# SOFTWARE ARCHITECTURE FOR PhotoN WEBSITE

**Anudeep Rao Perala (CS21BTECH11043)**
**Asli Nitej Reddy Busireddy (CS21BTECH11011)**
**Narsupalli Sai Vamsi (CS21BTECH11038)**
**Pranav Varma Pericherla (CS21BTECH11044)**

# Contents

# 1 Overview

## 1.1 System Overview

PhotoN is a user-friendly online platform for photo management and sharing. It provides essential functionalities, including secure account management, photo uploads with smart tagging, social interaction, and a marketplace for showcasing photos. PhotoN simplifies photo organization and makes it easy for search.

## 1.2 System Context

PhotoN acts as a hub for photographers and other artists to upload, manage, and share their photographic content. Users are the primary contributors and the main users of the application, they do tasks like uploading their photos, engaging with others' photos. External integrations include services for advanced features like image recognition and tagging, provided by APIs such as OpenAI which improve the user experience more. The system context is detailed in the SRS document so as how the system operates.

## 1.3 Stakeholders of PhotoN

The main stakeholders and their concerns of PhotoN are:

- **Users:** These stakeholders main concern is that how easy is the platform to use. This include how simple it is to find their way around, manage their photos, share content, interact with others, and to explore a wide range of photos.
- **Moderators:** These stakeholders are concerned with content governance, ensuring that the photos uploaded to the platform comply with the guidelines of the application.
- **External API's:** These stakeholders are concerned about providing API's for the functioning of the application. These stakeholders include OpenAI API, Location tagging API for geo-tagging of the photos.

## 1.4 Scope of this Document

In this document we describe two architectures for our application we consider only component and connector view architecture style for this, and then we compare those architectures to decide which architecture suites best for our application. We also describe the reason behind selecting a particular architecture.

## 1.5 Definitions

- **PhotoN**: The name of the photo storage, sharing, and exploration platform.
- **Marketplace**: This is the global feed of photos.
- **PhotoMap**: This is the map of photos, photos are displayed on the map.
- **Vault**: It serves as a collection to store and manage their photos securely.
- **Albums**: These are the vaults that are created by the user for organizing photos.
- **Folders**: These are the predefined vaults that are given by the application.

# 2 Architecture Design

## 2.1 Architecture 1: Client-Server 3-tier

### 2.1.1 Client Tier:

- **Client Interface Component:** Manages user interface and interaction, enabling operations like photo uploads, album and folder management, and social activities.

### 2.1.2 Server Tier:

#### Service Layer

- **User Management Service:** Manages user account operations. Interacts with the database for user data.
- **Photo Management Service:** Handles photo operations including upload, edit, and tagging.
- **Album Management Service:** Manages album operations: Create, Delete, Update, Add and Remove Photos and
  Interacts with external PhotoMap API for displaying maps within albums.
- **Folder Management Service:** Manages operations related to photo organization along with Locked, Shared, Bin etc.
- **Social Interaction Service:** Facilitates social functionalities; interacts with the database for user follow/unfollow and photo sharing.
- **Search Service:** Executes search operations; interacts with the database for searching photos and users.
- **Marketplace Interaction Service:** Communicates with the database for marketplace data i.e User Interested Feed and User Follow Feed.

#### Data Access:

- **Data Access Objects (DAOs):** DAO provides data operations without exposing database details.
- Key Responsibilities: CRUD Operations, Connection Management, Data Mapping.
- DAOs used: User, Photo, Tag, Follow, Like, Share

#### External Services APIs:

- **OpenAPI, PhotoMap API:** Provide specialized functionalities like tagging and mapping, used by respective services.

### 2.1.3 Database:

- **Database Server:** Central storage for all persistent data including users, photos, albums, interactions, and marketplace data.

### 2.1.4 Connectors:

- **HTTP/HTTPS Connectors:** Facilitate communication between the client and server tiers.
- **RPC:** Facilitate communication between Services and DAOs within server tier
- **API Connectors:** Enable services to utilize external functionalities.
- **Database Connectors:** Ensure server tier services can perform database operations through the Data Access Layer.

Architecture Diagram:

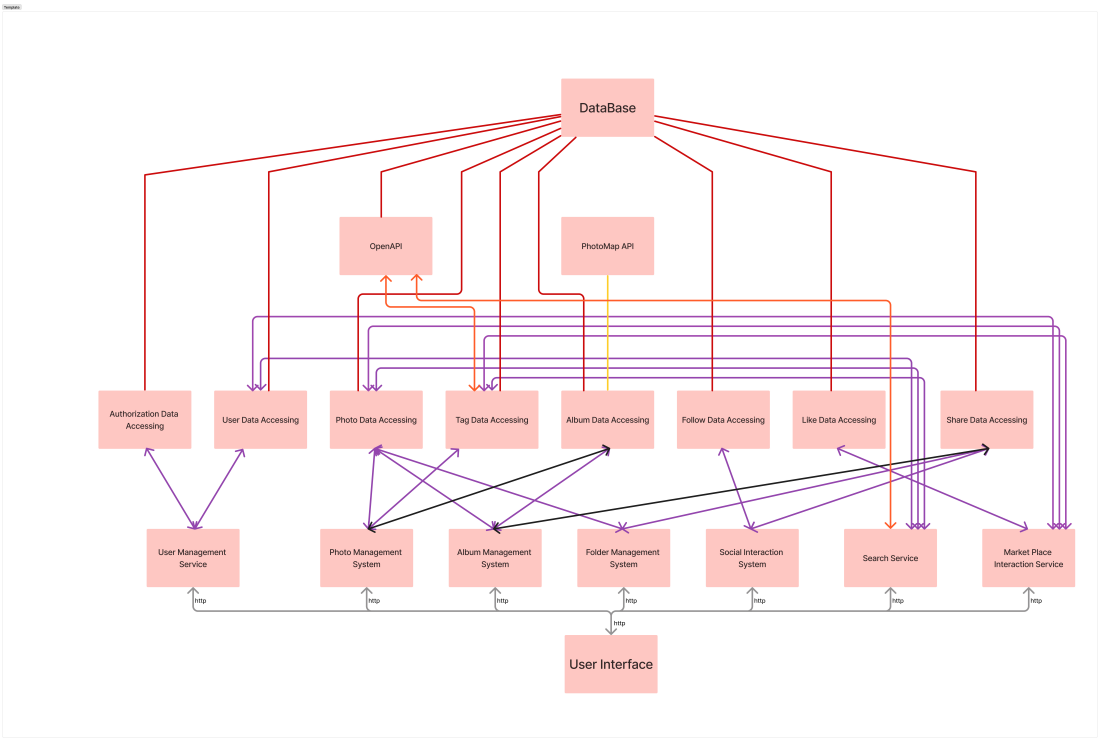


Figure 2.1: Client Server Architecture

## 2.2 Architecture 2: Client-Server 3-tier

The application is set up in a clear three-tier design, they are:

- **Presentation Tier:** The user interface is the primary interface between the user and the application.
- **Business Logic Tier (including Data Access Layer):** Central to the architecture is the Central Server, this acts as the command center for routing requests and managing the flow of operations.
- **Data Tier:** This tier is exclusively dedicated to data storage and management.

### 2.2.1 Connectors and Components:

Following are the main Components of the architecture:

| **Component** | **Component Type** | **Description** |
|---|---|---|
| User Interface | Frontend Component | The graphical interface through which users interact with the application. |
| Central Server | Server Component | Acts as the primary server that helps in the communication between different services, servers and the user interface. |
| Authentication Server | Server Component | Manages user authentication, session management, and security to ensure that only valid users can access their accounts. |
| Photo and Album Server | Server Component | Handles the storage, retrieval, and management of photos and albums, supporting operations like upload, edit, and delete , sharing a photo or album and also saving/liking a photo from global feed. |
| Map Service | External Service | Provides mapping functionalities of the photo for PhotoMap, by using an API to map the geographical locations. |
| Data Access Layer | Middleware Component | Acts as an middle layer between the application's business logic and the database. |
| Database | Storage Component | Stores all the data, like user profiles, photos, albums. |
| Auto Tagging Open API | External API | Automatically tags photos with relevant tags. |
| Auto Tags for a Description Open API | External API | Automatically gives relevant tags for a given description. |
| Profile Management Service | Server Component | Allows users to manage their personal information. |

| Component | Component Type | Description |
|---|---|---|
| Social Interactive Server | Server Component | Includes social features such as following, searching a user and count of followers and other social interactions. |

Following are the Connectors of the architecture:

| Connector | Connector Type | Description |
|---|---|---|
| HTTP/HTTPS | Network Protocol | Utilized for communication between the user interface and the central server, and also between the central server, various other services and servers. |
| MongoDB Connector | Database Protocol | Connects the Data Access Layer to the MongoDB database, handling all CRUD operations. |
| Read (R) Connector | Database Access | Represents the operations that involve reading data from the database. |
| Write (W) Connector | Database Access | Represents the operations that involve writing data to the database |
| Read/Write (R/W) Connector | Database Access | Represents operations that involve both R/W to the database. |

Architecture Diagram:

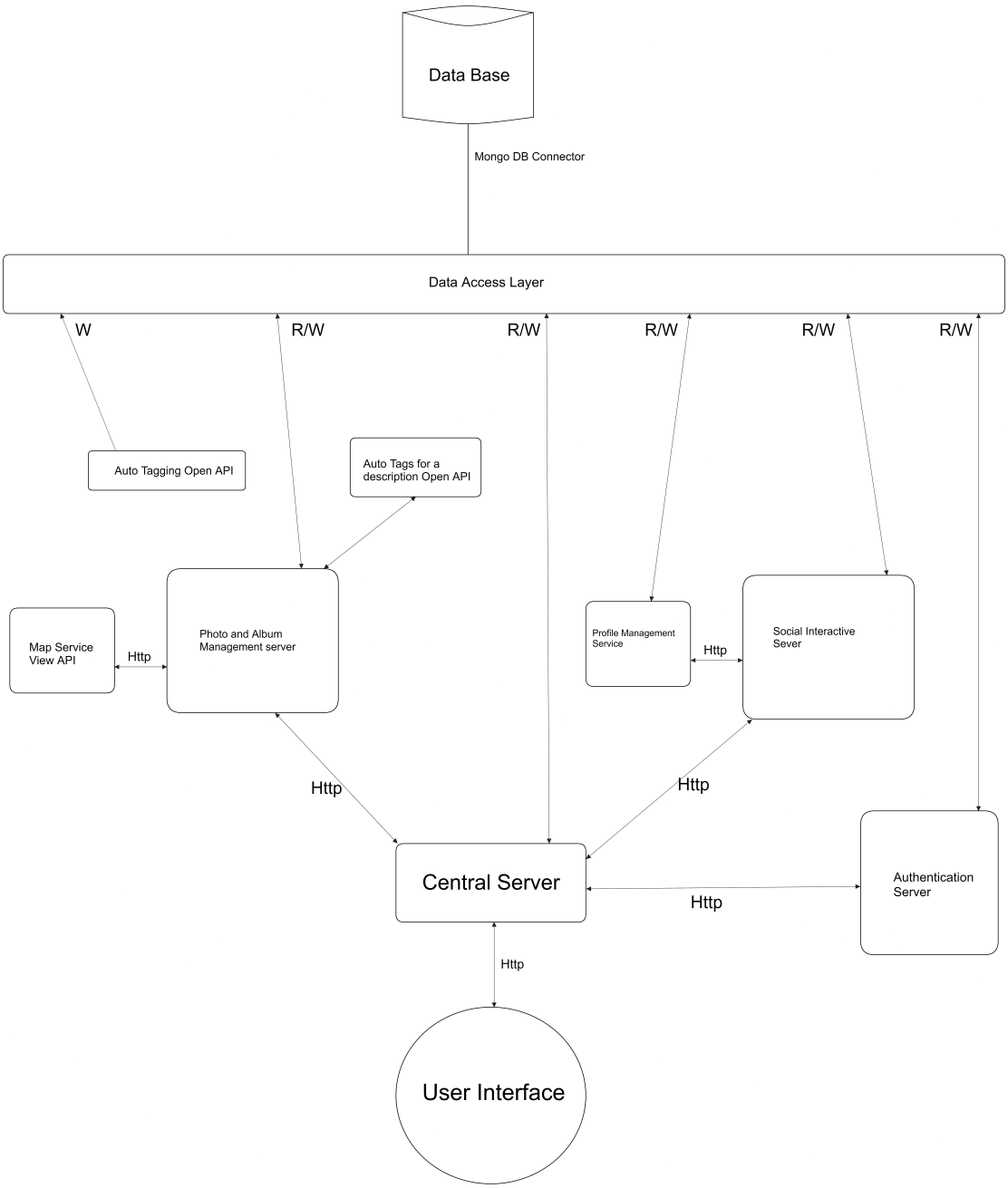


Figure 2.2: Client Server Architecture

# 3 ATAM Analysis

| **Criteria** | **Architecture 1** | **Architecture 2** |
|---|---|---|
| Response Time | High | Relatively Low, because of the central server |
| Scalability | Can be scaled horizontally | Scalability limited to central server |
| Read/Write to data repository | Code reusability | Relatively less code reusability |
| Changes to data repository | Only affects the data layer | Only affects the data layer |
| Data security | Since all the modules are isolated security is enhanced. | Centralization can be the single point of failure, as all the modules communicate with the central server |
| Reading and writing of code for developers | Easy | Hard |
| Folder and Album Management | Managed by a dedicated service | Managed by a dedicated server, which can lead to scalability issues |
| Maintenance and Updates | Easier to add new features due to modularity | Harder because of the central server |
| Debugging | Easier to isolate and address issues in each tier | Debugging can be more complex due to centralization |
| Cost effectiveness | Low | High, because of dedicated servers |

## 3.1 Choice of Architecture:

From the ATAM analysis we have concluded that **3-Tier Architecture** is the best architecture for our application. This architecture shows a very good flexibility for updating the application, and efficient code usage for data management to the data repository and can even address if we change the method of storing the photos, its layered approach also offers a more flexible way for scaling the system to support many users at a time.