NAME: Kasodariya Deep

REG NO: 221080029

EXPERIMENT 5

AIM: To make Software Requirements Specification Document.

THEORY:

Introduction to Software Requirements Specification (SRS)

- An SRS is a detailed, structured document that describes the software system's intended functionality, constraints, and interactions with external systems.
- It serves as a foundation for the entire development process, detailing the functional and non-functional requirements needed to create a successful product.
- The SRS is essential in aligning the vision between stakeholders, including clients, developers, testers, and project managers, by clearly specifying the scope and objectives of the software.
- Through the SRS, ambiguity is minimized, making it easier to understand the expected functionalities and constraints and providing a roadmap for the software lifecycle.

Importance and Benefits of an SRS

- The SRS document is crucial for project success because it forms the basis for system design, development, and testing. It provides clarity on what the software must do, which helps prevent scope creep and misunderstandings.
- For clients, an SRS gives assurance that their needs and requirements are understood and addressed.
- For developers, it provides a structured, step-by-step guide to follow, minimizing confusion and errors during implementation.
- Additionally, for testers, it serves as a benchmark against which the final product can be validated, ensuring that the software meets all specified requirements before release.

NAME: Kasodariya Deep

REG NO: 221080029

Software Requirements Specification:

Introduction: This document outlines the ‘Finance Management System’. The system aims to digitalize transaction management for users by providing web interface to add different transactions having various attributes like account and categories. It also automates the summary process by calculating detailed analysis of the transactions.

Index:

Revision History:

Name	**Date**	**Reason For Change**	**Version**
Kasodariya Deep	12-11-2024	Baseline Version	v1.0.0

1) **Functional Specifications**

    1.1) **User Specifications:**

NAME: Kasodariya Deep
REG NO: 221080029

1.1.1) **Create Transaction**:

- Users can create a new transaction by filling in required fields such as amount, date, payee, account, category, and description.
- Validation is applied to ensure all mandatory fields are completed correctly.

1.1.2) **Edit Transaction**:

- Users can edit an existing transaction to update its details, including amount, payee, account, category, and date.
- Any modification triggers a re-fetch of transaction data to display the updated record in the transactions table.

1.1.3) **Delete Transaction**:

- Users can delete unwanted transactions, which will be removed from the database and reflected in real-time on the transactions table.

1.1.4) **Add Account**:

- Users can create new accounts to categorize transactions (e.g., "Savings Account", "Credit Card", etc.).
- Each account will be saved in the database and available for selection in transaction creation and editing.

1.1.5) **Add Category**:

- Users can create new categories for organizing transactions (e.g., "Groceries", "Utilities", "Entertainment", etc.).
- Categories will be stored in the database and accessible in transaction forms.

NAME: Kasodariya Deep

REG NO: 221080029

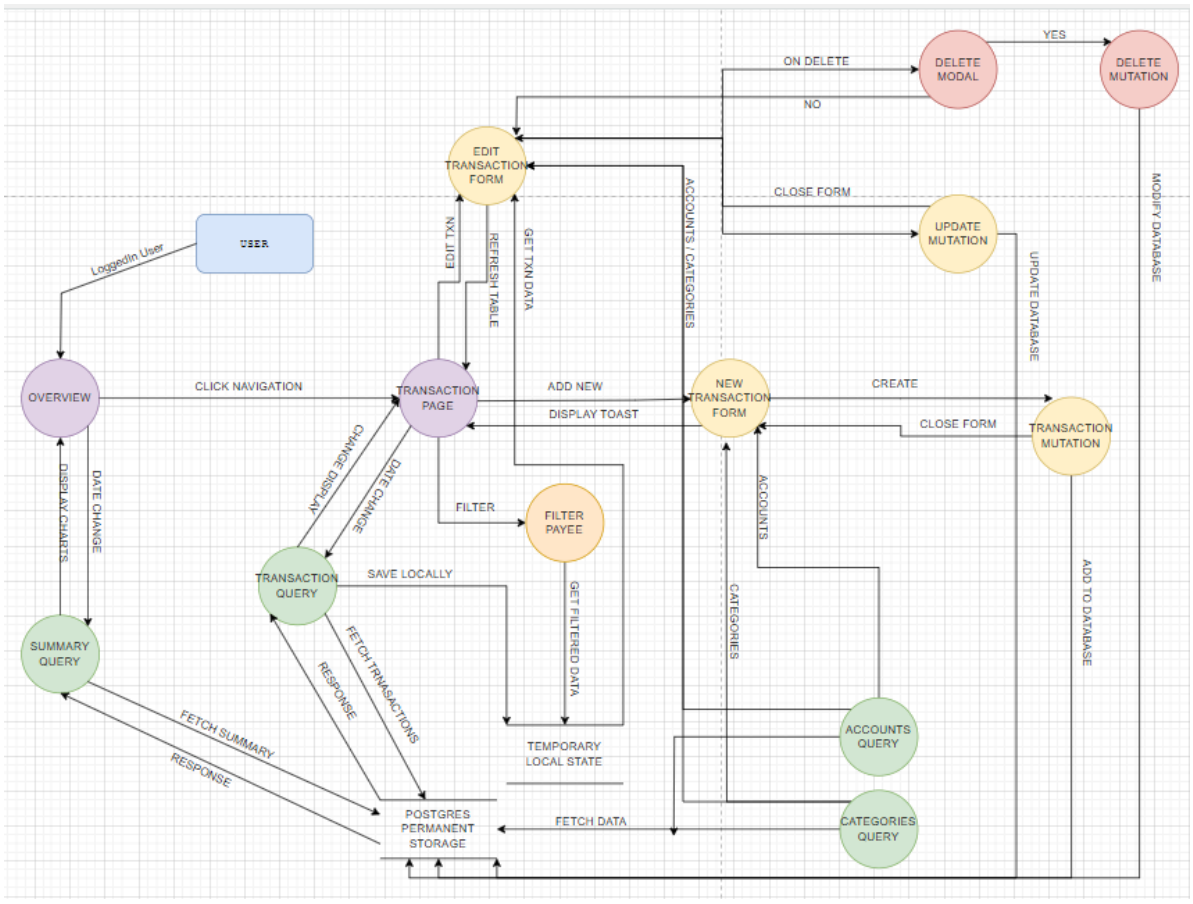
1.1.6) **Filter Transactions by Payee**:

- Users can filter transactions by entering a payee's name, narrowing down the displayed transactions in the transactions table.
- Filtered results should refresh instantly and reflect only transactions associated with the specified payee.

1.1.7) **Summary by Account**:

- Users can view a summary of transactions grouped by account, providing a quick overview of balances or total spending per account.
- Summaries should include total balances and a breakdown by categories if applicable.

1.2) **Data Flow Diagram:**



NAME: Kasodariya Deep

REG NO: 221080029

When a logged-in user visits the home page, a transaction summary is fetched from the database and displayed. Upon navigating to the transactions page, the user sees their transactions in a table, with data stored temporarily in local state for filtering. The user can create a new transaction or edit an existing one using the respective buttons. Before displaying the forms, account and category data are fetched for selection options, and the edit form retrieves the current transaction data from local storage. Any modifications update the database using mutations and display toast notifications for feedback. Forms close automatically after submission, triggering real-time data re-fetching to update the displayed information.

1.3) **Sequence Diagram:**

When the user visits the home page, a transaction summary is fetched from the database and displayed. Navigating to the transactions page triggers a fetch of all transactions, which are displayed in a table and stored in local state for filtering. If the user chooses to create or edit a transaction, account and category data are fetched for form selection options, and any existing transaction data is retrieved from local storage for editing. Upon form submission, the database is updated through mutations, and success or error toast notifications inform the user of the result. The form closes automatically, and the transactions page re-fetches the updated data to refresh the table in real time.

2) Non Functional Requirements:

2.1) **Performance Constraints**

- The transaction management application should be optimized to run efficiently on a standard system with a minimum of a dual-core processor, 4GB of RAM, and 5GB of available storage. It should maintain an average response time of

under 300 milliseconds for core operations such as fetching transaction data, creating new transactions, and updating existing ones. The system should be capable of supporting up to 500 concurrent users, ensuring smooth access and interaction without noticeable delays.

2.2) **Security Constraints**

- The database and user data should be protected against common security threats, including SQL injection, cross-site scripting, and other malicious attacks. Security protocols such as HTTPS for data in transit and encrypted storage for sensitive data should be implemented.

2.3) **Reliability**

- The application should maintain high reliability, with minimal downtime and robust error-handling mechanisms. It should efficiently handle data processing tasks, including transaction creation, updates, and real-time synchronization, ensuring data integrity and consistency across user sessions.

3) **Use Cases:**

1.1) **Creating Transaction**

- **Actor**: User
- **Description**: The user can create a new transaction by providing details such as amount, date, payee, account, category, and a description. The system validates the input and stores the transaction in the database. Once the transaction is successfully created, the user receives a confirmation notification.
- **Preconditions**: The user is logged in, and account and category data are available.

NAME: Kasodariya Deep
REG NO: 221080029

- **Postconditions**: A new transaction is stored in the database and displayed in the transactions list. A success notification is shown to the user.
- **Steps**:
    1. The user navigates to the transaction creation form.
    2. The user fills in the required transaction details.
    3. The system validates the input.
    4. The transaction is saved to the database.
    5. A success message is displayed.

1.2) **Searching for Transactions**

- **Actor**: User
- **Description**: The user can search for transactions by specific criteria payee. The system filters and displays matching transactions in a table format.
- **Preconditions**: The user is logged in, and transaction data exists in the local database.
- **Postconditions**: A filtered list of transactions is displayed, matching the search criteria.
- **Steps**:
    1. The user navigates to the transactions page.
    2. The user enters search criteria.
    3. The system filters transactions based on the input.
    4. The filtered transactions are displayed to the user.

1.3) **Summary by Accounts**

- **Actor**: User
- **Description**: The user can view a summary of transactions grouped by account. The system aggregates and displays transaction totals for each account, providing an overview of the account balances or total spending.
- **Preconditions**: The user is logged in, and transaction data is available for the relevant accounts.

NAME: Kasodariya Deep

REG NO: 221080029

- **Postconditions**: A summary of transactions by account is displayed to the user.
- **Steps**:
    1. The user navigates to the transaction summary section.
    2. The user selects the "Summary by Account" option.
    3. The system groups transactions by account and calculates totals.
    4. The summary of transactions by account is displayed to the user.

Observation: The software requirements specification Document (SRS) proves to be a very critical part of Software Development. It serves as a backbone and a guide during each phase as it makes it clear what is to be made.