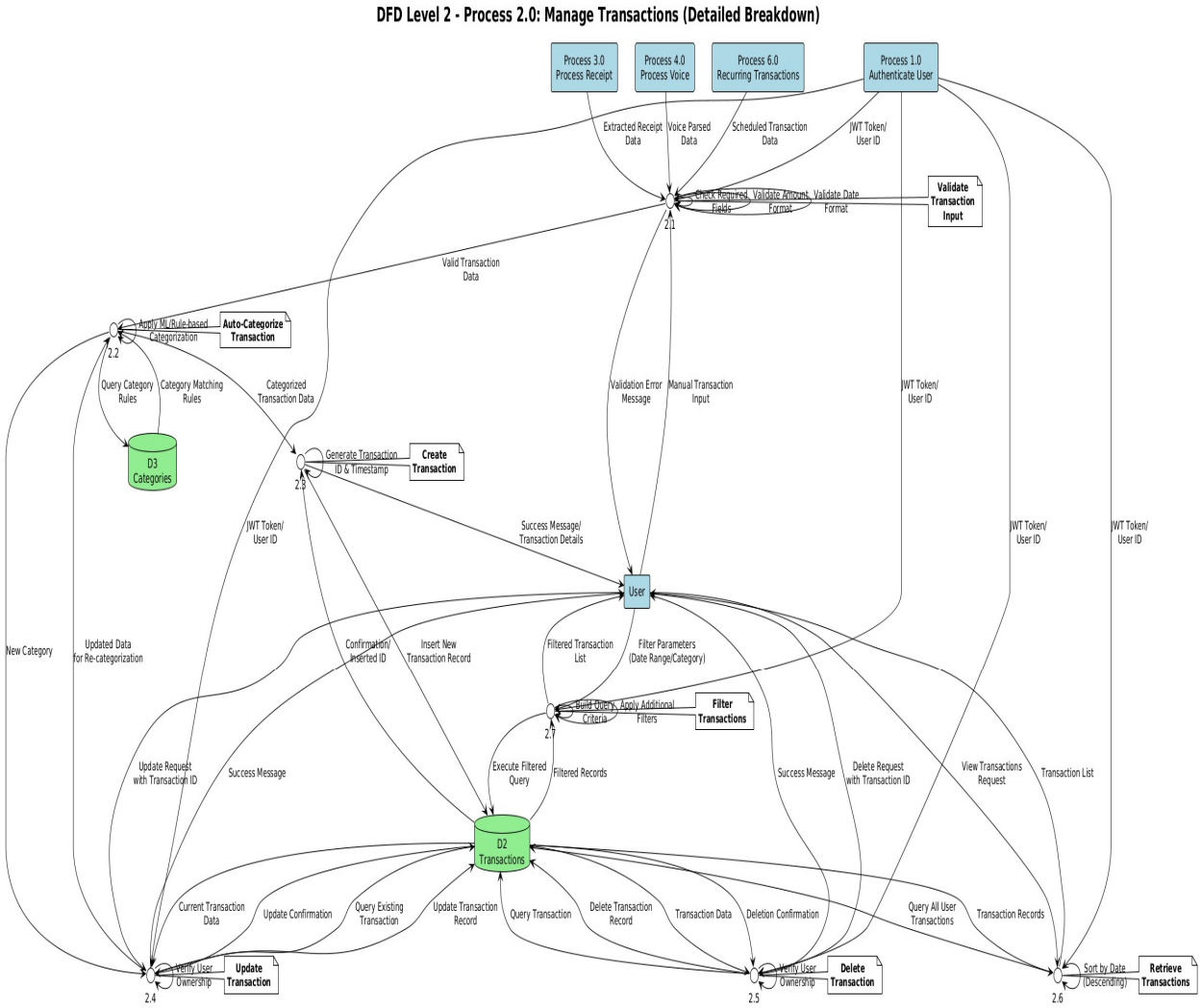
2.3.1 DFD LEVEL 0



2.3.2 DFD LEVEL 1



2.3.3 DFD LEVEL 2



2.4 Software Requirement Specification in IEEE

A CASE STUDY (IEEE Format)

Software Requirements Specification Document

Version 1.0

AI EXPENSE TRACKER

Table of Contents

1. Introduction

1.1 Purpose of this Document

The primary purpose of this SRS is to meticulously document the requirements for the **AI Expense Tracker** system. This document serves as a foundational contract between the development team and stakeholders, providing a precise, unambiguous, and complete definition of the system's objectives, scope, functional capabilities, and performance criteria. It aims to deliver a clear vision for an innovative solution that simplifies personal and business finance management.

1.2 Scope of the Development Project

The software must be able to perform the following operations:

1. User Authentication and Profile Management:
I must allow secure login and profile management, storing user personal details, authentication credentials, and preferences. Based on this, the system will deliver personalized financial insights and expense tracking recommendations tailored to each user's spending patterns.

2. Voice-Enabled Transaction Entry
It must help users create and manage financial transactions by supporting voice input through the Web Speech API, allowing natural language entry of transactions with automatic parsing of amounts, categories, merchants, and payment methods. The system should intelligently extract transaction details from conversational input.

3. AI-Powered Receipt Scanning
It must allow users to upload receipt images and automatically extract transaction details using AI/OCR technology. The system must identify merchant names, amounts, dates, categories, and payment methods from scanned receipts and automatically populate transaction records for easy tracking.

4. Smart Transaction Management
It must record all financial transactions with automatic categorization into predefined categories (groceries, dining, transportation, utilities, entertainment, shopping, healthcare, travel, housing, income, investments). The system should support creating, editing, deleting, and organizing transactions with flexible filtering options.

5. Advanced Analytics and Reporting
It must provide comprehensive financial insights such as category-wise spending

breakdown, income vs. expense trends, budget tracking, and cost optimization recommendations. The system should generate automated monthly reports with AI-powered insights emailed directly to users.

6. Recurring Transactions and Automation
It must support setting up recurring transactions with configurable frequencies (daily, weekly, monthly, yearly) and automatically create these transactions using scheduled cron jobs. The system should handle next occurrence calculation and provide users with automated expense tracking for subscriptions and regular payments.

7. Bulk Data Management
It must allow users to import and export transaction data in CSV format for bulk operations and data portability. The system should provide seamless migration capabilities, support large datasets (up to 300 transactions), and maintain data integrity during import/export operations.

8. Real-Time Financial Dashboard
It must provide a real-time overview of financial health through interactive charts and visualizations. The dashboard should display expense breakdowns, income vs. expense trends, recent transactions, and key financial metrics to help users make informed spending decisions.

1.3 Definitions, Abbreviations, and Acronyms

Term	Definition
SRS	Software Requirements Specification
AI	Artificial Intelligence
OCR	Optical Character Recognition (The technology underlying the AI Receipt Scanning)
JWT	JSON Web Token (The standard for secure authentication)
CRUD	Create, Read, Update, Delete (The fundamental database operations)
Cron Job	A time-based job scheduler for automated, recurring tasks in the system
MongoDB	The NoSQL database used for data persistence
CSV	Comma Separated Values (Used for bulk data import/export)

---

2. Overall Description

2.1 Product Perspective
The AI Expense Tracker is an **independent, cutting-edge web application** built entirely on a modern, scalable MERN-based stack (React, Node, Express, MongoDB).

1. **Frontend (UI):** Developed with **React.js** for a dynamic and responsive user experience.
2. **Backend (API):** Built with **Node.js and Express.js**, providing a high-performance RESTful API.

3. **Database (Persistence): MongoDB** is used for its flexibility and scalability, perfectly supporting the analytical needs.
4. **Key Integration:** Crucial integration with a **Cloud-based Vision API** powers the AI/OCR capabilities for receipt scanning.

2.2 Product Functions (Key Features)

The AI Expense Tracker provides the following transformative functions:

Function ID	Feature Name	Description
F-01	**Secure Authentication**	Provides high-security user access using **JWT-based** email and password authentication.
F-02	**AI Receipt Scanning**	Allows users to upload a receipt image and uses OCR to **automatically extract** the merchant, amount, and date.
F-03	**Voice Assistant**	Allows users to record their voice and **automatically extract** the merchant, amount, and date.
F-04	**Smart Transaction Mgmt.**	Full CRUD capability for transactions, including automatic, intelligent categorization (e.g., Groceries, Rent, Travel).
F-05	**Advanced Analytics**	Visualizes financial data using **MongoDB aggregate pipelines** to generate interactive pie charts (expense breakdown) and line charts (income/expense trends).
F-06	**Recurring Transactions**	Automated creation of scheduled entries (e.g., monthly rent, subscription fees) using **Cron Jobs**.
F-07	**Monthly Reports**	Auto-generation of comprehensive financial summary reports, delivered directly to the user's email via an automated service.
F-08	**Flexible Filtering**	Robust filtering options for transactions by predefined ranges (e.g., Last 30 Days) and custom date periods.
F-09	**CSV Import/Export**	Allows for efficient bulk management of transactions through standard CSV file formats.

2.3 User Characteristics

The system caters to two main user groups, requiring only moderate digital literacy:

1. **General Users (Finance Managers):** Individuals or business owners focused on tracking their personal or company finances. They require an intuitive interface to manage transactions, upload receipts, and interpret analytical reports.
2. **Administrators (Future Scope):** Users responsible for system health, user support, and data integrity monitoring (potential future version).

2.4 General Constraints, Assumptions, and Dependencies

1. **Constraint:** The application is strictly a web-based platform in its initial release.
2. **Dependency (External API):** The core value proposition relies on the stable, performant operation of the **Cloud-based Vision API** for OCR/AI functions.

3. **Dependency (Automation):** The reliable scheduling of recurring entries (F-05) and report generation (F-07) is entirely dependent on the continuous operation and accuracy of the **Cron Job** mechanism.
4. **Assumption (Internet/Browser):** Users must have a stable internet connection and be using one of the latest versions of major web browsers (Chrome, Firefox, Safari, Edge).

---

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces (UI)

1. **Dashboard:** A highly scannable page presenting key metrics, total balances, and the primary analytical charts (F-05).
2. **Transaction View:** A sortable, searchable table with options for editing, deleting, and manually creating new entries (F-04).
3. **Receipt Upload:** A dedicated, simple interface for drag-and-drop or file selection for AI scanning (F-03).
4. The entire UI is built using **React.js** and must adhere to modern responsive design principles.

3.1.2 Software Interfaces

1. **Backend API:** The **Node.js/Express.js** API must expose secure, well-defined REST endpoints for all CRUD operations and data retrieval for analytics.
2. **OCR/AI Integration:** A secure client library must be used to communicate with the **Cloud-based Vision API** to pass receipt images and receive extracted JSON data.
3. **Email Service:** An integrated SMTP or third-party email service must be configured to reliably send the monthly reports (F-07).

3.2 Functional Requirements

The system must support the following detailed functional requirements:

ID	Requirement Detail	Priority
FR-1.1	The system shall register a new user with email and securely hashed password.	High
FR-1.2	The system shall authenticate returning users using **JWT** and maintain a secure session.	High
FR-2.1	The system shall process an uploaded image and return **Merchant Name**, **Transaction Amount**, and **Date of Transaction** via OCR.	High
FR-3.1	The system shall allow manual creation and categorization of transactions.	High
FR-3.2	The system shall automatically suggest a category for any newly created or scanned transaction.	Medium
FR-4.1	The system shall use **MongoDB aggregate pipelines** to calculate data for the expense breakdown pie chart.	High
FR-5.1	The system shall allow users to define a transaction as **recurring** with frequency (e.g., weekly, monthly).	High
FR-5.2	The system shall use a **Cron Job** to automatically create the recurring transaction entry on the scheduled date.	High

ID	Requirement Detail	Priority
FR-6.1	The system shall use a **Cron Job** to generate a comprehensive summary report at the end of every month.	High
FR-6.2	The system shall automatically email the generated report to the user's registered email address.	High
FR-7.1	The user interface shall include dynamic controls to filter transactions by **custom date ranges**.	Medium
FR-8.1	The system shall accept a **CSV file** and import bulk transactions into the database.	Medium

3.2 Performance Requirements

To ensure a highly responsive user experience, the system must meet the following performance benchmarks:

1. **Latency (Authentication):** User login must complete in under **1 second** to ensure a swift start.
2. **Processing Time (AI):** The entire **AI Receipt Scanning** process (upload to data display) must be completed within a maximum of **5 seconds** per image, even during peak load.
3. **Load Time (Analytics):** The main dashboard with all analytical charts must load completely in under **3 seconds** by optimizing the MongoDB queries and data transfer.

4. **Scalability:** The architecture must be capable of handling **1,000 active users** without degradation in the above performance metrics.

3.4 Security Requirements

1. **Authentication:** All password data must be securely **hashed and salted** before storage.
2. **Authorization:** All API endpoints must be protected with **JWT validation**, ensuring only the authenticated user can access their own financial data.
3. **Data Security:** The application must utilize **HTTPS/SSL** for all data transfer between the client and the server.
4. **Input Validation:** Robust server-side validation must be implemented on all user inputs to prevent injection attacks.

3.5 Quality Attributes

Attribute	Description
Usability	The UI must be **intuitive** and require minimal training, focusing on a clean design to make complex financial data easy to understand.
Reliability	The automated **Cron Jobs** must be highly reliable, ensuring that users never miss a scheduled transaction or a monthly report.
Portability	The React.js frontend ensures the application is **cross-browser compatible** and runs effectively on all major operating systems.
Maintainability	The modular architecture (**React.js** frontend, **Node.js/Express.js** backend) promotes ease of future maintenance, bug fixing, and feature additions.
Scalability	Utilizing **MongoDB** and a microservices-friendly Node.js architecture allows the system to scale efficiently as the user base grows.

4. Change History

Version	Date	Author	Description
1.0	November 2025	CodeFusion	Initial Release of the Software Requirements Specification.

5. Document Approvers

2.5 User Stories and Story Cards

#0001 NEW USER REGISTRATION

User Story	As a new user, I want to create a secure account using my email and password, so that I can access personalized features and save my financial data.
Story Points:	3
Mockup Sketch	
Confirmation	**1. Success** – New account is **created** using **JWT** authentication, user is automatically logged in and referred to the main dashboard. A welcome email is dispatched. **2. Failure** – Display specific error message: a) "Invalid email format. Please enter a valid email." b) "Password is too weak. Please include uppercase, lowercase, numbers, and symbols." c) "This email is already registered. Please log in instead." d) "Registration is currently unavailable. Please try again later."

#0002 AI RECEIPT SCANNING

User Story	As a user with a receipt, I want to upload a photo of it so that the AI automatically extracts the amount, merchant, and date, eliminating manual data entry.
Story Points:	8
Mockup Sketch	
Confirmation	**1. Success** – Transaction details are successfully extracted, displayed in the form, and saved to the database upon confirmation. **2. Failure** – Display specific error message: a) "Image not clear enough. Please re-upload a clearer image." b) "Vision API failed to extract key details. Please enter manually." c) "Upload failed. Please check your connection and try again." d) "Transaction amount extracted is zero. Please correct or enter manually."

#0003 VOICE ASSISTANT ENTRY

User Story	As a busy user, I want to use the voice assistant to quickly add an expense, so that I can track spending without needing to type anything.
Story Points:	5
Mockup Sketch	
Confirmation	**1. Success** – Voice command is transcribed and parsed; the transaction is confirmed and saved to the database. A success toast notification is displayed. **2. Failure** – Display specific error message: a) "Could not hear command. Please speak louder." b) "Speech not clearly parsed. Please try again or enter manually." c) "Transaction incomplete (missing amount or merchant). Please retry." d) "Microphone access denied. Please grant permission in settings."

#0004 RECURRING TRANSACTION SETUP

User Story	As a user with repeating bills, I want to set up an expense as recurring, so that the system automatically logs the transaction every month without my input.
Story Points:	5
Mockup Sketch	
Confirmation	**1. Success** – The transaction is saved as a recurring template, and the **Cron Job** is configured to execute the entry on the specified schedule. **2. Failure** – Display specific error message: a) "Cannot set frequency without a Next Due Date." b) "Failed to save recurring template. Please try again later." c) "The provided date is in the past. Please select a future date."

#0005 VIEW ADVANCED ANALYTICS

User Story	As a finance-conscious user, I want to view dynamic charts and graphs of my spending, so that I can clearly see where my money is going and adjust my budget.
Story Points:	8
Mockup Sketch	
Confirmation	**1. Success** – Charts are rendered within 3 seconds, showing a clear visualization of the financial data for the selected period, powered by the **MongoDB aggregation pipeline**. **2. Failure** – Display specific error message: a) "No transactions found for the selected period." b) "Failed to retrieve data from the server. Please check your connection." c) "Chart rendering error. Data integrity compromised."

#0006 FLEXIBLE FILTERING

User Story	As a user, I want to filter my transaction list by custom date ranges and categories so that I can quickly find specific expenses or review spending for a certain period.
Story Points:	3
Mockup Sketch	
Confirmation	**1. Success** – The transaction list instantly updates to display only transactions matching the **custom date range** and/or selected category filter. **2. Failure** – Display specific error message: a) "Start date cannot be after end date for custom range." b) "No transactions match your selected filters." c) "Filtering service temporarily unavailable. Please try again."

#0007 SMART TRANSACTION MANAGEMENT (EDIT/DELETE)

User Story	As a user, I want to easily edit or delete any recorded transaction, so that I can correct errors and maintain accurate financial records.
Story Points:	2
Mockup Sketch	
Confirmation	**1. Success (Edit)** – The transaction details are successfully updated in the database, and the analytics dashboard reflects the change instantly. **1. Success (Delete)** – The transaction is permanently removed from the database, and the user receives a confirmation message. **2. Failure** – Display specific error message: a) "Failed to save changes. Please ensure all fields are valid." b) "Cannot connect to the database to process the deletion. Please try again."

#0008 USER LOGIN (EXISTING USER)

User Story	As an existing user, I want to log in to my account securely with my email and password, so that I can access my saved financial data and profile information.
Story Points:	2
Mockup Sketch	
Confirmation	**1. Success** – Valid user logged in, a **JWT token** is issued, and the user is referred to the main dashboard. **2. Failure** – Display specific error message: a) "Invalid email format. Please enter a valid email." b) "No account found with this email. Please register." c) "Incorrect password. Please try again." d) "Login service is temporarily unavailable. Please try again later."