INSTAGRAM:

1. Requirements

1. Users want to upload photos and videos from their phone gallery and apply filters before sharing.

2. Introducing Stories that disappear after 24 hours to increase user engagement.

3. The app must handle massive traffic spikes, like on New Year, without crashing.

4. Users should be able to set their profiles to private, allowing only approved followers to view their posts.

5. Direct Messaging feature is needed so users can send private messages, photos, and videos to friends.

2. Analysis

1. Evaluating if real-time filters can be applied to photos quickly without slowing down the app.

2. Dividing the app into modules like user management, media processing, and notifications for separate development.

3. Identifying potential risks such as data breaches and planning security measures like two-factor authentication.

4. Confirming the photo-to-posting process should be simple and quick, ideally not taking more than three steps.

5. Documenting requirements clearly, specifying support for various media formats like JPG, PNG, and MP4.

3. Design

1. Designing a scalable architecture using microservices for different parts of the app like photo uploads and user profiles.

2. Creating a seamless user interface with features like infinite scroll on the home feed for continuous content viewing.

3. Designing the database to efficiently store user data, including tables for profiles, posts, comments, and likes.

4. Detailing how images will be compressed and stored to optimize space and performance.

5. Ensuring the search bar retrieves relevant results like users, hashtags, and locations quickly.

4. Implementation

1. Developing the photo upload feature first, followed by Stories.

2. Writing backend code to handle user authentication and profile management securely.

3. Integrating the photo upload function with user profiles to ensure new posts appear immediately.

4. Conducting code reviews for features like photo editing to ensure filters are applied correctly.

5. Using GitHub for version control, with branches for different features like Direct Messaging and merging them after testing.

5. Testing

1. Testing individual features to ensure photo filters work correctly on various types of photos.

2. Ensuring combined features work well together, such as verifying that uploading a photo updates both the user's feed and profile correctly.

3. Simulating user journeys from account creation to posting and interacting with content to check overall functionality.

4. Testing app performance under high load by simulating thousands of users uploading photos simultaneously.

5. Conducting a beta test with selected users to gather feedback on usability and performance before the official release.

Requirements Specification

This involves gathering and documenting what the users and stakeholders need from the system. For example, Instagram users want to upload photos, apply filters, and share Stories that disappear after 24 hours.

Architectural, Component, & Detailed Designs

Creating high-level architecture, breaking down the system into components, and detailing each component's design. For instance, designing Instagram's scalable microservices architecture to handle user profiles, photo uploads, and notifications separately.

Implementation

Developers write the actual code based on the design specifications. For example, coding Instagram's photo upload feature to allow users to apply filters and post images.

Unit, Integration, and Acceptance Testing

Testing ensures that individual components (unit testing), combined parts (integration testing), and the overall system (acceptance testing) work correctly. For instance, testing Instagram's filter application feature individually and then verifying it works with the photo upload and sharing functionalities.

Installation and Maintenance

Deploying the system to the user's environment and ensuring it runs smoothly. For example, launching Instagram on app stores and providing regular updates and bug fixes based on user feedback.