21Summer

Jamii App

Samuel Gachunga

Runtime elements

Handlers and Middleware

Feed Generation Middleware: Recommender System

Purpose: Generates personalized feeds for users based on their preferences, behavior, and location.

Functions:

- Analyze user interactions and preferences.
- Fetch relevant content from the database.
- Apply filtering and sorting algorithms.

Direct Messaging Handler

Functions:

- Send: Handles sending messages between users.
- Receive: Handles receiving messages and updating user inboxes.

Friend Follow Relationship Handler

Functions:

- Follow: Manages follow requests, updates follower and following counts.
- Unfollow: Manages unfollow requests, updates follower and following counts.

Media Upload Middleware

Purpose: Facilitates the uploading of media content (images, videos) by users.

Functions:

- Handle file uploads.
- Validate file types and sizes.
- Store files in cloud storage.
- Generate and store URLs for media access.

Session and State Management

Session management

- **User Session Management:** Tracks active user sessions, handles login/logout, and session timeouts.
- **Online Status Management:** Monitors and updates users' online/offline statuses.

User Current Location on the Site

Purpose: Tracks and manages the current location or view of the user within the app.
Functions:

- Update user interface based on user navigation.
- Track likes, follows, and other interactions.

Scheduler and Dispatchers

User Feed Refresh

Purpose: Periodically refreshes the user feed to display the latest content.
Mechanism: Uses timed intervals or event-based triggers to refresh content.

Promotional Email Dispatcher

Purpose: Sends promotional emails and messages via various platforms.
Platforms: WhatsApp, Gmail, Facebook, X (formerly Twitter).

Notification Dispatcher

Purpose: Sends notifications to users based on certain events.
Events: Likes, comments, follows, orders, etc.

Content Cache Refresh Scheduler

Purpose: Regularly updates cached content to ensure it is current.
Mechanism: Uses timed intervals to clear and refresh cache.

Management of Resources

CPU

Optimization Algorithms and Code Efficiency: Implement algorithms and coding practices to maximize CPU efficiency.
Background Processing for Non-Urgent Tasks: Offload non-critical tasks to background processes.
Lazy Loading for Content: Load content as needed rather than all at once.
Caching Mechanisms: Use caching to reduce redundant processing.

RAM

Memory Management: Optimize the allocation and use of RAM.
Object Pooling: Reuse frequently created and destroyed objects to save memory.
Optimize Data Structures: Use efficient data structures to minimize memory usage.
Limit Number of Concurrent Operations: Prevent memory overload by limiting concurrent tasks.

Storage

Data Compression: Compress data to save storage space.
Cloud Storage: Use cloud storage solutions for scalable data storage.
Regular Cache and Temporary Files Clearing: Periodically clear cache and temporary files to free up space.

Battery
Minimize Background Processes: Reduce background activities to conserve battery.
Optimize Network Requests: Efficiently manage network requests to save power.
Use Push Notifications: Use push notifications instead of constant polling to conserve battery.
Dark Mode and Power Saving Features: Implement dark mode and other power-saving options.

Money

Optimize Database Queries: Write efficient queries to minimize database costs.
Hybrid Cloud: Use a combination of public and private cloud solutions for cost-effectiveness.

Data

Compress Data Before Transmission: Compress data to reduce bandwidth usage.
Efficient Data Syncing Mechanism: Implement efficient data syncing to minimize data transfer.
Lazy Loading: Load data as needed to reduce initial load times.

Binding Time

Flutter Data Binding

Technologies: Riverpod, Provider.
Purpose: Manage state and data binding efficiently within the app.

Define Compile-Time Constants

Purpose: Use compile-time constants for better performance and consistency.

Firebase Security Rules at Deploy Time

Purpose: Apply security rules during deployment to ensure data integrity and security.

Dependency Injection

Purpose: Manage dependencies effectively, making the codebase more modular and testable.

Decorators in Aspect-Oriented Programming for Python Cloud Functions

Purpose: Use decorators to add functionality to cloud functions in a clean and modular way.

Asynchronous Programming for I/O Bound Operations
Purpose: Implement asynchronous programming to handle I/O operations efficiently, improving app responsiveness.