Software Requirements Specification (SRS) Document

Project Name – Finease
Team Number – 2
Team Members:

- Swam Singla
- Kushagra Trivedi
- Ronak Gaur
- Ayush Gupta
- Nidhi Vaidya

Brief problem statement

Finease is a comprehensive platform that streamlines business operations such as data entry and regulatory compliance for ITR, GST, and PF filings. The platform leverages WhatsApp and a custom chatbot for user interaction, automates document processing, and provides functionalities for generating E-Waybills, delivery challans, and displaying key financial metrics on a dashboard.

System requirements

FinEase Project Technology Stack:

Frontend & Backend:

- Node.js
- Express.js
- React.js (Web App)
- React Native (Mobile App)
- MongoDB

APIs & Integrations:

- Gemini API (For general tax filing questions)
- NanoNets

Storage:

- Structured database for operational and compliance data (MongoDB Atlas)

Chatbot Integration:

- WhatsApp API with user authentication
- Custom app chatbot

Document Processing:

- File uploads for invoices, Form 16, and challans

Tools and Libraries:

Python Libraries:

- py-tesseract for OCR
- PIL (Python Imaging Library) for image processing

Frontend Styling:

- TailwindCSS

Tools:

- Postman (API Testing)
- Ngrok (Local hosting)
- Twilio (Webhook)
- NanoNets (Model Training)
- Expo Go (Mobile App Testing)

Collaboration Tools:

- GitHub for version control

Deployment:

- Vercel (Frontend)
- Render (Backend)
- Docker (Backend)

User Profiles

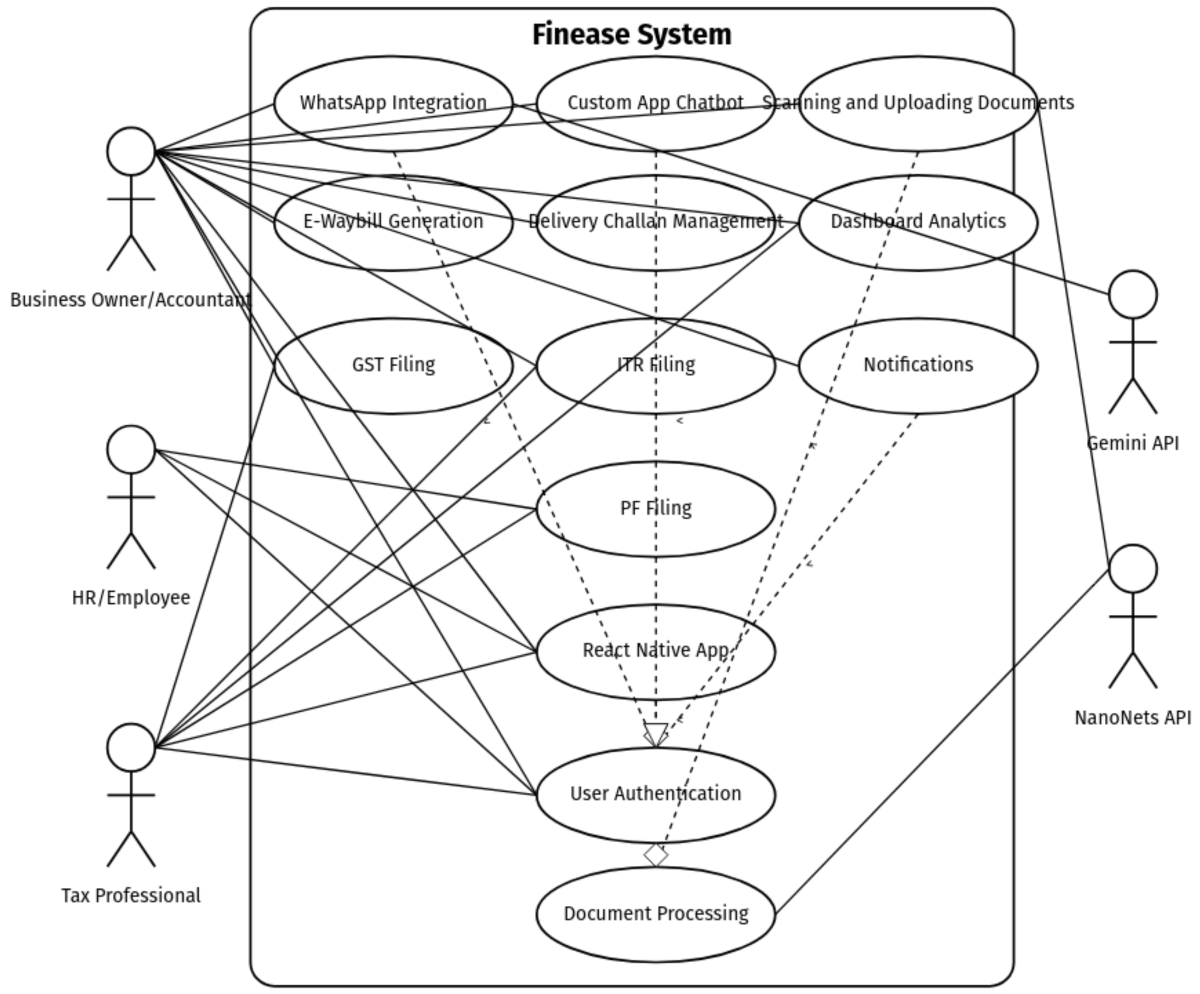
1. **Business Owners & Accountants**
    - Familiarity Level: Basic-to-Medium
    - Role: Utilize automation for tax filings, invoice generation, and review dashboard metrics. Reduced manual effort due to AI-powered processing.
2. **Employees & HR Personnel**
    - Familiarity Level: Basic
    - Role: Submit financial records and compliance documents. Now also interact with document classification and label extraction features.
3. **Tax Professionals**
    - Familiarity Level: Advanced

- Role: Leverage automated insights and dashboard analytics to streamline filings instead of manual data processing.

Feature requirements (described using use cases)

No.	User Case Name	Description	Release	Status
1	Whatsapp Integration	Collects invoices, purchase records from users via WhatsApp with added user authentication. Integrated with Gemini API for general tax filing questions	R1	Completed
2	Custom App Chatbot	Guides users through the website for seamless navigation	R1	Completed
3	E-Waybill Generation	Automates E-Waybill creation through API integration	R1	Completed
4	Delivery Challan Management	Generates challans for goods transport	R1	Completed
5	ITR Filing	Displays data in JSON format ready for filing. Direct API integration for filing was not implemented due to API unavailability	R2	Modified
6	GST Filing	Displays data in JSON format ready for filing. Direct API integration for filing was not implemented due to API unavailability	R2	Modified
7	PF Filing	Enables employees to add information about their employees and generate ECR documents	R2	Modified
8	Dashboard Analytics	Displays key metrics such as lifetime earnings and tax saved on a dashboard interface	R2	Added
9	Scanning and uploading Documents	Scan and upload documents for GST/PF/ITR filing and classify them and extract data from them using external API	R1	Completed
10	Notifications	Sending automated notifications through the website interface	R2	Modified
11	React Native App	Cross-platform application that works on both web version and Expo Go	R2	Added

Use Case Diagram



Use case descriptions

Use Case Number: UC-01

Use Case Name: GST Filing
Overview:
This use case describes how users can prepare GST returns (GSTR-1 and GSTR-3B) through the system. The system displays the processed data in JSON format that is ready for filing. Due to unavailability of government APIs, direct submission functionality was not implemented.

Actors:

- User (Business Owner/Accountant)
- Website
- Backend System

Pre-conditions:

- The user must have a valid GSTIN.
- Sales and purchase invoices must be recorded in the system.

Flow: **Main (Success) Flow**:

1. The user logs into the website and navigates to the GST Filing section.
2. The system retrieves relevant sales and purchase data.
3. The system validates GSTIN, HSN codes, and tax amounts.
4. The processed data is displayed in JSON format ready for filing.
5. The user can download the JSON data for manual submission to the GST portal.

Alternate Flows:

- Invalid Data: If incorrect GSTIN, HSN codes, or tax amounts are detected, the system prompts the user for corrections.

Post-condition:

- The GST return data is successfully processed and displayed in JSON format.
- The user can download the data for manual submission.

Use Case Number: UC-02

Use Case Name: WhatsApp Integration
Overview:
This use case describes the process of integrating WhatsApp as a platform for user interaction, allowing users to submit invoices, purchase records, and other financial data, request filings about their tax and compliance activities. The system includes user authentication and Gemini API integration for answering general tax filing questions.

Actors:

- User (Business Owner/Accountant)
- WhatsApp Chatbot
- Backend System
- Gemini API

Pre-condition:

- The user must have an active WhatsApp account.
- The chatbot must be authorized and integrated with the system.
- Required APIs (WhatsApp Business API, Gemini API, backend processing) must be configured.

Flow: **Main (Success) Flow**:

1. The user initiates a conversation with the WhatsApp chatbot.
2. The chatbot prompts the user to log in by providing credentials.

3. The user enters their login credentials.
4. The system validates the credentials against the stored user data.
5. If valid, the system generates an authentication token and grants access to the WhatsApp chatbot services.
6. The chatbot presents available options (e.g., GST filing, PF filing, ITR filing, document classification, general tax questions).
7. The user selects an option:
    - For filing-related options: The user submits relevant data (e.g., invoice details, Form 16 upload).
    - For general tax questions: The query is forwarded to the Gemini API for processing.
8. The chatbot validates the data and stores it in the backend or receives responses from Gemini API for tax questions.
9. The system processes the request (e.g., prepares GST filing data).
10. The user receives a confirmation message with details and next steps.

Alternate Flows:

- Invalid Data: If the submitted data is incomplete or incorrect, the chatbot prompts the user to provide corrections.
- Authentication Failure: If the user enters incorrect credentials, the system displays an error message and prompts for reentry.
- Query Not Recognized: If the query is not recognized as tax-related, the system may request clarification or direct the user to appropriate resources.
- Gemini API Unavailability: If Gemini API is unavailable, the system notifies the user.

Post Condition:

- The user's request is successfully processed, and the relevant data is stored.
- The user receives informative answers to their general tax-related questions.
- The authentication status is maintained throughout the user's session.

Use Case Number: UC-03

Use Case Name: Custom App Chatbot
Overview:
This use case describes the functionality of a custom app chatbot integrated within the website. The chatbot assists users by providing basic guidance on how to file GST, ITR, and PF filings, as well as how to generate invoices, E-Waybills, and delivery challans on the website.

Actors:

- User (Business Owner/Accountant)
- Custom App Chatbot
- Website

Pre-condition:

- The user must have access to the website where the chatbot is integrated.
- The chatbot must be active and capable of providing responses based on user queries.

Flow: **Main (Success) Flow**:

1. The user accesses the website and initiates a conversation with the chatbot.
2. The user asks a question related to filing GST, ITR, PF, or generating invoices/E-Waybills/delivery challans.
3. The chatbot provides relevant guidance in text format (e.g., steps to follow, required documents).
4. The user follows the chatbot's guidance and completes the process within the website.

Alternate Flows:

- User Asks an Unsupported Question: If the chatbot does not recognize the question, it responds with alternative suggestions such as:
  - "I'm sorry, but I don't have an answer to that. Could you please ask something else?"

Post Condition:

- The user is successfully guided to the relevant section within the website for filing or document generation.
- The chatbot ensures the user understands how to proceed but does not directly handle the filing process.

Use Case Number: UC-04

Use Case Name: E-Way Bill Generation
Overview:
The E-Way Bill Generation use case describes how users can generate an E-Way Bill through the website. The system provides a structured interface where users can input relevant details. Once submitted, the system processes the request and provides the generated E-Way Bill for download.

Actors:

- User (Business Owner/Transporter/Accountant)
- Website

Pre-conditions:

- The user must have access to the website.
- The required details (e.g., invoice details, consignee/consignor information, HSN codes) must be available.

Flow: **Main (Success) Flow**:

1. The user accesses the website and navigates to the E-Way Bill Generation section.
2. The system prompts the user to enter details such as consignee/consignor details, HSN codes, invoice reference, and transport details.
3. The user submits the required information.
4. Upon successfully submitting, the system generates the E-Way Bill.
5. The E-way Bill gets downloaded.

Alternate Flows: NONE

Post-condition:

- The E-Way Bill is successfully generated and gets downloaded.
- The user receives confirmation via the website.

Use Case Number: UC-05

Use Case Name: Delivery Challan Management
Overview:
This use case describes the process of generating and managing delivery challans for various scenarios such as goods transportation, returns, or samples. The system allows users to create and store challans for compliance purposes.

Actors:

- User (Business Owner/Transporter/Accountant)
- Website
- Backend System

Pre-condition:

- The user must have shipment details, including invoice references, quantity, and consignee/consignor details.
- The system must have the ability to generate and store delivery challans.

Flow: **Main (Success) Flow**:

1. The user initiates a request to generate a delivery challan via the app interface.
2. The system prompts the user to enter the required details, including:
    - Goods description and quantity
    - Shipment details (consignee/consignor information)
3. A delivery challan is generated and stored in the system.
4. The system provides the user with a downloadable and shareable PDF of the delivery challan.
5. The user receives a confirmation message via website.

Alternate Flows:

- Challan Retrieval: Users can retrieve previously generated challans.

Post Condition:

- The delivery challan is successfully generated and stored in the system.
- The user receives a confirmation along with a downloadable document.

Use Case Number: UC-06

Use Case Name: Income Tax Return (ITR) Filing
Overview:
This use case describes how users can prepare Income Tax Returns (ITR) by extracting data from Form 16 and other financial documents. The system processes and displays the data in JSON format ready for filing. Due to unavailability of government APIs, direct submission functionality was not implemented.

Actors:

- User (Taxpayer/Accountant/Business Owner)
- Website
- Backend System

Pre-conditions:

- The user must have valid financial documents, including Form 16, salary details, and tax deduction records.
- The system must support data extraction and validation for ITR filing.

Flow: **Main (Success) Flow**:

1. The user logs into the website and navigates to the ITR Filing section.
2. The system prompts the user to upload Form 16 and other necessary financial documents.
3. The system extracts relevant data and processes it into JSON format.
4. The system validates the extracted information, checking for errors or missing details.
5. The user reviews the filing details, can also manually change the details if needed.
6. The system provides the processed data in JSON format for the user to download.
7. The user can manually submit this data to the IT Portal.
8. The user receives a confirmation message about the successful data processing.

Alternate Flows:

- Invalid Data: If missing or incorrect income, deductions, or tax details are detected, the system prompts the user for corrections.

Post-condition:

- The ITR data is successfully processed and displayed in JSON format.
- The user receives confirmation upon successful data processing.

Use Case Number: UC-07

Use Case Name: Provident Fund (PF) Filing
Overview:
This use case describes how users can prepare Provident Fund (PF) contributions by adding employee information and generating Employee Contribution Returns (ECR) documents.

Actors:

- User (HR/Employer/Accountant)
- Website
- Backend System

Pre-conditions:

- The user must have valid employee salary and PF contribution details.
- The system must be capable of generating ECR documents in the required format.

Flow: **Main (Success) Flow**:

1. The user logs into the website and navigates to the PF Filing section.
2. The system provides an interface for the user to add employee information, including salary details and PF contributions.
3. The user enters or uploads employee data.
4. The system validates the data and calculates PF contributions accordingly.
5. The system generates an ECR document based on the provided information.
6. The user reviews the details, can manually adjust if needed, and confirms generation.
7. The user receives a confirmation message upon successful document generation.

Alternate Flows:

- Invalid Data: If missing or incorrect salary, PF contribution, or employee details are detected, the system prompts the user for corrections.

Post-condition:

- The ECR document is successfully generated.
- The user receives a confirmation message upon successful document generation.

Use Case Number: UC-08

Use Case Name: Dashboard Analytics
Overview:
This use case describes how the system displays key financial metrics such as lifetime earnings and tax saved on a dashboard interface, providing users with at-a-glance insights into their financial status.

Actors:

- User (Business Owner/Accountant)
- Website

- Backend System

Pre-condition:

- The user must have financial data stored in the system.
- The system must be capable of calculating and displaying analytics.

Flow: **Main (Success) Flow**:

1. The user logs into the website and accesses the dashboard.
2. The system retrieves relevant financial data from the database.
3. The system calculates key metrics such as lifetime earnings and tax saved.
4. The dashboard displays these metrics in an easy-to-understand format.
5. The user can view the metrics and use them for financial decision-making.

Alternate Flows:

- Insufficient Data: If there is insufficient data to calculate certain metrics, the system displays placeholder values or informative messages.

Post-condition:

- The user gains insights into their financial status through the dashboard metrics.
- The system successfully displays the calculated analytics.

Use Case Number: UC-09

Use Case Name: Scanning and Uploading Documents
Overview:
This use case describes the process of scanning and uploading financial and compliance-related documents, such as invoices, Form 16, challans, and other necessary paperwork. The system ensures proper categorization and storage of uploaded files for further processing.

Actors:

- User (Business Owner/Accountant)
- WhatsApp Chatbot
- Website
- Backend System

Pre-condition:

- The user must have documents ready for upload.
- The system must support document scanning, categorization, and storage.

Flow: **Main (Success) Flow**:

1. The user initiates a document upload request via WhatsApp chatbot or the app interface.

2. The system prompts the user to select the document type (e.g., invoice, Form 16, challan, tax receipt).
3. The user scans or selects an existing document and uploads it.
4. The system processes the document using OCR (if applicable) to extract relevant information.
5. The system validates the document format and checks for required fields using external API (NanoNets).
6. If the document is valid, it is stored in the appropriate category (e.g., compliance, transactions).
7. The user receives a confirmation message via WhatsApp or the app interface.

Alternate Flows:

- Invalid Document Format: If the uploaded document is in an unsupported format, the system prompts the user to retry with a valid format.
- Failed Upload: If the upload fails due to network issues, the system notifies the user and allows a retry.
- Manual Data Entry: If API processing fails, the system prompts the user to manually enter the extracted data.

Post Condition:

- The document is successfully uploaded and stored for future reference.
- The user receives a confirmation.

Use Case Number: UC-10

Use Case Name: Notifications
Overview:
This use case describes the process of sending automated notifications to users regarding important updates such as filing deadlines and document uploads through the website interface.

Actors:

- User (Business Owner/Accountant)
- Website
- Backend System

Pre-condition:

- The user must have an active account with contact details registered.
- The system must have access to relevant compliance and financial data.

Flow: **Main (Success) Flow**:

1. The system detects an event that requires user notification (e.g., tax filing due date).
2. Based on predefined triggers, the system generates a notification message.
3. The system displays the notification on the website interface when the user logs in.

4. The user receives the notification and can take necessary action.

Alternate Flows:

- User Does Not View Notification: If the user doesn't log in to view the notification, it remains visible until marked as read.

Post Condition:

- The user is informed of important updates related to financial and compliance processes through the website interface.
- The system ensures timely actions by displaying alerts to prevent delays or penalties.

Use Case Number: UC-11

Use Case Name: React Native App
Overview:
This use case describes the functionality of the React Native app that works on both web version and Expo Go, providing cross-platform accessibility to Finease features.

Actors:

- User (Business Owner/Accountant)
- React Native App
- Backend System

Pre-conditions:

- The user must have access to a web browser or the Expo Go app.
- The user must have valid credentials for authentication.

Flow: **Main (Success) Flow**:

1. The user opens the React Native app on their web browser or through Expo Go on their mobile device.
2. The user logs in using their credentials.
3. The app authenticates the user and displays the dashboard with available features.
4. The user navigates through the app to access various functionalities such as GST filing, ITR filing, PF filing, E-Waybill generation, etc.
5. The app synchronizes with the backend system to retrieve and update data as needed.

Alternate Flows:

- Authentication Failure: If the user enters incorrect credentials, the app displays an error message and prompts for reentry.
- Connectivity Issues: If there are connectivity problems, the app notifies the user and attempts to reconnect.

Post Condition:

- The user can access all Finease features through a cross-platform application.
- The app provides consistent functionality across web and mobile platforms(Android and IoS).