PROJECT DOCUMENTATION

(Submitted as part of the Business Analyst Assignment for------)

TITLE OF THE PROJECT

ONLINE INTERNET BANKING SOLUTION

(A Professional Business Analysis & System Design Report)

Submitted By

Name: Shivam Kumar Mahto

Role Applied For: Business Analyst

Organization:

Email ID: shivammahto2105@gmail.com

Contact No.: 9013289135

Supervised / Assigned By

Evaluator / Department: HR & Project Evaluation Team

Company:

Location:

Title Page

Project Title: Online Internet Banking Solution

Prepared By: Shivam Kumar Mahto — Business Analyst

Organisation:

Version: 1.0

Date:

Document Type: Business Requirements Document (BRD)

Confidentiality Level: INTERNAL / FOR REVIEW

Document Purpose:

This document outlines the **business requirements** for the Online Internet Banking Solution. It serves as a formal agreement between **business stakeholders** and the **development team** to ensure a clear understanding of the project scope, objectives, and functionality.

Intended Audience:

- Product Owner / Client Bank
- Project Managers
- Development Team (Front-end, Back-end, Database)
- QA / Testing Team
- Security & Compliance Team
- Customer Support / Operations Team

Document Control / Version History:

This document is **version controlled** to track changes and updates. Any modifications after approval will follow the **change management process** to maintain document integrity.

Revision History

Version	Date	Author Name	Change Summary / Description
1.0	05/11/2025	Shivam Kumar Mahto	Initial draft of BRD created with complete project structure and business requirements.
1.1	05/11/2025	Shivam Kumar Mahto	Incorporated review feedback, refined project scope, and updated stakeholder details.
1.2	05/11/2025	Shivam Kumar Mahto	Final version approved with all sections validated and ready for submission.

Notes for Revision History:

- Each new **update or change** should be added as a **new version**.
- Always include **Date, Author**, and a **brief description** of the change.
- Helps **stakeholders track changes** and know what was added or updated.
- Recommended to maintain **versioning in the format** 1.x (1.0, 1.1, 1.2…) for clarity.

Part 1: Business Requirements Document (BRD)

Table of Contents

Introduction & Purpose

3.1 Introduction

The **Online Internet Banking Solution** is a web-based system that allows customers to access their bank accounts using the internet. Through this system, users will be able to check their **account balance**, **view recent transactions**, **transfer money** within the **same bank**, and **request services** like **cheque books** or **address changes**. The goal is to provide customers with a quick, safe, and easy way to manage their banking activities without visiting the bank branch.

3.2 Purpose

The purpose of this **BRD** is to explain what the system needs to do from a business point of view. It helps everyone involved in the project, such as **developers**, **designers**, **testers,** and **managers**, understand the requirements clearly before starting the development work. This document makes sure that the final system matches the business needs, is easy for customers to use, and follows security rules to protect **financial information.**

Project Scope

4.1 In-Scope (What the System WILL do):

1. The system will allow customers to **log in** using a username and password.
2. After logging in, customers can see **all their bank accounts** on one screen.
3. Customers can **check their account balance** anytime.
4. Customers can **view their transaction history**, like money they sent or received.
5. Customers can **transfer money** from one account to another account within the **same bank.**
6. Customers must enter a **transaction password** to confirm the money transfer for security.
7. Customers can submit **service requests online**, such as:
    - Requesting a **new cheque book**
    - Requesting **address change**
    - **Stop cheque payment** request
8. Customers can **download account statements** monthly or yearly.
9. The system will **lock the account** after 3 wrong login attempts.
10. The system will show **notifications** for successful or failed transactions.
11. All actions (login attempts, transfers, service requests) will be **saved in logs** for security.

4.2 Out-of-Scope (What the System WILL NOT do right now):

1. No money transfer to accounts in **other banks** (no NEFT / RTGS / IMPS).
2. **ATM or Debit Card** services like PIN change or card blocking are not included.
3. **Loan, investment, and KYC** processes are not included.
4. No **mobile app** in this phase — only **web application**.
5. No **international transfers** or currency exchange features.

Stakeholders & Roles

5.1 Who is Involved in the Project (Stakeholders):

5.1.1 Product Owner / Client

The **person** or **organization** who wants this banking system. They give the main **requirements** and final approvals.

5.1.2 Customers / End Users

These are the people who will use the online banking website to manage their accounts.

5.1.3 Business Analyst (Shivam Kumar Mahto)

Prepare documents, collect requirements, and make sure everyone understands the project clearly.

5.1.4 Development Team (Developers)

They will create the system (front-end, back-end, and database).

5.1.5 UI/UX Designers

They make sure the website looks good and is easy for customers to use.

5.1.6 QA / Testing Team

They check if the system works properly and does not have errors.

5.1.7 Customer Support / Service Team

They help customers if their account gets locked or if they have any issues.

5.1.8 Security & Compliance Team

They ensure the system follows security rules to protect financial data.

5.2 Roles (What Each Group Does):

- **Product Owner:** Gives requirements and approves final output.
- **Business Analyst:** Documents and explains requirements.
- **Developers:** Code and build the system.
- **UI/UX Designers:** Design simple and user-friendly screens.
- **QA/Testers:** Test system and fix issues before release.
- **Customer Support:** Helps users with problems.
- **Security Team:** Protects system from security threats.

Business Objectives & Success Criteria

6.1 Business Objectives (What the Project Wants to Achieve):

The main objective of the **Online Internet Banking System** is to provide customers with a **simple and convenient** way to manage their banking activities online. Instead of visiting the bank branch for small tasks, customers should be able to **check balances, view transactions, transfer money, and submit service requests** from their home or office. This improves **customer comfort** and reduces waiting time in the bank. At the same time, the bank can handle work more efficiently because **less manual work** and **fewer physical visits are required**.

6.2 Success Criteria (How We Know the Project is Successful):

The system will be considered successful if customers are able to **log in and use the banking features** easily without confusion. Most common tasks like **balance check, transaction view, fund transfer, and statement download** should work **smoothly and quickly, without delays**. The number of customers visiting the bank for small requests should **reduce**, showing that online services are being used effectively. The system must also be **secure**, **protect customer data**, prevent unauthorized access, and remain available and stable with **minimum downtime**, ensuring a **reliable and trustworthy user experience**.

High-Level Business Requirements

The **Online Internet Banking System** allows a customer to access their bank account securely through the internet. The **user logs in, views account details, transfers money**, and **makes service requests** — all without visiting the bank. The system ensures security, availability, and accurate record tracking for every action performed.

In simple words:

*"The system helps customers do **banking tasks** online easily and safely."*

7.1 Workflow :

7.1.1 User Login

- User enters **username and password**
- System verifies details
- If password **incorrect 3 times → Account Lock**
- If correct → **User goes to Dashboard**

7.1.2 Dashboard / Account View

- Shows **Account Balance**
- Shows **Last Transactions**
- Provides menu for **Transfers, Statements, and Service Requests**

7.1.3 Fund Transfer (Same Bank)

- User selects **Source Account**
- Enters **Recipient Account Number**
- Enters **Amount**
- Confirms using **Transaction PIN/Password**
- System verifies:
    - Balance Availability
    - System processes the transfer and shows **Success / Failure Message**

7.1.4 Service Requests

Customer can request online services such as:

- Cheque Book Issue
- Address Update
- Stop Payment Order

System generates a **Request Reference Number** for tracking.

7.1.5 Account Statement

- User selects **Date Range / Month / Year**
- System generates **PDF or On-Screen Statement**
- User can **Download / Print**

7.1.6 System Log & Security Control

- All activities (login, transfers, requests) are **stored in logs**
- Helps in monitoring, tracking errors, and preventing fraud

7.2 High-Level Process Flow Diagram (Text Form) :

Login → Dashboard → (View Account / Transfer Money / Request Services / View Statement) → Logout

Business Process Overview

The **Online Internet Banking System** works in a **simple** and **secure way**. The customer first logs in using their **username** and **password**. If the login details are correct, the customer enters their **account dashboard**. If the password is entered incorrectly multiple times, the account will be locked for safety, and the customer will need to contact support to unlock it.

8.1 Login Process

- Customer enters **username and password**
- System checks if the details are correct
- If the password is entered wrong **3 times**, the system **locks the account**
- If details are correct, customer moves to the **Account Dashboard**

8.2 Viewing Accounts and Balance

After logging in, the customer can:

- See all the **bank accounts** linked to their profile
- View **current balance**
- View **recent transactions** (money received/spent)

8.3 Fund Transfer (Within Same Bank)

To transfer money:

- Customer selects **source bank account**
- Enters **receiver's account number** (same bank)
- Enters **amount** to transfer
- Enters **transaction password** to confirm

System checks:

- Sufficient balance
- Correct transaction password
    - System shows **Success or Error Message**
    - Transaction is **recorded in logs**

8.4 Service Requests (Online Forms)

The customer can request:

- **Cheque Book**
- **Address Change**
- **Stop Cheque Payment**

When a request is submitted:

- System checks details
- Generates a **Reference Number**
- Customer Support Team processes the request

8.5 Statement Download

- Customer selects month or year
- System generates **PDF statement**
- Customer can **view or download it**

8.6 System Logging (Security and Record Keeping)

The system **records**:

- Login attempts
- Fund transfers
- Service requests
- Any account lock/unlock actions

This helps in **security, tracking, and auditing.**

Non-Functional Requirements (Quality Requirements)

Non-Functional Requirements describe how the system should work, not what it should do. These focus on quality, performance, security, and user experience of the system.

9.1 Performance

The system should respond **quickly** and handle many users at the same time.

- Page loading time should be **fast**
- Fund transfer and balance checking should take **minimum delay**
- System should support **multiple users at once** without slowdown

9.2 Security

Banking system must be **highly secure** to protect customer data.

- Password must be **encrypted**
- Incorrect password attempts should **lock the account**
- Transactions require **additional authentication** (Transaction PIN / OTP)
- All user actions should be **recorded in logs**

9.3 Availability

The system should be available to users **24/7**.

- Only short downtime allowed for **maintenance**
- Bank customers should be able to use services **anytime, anywhere**

9.4 Usability

System must be **easy to understand and use**, even for new users.

- Simple and clear **user interface**
- Instruction messages should be **easy to follow**
- Important buttons (Transfer / Logout / Help) must be **easily visible**

9.5 Reliability

System should work **without errors** and ensure accurate processing.

- Transactions should be **correct and consistent**

- No loss of data even if **network fails temporarily**
- Backup of system data should be taken **regularly**

9.6 Maintainability

System should be easy to **update and improve** in the future.

- Code should be **well structured**
- System should allow **module-wise updates**
- Technical errors should be **easy to identify and fix**

9.7 Privacy

The customer’s data must remain **confidential**.

- Personal details should not be visible to others
- Employees should have **limited access** based on their role

Assumptions and Dependencies

10.1 Assumptions

These are the conditions we **believe** to be true while designing the Online Internet Banking System:

1. **User has a stable internet connection** to access the online banking services.
2. The customer already has an **active bank account** registered with the bank.
3. The **user's device (mobile/PC)** supports the web browser used to access the system.
4. The **bank database is already maintained** with correct customer and transaction data.
5. The user will keep their **login details and transaction password private** and will not share them with others.
6. Bank staff is available to help customers in case of **account lock, technical issues, or service requests.**

10.2 Dependencies

These are the **external systems or conditions** the banking system depends on to work correctly:

1. **Internet connectivity** — Without the internet, the system cannot be accessed.
2. **Bank Database System** — All account and transaction records depend on the accuracy and availability of the database.
3. **Server / Hosting Environment** — The system must be hosted on a server that stays online 24/7.
4. **Authentication System (Security Protocols)** — Password encryption, OTP generation, and login validation systems must work properly.
5. **Browser Compatibility** — The system depends on users accessing it through a supported web browser (Chrome, Edge, Firefox, etc.).
6. **Backup and Recovery System** — Automatic backups are needed to protect data from loss or corruption.

Constraints

Constraints are the **limitations or conditions** that may affect how the Online Internet Banking System is designed and used.

1. **Only same-bank transfers allowed:**

   The system will allow **fund transfer only within the same bank**. Transfers to accounts in other banks (NEFT / RTGS / IMPS) are **not included** in this version.

2. **Web Application Only:**

   The system is designed only for **web-based access** through a browser. There is **no mobile app** in this phase.

3. **Limited System Access:**

   The system can only be used by customers who already have a **bank account and are registered** for online banking services.

4. **Security Rules Must Be Followed:**

   Passwords should be **strong and kept secret**. The system must follow bank security policies, which may limit how data is shown or stored.

5. **Server and Internet Required:**

   The system works only if the **server is running** and the user has a **stable internet connection.**

6. **Data Privacy Must Be Maintained:**

   Customer personal and financial data must be kept **confidential**. The system must follow **data privacy and protection guidelines.**

Risks and Mitigation

1 Risk: User Account Lockout

If a user enters the wrong password multiple times, their account may get locked. **Mitigation**: Display a clear warning after each wrong attempt and allow customer care to easily verify and unlock the account.

2 Risk: Unauthorized Access / Hacking Attempts

Someone may try to access the system using stolen or guessed passwords. **Mitigation**: Use strong password rules, encryption, and security logging to track and block suspicious activity.

3 Risk: Users Sharing Their Passwords

Some users may unknowingly share their credentials or save them in insecure places. **Mitigation**: Display security awareness messages, encourage password secrecy, and send alerts when passwords are changed.

4 Risk: Server Downtime / System Unavailability

The system may temporarily stop working due to server or network issues. **Mitigation**: Use a reliable hosting server, maintain backups, and monitor system performance continuously.

5 Risk: Delay in Transaction Records Updating

Sometimes transactions take time to appear in the account history or balance. **Mitigation**: Use real-time data updates where possible, and show a message like *"**Transaction is being processed**"* to avoid confusion.

6 Risk: Incorrect Information Entered in Service Requests

A user may enter the wrong address or cheque number while submitting a service request.

Mitigation: Add form validations and show a final confirmation screen before submitting.

Acceptance Criteria

The **Acceptance Criteria** define how we know that the Online Internet Banking System is **working correctly** and meets the business requirements. These are the conditions that must be satisfied before the system is considered **complete and ready for use.**

A. Users must be able to **log in successfully** with correct username and password.
B. The system should **lock the account** after 3 wrong login attempts and allow **customer support** to unlock it.
C. Users must be able to **view their account balance** and **last n transactions**.
D. Users must be able to **download monthly or yearly account statements**.
E. Fund transfers **within the same bank** must work correctly and require a **transaction password** for confirmation.
F. Service requests like **cheque book, address change, or stop cheque** should generate a **reference number** for tracking.
G. All **security checks** like password encryption, TLS/HTTPS, and logging should pass successfully.
H. The system must be **available, fast, and reliable** with minimal downtime.

Approval

This Business Requirements Document (BRD) for the **Online Internet Banking System** has been prepared by **Shivam Kumar Mahto**, Business Analyst at_____.

The purpose of this approval page is to confirm that all stakeholders have **reviewed, understood, and agreed** with the requirements, scope, objectives, and functionality described in this document. Approval ensures that the development team can start implementation with **clear and agreed-upon expectations**.\

By signing below, the stakeholders confirm that:

- They have **reviewed all sections** of the BRD.
- The **business requirements are accurate** and complete to the best of their knowledge.
- They **accept the scope, objectives, and constraints** mentioned in this document.
- Any **changes to the requirements** after approval will follow the **change management process**.

Prepared By:

Shivam Kumar Mahto — Business Analyst
Date: 06 November 2025

Approved By:

Product Owner: _____ Date: _____

Project Manager: _____ Date: _____

Security / Compliance Lead: _____ Date: _____

Customer Support Lead: _____ Date: _____

Additional Notes:

- Signatures can be **digital or handwritten.**
- This approval confirms that the BRD is **final and ready for development.**

Title Page

Project Title: Online Internet Banking Solution

Prepared By: Shivam Kumar Mahto — Business Analyst

Organisation:

Version: 1.0

Date: 06 November 2025

Document Type:

- **SRS** (Software Requirements Specification)
- **FRS** (Functional Requirements Specification)

Confidentiality Level: INTERNAL / FOR REVIEW

Document Purpose:

This document defines the **software/functional requirements** for the Online Internet Banking Solution. It provides a detailed description of the system's **functional and non-functional requirements**, business rules, and constraints. The purpose is to serve as a **reference for developers, testers, and stakeholders** to ensure that the system is built correctly and meets business needs.

Intended Audience:

- Product Owner / Client Bank
- Project Managers
- Development Team (Front-end, Back-end, Database)
- QA / Testing Team
- Security & Compliance Team
- Customer Support / Operations Team

Document Control / Version History:

This document is **version controlled** to track changes and updates. Any modifications after approval will follow the **change management process** to maintain document integrity.

Revision History

Version	Date	Author Name	Change Summary / Description
1.0	06/11/2025	Shivam Kumar Mahto	Initial draft of SRS/FRS prepared, including functional and non-functional requirements.
1.1	06/11/2025	Shivam Kumar Mahto	Added UML diagrams and system architecture details for better visualization.
1.2	06/11/2025	Shivam Kumar Mahto	Updated wireframes, DFD levels, and module descriptions.
1.3	06/11/2025	Shivam Kumar Mahto	Final reviewed version approved and formatted for submission.

Notes for Revision History:

- Each new **update or change** should be added as a **new version**.
- Always include **Date, Author**, and a **brief description** of the change.
- Helps **stakeholders track changes** and know what was added or updated.
- Recommended to maintain **versioning in the format** 1.x (1.0, 1.1, 1.2…) for clarity.

Part 2: Software Requirements Specification (SRS) & Functional Requirements Specification (FRS)

Table of Contents

Introduction

3.1 Purpose

The purpose of this document is to describe the complete functional and non-functional requirements for the ***Online Internet Banking Solution***.

It explains what the system will do, how it will work, and what **features** it will provide to **customers** and **administrators**.

This document will be useful for **developers, testers**, and **business analysts** during the **system's design, development**, and **testing phases**.

3.2 Scope

The **Online Internet Banking Solution** allows customers to manage their **bank accounts digitally.**

Users can view **account balances**, **transfer money, pay bills**, and **check transaction history** from anywhere using a **secure web platform**.

3.2.1 The system will include features such as:

- Account creation and login
- Balance inquiry
- Fund transfer
- Mini statements and transaction history
- Bill payments and service requests

The main goal is to provide customers with a fast, secure, and user-friendly experience.

3.3 Overview

This **SRS** & **FRS** document provides details about the system design, functional modules, and user interface.

It also includes diagrams such as **UML**, **Data Flow Diagrams**, and **Wireframes** to help understand how different parts of the system work together.

3.4 Intended Audience

3.4.1 This document is prepared for:

- **Developers –** to design and build the system
- **Testers –** to create and execute test cases
- **Project Managers –** to track progress and ensure requirements are met
- **Stakeholders/Clients –** to verify that the final product meets business goals

3.5 Definitions, Acronyms, and Abbreviations

Terms	Description
SRS	Software Requirements Specifications
FRS	Functional Requirements Specification
UI	User Interface
DB	Database
UML	Unified Modelling Language
CRUD	Create , Read, Update , Delete

System Overview

4.1 System Description

The **Online Internet Banking Solution** is a secure web-based application that allows bank customers to perform basic banking operations from anywhere, anytime.

It is built using **modern web technologies** to ensure data protection, quick response, and an easy user experience.

The system ensures that only **authorized users** with **valid credentials** can access their accounts.

4.1.1 The solution offers the following key services:

- Balance inquiry
- Funds transfer between accounts within the same bank
- Service requests (cheque book, address change, stop cheque payment)
- Monthly and annual statements

4.2 Technology Used

Layer	Technology
Frontend	HTML , CSS , JS
Backend	Node.Js
Database	MongoDB
Server	Flask
Development Tools	Vs Code, MongoDB
Hosting	AWS/ Local Host

4.3 System Users

The main users of this application are:

User Type	Description
Customer	Performs online banking activities like viewing balance, transferring funds, and requesting services.
Admin/Bank Staff	Manages user accounts, monitors transactions, and handles requests such as account unlocking or verification.

4.4 System Environment

The application runs in a **web browser** and can be accessed through desktop or mobile devices connected to the internet.

It supports secure **HTTPS communication** and works best with updated browsers like Chrome, Edge, or Firefox.

4.5 Overall System Flow

1 The user opens the banking website.

2 The user logs in using valid credentials (username & password).

3 The system verifies credentials against the database.

4 Upon successful login, the user dashboard appears showing all linked accounts.

5 The user can:

- View **balance** and **transaction history**
- **Transfer funds** between accounts
- Make **service requests**
- View **statements**

6 The system validates every transaction with a transaction password for security.

7 After use, the user logs out, and the session ends securely.

4.6 System Features

The system offers the following major features:

Feature	Description
• User Registration & Authentication	Customers can register and log in securely using encrypted credentials.
• Dashboard Overview	Displays balance summary, recent transactions, and quick links to actions.
• Fund Transfer	Transfer money within accounts or to other registered beneficiaries.
• Transaction History	View, search, and download detailed transaction records.
• Profile Management	Update contact details, email, and password.
• Service Requests	Raise requests for cheque books, address change, etc.
• Notifications	Send transaction alerts and system updates to users.

4.7 Assumptions and Dependencies

1. Users must have a **valid bank account** and **registered** email/mobile number.
2. Internet connectivity is **stable** for performing transactions.
3. The core banking system **APIs** are always available.
4. Browser must support **JavaScript** and **cookies**.
5. Third-party payment gateway availability is **consistent**.

4.8 Design and Implementation Constraints

1. System must comply with **banking data privacy laws** and **RBI cybersecurity guidelines.**
2. Sensitive data (passwords, account numbers) must **be encrypted using AES-256**.
3. All user sessions should **auto-expire after 10 minutes of inactivity**.
4. The application should be **responsive and mobile-friendly.**
5. The backend must support **scalability for concurrent user sessions**.
6. The system should handle **up to 10,000 simultaneous requests** without downtime.

Functional Requirements

5.1 Overview

The functional requirements describe what the system **must do** to meet business objectives.

Each function represents a feature that the user can perform through the online banking portal.

These requirements ensure that the customer can carry out all necessary banking operations securely and conveniently.

5.2 Functional Modules

Module Name	Description
• **User Authentication**	Allows customers to log in using a **valid username** and **password**. Three failed attempts will **lock** the account.
• **Account Overview**	Displays a list of all **customer accounts** and their **current balance**.
• **Transaction History**	Provides details of the **last *n* transactions** performed by the user.
• **Funds Transfer**	Enables users to **transfer funds** between accounts within the **same bank** using a **transaction password**.
• **Service Requests**	Allows customers to **request cheque books, change address**, or stop cheque payments.
• **Statements**	Provides **downloadable** monthly and annual account statements.
• **Notifications & Alerts**	Displays **success** or **failure messages** for transactions and sends alerts for important activities.

5.3 Detailed Functional Requirements

5.3.1 Login and Authentication

- User must enter a **valid username** and **password** to access the system. The system must **verify credentials** from the **database** before login.
- If an **incorrect password** is entered three times consecutively, the account is locked and cannot be accessed.
- System should allow users to contact customer care for unlocking their account through a secure process.
- **Login** and **transaction passwords** are separate for additional security and to protect against fraudulent activities.

5.3.2 Account Dashboard

- After login, the system displays all accounts linked to the user for easy access and overview.
- On selecting an account, the balance and account number will be shown clearly on the screen.
- System must **fetch data** in real-time from the database to ensure information accuracy.
- The dashboard should also show quick links to transfer funds or request services.

5.3.3 Funds Transfer

- Users can **transfer money between accounts** within the same bank through a secure transaction interface.
- A **transaction password** is required to authorize the transfer and confirm user identity.
- System must validate recipient account details before processing to prevent errors.
- **Successful transactions** generate a **confirmation message;** failed transactions show **error messages** with reasons for failure.
- All transaction details must be logged for audit and reporting purposes.

5.3.4 Service Requests

- **Customers can raise requests for:**
    - Cheque book issuance
    - Address change

    - Stop payment of cheques
- Each request must be saved in the database and visible to the admin panel for approval or rejection.
- Once processed, the system should notify the customer of the request status via message or alert.

5.3.5 Statements

- Users can **generate** and **download** monthly or annual statements in PDF format from their account page.
- The statement includes transaction ID, date, description, debit/credit amount, and balance for each record.
- The system should allow viewing statements for multiple accounts owned by the same customer.
- Users can choose custom date ranges to generate specific reports if needed.

5.3.6 Security & Session Management

- All user data and transactions are encrypted using secure protocols (e.g., HTTPS) for safety.
- The system should automatically log out after a fixed period of inactivity to prevent misuse.
- Passwords must not be stored in plain text within the database, but in encrypted format.
- System should prevent simultaneous login from multiple devices for one account session.

Non-Functional Requirements

6.1 Overview

The **Non-Functional Requirements (NFRs)** define the overall quality, performance, and operational constraints of the **Online Internet Banking Solution**.

They describe how the system should behave rather than what it should do. These requirements ensure the solution is **secure**, **reliable**, and **efficient** for all users.

6.2 Performance Requirements

- The system should respond to user actions within **3 seconds** under normal network conditions.
- The web application must support at least **1000 concurrent users** without performance degradation.
- **Database queries** should be optimized to handle multiple transactions simultaneously.
- The system should minimize **downtime** and maintain smooth performance even during peak hours

6.3 Security Requirements

- All user data must be protected using **SSL encryption (HTTPS)** for secure communication.
- The system must enforce strong password policies including **minimum 8 characters**, uppercase, lowercase, and digits.
- Users' passwords and sensitive information should be **hashed and salted** before storage in the database.
- The application must have protection against **SQL Injection, XSS, and CSRF** attacks.
- Failed login attempts beyond three times must **lock the account** automatically to prevent unauthorized access.

6.4 Reliability Requirements

- The system must ensure **99.5% uptime** excluding scheduled maintenance.
- In case of a system crash, the **backup database** should restore within **30 minutes**.

- Each transaction should be **atomic**, meaning it should either complete fully or not at all to avoid inconsistency.
- **Error handling mechanisms** must detect and report system faults immediately.

6.5 Usability Requirements

- The interface should be **simple, clean, and user-friendly**, suitable for users with basic computer knowledge.
- All buttons and menus must have **clear labels** to make navigation easier.
- The design should be **responsive**, adjusting to both desktop and mobile screens.
- **User feedback messages** should be displayed for each action like login, transfer, or logout.

6.6 Availability Requirements

- The banking application should be available **24x7 f**or customers.
- Planned maintenance activities must be announced in advance via **alerts or notifications**.
- In case of server downtime, a **backup server** should take over automatically.
- Availability must comply with banking standards and **service-level agreements (SLA**).

6.7 Scalability Requirements

- The system should support future **expansion of user base** without affecting performance.
- It should allow easy **integration of new modules** like credit card management or bill payment.
- The architecture must support **horizontal and vertical scaling** as per traffic demand.
- The database should handle large volumes of transaction records efficiently.

6.8 Maintainability Requirements

- The codebase should follow **modular structure** for easy maintenance and debugging.
- **Documentation** must be provided for all major functions and components.
- The system should support **version control** to track changes and updates.

- Maintenance activities must not disrupt normal user operations.

6.9 Portability Requirements

- The system should run on all major browsers including **Chrome, Edge, and Firefox**.
- It must be compatible with multiple operating systems like **Windows, macOS, and Linux**
- The application should be deployable on different servers without configuration conflicts.
- Data should be easily **migratable** if the bank switches to a new environment.

Use Case Diagrams

7.1 Purpose

The **Use Case Diagram** visually represents how **users (actors)** interact with the system.

It shows all **major functionalities** offered to customers and administrators, and helps in understanding the system's behavior at a high level.

Each use case defines a **specific goal** that an actor wants to achieve through the system.

7.2 Actors

Actor Name	Description
• **Customer**	The end user who accesses online banking features such as login, viewing balance, and transferring funds.
• **Admin/Bank Staff**	Handles customer service requests, monitors transactions, and manages account status.
• **System Database**	Stores and retrieves data for login, transactions, requests, and reports.

7.3 Use Cases

Use Case ID	Use Case Name	Description
UC-01	**User Login**	Customer logs in using username and password; system verifies credentials.
UC-02	**View Account Balance**	Displays list of accounts and current balance for the logged-in user.
UC-03	**Transfer Funds**	Allows customer to transfer money between accounts using a transaction password.
UC-04	**View Transaction History**	Customer checks recent transactions for a selected account.

Use Case ID	Use Case Name	Description
UC-05	**Request Cheque Book/Change Address/ Stop Cheque**	Customer sends a service request to the bank for processing.
UC-06	**View Statements**	Customer views or downloads monthly and annual statements.
UC-07	**Manage Customer Requests (Admin)**	Admin verifies, approves, or rejects customer service requests.
UC-08	**Unlock Account (Admin)**	Admin unlocks user accounts locked due to incorrect password attempts.

7.4 Relationships

Include:

Transfer Funds → includes *User Login* and *Transaction Password Verification*

Extend:

Request Service → may extend to *Notify Customer Status*

Association:

Customer interacts directly with ***User Login***, ***Balance Inquiry***, ***Fund Transfer***, ***Service Requests***, and ***Statements***.

7.5 Use Case Diagram

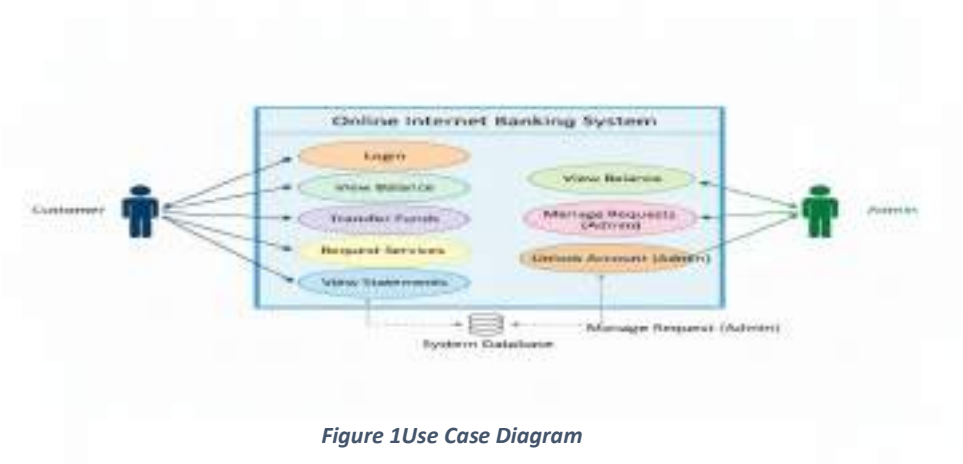

Figure 1Use Case Diagram

UML Diagrams

8.1 Purpose

The **UML (Unified Modelling Language)** diagrams represent the **structure** and **behaviour** of the Online Internet Banking System.

They help visualize how different parts of the system interact with each other — including data flow, classes, and operations.

This section includes **Class Diagram**, **Sequence Diagram**, and **Activity Diagram** to describe the system clearly.

8.2 Class Diagram

Description:

The **Class Diagram** shows the **static structure** of the system by representing classes, their attributes, and the relationships between them.

It helps developers understand how the system's data and functions are organized.

Main Classes and Attributes:

Class Name	Attributes	Methods / Functions
User	user_id, name, email, username, password, address, contact_no	login(), logout(), updateProfile()
Account	account_no, account_type, balance, user_id	viewBalance(), getTransactionHistory()
Transaction	transaction_id, date, amount, type, status, from_account, to_account	transferFunds(), getTransactionDetails()
ServiceRequest	request_id, user_id, request_type, request_date, status	createRequest(), updateStatus()
Admin	admin_id, name, email, password	approveRequest(), unlockAccount()

Relationships:

- **User → Account**: One-to-Many (One user can have multiple accounts)
- **Account → Transaction**: One-to-Many (One account can have many transactions)
- **User → ServiceRequest**: One-to-Many (A user can make multiple requests)
- **Admin → ServiceRequest:** One-to-Many (Admin manages multiple requests)

Figure 2 Class Diagram Of The System

8.3 Sequence Diagram

Description:

The **Sequence Diagram** explains how objects interact over time to complete a process, showing the **order of message flow** between the user and system.

Below is the main **Login and Funds Transfer** sequence.

Objects Involved:

- **User**
- **Login Page**
- **System / Server**
- **Database**

Flow:

1. **User** enters username and password.
2. **Login Page** sends data to **System** for verification.
3. **System** checks credentials from **Database**.

4. **Database** returns success or failure message.
5. On success, **System** loads **Dashboard** with user details.
6. For **Funds Transfer**, user enters receiver account and amount.
7. System verifies transaction password → updates balances → confirms transaction.

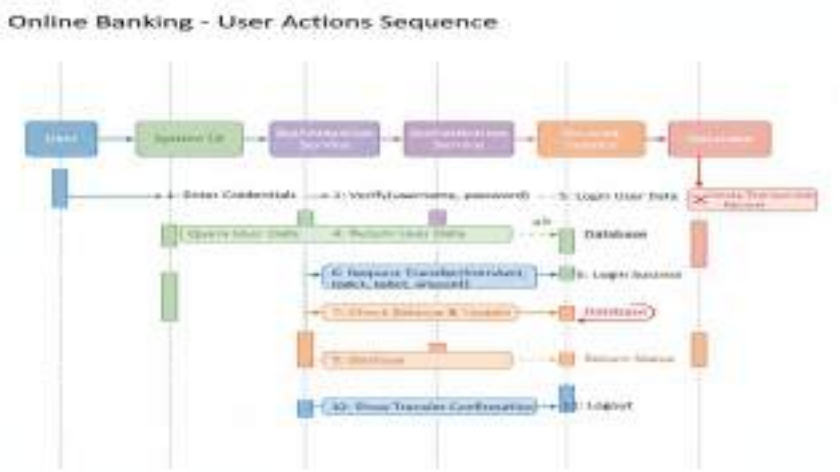

Figure 3 Sequence Diagram Transfer Funds

8.4 Activity Diagram

Description:

The **Activity Diagram** represents the **workflow** of the banking process — from login to logout.

It shows step-by-step actions and decisions performed by the user and system.

Flow of Activities:

1. **Start**
2. **User Login → [Valid Credentials] → Go to Dashboard**
   → [Invalid Credentials] → Show Error → Retry
3. **Select Operation**
   a. View Balance → Display Account Info
   b. Transfer Funds → Enter Details → Validate → Confirm → Show Result
   c. Request Service → Fill Form → Submit → Notify User
4. **Logout**
5. **End**

Decisions and Conditions:

- If login fails **3 times**, account gets **locked**.
- Funds transfer is only possible if **transaction password** is correct.
- Requests are stored until **Admin Approval**.

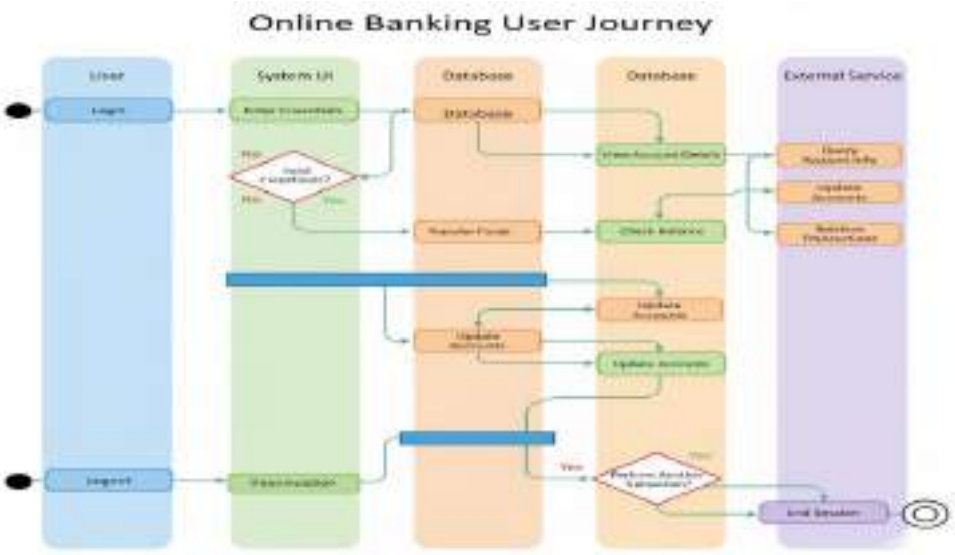

Figure 4 Activity Diagram for Online Banking Workflow

8.5 Diagram Summary

Diagram Type	Purpose	Represents
Use Case Diagram	Shows system interactions	Functional relationships
Class Diagram	Displays static structure	Classes and associations
Sequence Diagram	Represents process flow	Message order & logic
Activity Diagram	Shows dynamic behavior	Step-by-step workflow

8.5 Use Case Diagram

Figure 5 Use Case Diagram for System Functions

8.6 Component Diagram

Figure 6 Component Diagram of System Architecture

System Architecture Design

9.1 Purpose

The **System Architecture Design** explains how the application is structured — the components, their responsibilities, and how they communicate.

This helps developers and operations teams build, deploy, and scale the system correctly.

9.2 Architectural Style

Recommended style: Three-tier architecture (Presentation → Application → Data). This separates concerns, improves maintainability, and supports scalability

9.3 High-level Components

Component	Responsibility
Client / Presentation Layer	HTML, CSS, JavaScript — renders UI (Login, Dashboard, Transfer pages) in the browser.
Web Server (Reverse Proxy)	**Apache** (or Nginx) — serves static files, provides SSL termination, and forwards API requests to the backend.
Application Server / API Layer	**Node.js** — contains business logic, authentication, transaction handling, and exposes RESTful APIs.
Database Layer	**MongoDB** — stores users, accounts, transactions, service requests, and logs.
Authentication Service	Module for validating login and transaction passwords; can be part of Node.js or a separate microservice.
Notification Service	Sends email/SMS alerts (can integrate with external SMTP/SMS provider).
Admin Interface	Web pages or a separate portal for bank staff to manage requests and unlock accounts.
Logging & Monitoring	Centralized logs (e.g., ELK stack) and monitoring (Prometheus/Grafana) for health and audit.

Component	Responsibility
Backup & Recovery	Automated DB backups and restore procedures to meet RTO/RPO targets.

9.4 Deployment (Physical) Diagram

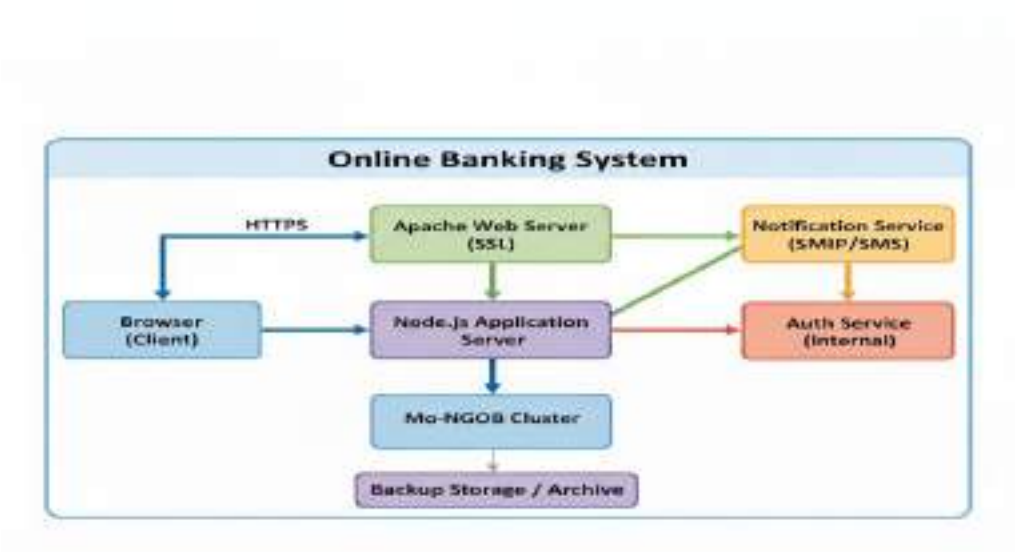

Figure 7 Deployment Diagram

- **Load Balancer** can be placed before Apache for high availability.
- **Failover DB replica** recommended for production.

9.5 Data Flow Summary

1. **User** sends request from browser → **Apache** (SSL terminates).
2. Apache forwards API request to **Node.js**.
3. **Node.js** validates session and transaction password (via Auth Service).
4. If DB access needed, Node.js queries **MongoDB**.
5. Results returned to **Node.js**, which builds response → Apache → **Browser**.
6. Notifications (email/SMS) sent asynchronously via **Notification Service**.

9.6 Security Considerations

1. Use **HTTPS/TLS** for all client-server communication.
2. Store **passwords** hashed (bcrypt) with salt; never store plain text.
3. Use **role-based access control (RBAC)** for Admin functions.
4. **Input validation & sanitization** on all endpoints to prevent XSS/NoSQL injection.

5. Implement **rate-limiting** and **account lockout** after failed attempts.
6. Encrypt sensitive data at rest if required (e.g., account numbers partially masked).

9.7 Scalability & Availability

1. **Horizontal scaling**: run multiple Node.js instances behind a load balancer.
2. **MongoDB**: use replica set for high availability and sharding when data grows.
3. Use **stateless API servers**; store sessions in Redis (optional) for session sharing.
4. Plan **automated CI/CD** pipelines for safe deployments.

9.9 Appendix: Simple Server Configuration Notes

1. **Apache**: acts as SSL terminator and reverse proxy to Node.js apps (ports 80/443 -> 3000).
2. **Node.js**: run with a process manager (PM2) and behind a load balancer.
3. **MongoDB**: enable authentication, backup, and monitoring.

Wireframe Designs

10.1 Purpose

The **Wireframe Designs** represent the **visual layout** of the main web pages before actual development begins.

They help designers and stakeholders understand how information, buttons, and features will appear on the screen.

Wireframes act as a **blueprint** for the final user interface.

10.2 Objective

The main goal of wireframes is to:

- Show **navigation flow** between pages.
- Provide a **clear structure** for each module (Login, Dashboard, Fund Transfer, etc.).
- Help developers understand the **UI layout** and **user interactions**.
- Ensure a **user-friendly design** that is simple and easy to use.

10.3 Tools Used

The wireframes were created using:

- **Figma** / **Balsamiq** / **Adobe XD** (any preferred tool can be mentioned).
- Basic shapes (rectangles, text boxes, buttons) to represent screen elements.

10.4 Wireframe Pages

Page No.	Wireframe Name	Description
1	**Login Page**	Allows users to log in using their username and password. Includes options like “Forgot Password” and “Create Account.”
2	**User Dashboard**	Displays account summary, quick transfer options, and navigation menu for different banking services.

Page No.	Wireframe Name	Description
3	**Fund Transfer Page**	Enables users to send money between accounts or to other users, with recipient details and amount fields.
4	**Transaction History Page**	Shows a list of recent transactions, including dates, amounts, and transaction types.
5	**Bill Payment Page**	Provides utilities and merchant payment options for users to pay bills online.
6	**Admin Panel**	Used by administrators to manage user requests, view reports, and maintain account security.

10.5 Sample Wireframes

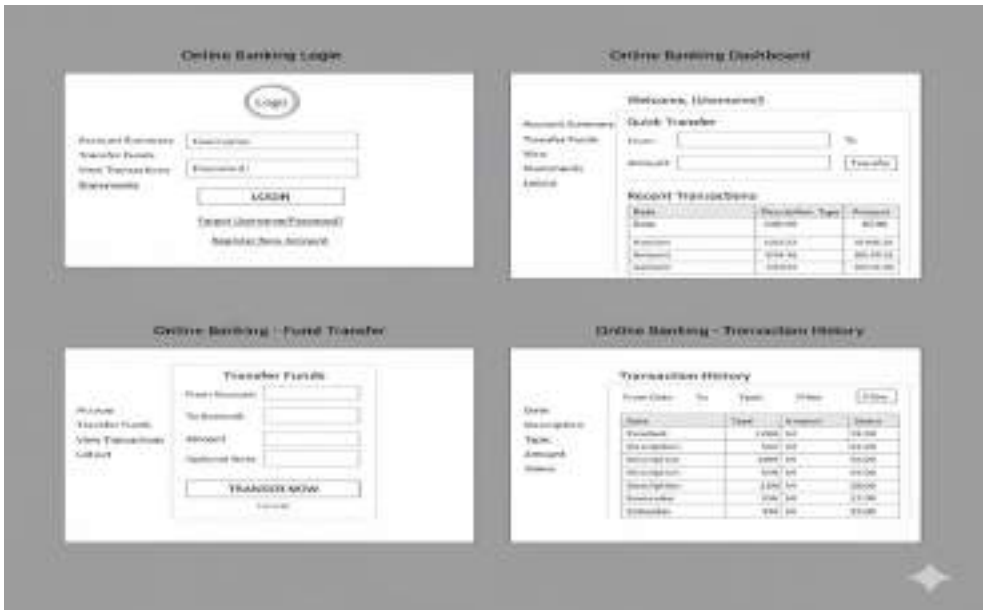

Figure 8 Sample Wireframes Login Page , User Dashboard , Fund Transfer , Transaction History

10.6 Notes

- The design will later be converted into real web pages using **HTML, CSS, and JavaScript**.
- The focus here is **layout**, not colors or detailed graphics.
- Wireframes should match functional requirements from the SRS/FRS.

Database Design (ER Diagram / Schema)

11.1 Purpose

The **database design** defines how data is stored, managed, and related within the system. It ensures data integrity, reduces redundancy, and allows efficient data retrieval.

11.2 Description

The Online Internet Banking System uses a **centralized database** to manage all customer, account, and transaction information.

The design supports modules like login, fund transfer, statement generation, and user requests.

11.3 Entities and Relationships

Main entities include:

- **Customer:** Stores customer details such as name, email, address, and contact.
- **Account:** Holds account number, type, and balance linked to the customer.
- **Transaction:** Records debit/credit details, date, and amount.
- **Request:** Handles cheque book, address change, or stop payment requests.
- **Admin:** Manages users, monitors system, and handles locked accounts.

11.4 ER Diagram

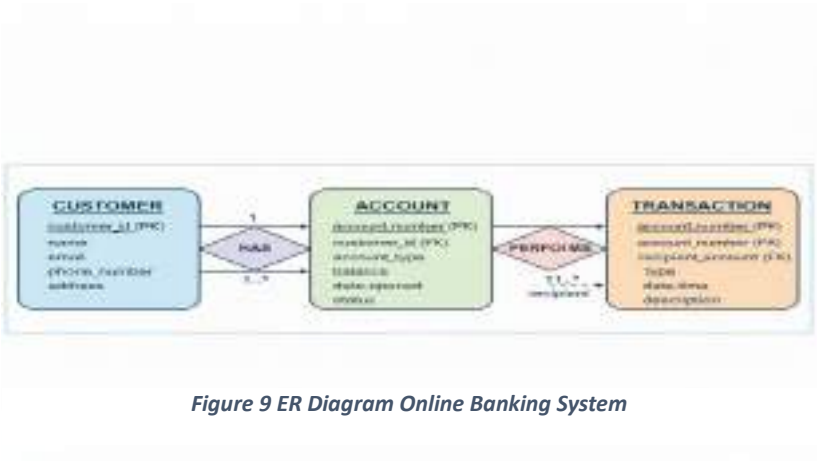

Figure 9 ER Diagram Online Banking System

Data Flow Diagram (DFD Level 0, 1, 2)

12.1 Purpose

The **Data Flow Diagram (DFD)** illustrates how information flows through the **Online Internet Banking System**.

It identifies the main processes, the data exchanged between them, and the external entities interacting with the system.

DFDs help developers and analysts understand the **logical flow of data** and system functionality at different levels of detail.

12.2 DFD Level 0 – Context Diagram

The **Level 0 DFD** provides a high-level overview of the entire banking system as a single process.

It shows the external entities such as **Customer** and **Admin**, and how they interact with the system.

Description:

- The **Customer** logs into the system, views balance, requests services, and performs transactions.
- The **System** processes all these actions and communicates with the **Database**.
- The **Admin** monitors customer activities, manages accounts, and handles system maintenance.
- The **Database** stores information like customer details, accounts, and transactions.

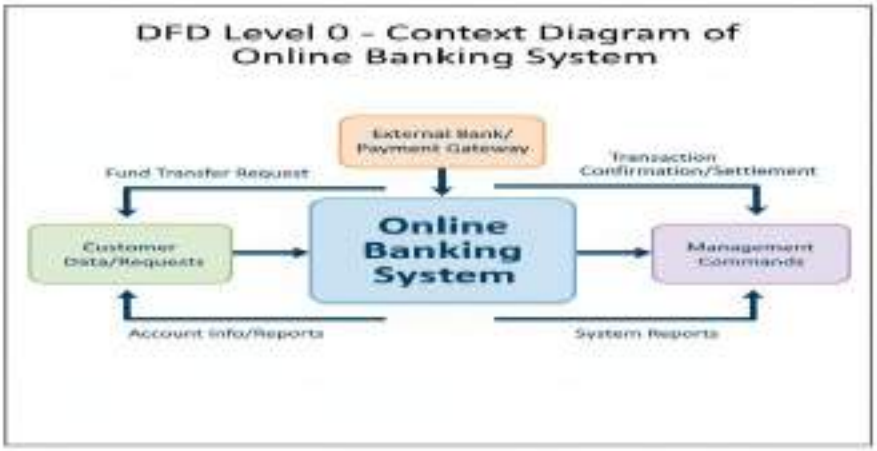

Figure 10 DFD Level 0 – Context Diagram of Online Banking System

12.3 DFD Level 1 – System Processes

The **Level 1 DFD** expands the main system process into several smaller sub-processes. It shows how major functions work independently and exchange data.

Main Processes Include:

1. **Login Process:** Validates username and password.
2. **Account Management:** Fetches and displays account details and balance.
3. **Transaction Processing:** Performs fund transfers and records transactions
4. **Request Handling:** Manages cheque book and address change requests.
5. **Report Generation:** Produces monthly and annual statements.

Each process interacts with the **Database** for reading or updating information

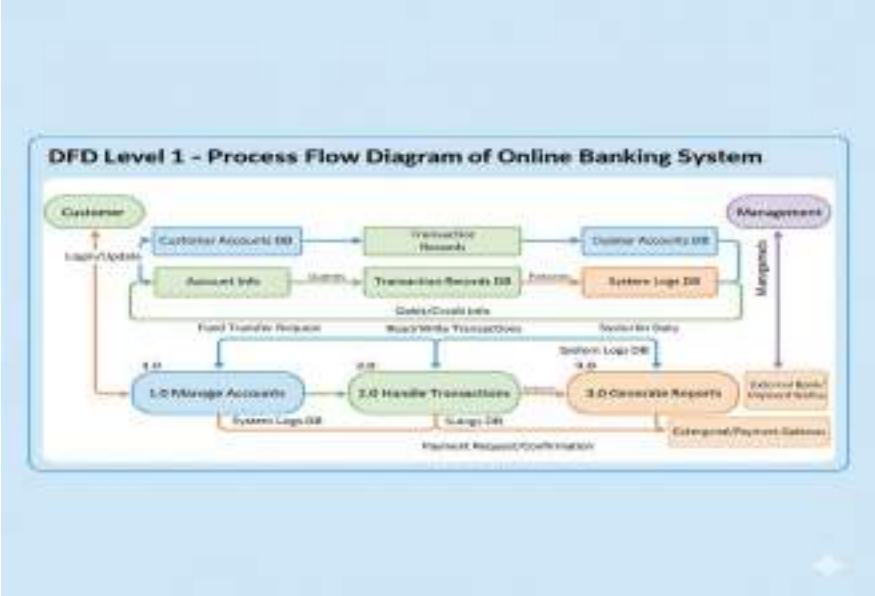

Figure 11 DFD Level 1 – Process Flow Diagram

12.4 DFD Level 2 – Detailed View (Fund Transfer Example)

The **Level 2 DFD** provides a deeper breakdown of one specific process — for example, **Fund Transfer**.

Steps Involved:

1. The user enters recipient details and the transfer amount.
2. The system verifies the transaction password.

3. The sender’s balance is checked and updated.
4. The receiver’s account is credited.
5. A confirmation message and transaction record are generated.

Figure 12 DFD Level 2 – Fund Transfer Process Diagram

System Modules Description

13.1 Purpose

The **System Modules Description** provides an overview of the key modules that make up the **Online Internet Banking System**. Each module is designed to perform specific functions that work together to deliver smooth and secure banking operations. Modular design improves maintainability, scalability, and clarity of the system's functionality.

13.2 Overview of Modules

The system is divided into several modules that interact with each other through a common database. Each module handles its own part of the application logic while ensuring seamless user experience.

13.3 Main Modules

1. Login & Authentication Module

- This module is responsible for verifying user credentials such as username and password.
- It ensures secure access by implementing multiple checks and limits login attempts.
- If incorrect credentials are entered three times, the account is locked and can be reactivated only by the admin.

2. Account Information Module

- Displays all account-related details of the customer, including account number, type, balance, and status.
- This module retrieves live data from the database to ensure accuracy and up-to-date information.

3. Fund Transfer Module

- Allows users to transfer money to another account within the same bank.
- The module verifies the transaction password, checks account balance, updates both accounts, and generates a transaction receipt.
- In case of insufficient balance or incorrect details, appropriate error messages are displayed.

4. Request Handling Module

- Manages customer service requests such as cheque book issuance, address change, or stop payment of cheques.
- Each request is recorded in the system and forwarded for processing by the admin or support team.

5. Transaction & Statement Module

- This module maintains transaction history and allows users to view their last *n* transactions or generate monthly/annual statements.
- It ensures transparency and enables customers to track their financial activities efficiently.

6. Admin Management Module

- Used by the admin to manage customer accounts, unlock locked users, and monitor system activity.
- The admin also reviews customer requests and ensures data integrity across the system.

13.4 Module Interaction

- All modules are interconnected through a **centralized database** (MongoDB).
- For example, the Fund Transfer module updates transaction data, which is instantly reflected in the Account and Statement modules.
- This ensures consistency and reliability of data throughout the application.

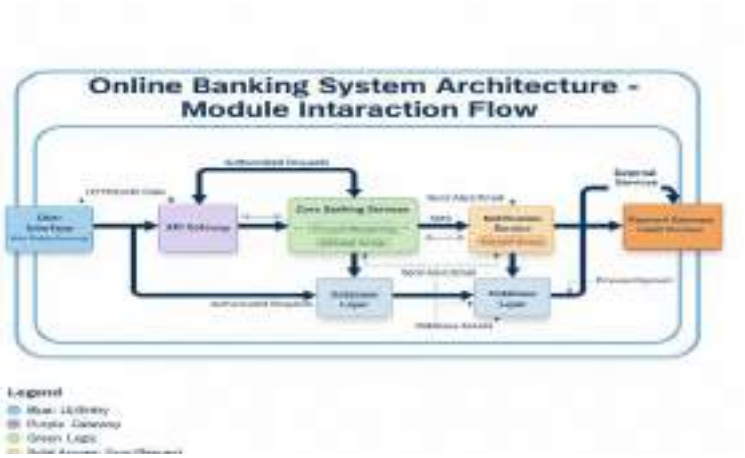

Figure 13 Module Interaction Flow Diagram

Testing Strategy & Test Cases

14.1 Purpose

Testing ensures that the developed **Online Internet Banking System** is reliable, secure, and performs all intended operations correctly.

It helps identify errors, security issues, and usability problems before deployment.

The main goal is to verify that each module functions as expected and that all integrated modules work seamlessly together.

14.2 Testing Strategy

The testing process is divided into multiple levels to ensure comprehensive coverage. Each level focuses on a different aspect of the system's functionality and performance.

1. Unit Testing

- Conducted for individual modules such as Login, Fund Transfer, and Account Management.
- Ensures that each component performs its designated task correctly.
- Example: Verifying that incorrect login attempts are handled properly.

2. Integration Testing

- Performed after unit testing to check the interaction between different modules.
- Ensures that data flows correctly from one module to another without conflicts or data loss.
- Example: Verifying that a fund transfer reflects immediately in the Transaction History module.

3. System Testing

- The complete system is tested as a whole to validate overall functionality.
- Focuses on end-to-end workflows, performance, and security checks.
- Example: Testing login, fund transfer, and statement generation in sequence.

4. User Acceptance Testing (UAT)

- Conducted with real users to ensure that the system meets their expectations.

- Helps evaluate usability, clarity of interfaces, and overall satisfaction.
- Example: Checking if users can easily request a cheque book or view account details.

14.3 Test Cases

Below are some sample test cases used during system testing:

TC01 – Login with valid credentials

- **Scenario:** User logs in with correct login details
- **Steps:** Enter valid username and password
- **Expected Result:** The user should be logged in successfully
- **Status:** Pass

TC02 – Login with invalid credentials

- **Scenario:** User enters wrong password multiple times
- **Steps:** Enter incorrect password three times
- **Expected Result:** Account should be locked
- **Status:** Pass

TC03 – Fund Transfer

- **Scenario:** Transfer money with sufficient balance
- **Steps:** Enter valid account number and amount
- **Expected Result:** Amount should be transferred successfully
- **Status: Pass**

TC04 – Fund Transfer with insufficient balance

- **Scenario:** Attempt transfer without enough balance
- **Steps:** Enter correct account details but low balance
- **Expected Result:** Error message should be displayed
- **Status:** Pass

TC05 – View Account Details

- **Scenario:** User checks account details
- **Steps:** Select account details from dashboard
- **Expected Result:** Correct account information should be displayed

- **Status:** Pass

TC06 – Generate Statement

- **Scenario:** User generates account statement
- **Steps:** Select time period and generate report
- **Expected Result:** Statement should be generated accurately
- **Status**: Pass

TC07 – Request for Cheque Book

- **Scenario:** User requests a cheque book
- **Steps:** Fill the request form and submit
- **Expected Result:** Request should be recorded successfully
- **Status:** Pass

14.4 Result Analysis

All modules passed the defined test cases successfully after minor debugging and improvements.

The system was found to be **stable, user-friendly, and secure** during testing, meeting both functional and non-functional requirements.

Future Enhancements

15.1 Overview

Future improvements will help the **Online Internet Banking System** stay aligned with **user needs and technology advancements**.

The goal is to **enhance user experience, security, and performance** continuously

15.2 Possible Enhancements

- **AI-Based Fraud Detection:** Integrate machine learning models to automatically identify unusual or fraudulent transactions.
- **Mobile Banking Application:** Launch a fully responsive mobile app for both Android and iOS to improve accessibility.
- **Chatbot Support:** Add 24/7 virtual assistant support to answer customer queries in real-time.
- **Multi-Language Support:** Provide services in regional languages to increase user reach.
- **Advanced Analytics Dashboard:** Introduce a visual dashboard for users to track spending patterns and savings insights.
- **Blockchain Integration:** Enhance data security and transaction transparency using blockchain technology.
- **Personalized Notifications:** Send alerts and reminders based on user preferences and transaction history.

15.3 Conclusion

These enhancements will make the system **more intelligent, user-friendly, and secure**, ensuring a **future-ready digital banking experience**

Conclusion

16.1 Summary

The **Online Internet Banking System** has been designed and developed to meet the increasing demand for digital financial services in today's technology-driven world. It provides a convenient, secure, and efficient platform for customers to perform day-to-day banking operations such as **fund transfers, balance inquiries, transaction history viewing, cheque book requests, and account management** without visiting the bank physically.

The system ensures that only authorized users can access their accounts using **unique login credentials**, thereby maintaining confidentiality and data integrity. It also includes mechanisms to prevent unauthorized access through **account lockout after multiple failed login attempts**.

16.2 Key Achievements

- The application offers an intuitive and **user-friendly interface**, making navigation simple even for first-time users.
- It ensures **data consistency and reliability** through proper validation and secure database management using **MongoDB**
- The implementation of **transaction authentication passwords** adds an extra layer of security to financial operations.
- Through proper use of **backend technologies like Node.js and Flask**, the system achieves high performance and scalability.
- The modular structure of the application allows for **easy maintenance and future enhancement**

16.3 System Benefits

Provides **24/7 accessibility** to banking services from any device with an internet connection.

Reduces manual intervention and **increases operational efficiency** for both customers and bank staff.

Ensures a **paperless environment**, promoting eco-friendly digital operations.

Enhances **customer satisfaction and trust** through transparency and real-time transaction feedback.

16.4 Future Vision

In the future, this system can be further enhanced with advanced technologies such as **AI-based fraud detection, voice-assisted banking, and biometric authentication**. Integration with **mobile applications and third-party financial services** can make the platform more comprehensive and customer-centric.

16.5 Final Note

Overall, the Online Internet Banking System achieves its primary goal of providing a **secure, reliable, and efficient online banking platform**. It stands as a step toward a smarter, faster, and safer digital banking future that aligns with modern technological advancements and customer expectations.

References

1. Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.
2. Sommerville, I. (2020). *Software Engineering* (10th ed.). Pearson Education Limited.
3. Fowler, M. (2018). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd ed.). Addison-Wesley.
4. IEEE Standards Association. (2014). *IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications*. IEEE.
5. MongoDB, Inc. (2024). ***MongoDB*** *Documentation*. Retrieved from https://www.mongodb.com/docs/
6. Node.js Foundation. (2024). ***Node.js*** *Documentation*. Retrieved from https://nodejs.org/en/docs/
7. Flask Documentation. (2024). ***Flask Framework*** *Overview*. Retrieved from https://flask.palletsprojects.com/
8. W3Schools. (2024). ***HTML***, ***CSS***, *and* ***JavaScript*** *Tutorials*. Retrieved from https://www.w3schools.com/
9. OWASP Foundation. (2024). *OWASP Secure Coding Practices*. Retrieved from https://owasp.org/
10. Reserve Bank of India. (2023). ***Internet Banking*** *Guidelines and Security Practices*. Retrieved from https://www.rbi.org.in/

Submission & Declaration

18.1 Declaration

I, **Shivam Kumar Mahto,** hereby declare that project titled **"Online Internet Banking System"** has been completed as part of an assignment given by_____.

This document represents my own understanding, analysis, and preparation of system design and documentation including **BRD, FRS/SRS, UML Diagrams, and Wireframes**, based on the problem statement provided.

All sources of information used in preparing this document have been properly acknowledged in the **References** section.

18.2 Submitted By

Name: Shivam Kumar Mahto

Position/Role (Applied For): Business Analyst

Organization:

Submission Date:

Version: 1.0

18.3 Submitted To

Company Name:

Department: Human Resources / Project Evaluation Team

Location: