# Development Roadmap for BandhuConnect+

We are building an MVP of **BandhuConnect+**, an application for Volunteers, Administrators, and Pilgrims/Attendees during large public gatherings. The app will have three modules:

---

**1. Development Method**

- Approach: **MVP-first Agile development** (start small, iterate fast).
- Focus: Core features first (auth, requests, task assignment, volunteer lifecycle, maps).
- Phases: Pilgrim app → Admin dashboard → Volunteer features → Real-time chat + maps → Enhancements (push notifications, multi-language).

---

**2. Tech Stack**

- **Frontend (Mobile Apps – Volunteers & Pilgrims):** React Native + Expo
- **Frontend (Admin Dashboard):** React.js + Tailwind CSS
- **Backend & Database:** Supabase (PostgreSQL, Auth, Realtime, Storage)
- **Authentication:** Supabase Auth (Phone OTP + Email/Password)
- **File Storage:** Supabase Storage (photos, lost child reports, sanitation reports)
- **Maps & Location:** Google Maps API (real-time volunteer/request tracking)
- **Notifications:** Expo Push Notifications (React Native)
- **Deployment:**
    - Web → Vercel (Admin Dashboard)
    - Mobile → Expo EAS Build / App Stores

---

**3. Database Schema (Supabase / PostgreSQL)**

**Important Terms**

- **PK (Primary Key):** A unique identifier for each row in the table.

- **FK (Foreign Key):** A reference to a primary key in another table, used to connect related records.

---

**users**

| Column | Type | Notes |
|---|---|---|
| id | uuid (**pk**) | Supabase default user ID |
| name | text | Full name |
| email | text | Unique, nullable |
| phone | text | Unique, required |
| role | enum | (‘volunteer’, ‘admin’, ‘pilgrim’) |
| skills | text[] | Array, only for volunteers |
| age | int | Nullable |
| lat | float8 | Current location (updated live) |
| lng | float8 | Current location (updated live) |
| created_at | timestamp | Default now() |

---

**requests**

| Column | Type | Notes |
|---|---|---|
| id | uuid (**pk**) | Unique request ID |
| user_id | uuid (**fk → users.id**) | Pilgrim who created request |
| type | enum | (‘medical’, ‘safety’, ‘lost_child’, ‘directions’, ‘sanitation’, ‘general’) |
| description | text | Nullable |

| Column | Type | Notes |
|---|---|---|
| photo_url | text | Optional |
| location | geography(point) | Lat/Lng |
| status | enum | (‘pending’, ‘assigned’, ‘in_progress’, ‘resolved’) |
| created_at | timestamp | Default now() |

---

**volunteer_tasks**

| Column | Type | Notes |
|---|---|---|
| id | uuid (**pk**) | Unique task ID |
| request_id | uuid (**fk → requests.id**) | Linked request |
| volunteer_id | uuid (**fk → users.id**) | Volunteer handling it |
| status | enum | (‘assigned’, ‘accepted’, ‘on_duty’, ‘completed’) |
| duty_start | timestamp | Set when volunteer checks in |
| duty_end | timestamp | Set when volunteer ends task |
| completed_by | uuid (**fk → users.id**) | Volunteer who ended task |
| completed_at | timestamp | When task was ended |

---

**messages (for chat)**

| Column | Type | Notes |
|---|---|---|
| id | uuid (**pk**) | Unique message ID |
| sender_id | uuid (**fk → users.id**) | From user |
| receiver_id | uuid (**fk → users.id**) | To user/group |
| content | text | Message body |

| Column | Type | Notes |
|---|---|---|
| created_at | timestamp | Default now() |

---

## 4. Task Lifecycle

1. **Assigned** → Admin assigns a volunteer.
2. **Accepted** → Volunteer accepts task.
3. **On Duty** → Volunteer checks in when starting.
4. **Completed** → Volunteer presses *End Task*.
    - Updates volunteer_tasks.status = completed
    - Auto-sets completed_by = volunteer_id and completed_at = now()
    - Related request.status = resolved

Admins **cannot** complete tasks. They can only assign/reassign.

---

## 5. Core Features

**Volunteers (Mobile App)**

- Sign up/login.
- Dashboard: Assigned tasks, duty status.
- Buttons: Accept Task → Check-In → End Task.
- Live map with current + task location.
- Chat (general volunteer chatroom).

**Admins (Web Dashboard)**

- See active/inactive volunteers.
- Track requests and their status.
- Assign/reassign volunteers to requests.
- Override skills, statuses (but not completion).

**Pilgrims/Attendees (Mobile App)**

- Request help (with auto GPS + description/photo).
- Lost child reports with photo + details.
- Report sanitation issues.
- Track volunteer ETA on map.
- Multi-language support.

---

## 6. API & Realtime Flows

- **Request Flow:** Pilgrim creates request → request goes to requests → Admin sees in dashboard (Realtime).
- **Assignment Flow:** Admin assigns volunteer → volunteer_tasks entry created → Volunteer app updates (Realtime).
- **Completion Flow:** Volunteer ends task → task + request auto-closed → Admin + Pilgrim see resolved status (Realtime).
- **Chat Flow:** Messages stored in messages table → live updates via Supabase Realtime.

---

## 7. Roadmap (Hackathon MVP order)

1. Setup Supabase project + schema.
2. Pilgrim app: Sign-up + request creation.
3. Admin dashboard: View requests + assign volunteers.
4. Volunteer app: Accept → Check-In → End Task.
5. Add Google Maps for location tracking.
6. Add Realtime updates + chat.
7. Add notifications + multi-language support.