K. J. Somaiya School of Engineering

Department of Computer Engineering

Batch: A-4	**Roll No.:** 16010122151
Experiment No:- 3	
Group No: 5	

Title: Prepare design document and Plan of project

Objective: Chapter No.3 of Mini Project Report will include detailed design document and plan of implementation of the project

Expected Outcome of Experiment:

	At the end of successful completion of the course the student will be able to
CO2	Identify various hardware and software requirements for problem solution
CO5	Prepare a technical report based on the Mini project.

Books/ Journals/ Websites referred:

1.
2.
3.

The students are expected to prepare chapter no 3 in the format given below

Chapter 3

Design Document and Project plan

*A **design document** is crucial in a software project because it serves as a blueprint that outlines the architecture, components, data flow, and technical specifications of the system before implementation **Clear Vision & Planning** will improve collaboration with in the team members. The*

Introduction

Purpose of the Document

This document serves as **Chapter 1 of the Mini Project Report** for the project **"Chat with PDF."** It aims to introduce the project by outlining its motivation, objectives, and significance in addressing challenges related to **efficient information retrieval from PDFs**. The document details the project's **functionality, scope, and technological foundation**, providing a clear understanding of how the system leverages **AI-powered document interactions** to enhance accessibility and productivity.

Expected Audience

The primary audience for this document includes:

- **Students & Researchers** – To help retrieve information from research papers, textbooks, and other academic materials.
- **Software Developers & Hackathon Participants** – To assist in analyzing problem statements and suggesting problem-solving approaches.
- **Legal & Corporate Professionals** – To extract relevant details from contracts, policies, and legal documents.

- **Educators & General Users** – To efficiently navigate and query large PDFs without manual searching.

Scope of the Project (Brief)

The **Chat with PDF** project focuses on enabling users to **upload PDFs and interact with them** using natural language queries. It extracts text from documents and utilizes **Groq's Llama 3 API** to generate intelligent responses beyond simple text retrieval. The chatbot enhances efficiency by:

1. **Providing context-based responses** rather than relying on basic keyword searches.
2. **Suggesting problem-solving strategies and tech stacks** for technical documents.
3. **Enhancing knowledge accessibility** in various domains, including education, law, and corporate sectors.

However, the system has limitations, such as **processing only text-based PDFs** (not scanned/image-based ones) and a **file size restriction of 200MB**.

Definitions, Acronyms, and Abbreviations

- **AI (Artificial Intelligence)** – Technology that enables machines to simulate human intelligence.
- **LLM (Large Language Model)** – A machine learning model trained to understand and generate human-like text.
- **Flask** – A Python-based web framework for developing backend applications.
- **PyPDF2/PdfPlumber** – Python libraries used for extracting text from PDFs.
- **Groq's Llama 3 API** – A cloud-based LLM API used for generating intelligent responses.
- **OAuth (Open Authorization)** – A protocol that allows secure user authentication via third-party services like Google and GitHub.

References

The section is drafted based on standard software documentation resources, including:

1. **Software Requirement Specification (SRS) Guidelines** – IEEE 830-1998 Standard for writing SRS documents.
2. **Flask Documentation** – Official Flask framework documentation for backend implementation.
3. **PyPDF2 & pdfplumber Docs** – Python libraries for text extraction from PDFs.

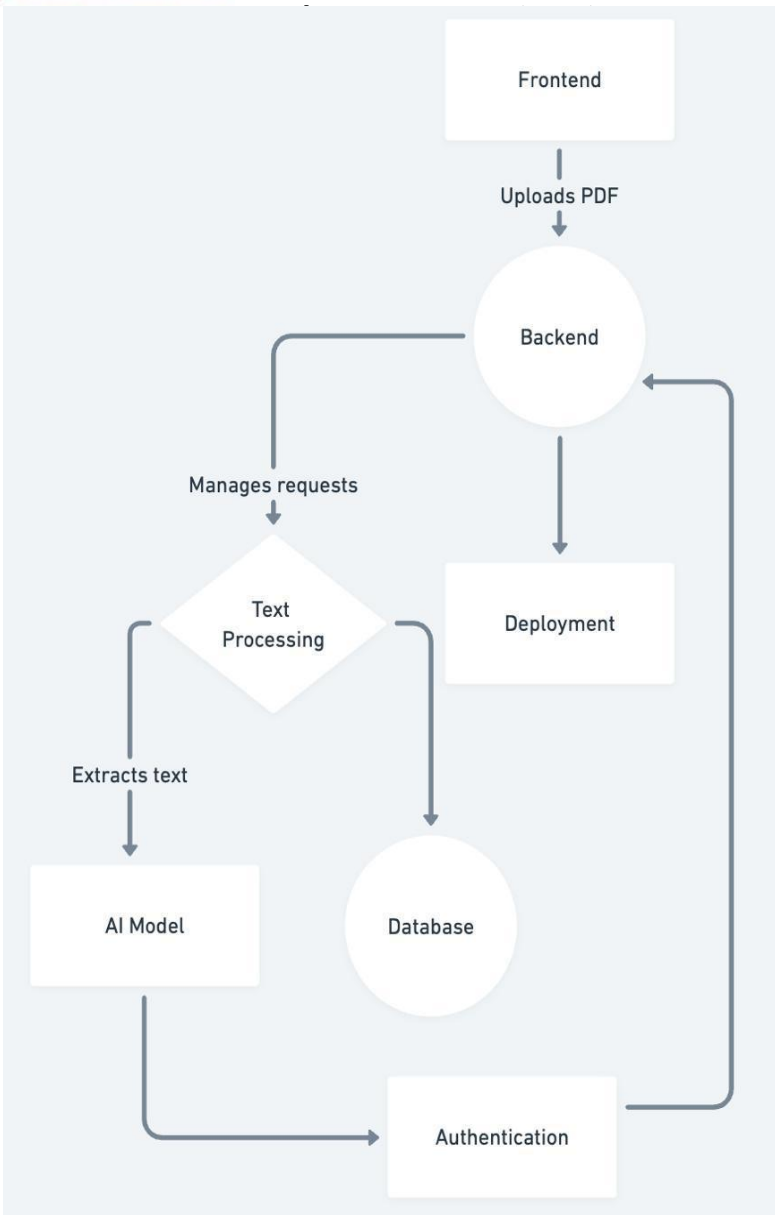
4. **Groq's Llama 3 API Documentation** – Reference material for integrating AI-based language models.

2.System Over View

2.1 System Architecture

The **Chat with PDF** system is designed as a **web-based AI chatbot** that allows users to interact with PDF documents using **natural language queries**. Below is the high-level **architectural diagram** and a description of how the components interact:

Architectural Diagram


Frontend
Uploads PDF
Backend
Manages requests
Deployment
Text Processing
Extracts text
AI Model
Database
Authentication

2.1 Design Goals

The system follows key **software engineering principles** to ensure robustness, efficiency, and scalability:

1. Scalability

- Uses **Flask** as a lightweight backend framework, making it easy to scale.
- Can be extended to support multiple users simultaneously.

2. Security

- Implements **OAuth authentication (Google/GitHub)** to prevent unauthorized access.
- Limits API access to prevent abuse.

3. Performance Optimization

- Uses **PyPDF2/pdfplumber** for **efficient text extraction**.
- Minimizes API response time with **optimized query processing**.

4. Maintainability

- Modular design separates **frontend, backend, and AI components** for easier debugging.
- Uses **well-documented APIs and libraries** for future maintainability.

3. Detailed Design

3.1 Module Description

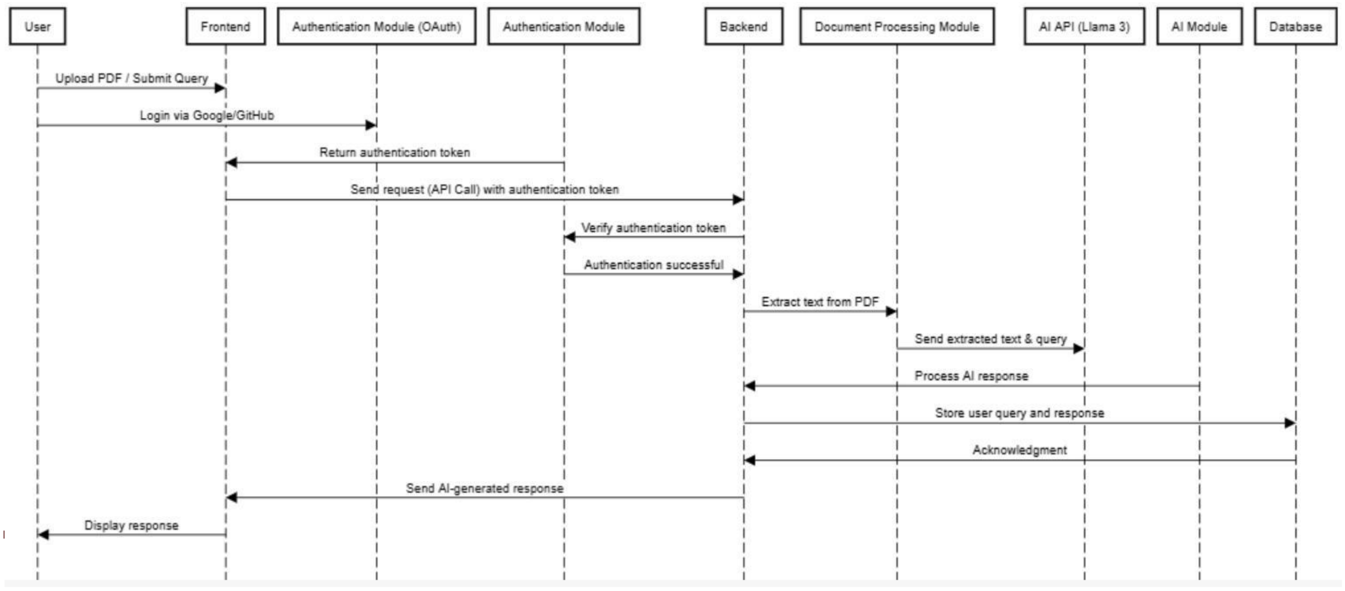
The **Chat with PDF** system is divided into several key modules, each with distinct responsibilities and interactions:

- **User Interface (Frontend)**
  - Developed using **HTML, CSS, JavaScript**.
    - Provides an intuitive interface for users to upload PDFs and input queries.
  - Sends user inputs to the backend via API calls.
- **Backend Server (Flask Web Application)**
  - Manages HTTP requests and routes data between components.
  - Handles authentication using OAuth (Google/GitHub).
  - Processes requests to extract text from PDFs and generate AI responses.
- **Document Processing Module**
  - Utilizes **PyPDF2/pdfplumber** for text extraction.
  - Formats and cleans extracted text for AI processing.
- **AI Processing Module**
  - Integrates with **Groq's Llama 3 API** for generating responses.
  - Sends extracted PDF content and user queries to the AI model.
- **Database Module**
  - Stores user query history and AI-generated responses.
  - Uses a relational database (PostgreSQL) for efficient data retrieval.
- **Authentication Module**
  - Implements OAuth-based authentication via **Google/GitHub**.
  - Ensures secure access to the chatbot.

3.2 Data Flow & Components

The system follows a structured data flow:

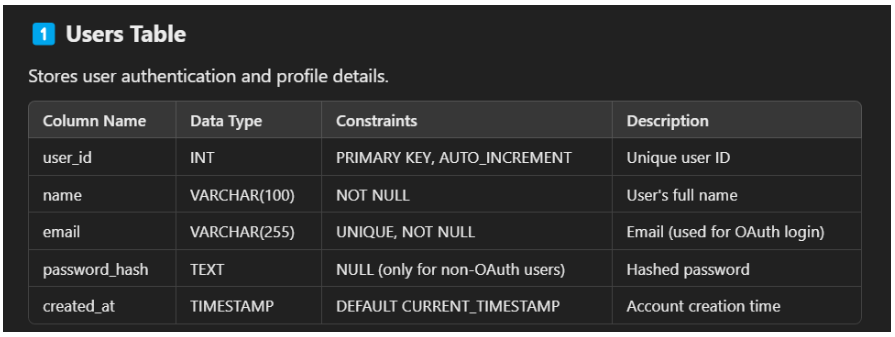
Sequence Diagram (Simplified)


User
Frontend
Authentication Module (OAuth)
Authentication Module
Backend
Document Processing Module
AI API (Llama 3)
AI Module
Database
Upload PDF / Submit Query
Login via Google/GitHub
Return authentication token
Send request (API Call) with authentication token
Verify authentication token
Authentication successful
Extract text from PDF
Send extracted text & query
Process AI response
Store user query and response
Acknowledgment
Send AI-generated response
Display response

3.3 Database Design

We will use a **relational database (PostgreSQL/MySQL)** to store previous chats and manage authentication using **Google OAuth and GitHub OAuth**. The key tables include:

1. **Users** → Stores user authentication details.
2. **Chats** → Stores chat history for each user.
3. **Messages** → Stores individual messages in each chat session.
4. **OAuth_Providers** → Tracks linked OAuth accounts (Google, GitHub).

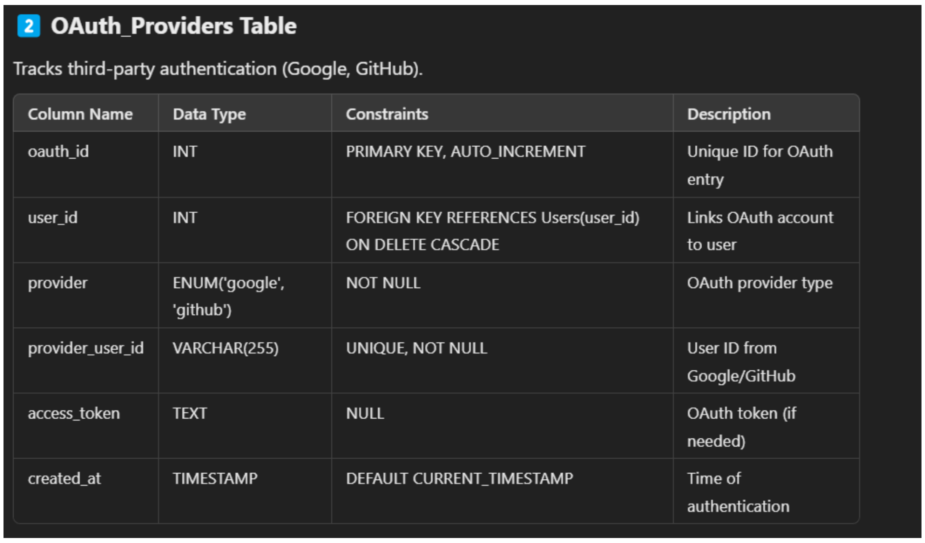


1 Users Table

Stores user authentication and profile details.

Column Name	Data Type	Constraints	Description
user_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique user ID
name	VARCHAR(100)	NOT NULL	User's full name
email	VARCHAR(255)	UNIQUE, NOT NULL	Email (used for OAuth login)
password_hash	TEXT	NULL (only for non-OAuth users)	Hashed password
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Account creation time

Indexing: UNIQUE(email) to ensure no duplicate accounts.

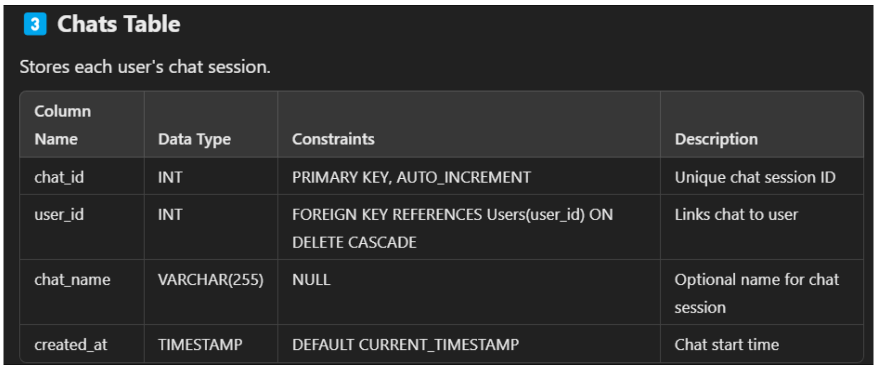


2 OAuth_Providers Table

Tracks third-party authentication (Google, GitHub).

Column Name	Data Type	Constraints	Description
oauth_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for OAuth entry
user_id	INT	FOREIGN KEY REFERENCES Users(user_id) ON DELETE CASCADE	Links OAuth account to user
provider	ENUM('google', 'github')	NOT NULL	OAuth provider type
provider_user_id	VARCHAR(255)	UNIQUE, NOT NULL	User ID from Google/GitHub
access_token	TEXT	NULL	OAuth token (if needed)
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Time of authentication

Indexing: INDEX(provider, provider_user_id) for quick lookup.

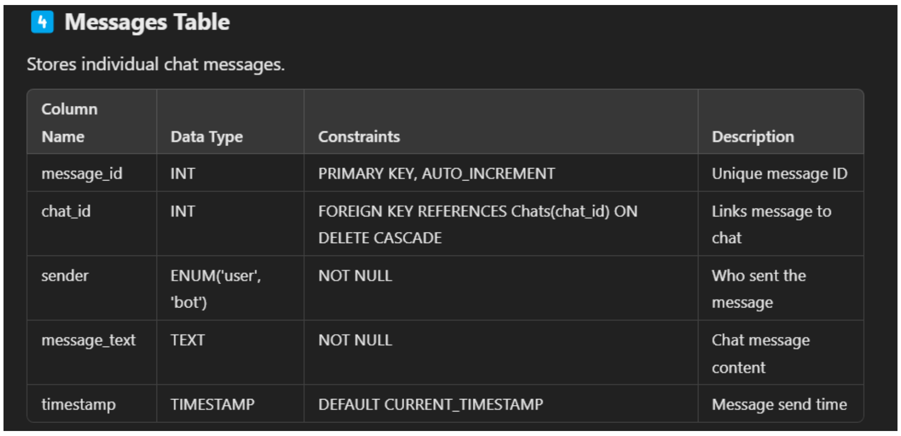


3 Chats Table

Stores each user's chat session.

Column Name	Data Type	Constraints	Description
chat_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique chat session ID
user_id	INT	FOREIGN KEY REFERENCES Users(user_id) ON DELETE CASCADE	Links chat to user
chat_name	VARCHAR(255)	NULL	Optional name for chat session
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Chat start time

Indexing: INDEX(user_id) for fast retrieval of user's chat history.

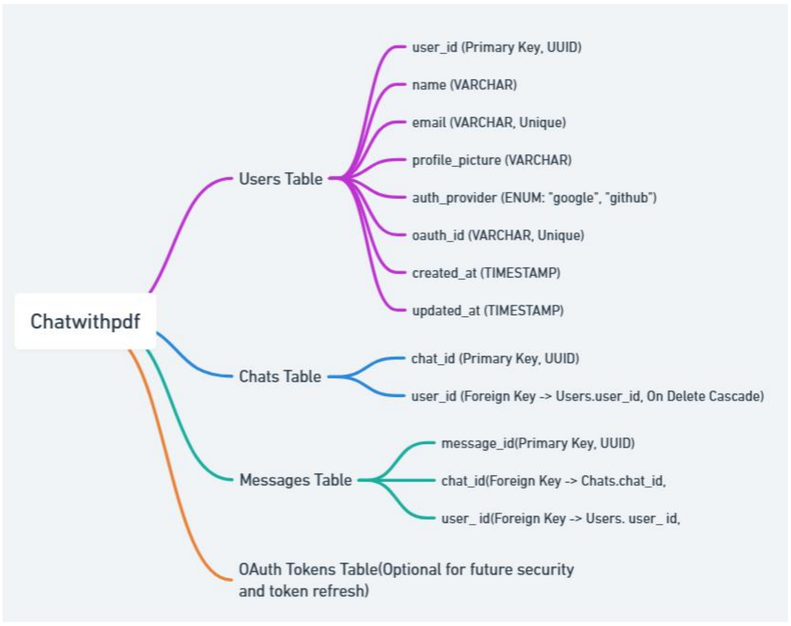


Messages Table

Stores individual chat messages.

Column Name	Data Type	Constraints	Description
message_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique message ID
chat_id	INT	FOREIGN KEY REFERENCES Chats(chat_id) ON DELETE CASCADE	Links message to chat
sender	ENUM('user', 'bot')	NOT NULL	Who sent the message
message_text	TEXT	NOT NULL	Chat message content
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Message send time

Indexing: INDEX(chat_id, timestamp) for retrieving messages in order.


Chatwithpdf
Users Table
user_id (Primary Key, UUID)
name (VARCHAR)
email (VARCHAR, Unique)
profile_picture (VARCHAR)
auth_provider (ENUM: "google", "github")
oauth_id (VARCHAR, Unique)
created_at (TIMESTAMP)
updated_at (TIMESTAMP)
Chats Table
chat_id (Primary Key, UUID)
user_id (Foreign Key -> Users.user_id, On Delete Cascade)
Messages Table
message_id(Primary Key, UUID)
chat_id(Foreign Key -> Chats.chat_id,
user_ id(Foreign Key -> Users. user_ id,
OAuth Tokens Table(Optional for future security and token refresh)

Wireframe Overview

- **Home Page**
  - Upload PDF button
  - Login/Signup (OAuth-based)
- **Chat Interface**
  - Text input box for user queries
  - Chat history sidebar for previous interactions

- **Results Display**
  - AI-generated responses formatted for readability

3.4 External Interfaces

The system integrates the following external APIs and services:

Component	Description
Groq's Llama 3 API	AI-powered text processing
PyPDF2 / pdfplumber	Extracts text from PDFs
OAuth (Google/GitHub)	Secure user authentication
PostgreSQL	Stores user queries & responses
Render	Cloud hosting for deployment

---

4. Project and Implementation Plan

4.1 Deliverables

The following deliverables will be provided as part of the **Chat with PDF** project:

- **Source Code**
  - Fully functional **frontend (HTML, CSS, JavaScript)**.
  - **Flask backend** code for API handling and document processing.
  - **AI integration module** for Groq's Llama 3 API.
- **User Documentation**
  - Instructions on how to use the chatbot.
  - Explanation of features and limitations.
- **Installation & Deployment Guide**
  - Steps for setting up the project on a local or cloud server.
  - Dependencies, configurations, and hosting instructions.
- **API Documentation** (If applicable)
  - Endpoints and their functionalities.

  - Sample requests and responses for integration.
- **Database Schema & Design Documents**
  - ER diagrams and table relationships.
  - Query optimization strategies.
- **Testing & Evaluation Reports**
  - Unit testing and integration testing results.
  - Performance benchmarks and expected response times.

4.2 Team Roles and Responsibilities and delivery schedule

Name of the Task	Developer	Tester	Approver	Date of Delivery
Frontend	Ronak	Vedant		
Backend	Hyder	Vedant		
Google OAuth, Github Auth	Vedant	Ronak		
Database	Vedant	Hyder		

4.3 Risk Management Plan

To mitigate potential risks, the following strategies will be implemented:

- **System Security Risks**: Implement OAuth authentication and encryption protocols to prevent unauthorized access.
- **API Downtime**: Use a fallback mechanism to handle API failures and minimize service disruptions.
- **Data Loss**: Implement regular backups of the database to ensure recovery in case of failure.
- **Scalability Issues**: Optimize the backend for concurrent users and increase server capacity as needed.
- **Performance Bottlenecks**: Perform load testing to identify and fix inefficiencies before deployment.

5. **Testing & Deployment Plan**

5.1 Testing Strategy

A comprehensive testing plan will be followed:

1. **Unit Testing**: Verify individual components such as document processing and AI integration.
2. **Integration Testing**: Ensure smooth interaction between frontend, backend, and AI modules.
3. **System Testing**: Validate full system functionality, including API responses and database operations.
4. **User Acceptance Testing (UAT)**: Conduct usability tests with real users to confirm that the system meets requirements.

.

Deployment will follow a structured process:

- **Deployment Environment**: The system will be deployed on **Render** for cloud-based access.
- **Installation Setup**:
  - Install required dependencies (Flask, PyPDF2, pdfplumber, PostgreSQL, etc.).
  - Configure OAuth authentication and API keys.
- **Rollback Strategy**: Maintain version control using GitHub to revert to a previous stable version in case of failures.

The next chapter, chapter no . 4 will explain test cases, test plan and test reports in detail