Team 20 Project Design

PurePost

Team Members:

Yifan Li, Yi Lin, Bao Phan, Liuqing Yang.

Index

Purpose

With the development of Computer Vision (CV) generative models, deepfake images and videos are becoming more and more common and convincing. The rise of deepfake technology poses significant risks to the authenticity of visual contents on social media platforms (e.g. X/Twitter, Instagram, Facebook), leading to such as misinformation, identity theft, defamation.

The purpose of this project is to resolve this newly arisen issue by creating an innovative and trustworthy social media platform. We aim to develop a social media platform PurePost, with not only excellent recommendation algorithms and user-friendly interface, but also equipped with robust deepfake detection tools, empowering users to identify potential deepfake images and videos.

(a) Functional requirements

I. User Account/Guidelines

1. As a user, I would like to view platform guidelines and privacy policies.
2. As a user, I would like to register an account.
3. As a user, I would like to log in to my account securely.
4. As a user, I would like to manage my user profile.
5. As a user, I would like to reset my password if I forget it.
6. As a user, I would like to verify my account through email.
7. As a user, I would like to sign in using third-party platform accounts (e.g. Google, Apple).
8. As a user, I would like to delete my account.

II. User Privacy/Networking

1. As a user, I would like to manage the visibility of my account, so only approved followers can see my profile and posts.
2. As a user, I would like to see other accounts' profiles and posts with permission.
3. As a user, I would like to block certain accounts.
4. As a user, I would like to manage who can follow me.
5. As a user, I would like to follow others with their permission.
6. As a user, I would like to view my followings and followers.
7. As a user, I would like to filter user groups who can message me.
8. As a user, I would like to send messages to others with their permission.

III. Posts/Contents

1. As a user, I would like to post texts, images and videos.
2. As a user, I would like to add captions and tags for my posts.
3. As a user, I would like to collaborate with others to create joint posts.
4. As a user, I would like to schedule my posts.
5. As a user, I would like to save my posts as drafts when I am not ready to publish them immediately, allowing me to return and complete them later.
6. As a user, I would like to edit or delete my posts after posting.
7. As a user, I would like to manage the visibility of my posts and still be able to change it after publication.
8. As a user, I would like to preview the layout of my post before publishing it.
9. As a user, I would like to pin my best posts to the top of my profile.
10. As a user, I would like to add a content disclaimer to my post such as "Fictional interpretation" or "Notes contain AI-generated content".
11. As a user, I would like the option to mark specific locations when posting.
12. As a user, I want to share my post across multiple platforms.
13. As a user, I want to see detailed analytics for my content (e.g., views, likes, shares).

IV. Interactions

1. As a user, I would like to “like” posts.
2. As a user, I would like to share posts.
3. As a user, I would like to reply to posts.
4. As a user, I would like to control who can comment on my posts.
5. As a user, I would like to view who liked, shared or replied to my posts.
6. As a user, I would like to receive notifications when my posts are liked, shared or commented on.
7. As a user, I would like to be able to manage notifications settings.
8. As a user, I would like to receive notifications when my posts are saved in others’ public folders.
9. As a user, I would like to receive notifications when followed.

V. Content Discovery & Organization

1. As a user, I would like to create folders and save posts.
2. As a user, I would like to customize my folders to be either public or private.

9. As a user, I would like to search for contents, users and tags.
10. As a user, I would like to receive personalized content recommendations.
11. As a user, I would like to see trending contents on the platform.
12. As a user, I would like to avoid certain types of content in my feed.
13. As a user, I would like to dislike notes or authors that do not interest me and receive fewer similar recommendations.
14. As a user, I would like to provide quick feedback on content by labeling it as "Ads" or "Repeated Content" to tailor my experience.
15. As a user, I would like to receive recommendations for tags most relevant to my content and view their usage statistics while editing.
16. As a user, I would like to discover nearby contents from local creators.
17. As a user, I would like to set and adjust my content preferences for the main categories.
18. As a user, I would like to access tips and resources for creating engaging content.

VI. Deepfake Contents Moderation

1. As a user, I would like to report disturbing or deepfake posts.
2. As a user, I would like to receive notifications on the status of my reports.
3. As a user, I would like to see reminders of potential deepfake posts.
4. As a user, I would like to appeal if my content is mistakenly flagged as a deepfake.
5. As a user, I would like to receive notifications on the status of my appeals.

VII. Customization

1. As a user, I would like to have multiple available languages on the platform.
2. As a user, I would like to have translation tools for posts and comments in foreign languages.
3. As a user, I would like to adjust font size and customize layout.
4. As a user, I want to customize my profile theme (e.g., colors, layouts) to better express my personality.

VIII. Feedback

1. As a user, I would like to provide feedback on the platform's features (e.g., surveys, suggestion boxes).

IX. Admin

1. As an admin, I would like to view, disable, or restore user accounts to address violations or errors.
2. As an admin, I would like to review, edit, or delete user-generated content (e.g., posts, images, videos).
3. As an admin, I would like to review reports submitted by users about content, accounts, or messages and take appropriate actions (e.g., warnings, content removal, account suspension).
4. [If time allows] As an admin, I would like to create and publish platform-wide announcements (e.g., new features, policy changes, or known issues).
5. [If time allows] As an admin, I would like to view analytics on user activity, interactions, and reports (e.g., active users, trending content, frequent violations) to make informed decisions about platform management.
6. [If time allows] As an admin, I want to publish transparency reports about report handling to build trust with users.
7. [If time allows] As an admin, I want to use multilingual tools to review content in different languages to effectively moderate a global user base.

(b) Non-functional requirements

1. Our application will load video feed for users within 2 seconds, even during peak usage times.
2. Our application will handle up to 10000 active users simultaneously without performance degradation.
3. Our application will work seamlessly on both iOS and Android devices across multiple screen sizes.
4. Our application will ensure users’ uploaded videos and personal data to be securely stored and encrypted.
5. Our application development will have modular code and clear documentation.

Design Outline

(a) Design Decisions

Our system is built on a client-server architecture, where the front-end application (client) communicates with the backend server through an API gateway. This structure ensures scalability, a clear separation of concerns, and centralized management of business logic.

The backend is designed as a collection of microservices (e.g., Auth Service, Content Moderation, Social Service), each handling a specific function. This approach enhances modularity, maintainability, and allows for independent deployment of services. However, during the initial development phase, our team will focus on building a monolithic version first to avoid over-engineering. This will serve as a foundation before transitioning to a fully microservices-based architecture.

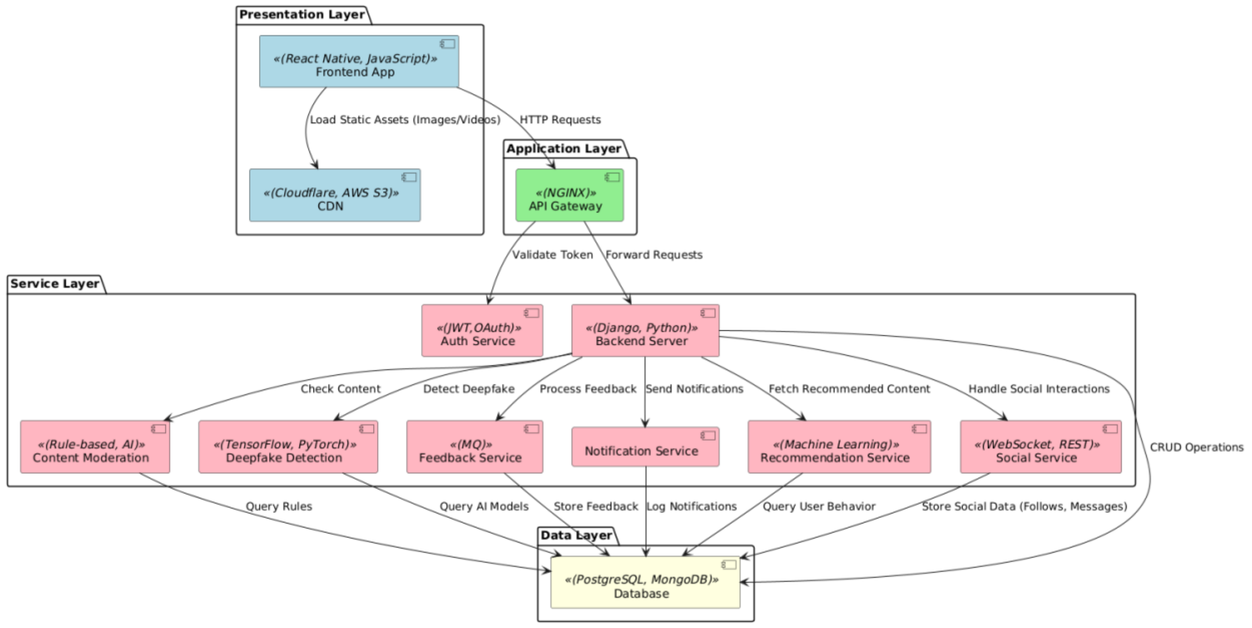
An API Gateway is used as the single entry point for all client requests, providing routing, load balancing, and security (e.g., token validation).

A centralized database is used to store all system data (e.g., user profiles, content, feedback, social interactions). If time allows, we plan to leverage multiple database types, including relational and NoSQL databases, to better suit different use cases. This hybrid approach ensures data consistency and simplifies data management while optimizing performance for specific scenarios.

AI-powered services (Deepfake Detection, Recommendation Service) are integrated into the system as a core function.

(b) Components

The following layer-based UML component diagram outlines our system components and some technologies chosen for several components.


Presentation Layer
«(React Native, JavaScript)» Frontend App
Load Static Assets (Images/Videos)
HTTP Requests
«(Cloudflare, AWS S3)» CDN
Application Layer
«(NGINX)» API Gateway
Validate Token
Forward Requests
Service Layer
«(JWT,OAuth)» Auth Service
«(Django, Python)» Backend Server
Check Content
Detect Deepfake
Process Feedback
Send Notifications
Fetch Recommended Content
Handle Social Interactions
CRUD Operations
«(Rule-based, AI)» Content Moderation
«(TensorFlow, PyTorch)» Deepfake Detection
«(MQ)» Feedback Service
Notification Service
«(Machine Learning)» Recommendation Service
«(WebSocket, REST)» Social Service
Query Rules
Query AI Models
Store Feedback
Log Notifications
Query User Behavior
Store Social Data (Follows, Messages)
Data Layer
«(PostgreSQL, MongoDB)» Database

Frontend App: where people interact with the platform. Users can perform actions such as uploading content, viewing posts, or sending messages.

API Gateway: the central entry point for all client requests. It manages routing, ensures load balancing, and handles security tasks like token validation.

Auth Service: manages user authentication and authorization. It’s responsible for generating and validating tokens to ensure secure access.

Backend Server: The core business logic layer, responsible for processing user requests and coordinating with other services.

Content Moderation: ensures that user-generated content follows the platform’s guidelines by detecting and flagging inappropriate material.

Deepfake Detection: Using AI models, this service identifies deepfake content in uploaded images and videos, helping maintain the platform's integrity.

Feedback Service: Processes and stores user feedback such as ratings, reviews, and comments

Notification Service: sending real-time notifications to users, this service alerts them about updates like new messages or content approvals.

Recommendation Service: Generates personalized content recommendations based on user behavior.

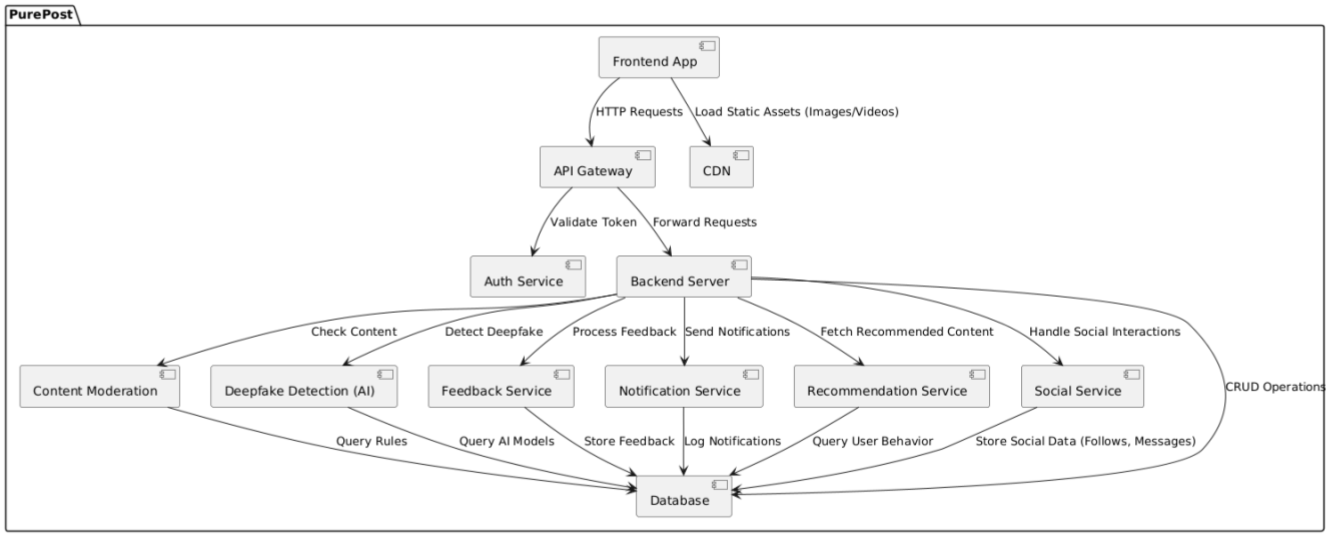
Social Service: Handles user social interactions like following others, sending messages, and managing friend requests.

Database: Stores all system data.

CDN: Optimizes the delivery of static assets (images, videos) to users.

(c) Interactions

Here is a more concise version of our PurePost component diagram.



- **Frontend s**ends HTTP requests to the **API Gateway** and loads static assets from the **CDN**.
- **API Gateway** forwards requests to the appropriate **backend services,** as well as validating user tokens with the **Auth Services**.
- **Auth Service** stores and retrieves user credentials from the **Database**.
- **Backend server** (Django) interacts with **Content Moderation**, **Deepfake Detection**, **Feedback Service**, **Notification Service**, **Recommendation Service**, and **Social Service**.
- **Database** provides data to all **services**

Design Issues

(a) Functional Issues

1. How should deepfake detection be integrated into the platform?
    - Option1: Automatic detection with real-time labels on posts.
    - Option2: Users manually request deepfake analysis on specific posts.
    - Option3: A combination of automatic detection and user-requested analysis.

    Choice: Option 3
    Justification: A hybrid approach balances accuracy and user control. Automatic detection ensures widespread identification, while manual requests allow users to verify specific content that they are uncertain about.

2. How should deepfake detection results be presented?
    - Option1: Simple binary classification (Real / Fake)
    - Option2: Confidence-based scoring (e.g., 0–100%)
    - Option3: Detailed analysis (e.g., highlighting manipulated areas)

    Choice: Option 2&3
    Justification: A confidence score gives users a better understanding of the detection certainty, while explainability features improve transparency and trust in the detection system.

3. What types of media should be supported for analysis?
    - Option1: Images only
    - Option2: Videos only
    - Option3: Both images and videos

    Choice: Option 3
    Justification: Since deepfakes exist in both formats, supporting both image and video analysis ensures comprehensive coverage.

4. Should users be able to challenge or appeal detection results?
    - Option1: No appeals, system decisions are final
    - Option2: Allow users to request a re-evaluation
    - Option3: Allow users to submit evidence to contest a result

    Choice: Option 2&3
    Justification: Users can request re-evaluation to promote fairness and user trust in the detection system.

5. How should flagged deepfake content be handled?

- Option1: Automatically remove flagged content
- Option2: Label flagged posts but allow users to view them
- Option3: Temporarily restrict flagged posts until review is complete.

Choice: Option 3
Justification: Temporarily restricting flagged posts prevents the spread of misinformation while ensuring fairness by allowing users to appeal before permanent removal.

6. What approach should be used for content moderation?
   - Option1: Fully automated moderation using AI-based filtering
   - Option2: A human moderation team manually reviews flagged content
   - Option3: A combination of AI-based detection and human review for flagged cases

   Choice: Option 3
   Justification: AI-based filtering ensures speed, but human review is necessary to prevent false positives and address nuanced cases, improving fairness and accuracy.

7. How should personalized content recommendations be generated?
   - Option1: Based on user interactions (likes, shares, follows)
   - Option2: Based on AI-driven behavioral analysis.
   - Option3: A combination of interaction-based and AI-driven recommendations

   Choice: Option 3
   Justification: Combining direct user interactions with AI-driven insights allows for more relevant and engaging recommendations while maintaining transparency and user control.

(b) Non-Functional Issues

8. What database architecture should be used?
   - Option1: Relational Database
   - Option2: NoSQL Database

   Choice: Option 1
   Justification: Since our platform requires structured data with relationships, relational database is a better choice. It ensures data integrity, ACID compliance, and supports advanced queries for content moderation. We can integrate NoSQL databases later if needed for handling AI-related metadata.

9. What backend language/framework should we use?
   - Option1: Django (Python)
   - Option2: Flask
   - Option3: FastAPI

   Choice: Option 1
   Justification: Django offers a full-featured framework with built-in authentication, admin panel, and ORM, speeding up development. It is also widely used by large-scale platforms like Instagram and Reddit. While Flask and FastAPI offer more lightweight alternatives, Django's scalability and maintainability make it the preferred choice.

10. What frontend language/framework should we use?
    - Option1: React Native
    - Option2: Flutter
    - Option3: Swift for iOS, Kotlin for Android

    Choice: Option 1
    Justification: React Native allows to maintain a single codebase for both iOS and Android, reducing development time and effort. It has a large developer community, strong support for UI components, and good performance. While native development provides better optimization, it is not feasible given our team size.

11. How should system scalability be handled?
    - Option1: Monolithic architecture with Django
    - Option2: Microservices-based architecture with separate AI and content services
    - Option3: Serverless architecture using cloud-based functions for AI and moderation

    Choice: Option 1

Justification: We will start with a monolithic architecture to simplify development and deployment. Django provides built-in features for handling authentication, routing, and database management. As the platform scales, we can migrate AI services and high-load components to a microservices-based architecture.

12. How should media content be stored?
    - Option1: Self-hosted storage
    - Option2: AWS S3
    - Option3: Google Cloud

    Choice: Option 2
    Justification: AWS S3 is a scalable and cost-effective solution for storing images and videos. Self-hosting storage would require additional maintenance and security measures, making it less practical at this stage.

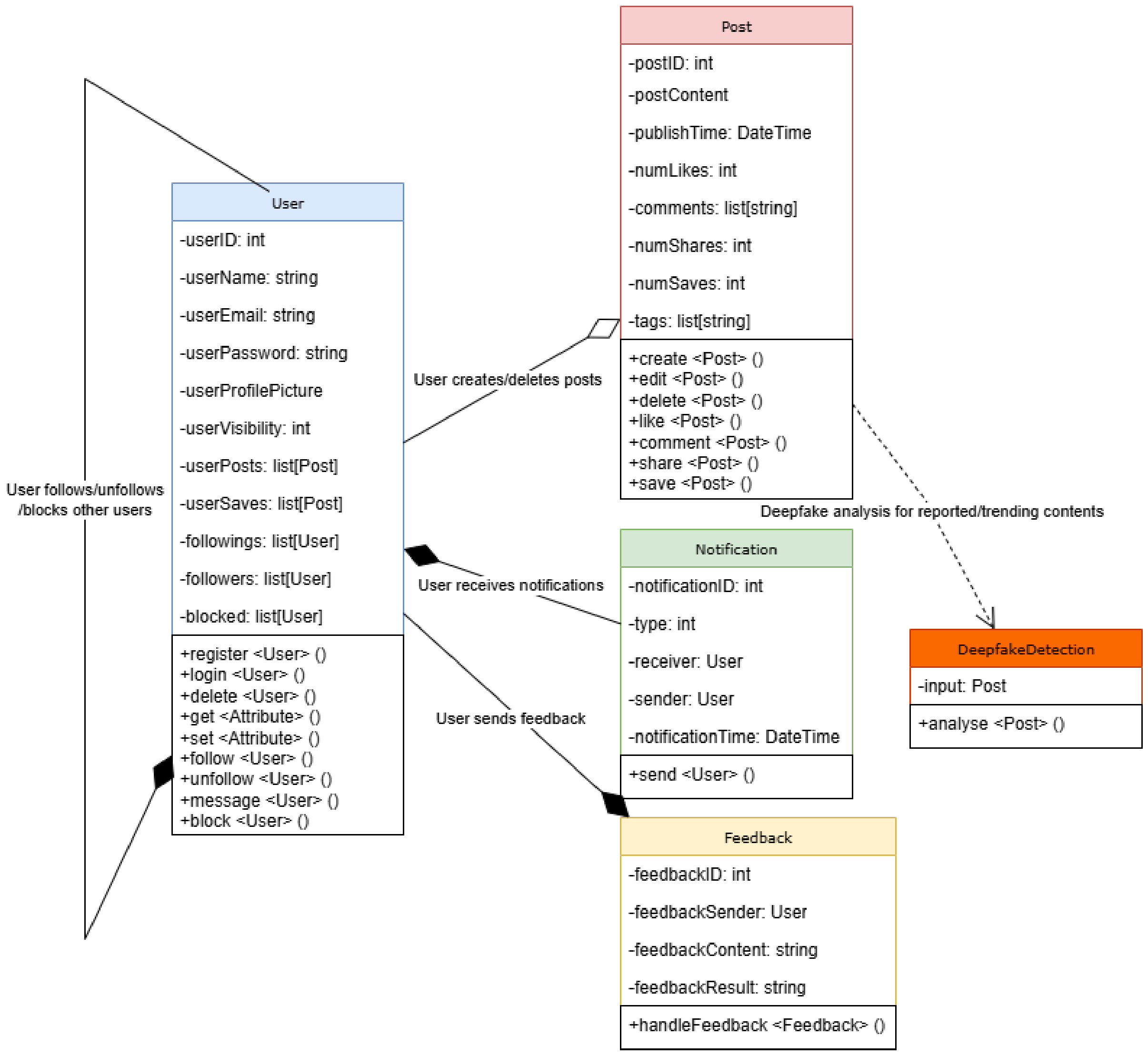
13. How should system performance be optimized for video processing?
    - Option1: Cloud-based processing with scalable resources
    - Option2: On-device processing to reduce server load
    - Option3: Hybrid processing with lightweight on-device filtering and cloud-based deepfake analysis

    Choice: Option 3
    Justification: Hybrid processing minimizes server strain by offloading simple tasks to devices while ensuring deepfake detection benefits from powerful cloud-based resources.

Design Details

(a) Class-level design

Here is a diagram of our class-level design for our platform.



(b) Class Descriptions and Interactions

Each class has a list of attributes representing the characteristics owned by each object, and the interactions between these classes define the overall system behavior.

We will first present the descriptions of each class.

I. User

1. A User object is created when someone signs up on the platform.
2. Each user is assigned a unique user ID.
3. A user's account stores credentials such as username (of their choice), email, and a password.
4. Each user can follow, unfollow or block other users.
5. The user profile includes additional information such as profile picture, followings, followers.
6. Each user can create posts, and these posts appear on the user's homepage.
7. Each user can create folders to save posts.
8. Users can control the visibility of the aforementioned information (number 5-7).
9. A User object is deleted when the user deletes the account.

II. Post

1. A Post object is created when a user submits it.
2. Each post is assigned a unique post ID.
3. Each post has a publishing time.
4. A post might contain texts, images and videos.
5. The uploader can add tags for the post.
6. A post might contain comments.
7. A post maintains its number of likes, shares and saves.
8. A Post object is deleted when the uploader deletes it.

III. Notification

1. A Notification object is created when a user gets likes, shares, saves, comments, messages or follows.
2. Each notification is assigned a unique notification ID.
3. Each notification has a notification time.

IV. Feedback

1. A Feedback object is created when a user submits it.
2. Each feedback is assigned a unique feedback ID.
3. Each feedback contains sender and feedback content.
4. The user receives feedback result after feedback is processed.

V. DeepfakeDetection

1. Deepfake detection is initiated when receiving multiple user reports/observing trending contents.

We will then discuss the interactions between classes.

I. When a user creates a post, that post is linked to the user's account.

II. Users receive notifications when their posts are liked/shared/saved/commented.

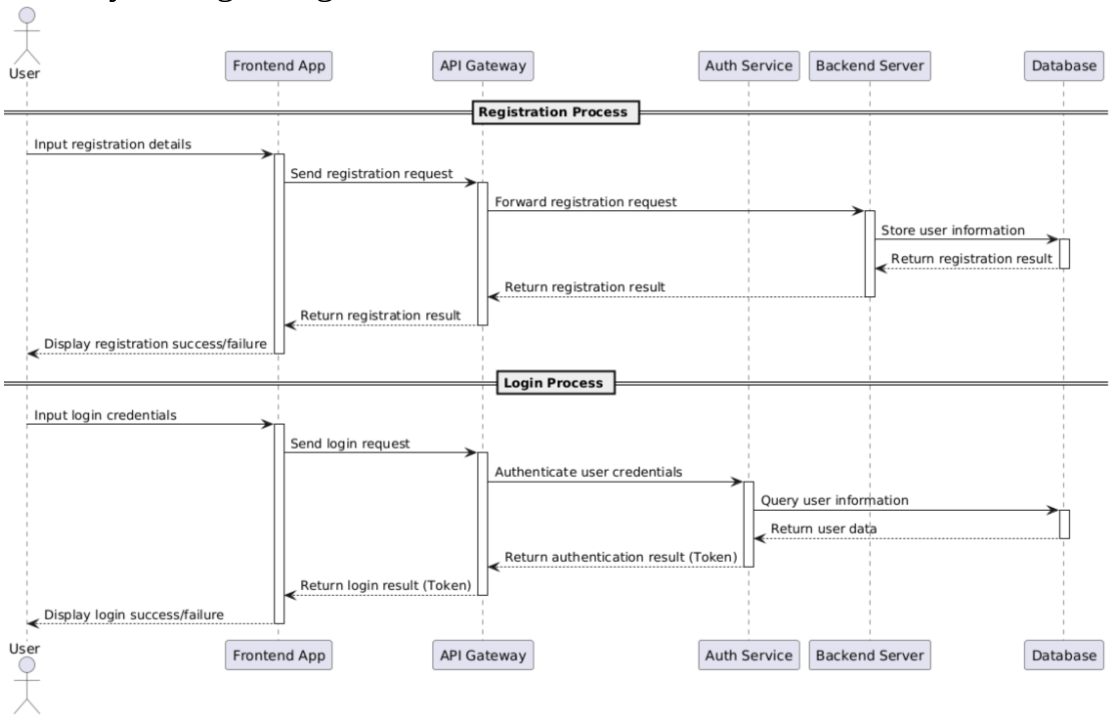
III. Users receive notifications when they get private messages or followed by other users with public following lists.

IV. The Deepfake Detector analyzes the media and returns a detection result, which is then used to update the Post's deepfake reminder.

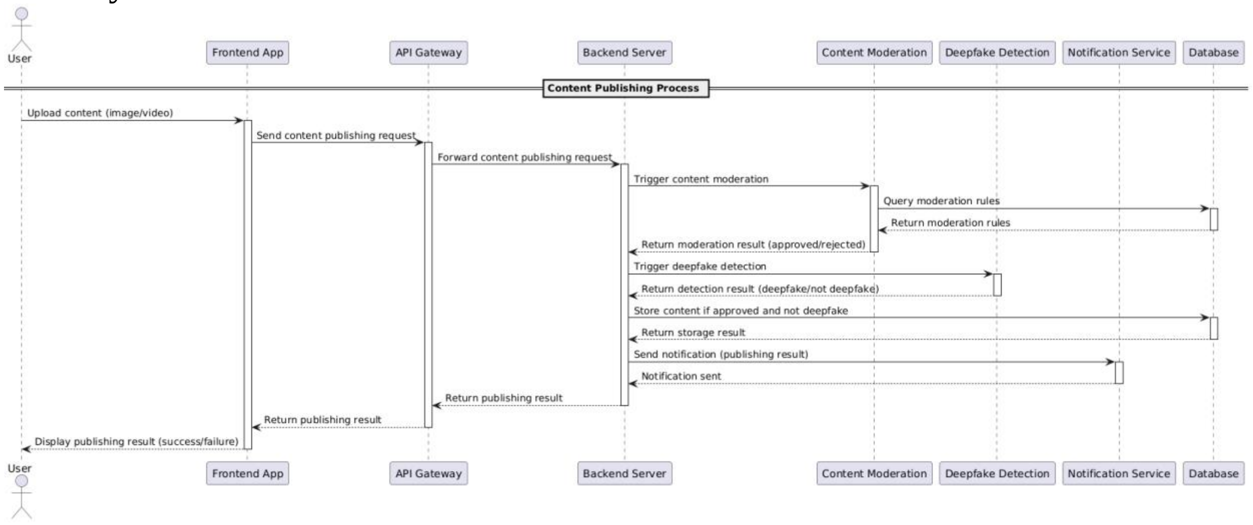
(c) Sequence Diagram

The following is a list of UML Sequence Diagrams, describing some of the usual user activities in our APP.

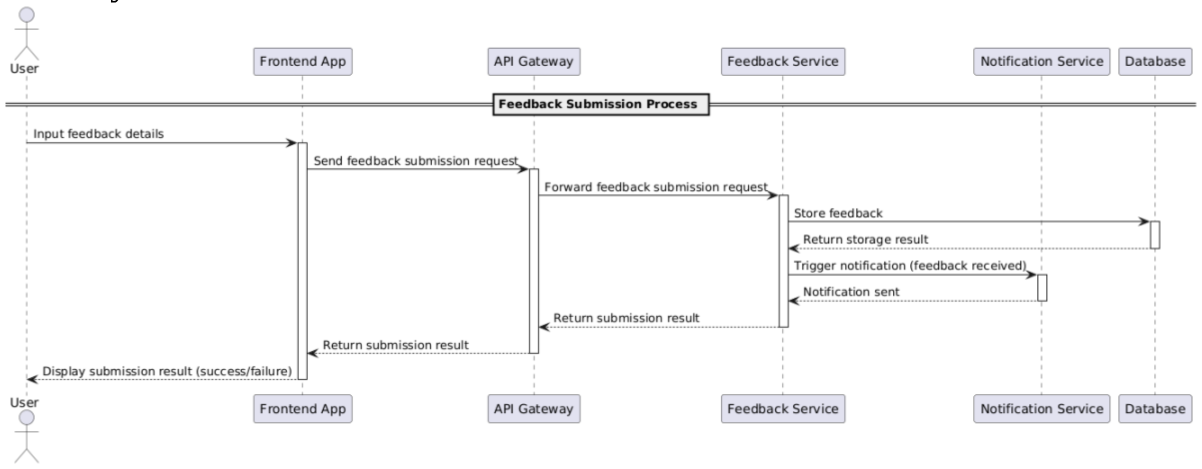
Activity 1: Login/Registration


User
Frontend App
API Gateway
Auth Service
Backend Server
Database
Registration Process
Input registration details
Send registration request
Forward registration request
Store user information
Return registration result
Return registration result
Return registration result
Display registration success/failure
Login Process
Input login credentials
Send login request
Authenticate user credentials
Query user information
Return user data
Return authentication result (Token)
Return login result (Token)
Display login success/failure

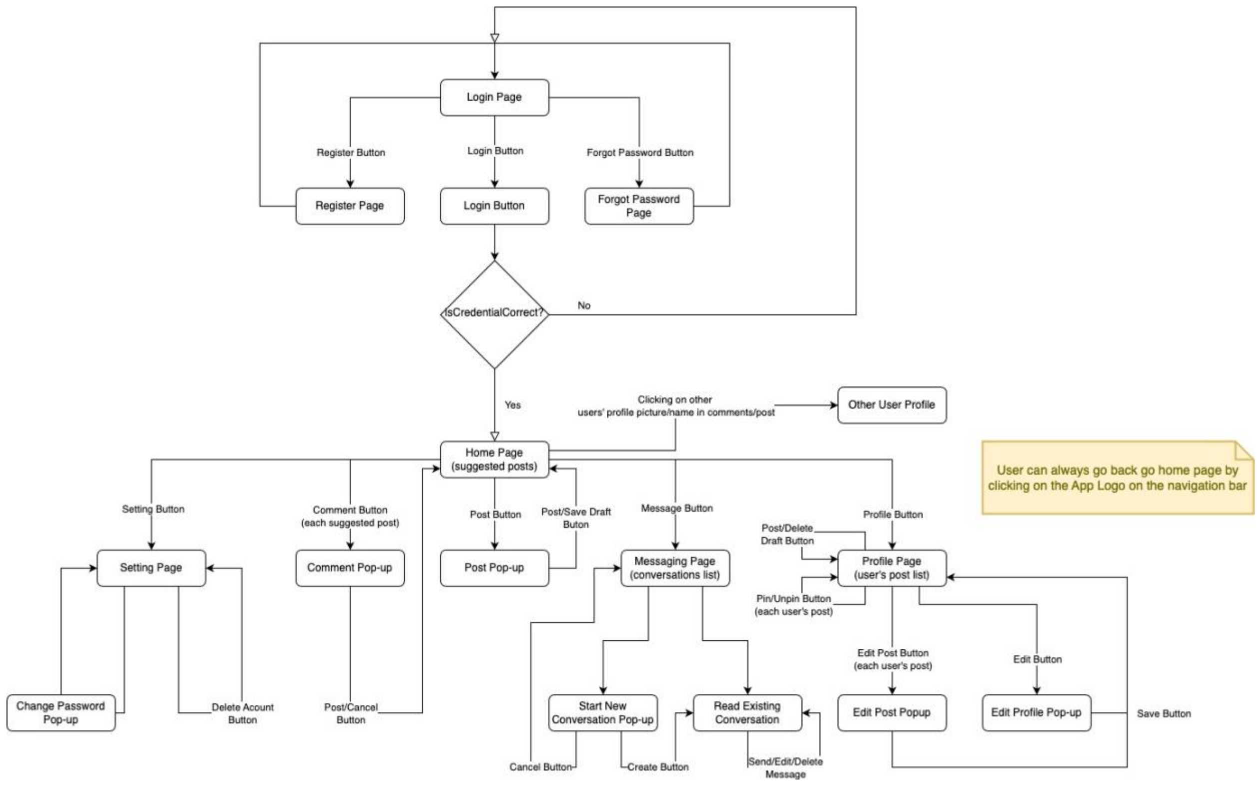
Activity 2: Post new content


User
Frontend App
API Gateway
Backend Server
Content Moderation
Deepfake Detection
Notification Service
Database
Content Publishing Process
Upload content (image/video)
Send content publishing request
Forward content publishing request
Trigger content moderation
Query moderation rules
Return moderation rules
Return moderation result (approved/rejected)
Trigger deepfake detection
Return detection result (deepfake/not deepfake)
Store content if approved and not deepfake
Return storage result
Send notification (publishing result)
Notification sent
Return publishing result
Return publishing result
Display publishing result (success/failure)

Activity 3: Users send feedback

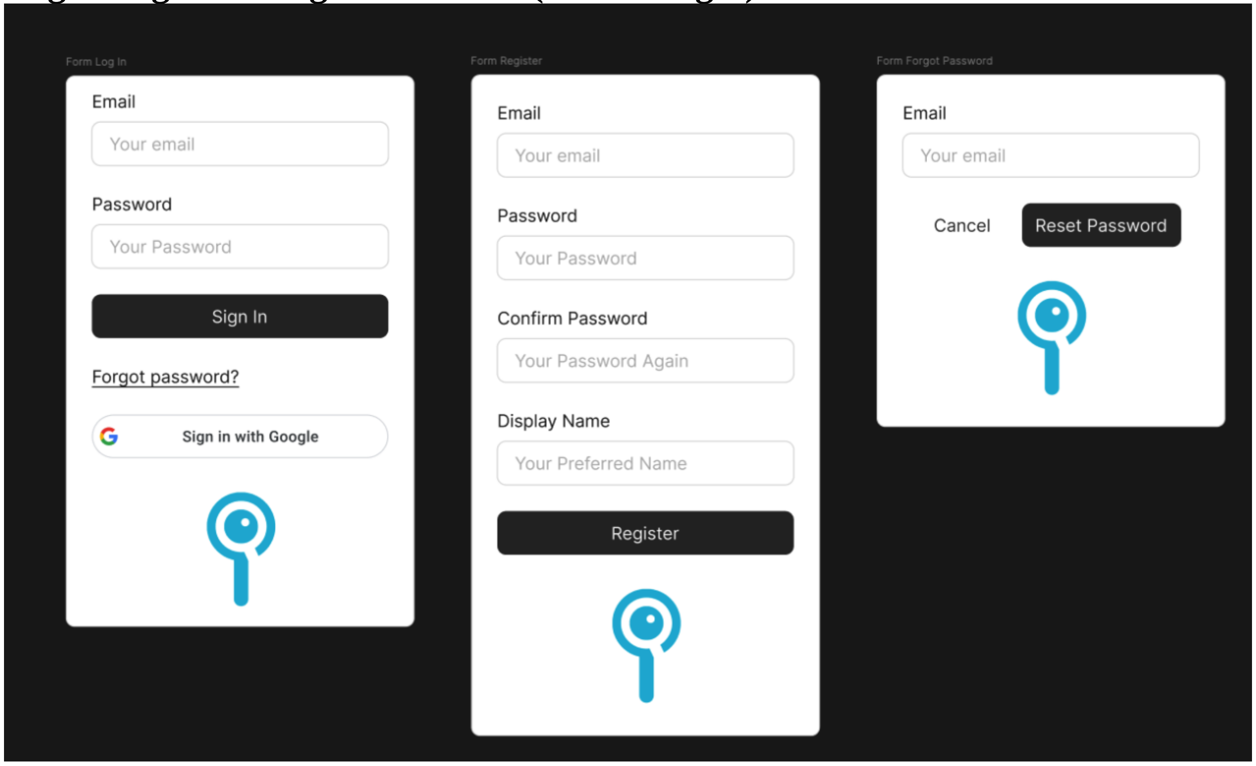


(d) Activity Diagram

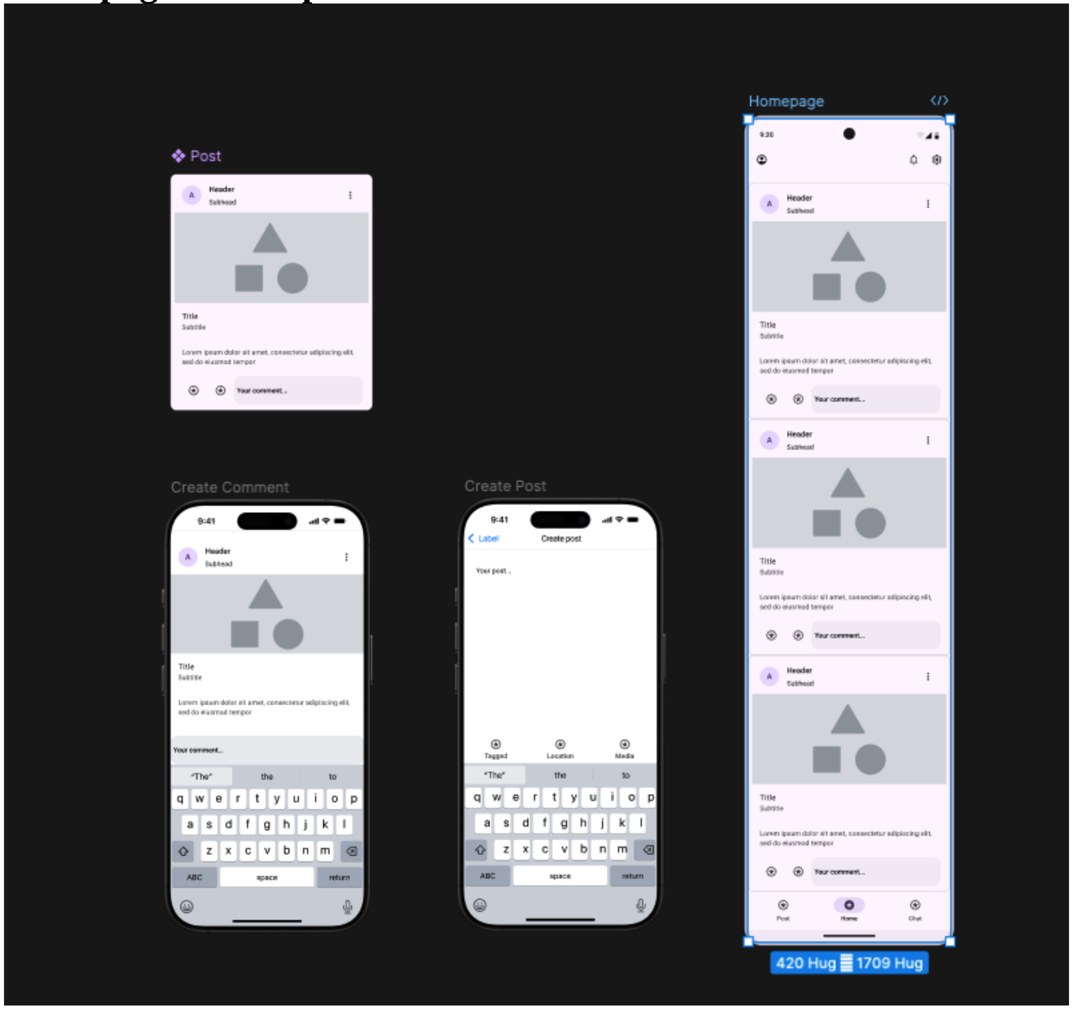


(e) **UI mockup (Figma)**

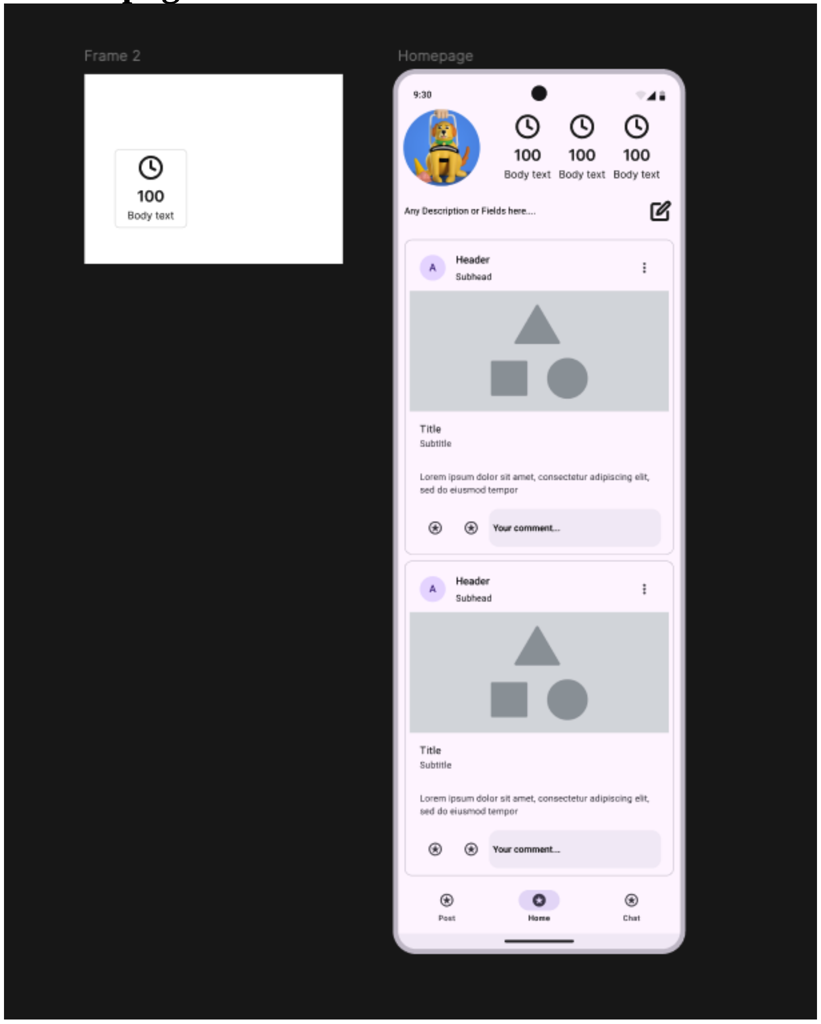
Login/Register/Forgot Password (Public Pages)



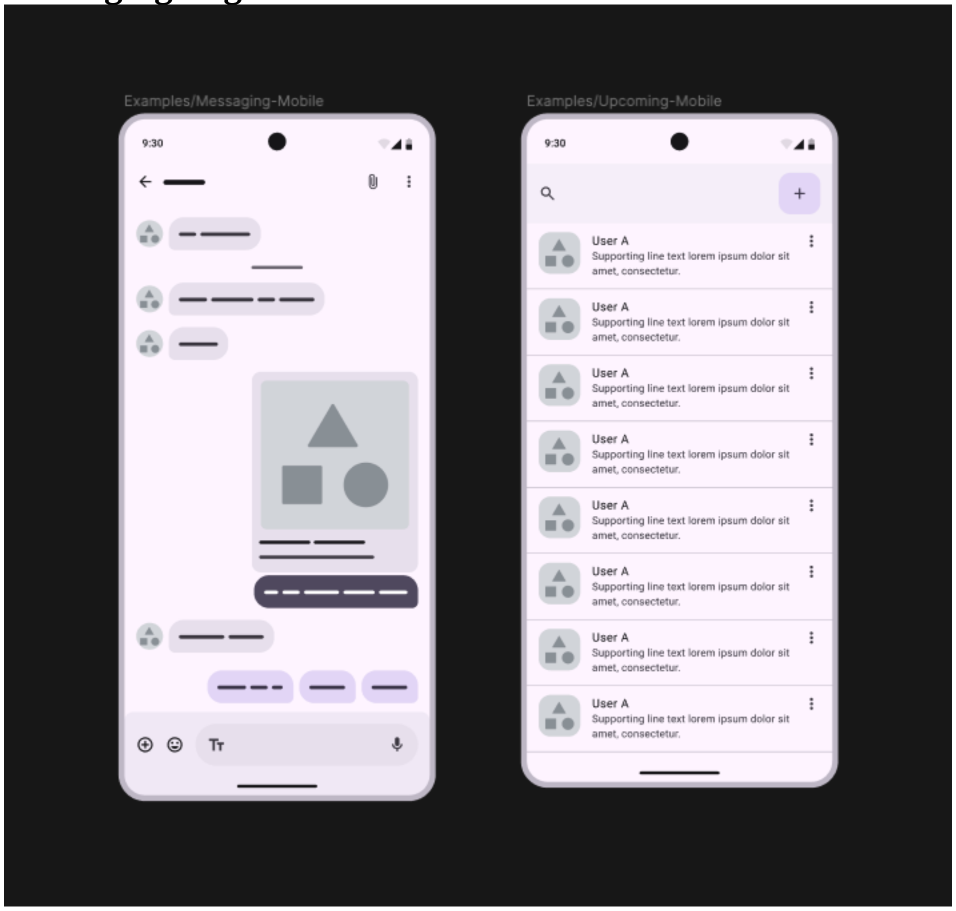
Homepage/Create post/Create comment



Profile page


Frame 2
100
Body text
Homepage
9:30
100
Body text
100
Body text
100
Body text
Any Description or Fields here....
A
Header
Subhead
Title
Subtitle
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
Your comment...
A
Header
Subhead
Title
Subtitle
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
Your comment...
Post
Home
Chat

Messaging Page


Examples/Messaging-Mobile
9:30
Tт
Examples/Upcoming-Mobile
9:30
User A
Supporting line text lorem ipsum dolor sit amet, consectetur.
User A
Supporting line text lorem ipsum dolor sit amet, consectetur.
User A
Supporting line text lorem ipsum dolor sit amet, consectetur.
User A
Supporting line text lorem ipsum dolor sit amet, consectetur.
User A
Supporting line text lorem ipsum dolor sit amet, consectetur.
User A
Supporting line text lorem ipsum dolor sit amet, consectetur.
User A
Supporting line text lorem ipsum dolor sit amet, consectetur.
User A
Supporting line text lorem ipsum dolor sit amet, consectetur.