Design Document for Connecta

1 Overview

This project aims to develop an online social networking system to connect users, foster meaningful interactions, and provide a dynamic platform for sharing content. Designed to enhance user experiences, the system incorporates various features, including profile management, content sharing and managing connections.

This document shows the DFD and the structured chart for the system with detailed information regarding them.

2 DFD of the System

The DFD of the system, which describes the various inputs and outputs and the flow of data in the system, is shown in Figure 1(a,b, and c). The DFD gives an overall idea of how this system processes the data and satisfies the various constraints.

2.1 Context Diagram (Level-0 DFD)

The context diagram for Connecta is shown in Fig 1a. All the primary inputs and outputs are shown in the diagram. Connecta only has one external entity (User) that plays a significant role in the system. Figure 1a shows the inputs and output based on the various functionalities of the system, including the aspects of registration and authentication, post management, profile management, connection handling, and search functionality. Each functionality has inputs and outputs to the only source/sink of the system.

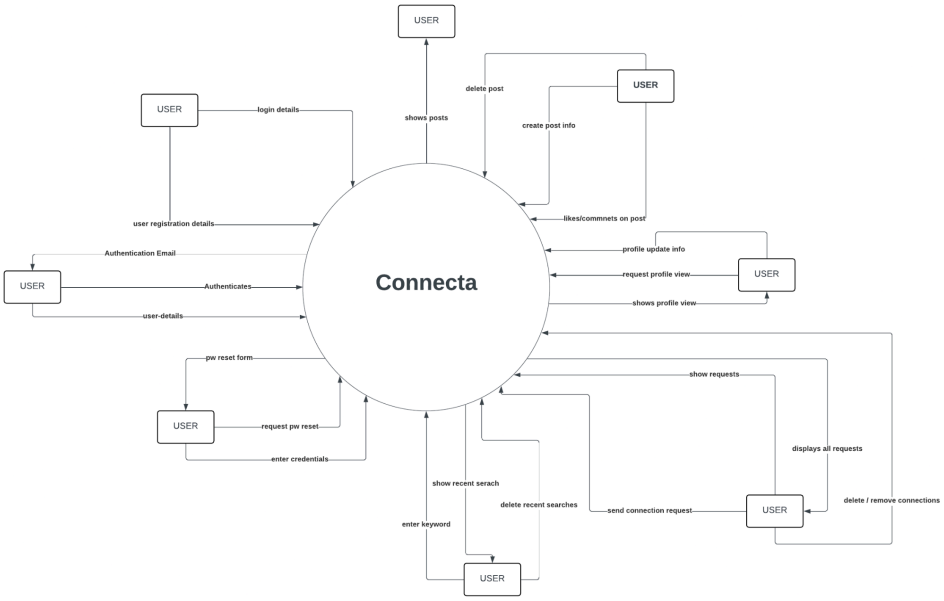

Fig1a: Level-0 DFD for Connecta

2.2 Level-1 DFD

The Level-1 DFD shown in Fig 1b encompasses the system's significant processes and major datastore. Some methods are abstracted, along with some data stores that come after a more profound analysis of the DFD. Fig 1b shows the 10 major processes (most abstracted) and 4 significant data stores to handle each system functionality.

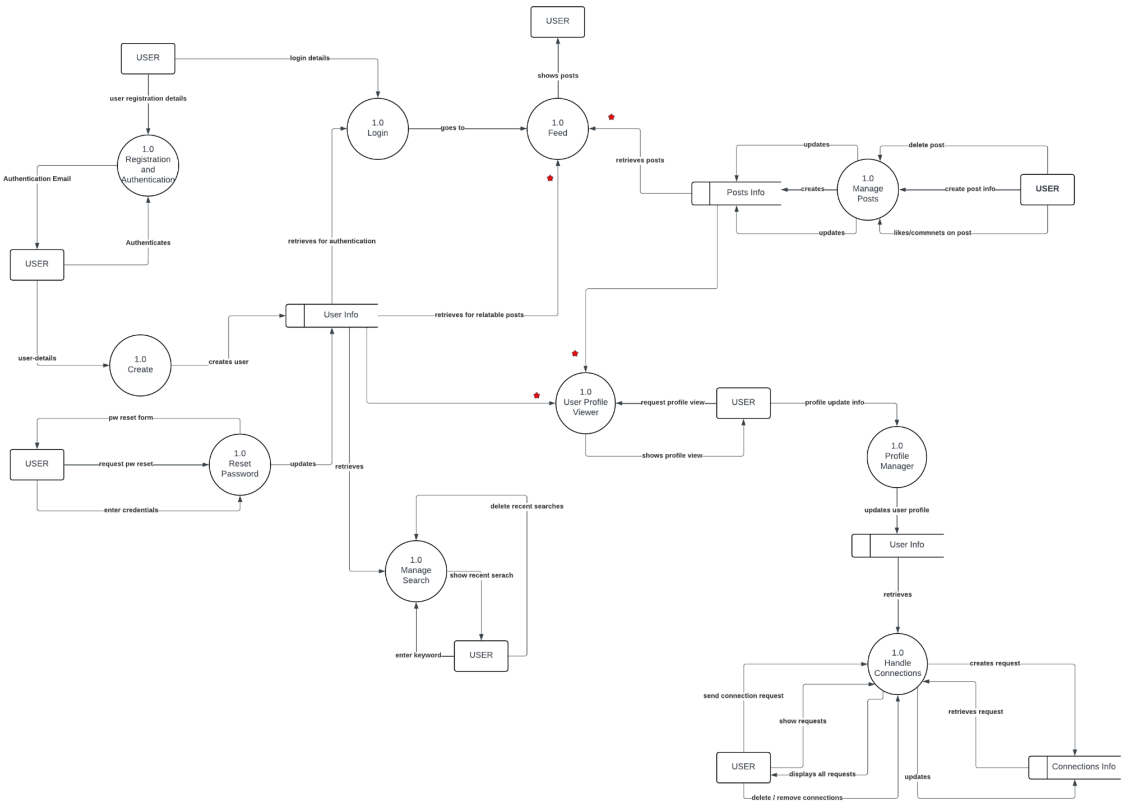

Fig1b: Level-1 DFD for Connecta

2.3 Level-2 DFD

The Level-2 DFD shown in Fig 1c is more thorough for the different processes shown in Fig 1b. It shows the 15 major processes and 5 major data stores to handle each functionality of the system. The handle connection, manage search, manage posts, and registration & authentication are shown in a more detailed manner.

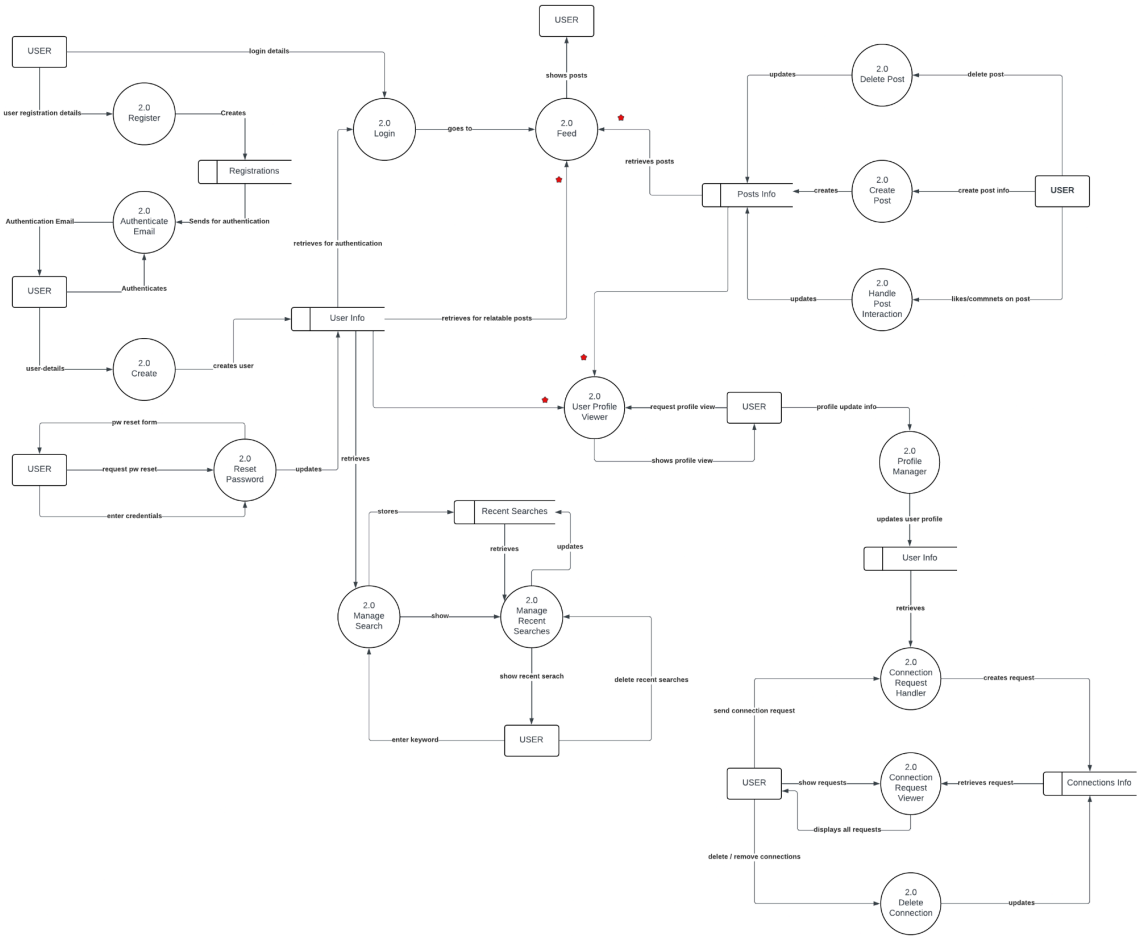

Fig1c: Level-2 DFD for Connecta

- Most Abstract Input

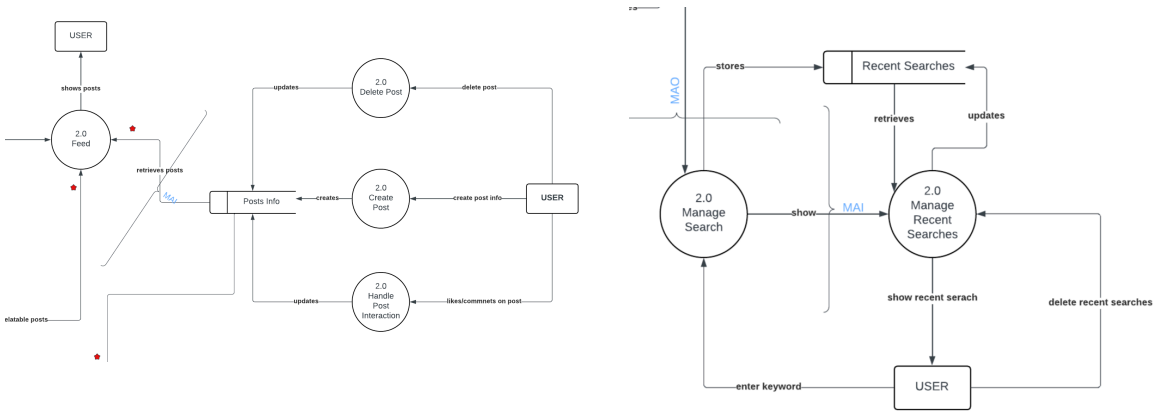

Fig2: MAI

The MAI in the system's level 2 DFD is the retrieval of posts in Feed and showing recent searches, as shown in Fig 2 above.

3 Structured Chart

The structure chart of the proposed system describes the overall structure of the initial design that was created in Figure 2. After analysis, this design was enhanced. Note that this design is consistent with the system's architecture in its software architecture document and the DFD shown above. Mapping the modules in this design to the components in the architecture is also relatively straightforward.

3.1 Factored Input Modules

The two factored input modules are shown in Fig 3 given below. In the initial step, the high-level structure of the system is analyzed, and its modules are thoroughly examined. This process involved factoring the input modules. From the standpoint of the Connecta, an input module's function is to supply data. Receiving data from external sources, including posts, reactions to the posts and recent searches.

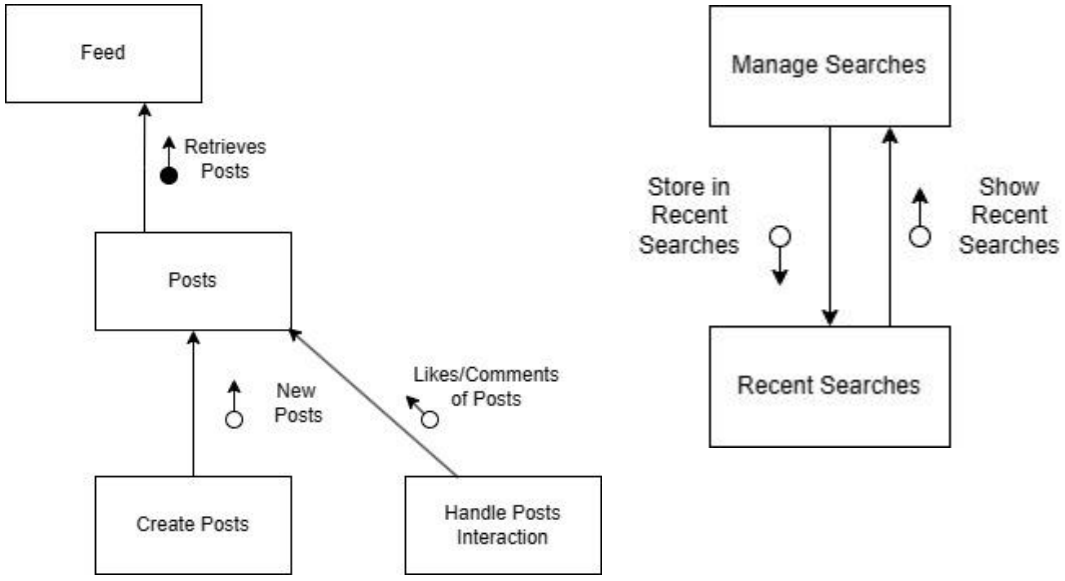

Fig 3: factored input modules

3.2 Factored Transform Module

Factoring of the transform module plays a crucial role in determining how data flows and is processed inside a system. They encompass essential functions and frequently specify the actions that follow in data processing or system behavior because of this, their implementation and design need to be well thought out in order to guarantee their robustness, efficiency, and compatibility with the larger system architecture. The fig 4 describes the factoring of how the login details are credentialed and verified as the factoring of the transform modules.

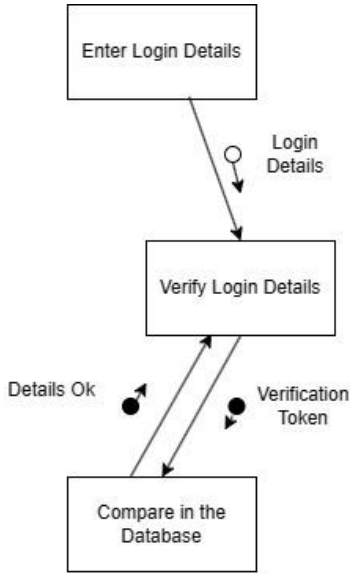

Fig 4: factored transform modules

3.3 Final Structure Chart

Figure 5 shows the final structured chart consisting of factored input and central transform. The primary functionality has three different routes through which users can interact. Registration is a one-time activity. Users can choose to request a password change. The central branch starts from the Login screen. Users log in to the system and can navigate through major processes, among which Feed and Manage Search have two MAIs that have been factored in. The registration and the login has a loop.

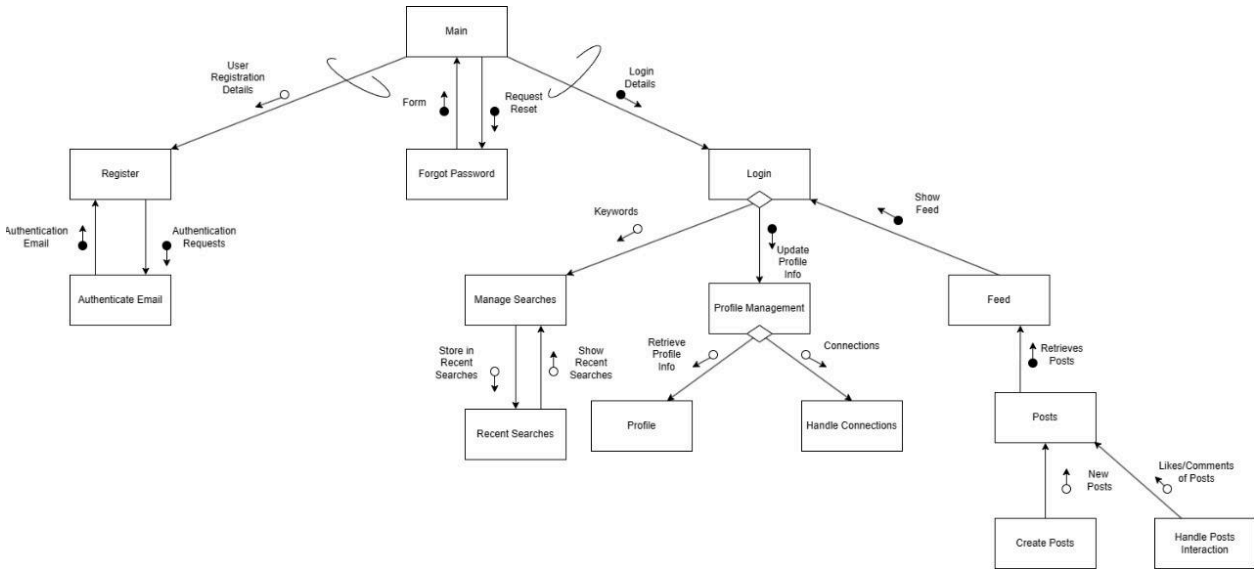

Fig 5: Structured Chart for Connecta

4 Design Analysis

4.1 Name of final factored modules, their types, cohesion and estimated size:

Module Name	Type	Cohesion	Estimated Size(LoC)
Registration	Input	Functional	500
Login	Input	Functional	300
Create Post	Input	Functional	800
Delete Post	Output	Functional	400
Create Comment	Input	Functional	600
User Profile Viewer	Coordinate	Communicational	1000
Feed	Transform	Functional	1200
Handle Feed Interaction	Transform	Functional	900
Manage Searches	Coordinate	Communicational	700
Connection Request Manager	Composite	Temporal	1500
Profile Manager	Transform	Functional	800

4.2 Justification for cohesion and coupling

Registration: This module has functional cohesion as it handles the specific task of user registration. It has low coupling with other modules as it only interacts with the Authentication Email module to send registration emails.

Login: This module exhibits functional cohesion as it is focused on the single task of user authentication. It has low to moderate coupling with the Feed module to retrieve and show the user's feed after successful login.

Create Post: This module displays functional cohesion as it is responsible for creating new posts, which is a specific and well-defined task. It has moderate coupling with the Feed module to add the new post to the user's feed and the User Info module to associate the post with the user

Delete Post: This module has functional cohesion as it deals with the specific task of deleting a post. It has low to moderate coupling with the Feed module to remove the deleted post from the feed and the User Info module to update the user's post count.

Create Comment: This module exhibits functional cohesion as it is focused on the single task of creating comments on posts. It has moderate coupling with the Feed module to add the comment to the respective post and the User Info module to associate the comment with the user.

User Profile Viewer: This module displays communicational cohesion as it coordinates and manages data from different sources (User Info, Posts Info, Recent Searches) to display the user's profile. It has moderate coupling with these modules to retrieve and display the required data.

Feed: This module has functional cohesion as it is responsible for displaying the user's feed, which is a specific task. However, it has high coupling with multiple modules (User Info, Posts Info, Handle Feed Interaction) as it needs to interact with them to fetch and display the feed content.

Handle Feed Interaction: This module exhibits functional cohesion as it handles interactions with the feed, such as likes and comments, which is a specific task. It has moderate to high coupling with the Feed module to update the feed based on user interactions and the User Info module to associate interactions with users.

Manage Searches: This module displays communicational cohesion as it manages and coordinates data from different sources (Recent Searches, User Info) to handle search functionality. It has moderate coupling with these modules to retrieve and update search data.

Connection Request Manager: This module has temporal cohesion as it deals with time-dependent data and processes related to connection requests. It has moderate coupling with the User Info module to retrieve and update user information and the Recent Searches module to store and retrieve recent connection requests.

Profile Manager: This module exhibits functional cohesion as it is responsible for managing user profiles, which is a specific task. It has moderate coupling with the User Info module to update user profile information and the Feed module to reflect profile changes in the user's feed.

4.3 Count of Module types

Module Type	Count
Input	4
Output	1
Coordinate	2
Transform	3
Composite	1

4.4 Most complex or error-prone modules

Input subsystem: The Create Post module can be complex due to handling various types of content (text, images, videos) and validating user inputs.

Transformation subsystem: The Feed module can be error-prone as it needs to handle and display different types of content (posts, comments, likes, etc.) from multiple sources while maintaining proper ordering and filtering.

Output subsystem: The Delete Post module can be error-prone due to the need for proper authorization checks and maintaining data integrity when deleting posts and associated comments or likes.

4.5 Top-3 modules in terms of fan-out and fan-in

Fan-out:

- Feed (fan-out: 4)
- Connection Request Manager (fan-out: 3)
- User Profile Viewer (fan-out: 2)

Fan-in:

- User Info (fan-in: 4)
- Feed (fan-in: 3)
- Profile Manager (fan-in: 2)

4.6 Total expected size and subsystem sizes

- Total expected size (LoC): 9200
- Input subsystem size (LoC): 2200
- Transformation subsystem size (LoC): 2900
- Output subsystem size (LoC): 400

5 Detailed Design Specification

Each class represents a final level factored module with its associated attributes and methods, as described in the design document. These interfaces can serve as a basis for implementing the functionality of the online social networking system.

1. class Registration {
   public:
   void input_data(string username, string email, string password);
   void process_registration();
   };

2. class Login {
   public:
   void input_credentials(string username, string password);
   void authenticate_user();
   };

3. class PostManager {
   public:
   void create_post(string content);
   void delete_post(int post_id);
   void edit_post(int post_id);
   };

4. class CommentManager {
   public:
   void create_comment(int post_id, string content);
   };

5. class UserProfileViewer {
   public:
   void view_profile(int user_id);
   };

6. class FeedManager {
   public:
      vector<string> get_feed(int user_id);
   };

7. class FeedInteractionHandler {
   public:
      void like_post(int post_id);
      void comment_on_post(int post_id, string comment_content);
   };
8. class FeedInteractionHandler {
   public:
      vector<string> search(string query);
   };

9. class ConnectionRequestManager {
   public:
      void send_request(int sender_id, int receiver_id);
      void handle_request(int request_id, string action);
   };

10. class ProfileManager {
   public:
      void edit_profile(int user_id, string new_data);
      void view_profile(int user_id);
   };