# Product Backlog
# Islamic Social Media and Spiritual Utility Application

## Overview

**Tech Stack:** React (Frontend), Node.js/Express (Backend), MongoDB (Database).

This document lists the product backlog organized into Epics, User Stories, and Tasks. Each User Story has an explicit Priority (High, Medium, Low).

## Epic 1: Core User & Infrastructure

### US 1.1 – User Registration & Login (Priority: High)

**User Story:** As a new or returning user, I want to sign up, log in, and log out securely so that I can access my personalized Islamic content and features.

**Tasks:**

1.1.1 Design auth flows and UX:
- Define screens: Signup, Login, Logout confirmation, Email verification, Forgot/Reset Password.
- Create wireframes and basic UI mockups.

1.1.2 Implement backend authentication APIs (Node.js):
- POST `/auth/signup`, POST `/auth/login`, POST `/auth/logout`.
- Store users in MongoDB with hashed passwords (bcrypt/argon2).
- Implement JWT-based auth (access + refresh tokens).

1.1.3 Implement email verification and password reset backend:
- Generate email verification and password reset tokens with expiry.
- Endpoints: GET `/auth/verify-email/:token`, POST `/auth/forgot-password`, POST `/auth/reset-password`.
- Integrate email service (e.g., SendGrid/SES) with templates.

1.1.4 Frontend integration for auth:
- React pages: Signup, Login, Forgot Password, Reset Password, Email Verified.
- Client-side validation for email and password strength.
- Connect to auth APIs and manage tokens securely.
- Implement global auth context/hook for session management.

1.1.5 Security & validation:
- Enforce password rules and rate-limit login attempts.
- Input validation & sanitization.
- Basic protections against XSS, CSRF (strategy), and brute force.

### US 1.2 – User Profile Management (Priority: High)

**User Story:** As a user, I want to view and edit my profile and control privacy settings so that I can manage my identity and visibility in the community.

**Tasks:**

1.2.1 Define profile data model in MongoDB:

- Fields: name, username, email, avatar, gender (optional), country, city, language, madhhab, bio, interests/tags, notification preferences, privacy level.

1.2.2 Implement backend profile APIs:
- GET `/users/me`, PATCH `/users/me`.
- GET `/users/:username` for public profile (respect privacy).
- PATCH `/users/me/privacy` for privacy settings.

1.2.3 Implement frontend profile UI:
- Profile page (self view) with editable sections.
- Public profile page for other users.
- Privacy settings screen.
- Avatar preview on client side.

1.2.4 Implement avatar upload:
- File upload endpoint `/users/me/avatar`.
- Integrate with object storage (e.g., S3) or local dev storage.
- Resize/compress images on backend or via service.

### US 1.3 – Basic CRUD APIs for Content Types (Priority: High)

**User Story:** As a backend developer, I want standardized CRUD APIs for all core content types so that the frontend can manage and display content consistently.

**Tasks:**

1.3.1 Define core content models in MongoDB:
- Post, Channel, IslamicContent, Deed, DailyItem with appropriate fields and references.

1.3.2 Implement generic CRUD services and controllers:
- Reusable service layer with validation.
- Standardized error handling and responses.

1.3.3 Implement REST APIs for Posts:
- POST `/posts`, GET `/posts`, GET `/posts/:id`.
- PATCH `/posts/:id`, DELETE `/posts/:id` with author/admin checks.

1.3.4 Implement REST APIs for Channels:
- POST `/channels`, GET `/channels`, GET `/channels/:id`.
- PATCH `/channels/:id`, DELETE `/channels/:id` (admin only, soft delete).

1.3.5 Implement REST APIs for IslamicContent:
- GET `/content` with filters.
- GET `/content/:id`.
- Prepare admin CRUD for future content management.

1.3.6 Implement pagination, sorting, and filtering:
- Common query params: `page`, `limit`, `sort`, `tag`, `type`.
- Standard pagination response format.

1.3.7 API documentation:
- Document endpoints using OpenAPI/Swagger.
- Provide Postman collection for testing.

### US 1.4 – Infrastructure & DevOps Setup (Priority: Medium)

**User Story:** As a dev team, I want a stable infrastructure and CI/CD so that we can deploy quickly and reliably.

**Tasks:**

1.4.1 Project scaffolding:
- Initialize React app and Node.js/Express server.
- Configure ESLint, Prettier, and optionally TypeScript.

1.4.2 Environment configuration:
- Manage `.env` for dev/stage/prod.
- Central configuration for DB URI, JWT secrets, email provider, external APIs.

1.4.3 MongoDB setup:
- Create DB, collections, and indexes.
- Seed minimal test data.

1.4.4 CI/CD pipeline:
- Set up git repo and branching strategy.
- Configure CI (e.g., GitHub Actions) for lint/test/build and deployment.

## Epic 2: Community & Social Features

### US 2.1 – Follower / Followee System (Priority: High)

**User Story:** As a user, I want to follow or unfollow other users so that I can see content from people I care about.

**Tasks:**

2.1.1 Design follow data model:
- Follow: `followerId`, `followeeId`, `createdAt`; with indexes.

2.1.2 Implement backend follow APIs:
- POST `/users/:id/follow`, DELETE `/users/:id/follow`.
- GET `/users/:id/followers`, GET `/users/:id/following`.
- Prevent self-follow and duplicates.

2.1.3 Implement frontend follow UI:
- Follow/Unfollow buttons on profiles and user cards.
- Followers/following lists with lazy loading.
- Integrate a basic "following feed" endpoint.

### US 2.2 – Islamic Community Channels (Priority: High)

**User Story:** As a mosque/society admin, I want to create and manage a channel so that I can share updates and spiritual content with my community.

**Tasks:**

2.2.1 Extend Channel model:
- Mosque/society name, address, timezone, prayer time source, logo, verification status, admins and roles.

2.2.2 Implement backend channel admin APIs:
- POST `/channels`, PATCH `/channels/:id`.
- POST `/channels/:id/admins` to add/remove admins.

2.2.3 Implement frontend channel management UI:
- Create channel form.
- Channel admin dashboard (stats, posts, subscribers).
- Admin list management.

### US 2.3 – Channel Updates (Salah Times, Halaqa, Lessons) (Priority: High)

**User Story:** As a channel admin, I want to post updates on salah times, halaqat, and events so that my community stays informed.

**Tasks:**

2.3.1 Extend post type for channel updates:
- Type: `channel_update`, category: `salah_time`, `halaqa`, `event`, `announcement`.
- Optional schedule/time range fields.

2.3.2 Implement backend APIs for channel posts:
- POST `/channels/:id/posts` (admin only).
- GET `/channels/:id/posts` with filters.

2.3.3 Implement frontend channel feed:
- Channel page with tabbed feed (All / Salah Times / Halaqa / Announcements).
- Structured forms for creating updates.

2.3.4 Implement basic notification hooks:
- Backend hooks to notify subscribers of new channel posts.

### US 2.4 – Channel Subscription / Unsubscription (Priority: High)

**User Story:** As a user, I want to subscribe or unsubscribe from channels so that I can receive updates from relevant mosques/societies.

**Tasks:**

2.4.1 Design subscription data model:
- Subscription: `userId`, `channelId`, `createdAt`.

2.4.2 Implement backend subscription APIs:
- POST `/channels/:id/subscribe`, DELETE `/channels/:id/subscribe`.
- GET `/users/me/subscriptions`.
- GET `/channels/:id/subscribers` (admin only).

2.4.3 Implement frontend subscription UI:
- Subscribe/Unsubscribe button on channel pages and cards.
- "My Channels" page listing subscriptions.
- Subscribed indicator on channel cards.

### US 2.5 – User Blogging / Post Creation (Priority: High)

**User Story:** As a user, I want to create Islamic posts and blogs so that I can share reflections, reminders, and knowledge.

**Tasks:**

2.5.1 Design post creation UX:
- Rich text editor with basic formatting.
- Tags (e.g., #Tafsir, #Hadith, #Ramadan, #Seerah).
- Optional association with a channel (if admin).

2.5.2 Wire backend authorization into Posts CRUD:
- Only authors or channel admins can edit/delete posts.

2.5.3 Implement post listing and detail views:
- Global feed with filters by tag, author, channel.
- Post detail page.

2.5.4 Add moderation hooks:
- Status: `published`, `pending_review`, `removed`.
- Admin endpoints to list and moderate content.

## Epic 3: Digital Quran & Knowledge Center

### US 3.1 – Quran Reader (Arabic + Roman Transliteration) (Priority: High)

**User Story:** As a user, I want to read the Quran with Arabic script and Roman transliteration so that I can recite and understand better.

**Tasks:**

3.1.1 Decide Quran data source:

- External API vs local dataset; must support surah, ayah, Arabic text, transliteration, and at least one translation.

3.1.2 Implement backend Quran APIs:
- GET `/quran/surahs`, GET `/quran/surahs/:id`, GET `/quran/ayah`.

3.1.3 Implement frontend Quran reader UI:
- Surah list page with search/filter.
- Reader view with Arabic, transliteration, translation.
- Font size and theme controls (day/night).

3.1.4 Implement bookmarks and last-read:
- Data model for bookmarks and last-read.
- APIs: GET/POST/DELETE `/quran/bookmarks`.
- UI for saving and resuming.

### US 3.2 – Downloadable Recitation Audio (Priority: Medium)

**User Story:** As a user, I want to listen to Quran recitation and download audio (where allowed) so that I can recite and memorize more easily.

**Tasks:**

3.2.1 Select and integrate audio source:
- Choose reciter/source and map surah/ayah to audio URLs.

3.2.2 Provide backend configuration/endpoints:
- Expose streaming URLs or configuration for frontend.

3.2.3 Implement reader audio controls:
- Play/pause, next/prev ayah, highlight playing ayah.
- Option to play full surah and download where licensing permits.

### US 3.3 – Tafsir, Hadith, Seerah, Dua & Adhkar (Priority: High)

**User Story:** As a user, I want to access tafsir, hadith, seerah, duas, and adhkar so that I can deepen my knowledge and practice.

**Tasks:**

3.3.1 Define content models and data sources:
- Tafsir linked to surah/ayah.
- Hadith (collection, number, texts, grading).
- Seerah (chapters/sections).
- Dua & Adhkar categories with Arabic, transliteration, translation.

3.3.2 Implement backend APIs:
- GET `/tafsir`, GET `/hadith`, GET `/seerah`, GET `/duas`, GET `/adhkar`.

3.3.3 Create frontend Knowledge Center hub:
- Sections: Quran/Tafsir, Hadith, Seerah, Dua & Adhkar.
- Search and filter UI.

3.3.4 Integrate with Quran reader:
- "View Tafsir" for each ayah.
- Link relevant hadith/duas where available.

### US 3.4 – Islamic Content/Education Module (Priority: Medium)

**User Story:** As a user, I want curated Islamic content (Masa'il, stories, lectures, nasheeds) so that I can learn and be inspired.

**Tasks:**

3.4.1 Define IslamicContent categorization:

- Type (masaala, story, lecture, nasheed), title, summary, body/media URL, language, scholar/reciter, tags, duration.

3.4.2 Implement backend APIs:
- GET `/content` with filters, GET `/content/:id`.

3.4.3 Implement frontend content hub:
- Sections for Masa'il, Stories, Lectures, Nasheeds.
- Cards and detail views with text or embedded player.

3.4.4 Add scaffolding for recommendations:
- Log user views/likes as events.

### US 3.5 – Daily Verse + Dua (Priority: High)

**User Story:** As a user, I want to receive a daily verse and dua so that I maintain daily spiritual reminders.

**Tasks:**

3.5.1 Design daily item generation logic:
- Pre-curated vs random with constraints.
- Define DailyItem model with date, verseRef, duaRef.

3.5.2 Implement backend API:
- GET `/daily` returning today's verse and dua (respect timezone).

3.5.3 Set up scheduling job:
- CRON job to pre-generate or cache daily items.

3.5.4 Implement frontend daily card:
- Home widget showing today's verse and dua.
- Share options and link to full surah/dua.

## Epic 4: Spiritual Utility & Personalization

### US 4.1 – Prayer Time Calculation & Reminders (Priority: High)

**User Story:** As a user, I want accurate prayer times with reminders so that I can pray on time wherever I am.

**Tasks:**

4.1.1 Implement user settings:
- Location (auto + manual), calculation method, madhhab, high-latitude rule.

4.1.2 Integrate prayer time calculation:
- Use library or API (e.g., Adhan.js).

4.1.3 Implement prayer time API:
- GET `/prayer-times` with date, lat/lng, method, madhhab; fallback to user preferences.

4.1.4 Implement reminders & notifications model:
- Reminder: userId, prayerName, offset, channels.
- APIs: GET/POST/PATCH `/prayer-reminders`.

4.1.5 Implement frontend prayer time UI:
- Daily times with next prayer highlight and countdown.
- Settings screen for calculation method and reminders.

### US 4.2 – Qibla Compass (Priority: Medium)

**User Story:** As a user, I want a Qibla compass so that I can easily find the direction of the Kaaba for prayer.

**Tasks:**

4.2.1 Implement Qibla bearing calculation:
- Use user's geolocation and Kaaba coordinates.

4.2.2 Implement frontend Qibla compass UI:
- Use device orientation where supported.
- Fallback arrow + degrees if orientation unavailable.

4.2.3 Add calibration and UX guidance:
- Instructions for when sensors are inaccurate or unavailable.

### US 4.3 – Forbidden Prayer Times Alerts (Priority: Medium)

**User Story:** As a user, I want alerts for forbidden prayer times so that I can avoid praying at discouraged times.

**Tasks:**

4.3.1 Define rules for forbidden times:
- Sunrise, zenith, sunset windows parametrized by location and prayer times.

4.3.2 Implement logic and API:
- Compute forbidden intervals from daily prayer times.
- GET `/prayer-times/forbidden`.

4.3.3 Implement frontend visualization and alerts:
- Show forbidden windows on prayer times screen.
- Optional in-app/local notifications.

### US 4.4 – Spirituality / Deeds Tracker (Priority: High)

**User Story:** As a user, I want to track my good deeds and spiritual habits so that I can monitor and improve my worship and character.

**Tasks:**

4.4.1 Define Deed and Habit models:
- Deed: userId, type, count/amount, date, notes, points.
- Habit templates for common deeds.

4.4.2 Implement backend APIs:
- POST `/deeds`, GET `/deeds`, GET `/deeds/summary`, GET `/deeds/habits`.

4.4.3 Implement frontend tracker UI:
- Quick logging for common deeds.
- Calendar/timeline view and progress charts/streaks.

4.4.4 Design gamification scaffolding:
- Points per deed type; basic level/badge structure.

### US 4.5 – AI Suggestion & Chatbot (Priority: Medium)

**User Story:** As a user, I want an AI assistant for Islamic questions and personalized guidance so that I can get quick, contextual reminders and suggestions.

**Tasks:**

4.5.1 Define AI scope, constraints, and disclaimers:
- Clarify allowed topics and limitations.
- Emphasize that it is not a Mufti and advise consulting scholars.

4.5.2 Integrate backend with LLM provider:
- Create service wrapper and system prompts.
- Implement POST `/ai/chat`.

4.5.3 Design contextual suggestion engine:
- Use deeds, prayer patterns, and content interests to generate suggestions.
- Define Suggestion model.

4.5.4 Implement frontend chatbot UI:

- Chat interface with message bubbles and quick question templates.
- Prominent disclaimers and “Ask a local scholar” CTA.

4.5.5 Implement logging and review:

- Store anonymized Q&A logs for quality monitoring.
- Flag sensitive topics for potential human review, respecting privacy.

## Prioritization Summary

**High Priority:** US 1.1, US 1.2, US 1.3, US 2.1, US 2.2, US 2.3, US 2.4, US 2.5, US 3.1, US 3.3, US 3.5, US 4.1, US 4.4.

**Medium Priority:** US 1.4, US 3.2, US 3.4, US 4.2, US 4.3, US 4.5.