# Bible App – Epic 3: User Profiles & Personalization

This document describes the implementation plan for Epic 3: User Profiles & Personalization. It focuses on using phone numbers as user identity, with SMS-based 6-digit verification codes. The approach enhances user experience and security for managing preferences and notes in the Bible app.

## 1️⃣ Overview

This implementation guide outlines how to build a secure and personalized user experience that includes identity management, note-taking, and reading preferences. User identity is based on phone numbers, verified through SMS, replacing traditional email-based registration.

## 2️⃣ User Stories & Acceptance Criteria

• User Story 3.1: As a user, I want to create an account using my phone number and verify it with a code, so that my preferences and notes are saved securely.

✅ Acceptance Criteria:

- Phone number entry form

- SMS-based 6-digit verification code

- Device verification and secure sign-up/login

- Secure session handling (e.g., JWT)

• User Story 3.2: As a user, I want to save notes on specific Bible verses or chapters so that I can reference them later.

✅ Acceptance Criteria:

- Notes linked to specific verses/chapters

- Ability to create, edit, or delete notes

- Notes tied to the authenticated user

• User Story 3.3: As a user, I want to personalize my reading experience (font, color themes, etc.) so that it’s comfortable for me.

✅ Acceptance Criteria:

- User profile settings for customization

- Settings persist across sessions

## 3️⃣ IAM Implementation Details

The identity management system will use phone numbers as primary user identifiers. Users verify their device through an SMS-based 6-digit code sent to their phone. The flow is as follows:

1. User enters phone number.

2. Server generates a 6-digit code and sends it via SMS.

3. User enters the code in the app.

4. Server verifies the code:

- If correct, create an account (if new) and generate a JWT for the session.

- If incorrect or expired, reject the login attempt.

SMS messages will be sent using an SMS gateway service (e.g., Twilio, Nexmo, Firebase Auth). Verification codes will be short-lived (5–10 minutes) and stored securely.

## 4️⃣ API Endpoint Definitions

• POST /api/auth/request-code – Send SMS verification code

• POST /api/auth/verify-code – Verify code and authenticate user

• GET /api/user/profile – Fetch user profile (secure)

• PATCH /api/user/profile – Update user profile (secure)

• POST /api/notes – Create note (secure)

• GET /api/notes – Fetch user’s notes (secure)

• PATCH /api/notes/:id – Edit note (secure)

• DELETE /api/notes/:id – Delete note (secure)

## 5️⃣ Database Schema Overview

• users table:

- id (UUID)

- phone\_number (string, unique)

- preferences (JSON)

• verification\_codes table:

- id

- phone\_number

- code

- expires\_at

• notes table:

- id

- user\_id

- verse\_reference

- content

## 6️⃣ Frontend Considerations

The frontend will include forms for phone number input, code verification, and user customization. It will securely store and use the session token to access protected APIs.

## 7️⃣ Security & Best Practices

✅ HTTPS for all traffic

✅ Rate-limiting for SMS verification code requests

✅ Short-lived codes (5–10 min)

✅ Avoid storing codes in plain text

✅ Minimal data collection for privacy compliance (e.g., GDPR, CCPA)

## 8️⃣ Next Steps

1. Choose and configure SMS provider (Twilio, Firebase Auth).

2. Implement backend API endpoints and database tables.

3. Build frontend UI for phone-based login and user customization.

4. Conduct end-to-end testing of phone-based authentication.

5. Deploy IAM system in production.