CHAT APP: REGISTRATION AND LOGIN WORKFLOW WITH DATABASE

INTRODUCTION TO USER REGISTRATION AND LOGIN FLOW

This chat application employs a database-only approach to simulate user authentication without relying on external authentication services. The primary objective is to manage user registration and login seamlessly through database operations combined with OTP (One-Time Password) verification for added security.

When a user opens the app for the first time, the system checks if their phone number already exists in the database. If it does, the user is informed that their number is already registered. Otherwise, the user is directed to a profile setup screen where essential personal information such as first name, last name, bio, and profile picture is collected.

After submitting profile details, an OTP is generated and displayed in the terminal for testing purposes. Upon successful OTP verification, the user's data is saved in the database, and they are redirected to the home screen.

Returning users logging in will have their phone number checked against the database. If the number is found, an OTP is generated and must be verified before accessing the home screen. Unregistered users attempting to log in will receive an appropriate notification. This document outlines the entire workflow, including screen transitions and data handling, ensuring a clear understanding of the registration and login process within this application.

USER REGISTRATION FLOW

When a new user accesses the chat application for the first time, the registration process begins by verifying the user’s phone number against the existing records in the database. This initial check is crucial to determine whether the user is already registered or needs to proceed with setting up a new profile.

STEP 1: PHONE NUMBER VERIFICATION

Upon launching the app, the user is prompted to enter their phone number. The system queries the database to see if this phone number is already associated with an existing user account. There are two possible outcomes:

- **Phone Number Found:** If the phone number exists in the database, the app notifies the user that they are already registered. In this scenario, the user is not required to register again and should be guided towards the login process or informed accordingly.
- **Phone Number Not Found:** If the phone number is not present in the database, it indicates that this is a new user. The application then redirects the user to the profile setup screen to collect personal details essential for creating their account.

STEP 2: PROFILE SETUP SCREEN

The profile setup screen serves as the next crucial stage in the registration workflow. Here, the user must provide the following information:

- **First Name:** The user's given name, used to personalize their profile and interactions within the app.
- **Last Name:** The user's family name or surname, supplementing the first name for identification.
- **Bio:** A short description or introduction written by the user to share more about themselves with other chat members.
- **Profile Picture:** An image chosen by the user to represent their account visually. The app provides an option to upload or capture this picture during setup.

These inputs are mandatory to complete the profile setup and ensure a rich user experience by allowing others to recognize and connect with the user effectively.

STEP 3: GENERATING AND DISPLAYING THE OTP

Once the user fills in all required profile details and taps the "Continue" button, the application triggers the OTP (One-Time Password) generation process. This OTP acts as a verification mechanism to confirm the phone number belongs to the user and prevents unauthorized registrations.

For testing purposes during development, instead of sending the OTP via traditional SMS gateways, the generated OTP is displayed directly in the application's terminal or console. This approach simplifies debugging and speeds up the testing cycle without involving external communication services.

```
Example terminal output:
Generated OTP for +1234567890 is 837492
```

STEP 4: OTP VERIFICATION

After receiving the OTP (displayed in the terminal), the user must enter it into the OTP verification screen. The app then:

- Compares the entered OTP with the generated one stored temporarily during the verification session.
- Validates that the OTP matches and is still valid (e.g., within a time limit or not already used).

If the OTP verification succeeds, the registration process continues to the final step. If verification fails, the app should prompt the user to retry or request a new OTP.

STEP 5: STORING USER DETAILS IN THE DATABASE

Upon successful OTP verification, the application commits the new user's profile data to the database. This includes:

- **Phone Number:** The primary identifier for the user account.
- **First Name and Last Name:** Stored as plain text fields.
- **Bio:** An optional descriptive text stored for display in user profiles.
- **Profile Picture:** The uploaded image is either stored directly in the database as a blob or saved in an external storage solution with a reference path saved in the database.

This entire dataset forms the user's account profile, enabling personalized experiences throughout the chat application.

STEP 6: REDIRECTING TO THE HOME SCREEN

Once the user's details have been successfully saved, the app automatically redirects the user to the home screen. The home screen acts as the main

interface for all chat interactions, displaying contacts, conversations, and other key features of the app.

From this point forward, the user remains logged in and does not need to go through registration again unless they uninstall the app or clear the app data. This persistent session ensures quick access and seamless user engagement.

USER LOGIN FLOW

The login process in this chat application is designed to provide a streamlined and secure way for returning users to access their accounts using just their phone number, backed by a lightweight OTP verification as a substitute for traditional authentication methods.

CHECKING USER EXISTENCE IN THE DATABASE

When an existing user attempts to log in, they are prompted to enter their phone number. The application immediately performs a database query to determine if the entered phone number corresponds to a registered user. This verification step is crucial to ensure that only recognized users proceed further into the login flow.

- **User Not Found:** If the phone number is not found in the database, the application informs the user with a clear message stating "User is not registered." This prevents unauthorized access attempts and guides users towards the registration process if needed.
- **User Found:** If the phone number exists in the database, the user is allowed to continue by initiating the OTP generation step, which serves as an additional security layer.

OTP GENERATION AND VERIFICATION

Once the system confirms that the phone number exists, it generates a One-Time Password (OTP). This OTP acts as a temporary access code, providing a simple yet effective means of verifying the user's identity without implementing a full-fledged authentication system.

During development and testing, the generated OTP is displayed directly in the application's terminal rather than being sent via SMS. This facilitates efficient testing and debugging of the OTP process.

```
Example terminal output:
Generated OTP for +1234567890 is 529374
```

After the OTP is generated, the user is redirected to the OTP verification screen where they must enter the received code. The application then compares the provided OTP against the generated one stored temporarily. The verification also ensures that the OTP has not expired and has not been used before.

- **Successful Verification:** If the OTP entered matches the generated code within the valid time frame, the user is authenticated and granted access, resulting in redirection to the home screen.
- **Failed Verification:** If the OTP does not match or expires, the user is prompted to retry the process, including the option to request a new OTP.

SIGNIFICANCE OF OTP IN THE LOGIN WORKFLOW

The use of OTP in this login flow replaces traditional authentication mechanisms such as password checks or third-party services. This approach offers several benefits:

- **Simplicity:** Users verify their identity solely based on possession of the registered phone number, simplifying the user experience.
- **Security:** OTP ensures that access is granted only if the user inputs the correct temporary code, effectively preventing unauthorized logins.
- **No External Dependencies:** By handling OTP generation and verification internally and displaying it in the terminal during development, the system avoids reliance on external SMS gateways and authentication providers.
- **Flexibility for Testing:** Displaying the OTP in the console streamlines the testing phase, allowing developers to validate the workflow efficiently before deploying production-ready SMS functionality.

Overall, the OTP-based login process balances security and usability, enabling quick access for verified users while minimizing complexity.

SESSION PERSISTENCE AND SCREEN NAVIGATION

After a user successfully completes the registration or login process and reaches the home screen, the application ensures that the user remains on

this screen during subsequent app launches, providing a seamless experience without repeated authentication steps. Since the app does not use traditional authentication tokens or session management, this persistence relies primarily on database checks and local data presence.

Specifically, when the app launches, it verifies the existence of the user's phone number in the database. If the phone number is found and the associated profile exists, the application assumes the user is already authenticated and allowed to access the home screen immediately. This approach mimics session persistence by using the database state as the source of truth rather than maintaining separate session tokens or in-memory states.

Conversely, if the user uninstalls the app or clears the application data, any locally stored information (such as cached phone numbers or flags) is lost. Upon reinstalling and launching the app again, the system treats it as a fresh start. Since no locally stored identifiers exist and the app cannot infer a logged-in user from cache, it redirects the user to the registration screen to either sign up or log in.

This logic can be summarized as follows:

- On app launch, check local storage or cache for a stored phone number or login flag.
- If found, verify the phone number’s existence in the database.
- If the user exists, navigate directly to the home screen, skipping registration or login.
- If the user does not exist or local data is missing (e.g., after uninstall), redirect to the registration screen.

Implementing this flow requires careful synchronization between the local data and database state to avoid inconsistencies. Developers can use lightweight local storage (such as SharedPreferences on Android or UserDefaults on iOS) to keep track of the logged-in user's phone number securely and verify it against the database during each launch.

By relying on database queries combined with local presence checks, this workflow maintains a simple yet effective session persistence mechanism that aligns with the database-only authentication model while ensuring users seamlessly remain on the home screen after authentication.

HANDLING PROFILE PICTURES AND ALTERNATIVE STORAGE SOLUTIONS

In the current implementation, profile pictures uploaded during user registration are stored using Firebase Storage. This approach leverages Firebase's scalable infrastructure, enabling efficient upload, retrieval, and management of images separate from the primary database. Firebase Storage simplifies handling large binary files and reduces database load, improving overall app performance.

However, depending on project requirements and resource constraints, an alternative strategy might be considered: storing profile pictures directly within the database as binary large objects (BLOBs) or base64-encoded strings. This method consolidates user data in one place and removes dependency on an external storage service.

COMPARING STORAGE OPTIONS

- **Firebase Storage:**
  - Pros: Scalable, optimized for media, supports CDN delivery, easy integration with Firebase services.
  - Cons: Requires additional setup and network calls; potential cost implications as storage scales.
- **Database Storage (BLOB/Base64):**
  - Pros: Simplifies data management by keeping all user info centralized, no external dependencies.
  - Cons: Can bloat the database size, potentially degrading query performance and increasing backup complexity.

Choosing the alternative approach could benefit smaller applications or scenarios with limited external services, where simplicity and reduced infrastructure dependencies are priorities. This method integrates naturally with the existing registration and login flows by saving the profile picture alongside other user details after OTP verification.

Ultimately, the storage decision depends on balancing scalability, performance, and operational complexity. Developers should consider anticipated user volume, image size, and cost constraints when selecting the appropriate method for managing profile images in the chat application.