

Chapter 3

Requirements and Analysis

3.1 Problem Definition

Traditional digital platforms handle either **media streaming** or **file sharing**, but not both together in a single integrated system. This separation causes several challenges:

- **Limited Functionality**
 - OTT platforms only focus on streaming content; they lack offline local file-sharing options.
- **Connectivity Issues**
 - OTT services require a stable internet connection, while file sharing often needs external apps.
- **Time Consuming**
 - Switching between multiple apps (OTT for streaming and another for sharing files) wastes time.
- **Privacy Concerns**
 - Using third-party apps for file sharing can risk personal data exposure.
- **Lack of Integration**
 - No single solution currently combines OTT streaming and local peer-to-peer file sharing.

The proposed “**Local Send**” project integrates **OTT media services** and **Local File Sharing** into a **single software platform**. It enables users to **stream and access digital media** while also **sharing files securely over local Wi-Fi or hotspot**, ensuring both entertainment and productivity in one application.

3.2 Requirement Specification

The requirements for the system are divided into **Functional** and **Non-Functional**.

Functional Requirements

- **User Management:** Register, login, profile update, subscription management.
- **OTT Module:** Browse, stream, and download content (movies, shows, media).
- **File Sharing Module:** Peer-to-peer file transfer across local devices.
- **Device Discovery:** Automatic detection of nearby devices for file transfer.

- **Transfer Management:** Pause, resume, or cancel ongoing file transfers.
- **Notifications:** Alerts for new OTT content, incoming files, or completed transfers.
- **Reports & Logs:** Track user activities, subscriptions, and file transfers.

Non-Functional Requirements

- **Performance:** Smooth video playback (OTT) and high-speed Wi-Fi file transfer.
- **Security:** Encrypted streaming and end-to-end secure file transfers.
- **Usability:** Unified interface for OTT streaming and file sharing.
- **Scalability:** Support for multiple users and devices.
- **Reliability:** Works offline for local file sharing and online for OTT streaming.
- **Maintainability:** Modular design for easy upgrades and bug fixes.

3.3 Planning and Scheduling

3.3.1 Gantt Chart

Project phases include:

Requirement Gathering → Design → OTT Module Development → Local Send Module Development → Integration → Testing → Deployment → Maintenance

GAMIT

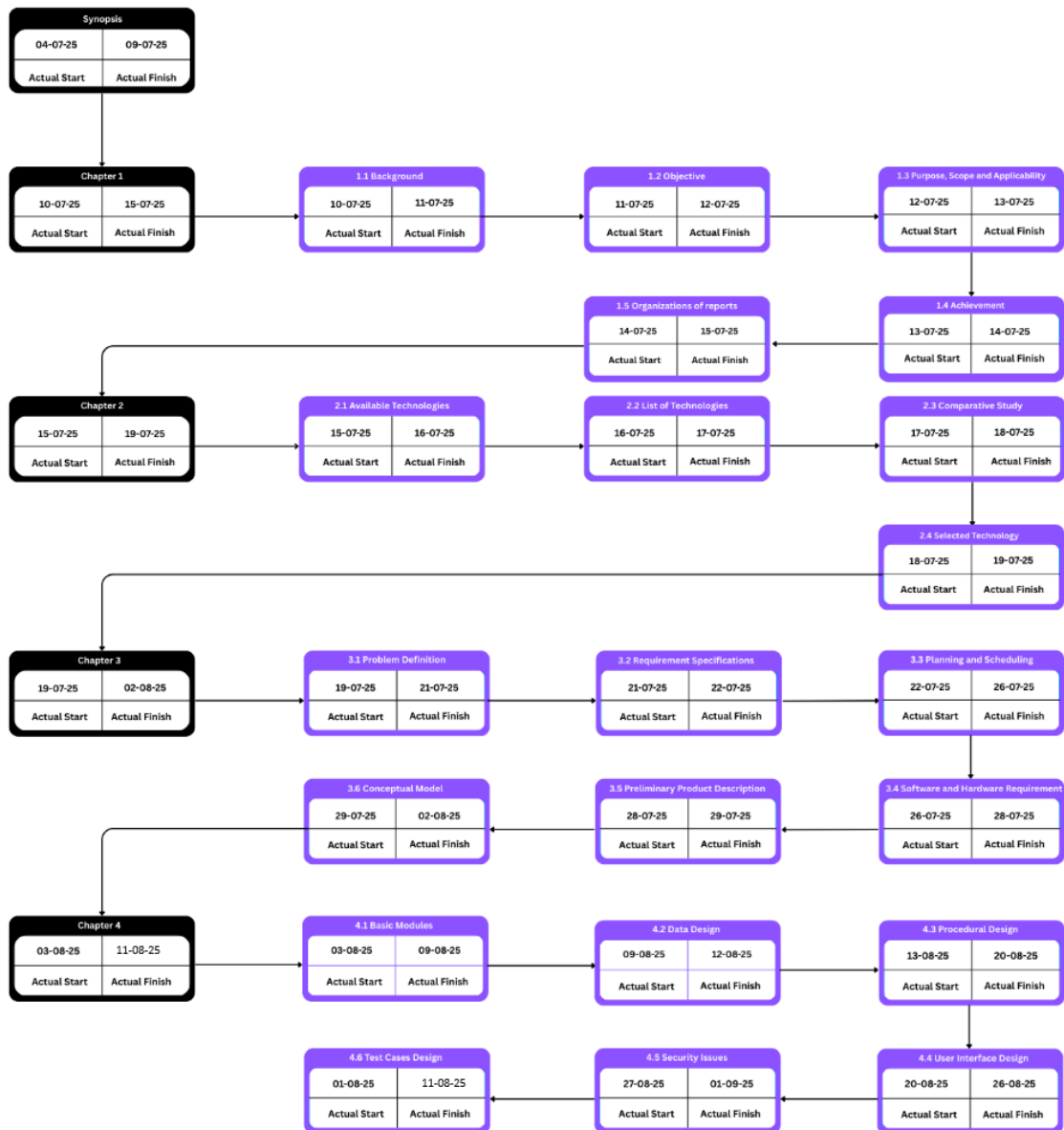
project

		2025																
Name	Begin date	End date	Week 27 29/06/25	Week 28 06/07/25	Week 29 13/07/25	Week 30 20/07/25	Week 31 27/07/25	Week 32 03/08/25	Week 33 10/08/25	Week 34 17/08/25	Week 35 24/08/25	Week 36 31/08/25	Week 37 07/09/25	Week 38 14/09/25	Week 39 21/09/25	Week 40 28/09/25	Week 41 05/10/25	
• Synopsis	04/07/25	08/07/25	[5 Day(s)]	Synopsis [4 Day(s)]														
• Chapter 1 : Intro...	04/07/25	07/07/25																
• Chapter 1 : Intro...	10/07/25	15/07/25																
• Chapter 1 : Intro...	10/07/25	14/07/25																
• 1.1 Background	10/07/25	11/07/25																
• 1.1 Background	10/07/25	11/07/25																
• 1.1 Background	10/07/25	10/07/25																
• 1.2 Objective	11/07/25	12/07/25																
• 1.2 Objective	11/07/25	12/07/25																
• 1.2 Objective	11/07/25	11/07/25																
• 1.3 Purpose, Sc...	12/07/25	13/07/25																
• 1.3 Purpose, Sc...	12/07/25	13/07/25																
• 1.3 Purpose, Sc...	12/07/25	12/07/25																
• 1.4-1.5 Achieve...	13/07/25	15/07/25																
• 1.4-1.5 Achieve...	13/07/25	15/07/25																
• 1.4-1.5 Achieve...	13/07/25	14/07/25																
• Chapter 2 : Surv...	15/07/25	19/07/25																
• Chapter 2 : Surv...	15/07/25	19/07/25																
• Chapter 2 : Surv...	15/07/25	18/07/25																
• 2.1 Available Te...	15/07/25	16/07/25																
• 2.1 Available Te...	15/07/25	16/07/25																
• 2.1 Available Te...	15/07/25	15/07/25																
• 2.2 List of Techn...	16/07/25	17/07/25																
• 2.2 List of Techn...	16/07/25	17/07/25																
• 2.2 List of Techn...	16/07/25	16/07/25																
• 2.3-2.4 Compar...	17/07/25	19/07/25																
• 2.3-2.4 Compar...	17/07/25	19/07/25																
• 2.3-2.4 Compar...	17/07/25	18/07/25																
• Chapter 3 Requi...	19/07/25	02/08/25																
• Chapter 3 Requi...	19/07/25	01/08/25																
• Chapter 3 Requi...	19/07/25	01/08/25																
• 3.1-3.2 Problem...	22/07/25	25/07/25																
• 3.1-3.2 Problem...	22/07/25	25/07/25																
• 3.1-3.2 Problem...	22/07/25	25/07/25																
• 3.3 Planning an...	26/07/25	28/07/25																
• 3.3 Planning an...	26/07/25	28/07/25																
• 3.3 Planning an...	26/07/25	27/07/25																
• 3.4 Software an...	28/07/25	29/07/25																
• 3.4 Software an...	28/07/25	29/07/25																
• 3.4 Software an...	28/07/25	28/07/25																
• 3.5-3.6 (Prelim...	29/07/25	02/08/25																
• 3.5-3.6 (Prelim...	29/07/25	01/08/25																
• 3.5-3.6 (Prelim...	29/07/25	01/08/25																
• Chapter 4 Syste...	03/08/25	04/08/25																
• Chapter 4 Syste...	03/08/25	03/08/25																
• 4.1, 4.2 (Basic I...	03/08/25	12/08/25																
• 4.1, 4.2 (Basic I...	03/08/25	11/08/25																
• 4.1, 4.2 (Basic I...	03/08/25	11/08/25																
• 4.3, 4.4(Procedu...	13/08/25	25/08/25																
• 4.3, 4.4(Procedu...	13/08/25	25/08/25																
• 4.3, 4.4(Procedu...	13/08/25	25/08/25																
• 4.5, 4.6(Security ...	27/08/25	04/09/25																
• 4.5, 4.6(Security ...	27/08/25	27/08/25																
• 4.5, 4.6(Security ...	27/08/25	03/09/25																

3.3.2 PERT Chart

The PERT chart will show dependencies such as:

- UI/UX design
- Backend setup
- OTT content integration
- File transfer protocol setup
- Testing and deployment



3.4 Software and Hardware Requirements

3.4.1 Developer Requirements

Hardware

- Laptop/PC with Intel i3 or above (i5 recommended)
- 8 GB RAM or higher
- 2 GB free storage for project files and database

Software

- OS: Windows/Linux
- IDE: Visual Studio Code / Android Studio
- Frontend: React / Flutter (for cross-platform UI)
- Backend: Node.js / Python Flask
- Database: MySQL / MongoDB
- Version Control: GitHub

3.4.2 User Requirements

Hardware

- Smartphone or PC with internet and Wi-Fi capability

Software

- Any modern web browser or dedicated mobile app for OTT + Local Send

3.5 Preliminary Product Description

The **Local Send OTT + File Sharing Platform** is a **cross-platform application** that integrates both entertainment and utility features.

Users can:

- Register/Login and manage subscriptions
- Stream OTT content (movies, TV shows, music, etc.)
- Discover nearby devices
- Send and receive files locally without internet
- Monitor transfer and streaming status

Admins can:

- Manage content library (OTT)
- Monitor transfer history
- Handle subscription and access control
- Configure updates and bug fixes




This ensures an **all-in-one platform** that provides both **media streaming** and **local file transfer**.

3.6 Conceptual Model

3.6.1 Use Case Diagram

The Use Case Diagram for **Local Send** includes interactions such as:

- User browsing and streaming OTT content
- User discovering nearby devices
- Sending and receiving files
- Admin managing content, subscriptions, and reports

Symbol	Name	Description
	Actor	An actor specifies a role played by a user or any other system that interacts with the subject. It could be a user, another system, or an external hardware device.
	Use Case	Represent the different uses that a user might have. These describe the actions performed by actors to achieve a specific goal.
	Association	Indicates that an actor participates in or interacts with a use case. It shows the interaction or communication between the actor and the system's functionality.

3.1 Use Case Diagram Symbols

Actors With Respect to Project:

1. User
2. Admin
3. Server

Use Cases for User:

1. Registration / Login
2. Browse Files
3. Send File
4. Receive File
5. Stream Content
6. Pause/Play Media
7. View Media Recommendations

Use Cases for Admin:

1. Login
2. Manage Users
3. Upload Content
4. Manage Content
5. Generate Reports
6. Monitor Server

Use Cases for Server:

1. Authenticate Users
2. Route Files
3. Stream Media
4. Transcode Media (if applicable)
5. Monitor Activity

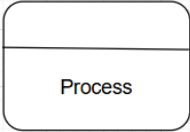
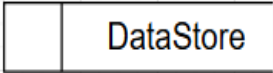


3.6.2 Data Flow Diagram

Definition:

A Data Flow Diagram (DFD) is a visual representation of the flow of data through a system, highlighting the inputs, processing, and outputs. A neat and clear DFD can depict a good amount of the system requirements graphically, it makes easy for the business requirements of applications by representing the sequence of process steps by step. It shows how data enters and leaves the system, what changes the information, and where data is stored. The DFD is also called as a data flow graph or bubble chart.

Components

Component	Symbols	Description
Names		
Process		Processes represent activities or functions that transform input data into output data. They are the actions performed on data within the system.
Data Source		Data Stores represent repositories where data is stored. They can be databases, files, or any other storage location. Data flows in and out of these stores through processes.

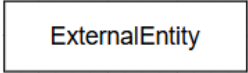

External Entity		External Entities are sources or destinations of data that interact with the system but are outside of it. They represent users, other systems, or external data sources.
Data Flow		Data Flows depict the movement of data between processes, data stores, and external entities. They indicate how data is transferred, processed, or stored within the system.

Table 3.3 Data Flow Diagram Symbols.

Entities Used:

1. User
2. Admin
3. External Media Sources

Processes:

1. Authentication Process (handles login).
2. File Sharing Process* (handles LAN file transfers).
3. Media Streaming Process(handles OTT streaming).
4. Content Management Process (administers media content).
5. Report Generation Process (generates reports for admin).

Data Stores:

1. User Database: Stores user data and preferences.
2. File Database: Stores file transfer information.
3. Media Database: Stores media content for streaming.
4. Report Database: Stores activity reports (optional).

Level 0 Diagram

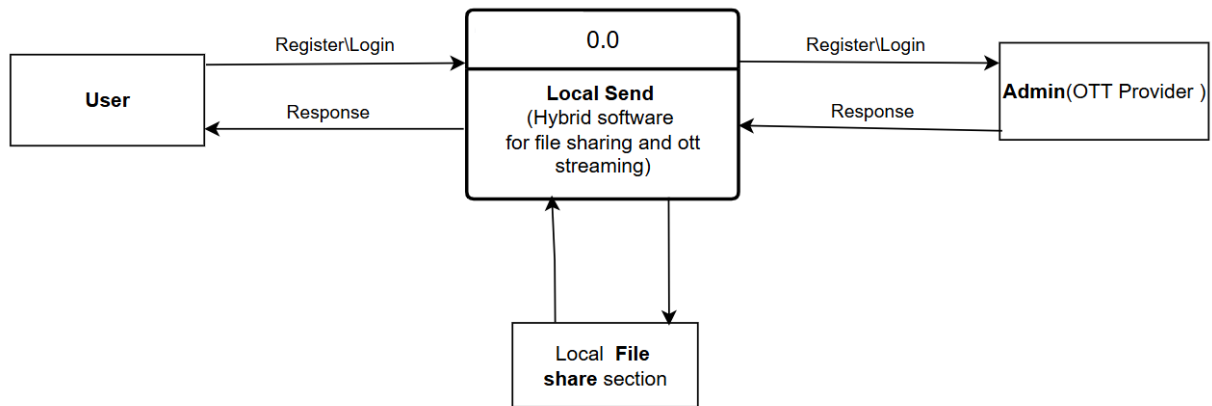


Fig 3.4 Data Flow Diagram Symbols (level 0)

Level 1 Diagram

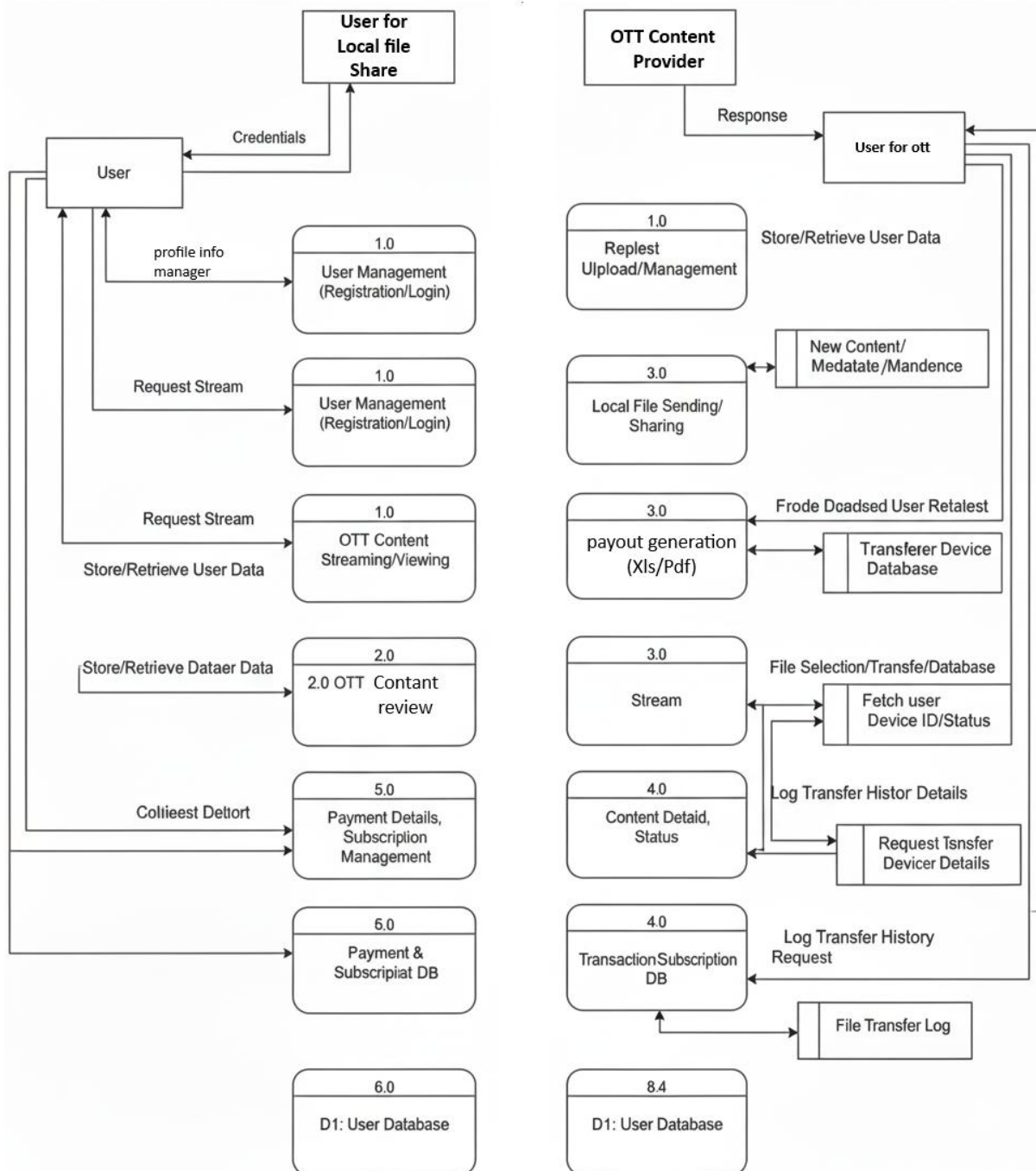


Fig 3.5 Data Flow Diagram Symbols (level 1)

Level 2 Diagram

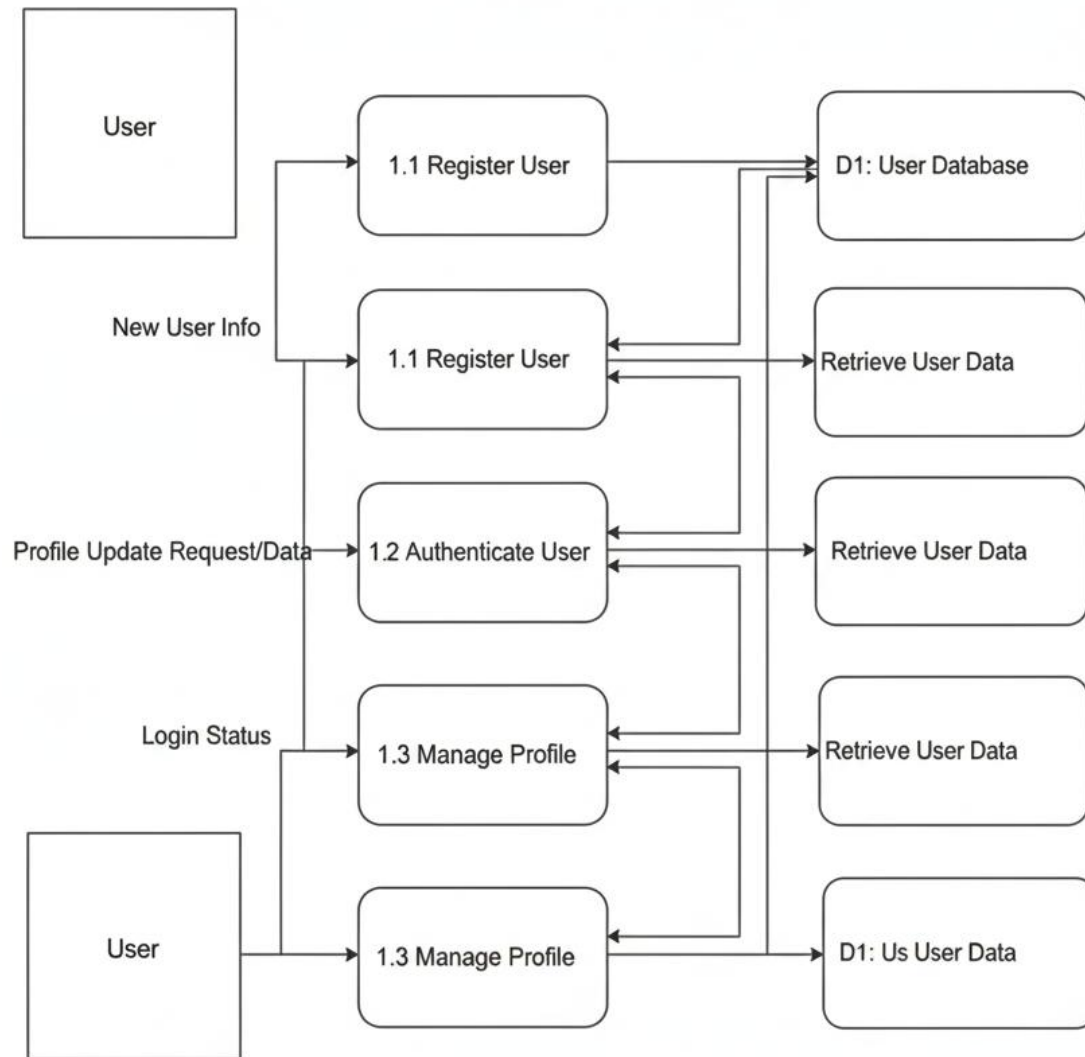


Fig 3.6 Data Flow Diagram Symbols (Level 2)

3.6.3 Activity Diagram

Definition :

An Activity Diagram is a type of UML (Unified Modelling Language) diagram that shows the dynamic behaviour of a system, highlighting the flow of control and data between activities. It's used to model the workflow, business processes, and operational procedures of a system.


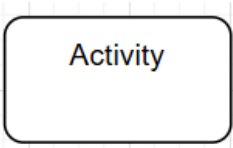
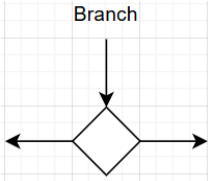
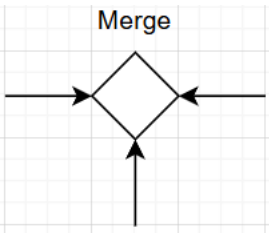

Components Component Names	Symbols	Description
Initial State		A small, filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.
Activity or Action State		A step in which the users or software performs a certain task. It represents an action that is going to take place at this stage of the software system.
Decision And Branching		A decisional node is one where there are multiple options available. Or there are two or more conditions which can be considered at the point of the software system.
Merge Node		The notation for a merge node is a diamond-shaped Symbols with two or more edges entering it and a single activity edge leaving it. Merge node and decision node combined. The functionality of merge node and decision node can be combined by using the same node Symbols.
Final State		An arrow pointing to a filled circle nested in another circle represents the final action state.

Table 3.3 Activity Diagram Symbols.

Activity Diagram 1

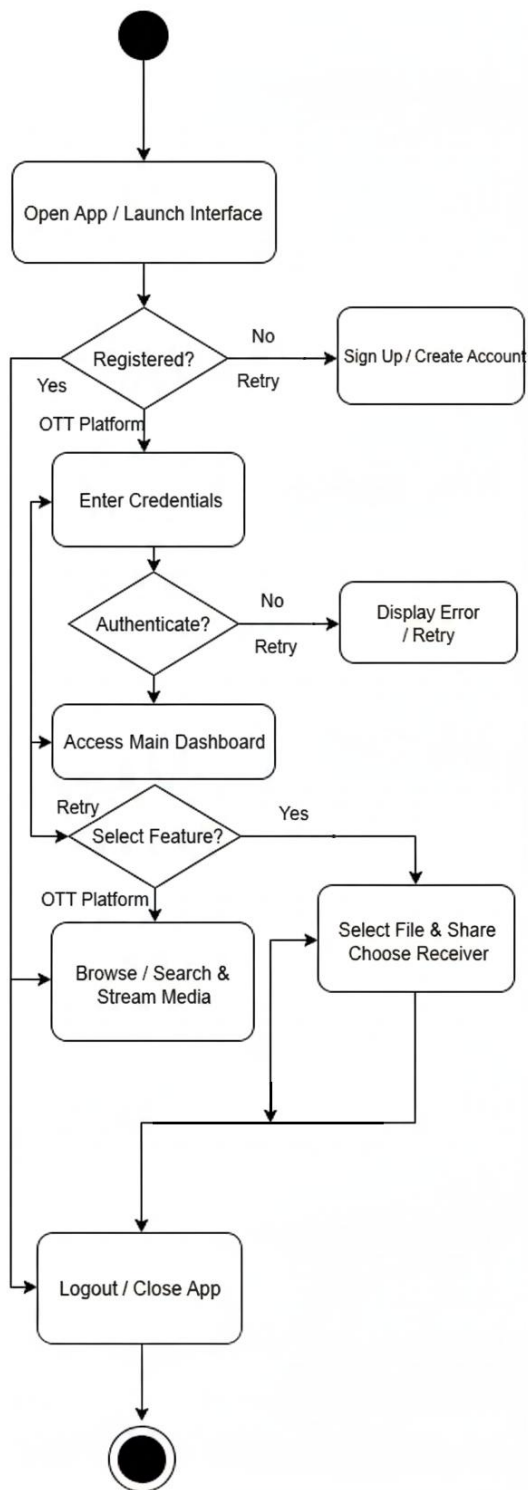


Fig 3.7 Activity Diagram Symbols 1

Activity diagram 2

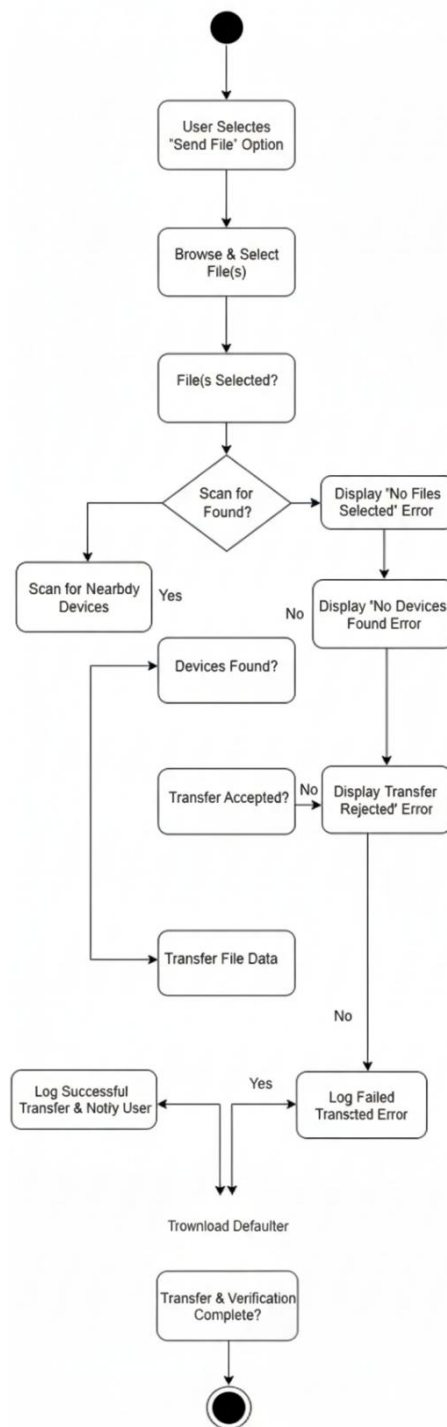


Fig 3.8 Activity Diagram Symbols 2

3.6.4 Entity-Relationship (ER) diagram

Definition

An Entity-Relationship (ER) diagram is a visual representation in database design that illustrates the relationships between various entities or objects within a system and how they interact with each other. It showcases the structure of a database by depicting entities as tables, attributes as columns, and relationships as connecting lines. ER diagrams help in designing and understanding the logical structure of a database system.

Components

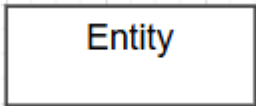


Component Names	Symbols	Description
Entities		Represented as a rectangle, an entity represents a real-world object, concept, or thing. For example: “Student” could be an entity.
One to Many Relationship		This relationship represents a scenario where one entity instance is associated with multiple instances of another entity, but each instance of the second entity is linked to only one instance of the first entity
Many to Many Relationship		In this type, multiple instances of both entities are connected, indicating that each instance from one entity can be associated with multiple instances from the other entity, forming a complex network.

Table 3.3 Entity-Relationship (ER) Diagram Symbols

Entity-Relationship (ER) diagram

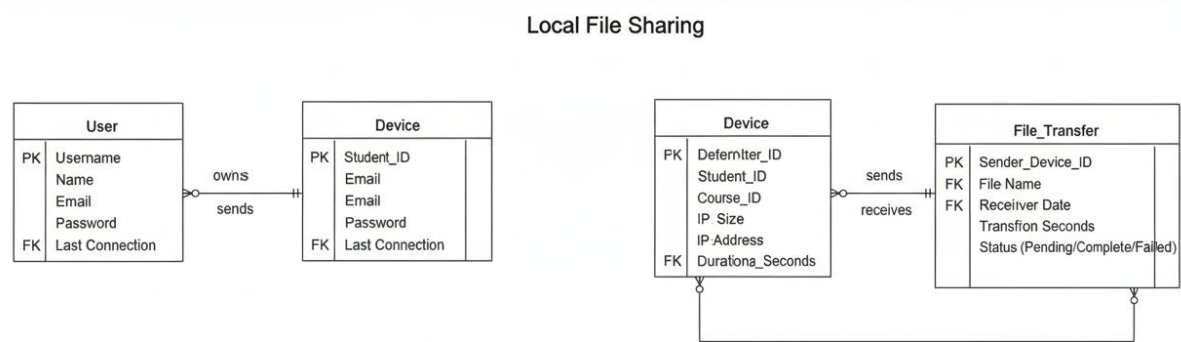
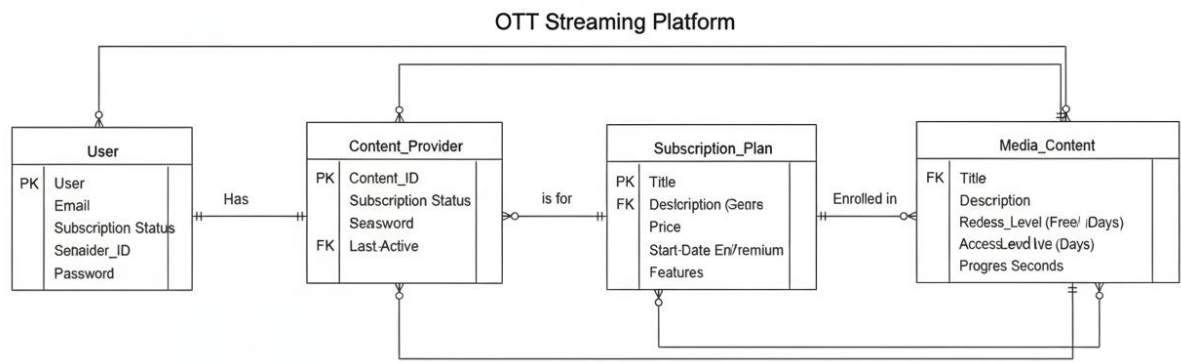


Fig 3.9 ER Diagram