ORTA DOĞU TEKNİK ÜNİVERSİTESİ

CNG 495 – Cloud Computing
Fall 2025

HelpConnect: Cloud-Based Emergency Assistance Platform (Flutter + AWS)

Capstone Project Final Report

Team Members:	**SIDs:**
Ece Kocabay	2591618
Noyan Saat	2453504
Ali Saadettin Yaylagül	2453637

Date of Submission: December 23, 2025

Contents

1 Introduction

HelpConnect is a cross-platform emergency assistance application designed to operate seamlessly on mobile (iOS), unified Flutter codebase. It connects individuals experiencing urgent but manageable emergencies with nearby community volunteers and responders.

HelpConnect addresses a critical gap in emergency response systems by providing a platform for community-driven assistance. Traditional emergency services (911, police, fire) are designed for life-threatening situations, leaving many individuals without adequate support for everyday urgent needs such as grocery assistance for the elderly, helping locate missing pets, or responding to minor environmental incidents. HelpConnect solves this problem by creating a peer-to-peer volunteer network where community members can request help and nearby volunteers can respond in real-time. The application leverages cloud-native AWS services to ensure high availability, automatic scaling, and secure data handling, while the Flutter framework enables deployment across iOS, Android, and web platforms from a single codebase.

1.1 Benefits and Novelties

HelpConnect introduces several innovative approaches that distinguish it from traditional emergency applications. Unlike conventional systems that rely solely on professional responders, HelpConnect empowers everyday citizens to become first responders for non-critical situations. The dual-role architecture allows users to seamlessly transition between seeking help and providing assistance, fostering a reciprocal community ecosystem. The proximity-based matching algorithm ensures that the nearest available volunteer is notified first, significantly reducing response times. Furthermore, the offer-based system gives help seekers control over who assists them, as they can review volunteer profiles, estimated arrival times, and personal notes before accepting an offer.

Key features and novelties include:

- **Community-Driven Response:** Connects people with volunteer responders for non-life-threatening situations.
- **Dual-Role System:** Users can switch between Help Seeker and Volunteer roles within the same app.
- **Broad Category Coverage:** Includes medical assistance, missing pets, environmental

incidents, and daily life support.

- **Proximity-Based Matching:** Volunteers view requests near their location with a configurable radius.
- **Request Lifecycle Management:** Full tracking from `OPEN` to `CLOSED` with real-time status updates.
- **Offer System:** Volunteers provide specific ETAs and notes, giving control back to the help seeker.
- **Serverless Architecture:** Pay-per-use AWS infrastructure eliminates server maintenance and enables infinite scalability.

1.2 Similar Projects

Several open-source projects address community assistance and emergency coordination. The following GitHub repositories represent similar efforts in this domain:

- **SOS Emergency App**
  A mobile/web emergency app providing SOS and location features.
  `https://github.com/Thabhelo/sos`
- **emergency-app-mobile**
  Mobile project for requesting immediate help in emergencies with media support.
  `https://github.com/melihsahtiyan/emergency-app-mobile`
- **public-emergency-app**
  Cross-platform emergency response application built with Flutter and Firebase.
  `https://github.com/ahmaddioxide/public-emergency-app`
- **SOS-Emergency-App (Android)**
  Android app for emergency SOS messages and location alerts.
  `https://github.com/samirsuroshe18/SOS-Emergency-App`
- **hazard_reporting_system**
  A system to report hazards and connect with volunteers.
  `https://github.com/skfarhad/hazard`$_r eporting_s ystem$

2 Project Statistics

The following tables summarize the technical metrics, system requirements, and development timeline for HelpConnect.

2.1 Programming Languages and Code Metrics

Language	**Component**	**Lines of Code**	**Percentage**
Dart	Flutter Frontend	$\approx$ 4,200	81%
Python	AWS Lambda Backend	$\approx$ 1,000	19%
Total		$\approx$ **5,200**	**100%**

Table 1: Programming Languages Breakdown

2.2 Memory and Storage Requirements

Component	**Requirement Type**	**Value**
Mobile App (iOS)	Minimum RAM	2 GB
Mobile App (iOS)	Recommended RAM	3+ GB
Mobile App	Storage Space	$\approx$ 80 MB
AWS Lambda	Function Memory	256 MB (configurable)
DynamoDB	Storage	On-demand (serverless)
S3	Image Storage	$\approx$ 2 MB per image

Table 2: Memory and Storage Requirements

2.3 Database and Infrastructure

Component	**Type**	**Details**
Primary Database	NoSQL (Key-Value)	Amazon DynamoDB
Object Storage	Blob Storage	Amazon S3
Authentication Store	Managed Identity	Amazon Cognito User Pool
Compute	Serverless Functions	AWS Lambda (Python 3.12)
API Layer	HTTP API	Amazon API Gateway v2
Deployment	Manual	AWS Management Console

Table 3: Database Types and Infrastructure

Dates	**Phase**	**Tasks & Milestones**	**Member**
Oct 27 – Nov 2	Scope & Design	Finalized scope; designed serverless architecture; created GitHub repo; defined DynamoDB schemas.	Ece, Ali, Noyan
Nov 3 – Nov 9	AWS Setup	Configured Cognito User Pool; created DynamoDB tables; set up S3 bucket; initialized Flutter project.	Ali, Noyan
Nov 10 – Nov 16	Backend API	Developed Create/List/Nearby Lambdas; configured API Gateway with JWT; resolved CORS.	Ece, Ali
Nov 17 – Nov 23	Frontend UI	Built Help Seeker & Volunteer Home Screens; Google Maps integration; Request Detail Screen.	Noyan, Ece
Nov 24 – Nov 30	Offers System	Developed Offer/Accept Lambdas; built My Requests & My Offers screens; integration testing.	Ece, Noyan
Dec 1 – Dec 14	Auth & Notify	Built Login, Register, Role Selection screens; developed SNS notification & image upload Lambdas.	Ali, Ece
Dec 15 – Dec 23	Admin & Final	Built Admin Dashboard UI; developed Admin Lambdas (Init, Reset, Backup, Modify); Final Report.	Ali, Noyan, Ece

Table 4: Detailed Project Timeline and Member Responsibilities

3 Structure of the Project

HelpConnect utilizes a serverless architecture across multiple AWS service layers to ensure scalability and security.

3.1 Utilized Cloud Services and Delivery Models

The system leverages SaaS, PaaS, and IaaS delivery models as summarized below.

Service	**Model**	**Purpose**	**How It's Used**
Amazon Cognito	SaaS	Authentication	Registration, login, and JWT token issuance.
Amazon DynamoDB	PaaS	Database	Stores requests, offers, and settings.
Amazon S3	IaaS	Object Storage	Stores request images and DB backups.
Amazon API Gateway	PaaS	REST API	Routes HTTP requests to Lambda.
AWS Lambda	PaaS	Compute	Executes Python business logic.
Amazon SNS	PaaS	Notifications	Email alerts for new help requests.
AWS IAM	PaaS	Access Control	Manages service-to-service permissions.

Table 5: AWS Services Overview and Delivery Models

3.2 Functional Diagrams

Below are the architectural abstractions representing the information flow of the system.

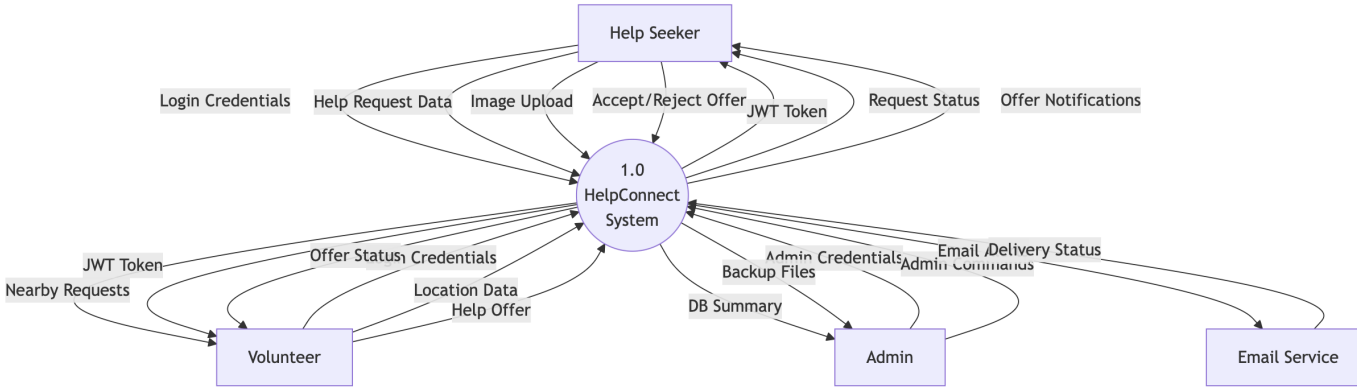

Figure 1: Context Level (Level 0) Data Flow Diagram

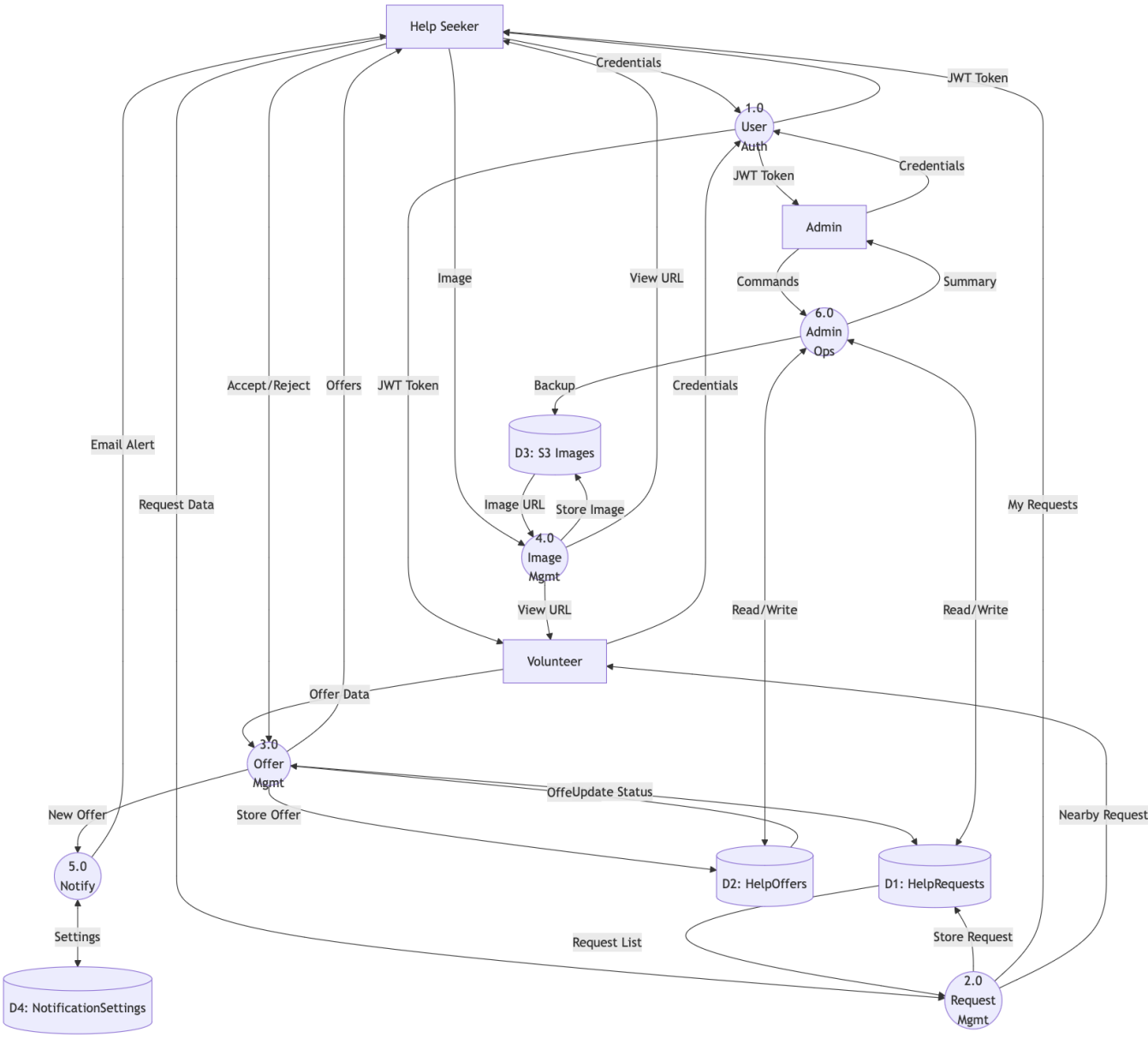

Figure 2: Level 1 Data Flow Diagram showing major sub-processes

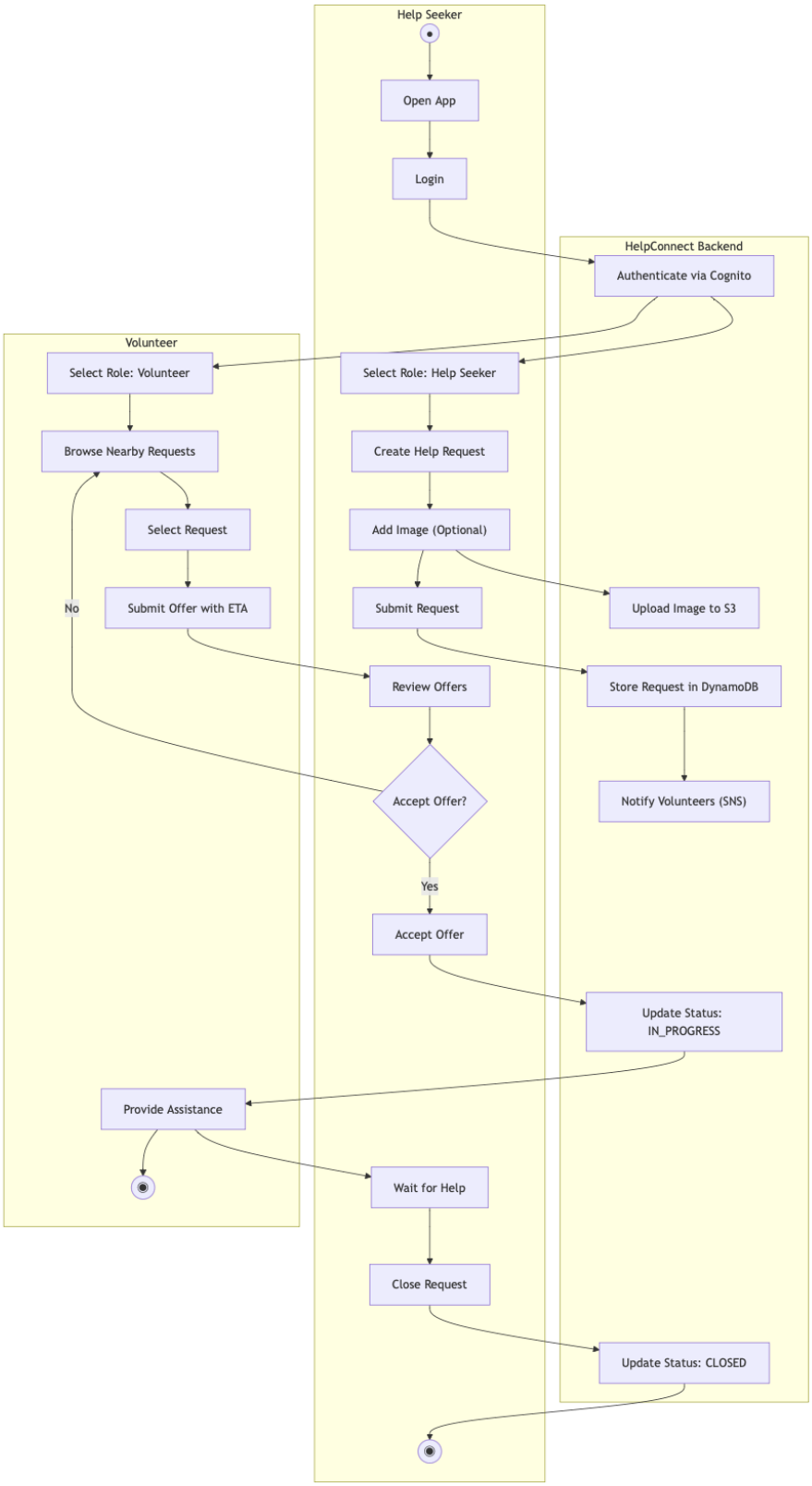

Figure 3: Activity Diagram illustrating the Help Request submission workflow

3.3 Detailed AWS Component Implementation

Feature	**Usage**
User Pool	Stores user accounts (email, password, name).
JWT Tokens	Sent with every API request for authorization.
User Groups	Admin group for administrative dashboard access.

Table 6: Amazon Cognito Role Management

Table	**Partition Key**	**Sort Key**	**GSI**
HelpRequests	request_id	-	help_seeker_id-index
HelpOffers	request_id	offer_id	volunteer_id-index
HelpConnectRequestImages	request_id	image_id	-
NotificationSettings	user_id	-	-

Table 7: DynamoDB Table Structure and Indexing

Lambda File Name	**Trigger**	**Purpose**
HelpConnectListEmergencies.py	GET /emergencies	Fetch all requests
HelpConnectGetNearbyEmergencies.py	GET /emergencies/nearby	Geospatial query
HelpConnectCreateHelpRequest.py	POST /help-requests	Create new request
HelpConnectGetHelpRequest.py	GET /help-requests/{id}	Fetch request details
HelpConnectCloseRequest.py	PATCH /.../close	Close a request
HelpConnectListMyRequests.py	GET /my-requests	List user's requests
HelpConnectOfferHelp.py	POST /offers	Volunteer offers help
HelpConnectListOffers.py	GET /offers	List offers for request
HelpConnectGetMyOffers.py	GET /my-offers	List volunteer's offers
HelpConnectAcceptOffer.py	POST /accept-offer	Accept an offer

Table 8: AWS Lambda Functions - Core Request & Offer Operations

Lambda File Name	Trigger	Purpose
HelpConnectGetUploadUrl.py	POST /images/upload-url	S3 pre-signed PUT URL
HelpConnectGetViewUrl.py	GET /images/view-url	S3 pre-signed GET URL
HelpConnectAttachRequestImage.py	POST /requests/{id}/images	Link image to request
HelpConnectListRequestImages.py	GET /requests/{id}/images	List request images
HelpConnectDeleteRequestImage.py	DELETE /.../images	Delete image
HelpConnectGetNotificationSettings.py	GET /notification-settings	Get preferences
HelpConnectUpdateNotificationSettings.py	POST /notification-settings	Update preferences
HelpConnectAutoSubscribeUser.py	Cognito Trigger	Auto-subscribe to SNS
AdminInitialize.py	POST /admin/initialize	Initialize database
AdminReset.py	POST /admin/reset	Reset database
AdminBackup.py	POST /admin/backup	Backup to S3
AdminView.py	GET /admin/view	View DB summary
AdminModify.py	PATCH /admin/modify	Modify records

Table 9: AWS Lambda Functions - Images, Notifications & Admin

3.4 Technical Implementation Details

This section explains the core algorithms and data flows that power the HelpConnect backend.

3.4.1 Geospatial Matching Algorithm

The proximity-based volunteer matching uses the Haversine formula to calculate distances between coordinates. The algorithm flow is:

1. **Request Input:** The Lambda function receives the volunteer's current latitude, longitude, and desired radius (in kilometers).
2. **Database Scan:** All `OPEN` and `IN_PROGRESS` requests are retrieved from DynamoDB.
3. **Distance Calculation:** For each request, the Haversine formula calculates the great-circle distance:

```
a = sin²(lat/2) + cos(lat1) × cos(lat2) × sin²(lng/2)
distance = 2 × R × arctan2(a, (1−a))
```

where R = 6,371 km (Earth's radius).

4. **Filtering:** Only requests within the specified radius are returned.
5. **Sorting:** Results are sorted by distance (nearest first) and urgency level.

3.4.2 S3 Pre-Signed URL Generation

Image uploads use pre-signed URLs to securely transfer files directly from the client to S3 without exposing AWS credentials:

1. **Step 1 - Request URL:** The Flutter app calls `POST /images/upload-url` with the desired file extension.
2. **Step 2 - Generate Key:** Lambda generates a unique S3 key: `images/{uuid}.{extension}`.
3. **Step 3 - Create Pre-Signed URL:** Using boto3's `generate_presigned_url()`, a PUT URL is created with 15-minute expiry.
4. **Step 4 - Client Upload:** Flutter uses the HTTP PUT method to upload the image directly to S3.
5. **Step 5 - Attach to Request:** After upload, Flutter calls `POST /requests/{id}/images` to link the image key to the request in DynamoDB.
6. **Step 6 - View URL:** When displaying images, `GET /images/view-url` returns a pre-signed GET URL valid for 1 hour.

3.4.3 JWT Token Authentication Flow

All API requests are authenticated using Cognito JWT tokens:

1. User logs in via Cognito and receives an ID Token and Access Token.
2. Flutter stores the Access Token securely and includes it in the `Authorization` header.

3. API Gateway validates the JWT signature against Cognito's public keys.

4. If valid, the request proceeds to Lambda with the user's `sub` (unique ID) available in claims.

5. Lambda extracts the `sub` to identify the user and enforce ownership checks (e.g., only close your own requests).

4 User Manual

HelpConnect provides two primary interfaces: the Mobile application for general users and the Admin Dashboard for system management. This section provides step-by-step tutorials for each user role.

4.1 Authentication Tutorial

4.1.1 Registration

New users must create an account before using HelpConnect.

1. Open the HelpConnect application

2. Tap **"Create Account"** on the login screen

3. Enter your **Full Name**, **Email Address**, and **Password**

4. Tap **"Create Account"** to submit

5. Check your email inbox for a 6-digit verification code

6. Enter the verification code on the confirmation screen

7. Upon successful verification, you will be redirected to the login screen

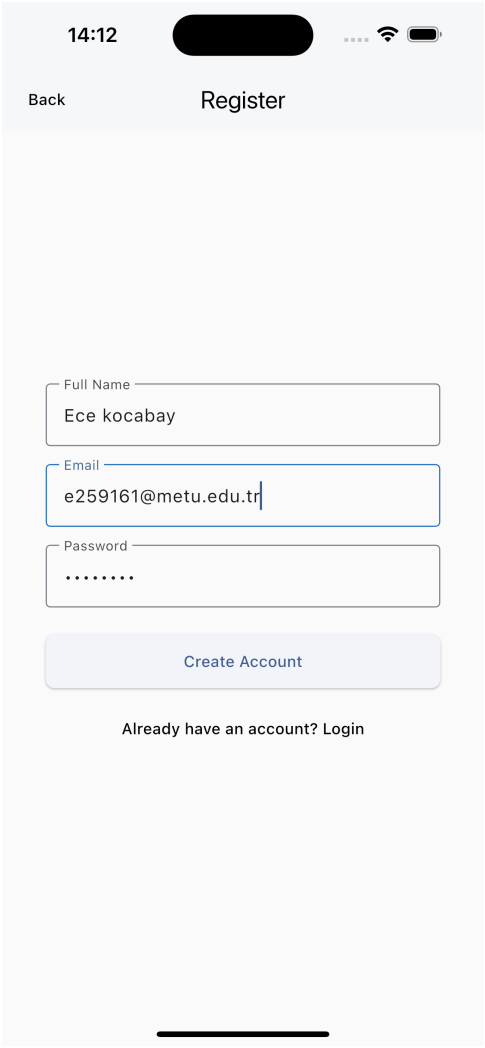

Figure 4: Registration Screen - Enter user details

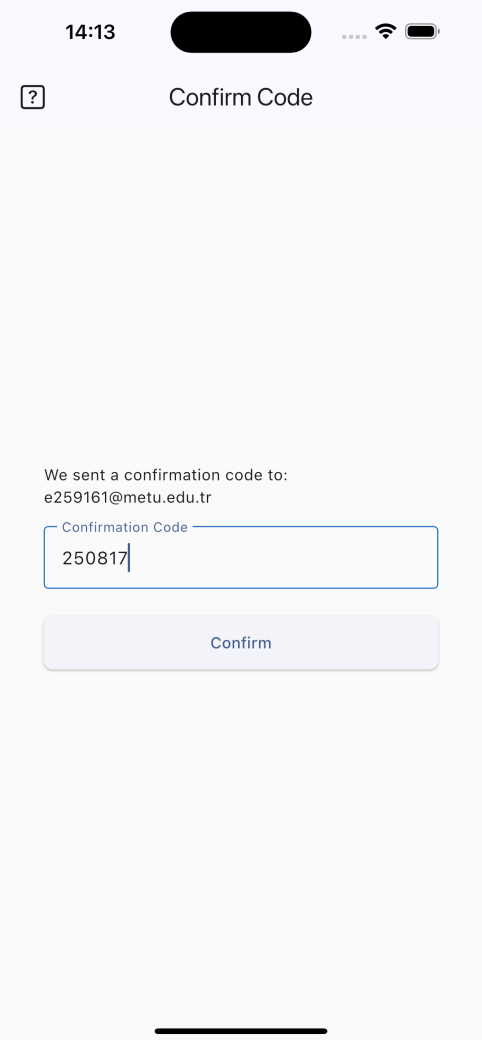

Figure 5: Email Verification Screen - Enter 6-digit code

4.1.2 Login

Registered users can access the application by logging in.

1. Open the HelpConnect application
2. Enter your registered **Email Address**
3. Enter your **Password**
4. Tap **"Login"** to authenticate
5. Upon successful login, you will be directed to the Role Selection screen

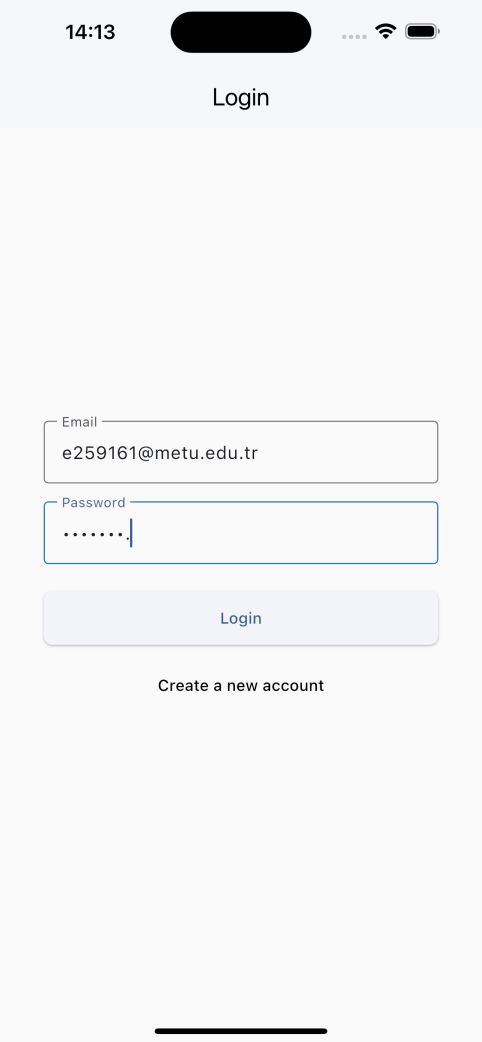

Figure 6: Login Screen - Enter credentials to access the app

4.1.3 Role Selection

After login, users must select their role for the current session.

1. View the available roles on the Role Selection screen
2. Tap **"Help Seeker"** if you need assistance
3. Tap **"Volunteer"** if you want to help others
4. Admin users will see an additional **"Admin"** option
5. Your selection is saved for future sessions

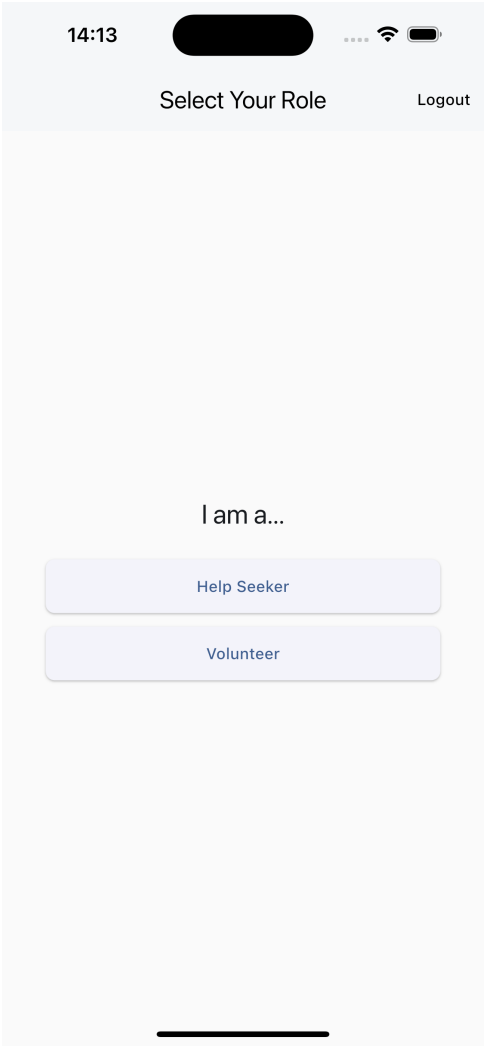

Figure 7: Role Selection Screen - Choose Help Seeker or Volunteer

4.2 Help Seeker Tutorial

Help Seekers can create assistance requests and manage responses from volunteers.

4.2.1 Help Seeker Home Screen

The home screen displays all active help requests on a map and list view.

- **Map View:** Shows request locations with interactive markers
- **List View:** Displays request cards with title, category, and status
- **New Request Button:** Floating button to create a new request
- **Navigation Options:** Switch Role, Refresh, My Requests, Profile

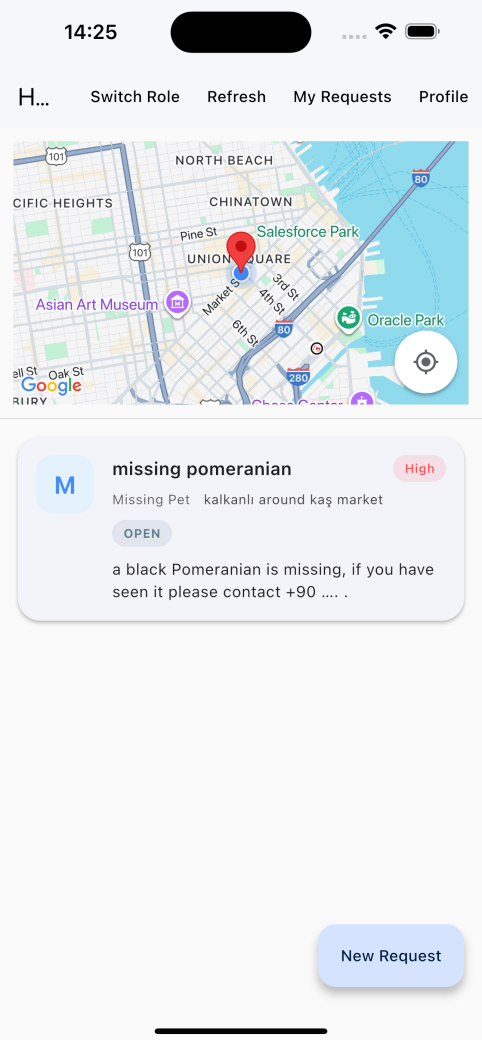

Figure 8: Help Seeker Home Screen - View requests on map and list

4.2.2 Creating a Help Request

To request assistance, follow these steps:

1. Tap the **"New Request"** button on the home screen
2. Enter a **Title** describing your need (e.g., "Need help with groceries")
3. Select a **Category**: Medical, Missing Pet, Environmental, or Daily Support
4. Choose an **Urgency Level**: Low, Medium, or High
5. Write a detailed **Description** of your situation
6. Set your **Location** using GPS or manual entry

7. Optionally, tap **"Add Photo"** to attach an image

8. Tap **"Submit Request"** to publish

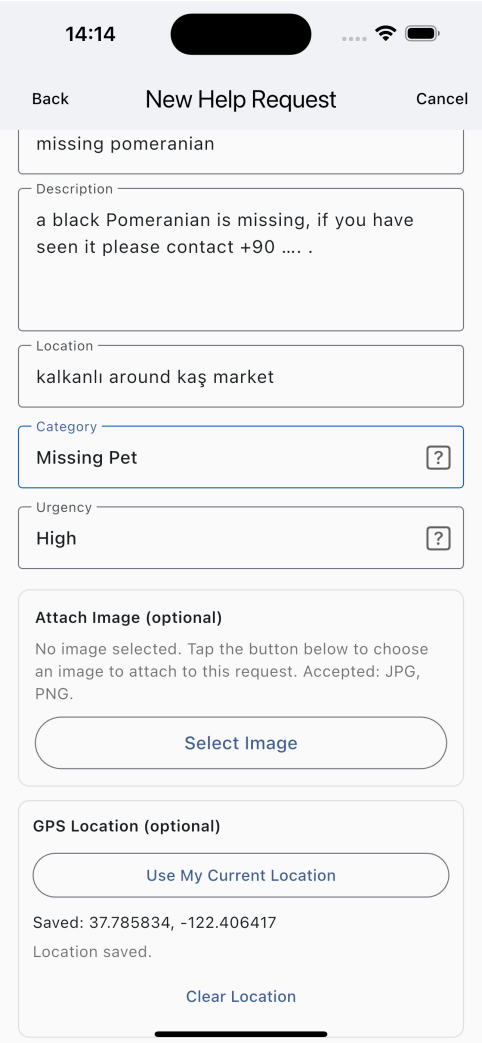

Figure 9: Create Request Screen - Fill in request details

4.2.3 Managing Your Requests

View and manage all your submitted requests.

1. Tap **"My Requests"** from the home screen

2. View the list of your requests with their current status

3. Tap a request to see its details and any volunteer offers

4. Status indicators:

- `OPEN` - Waiting for volunteer offers
- `IN_PROGRESS` - A volunteer is helping
- `CLOSED` - Request has been resolved

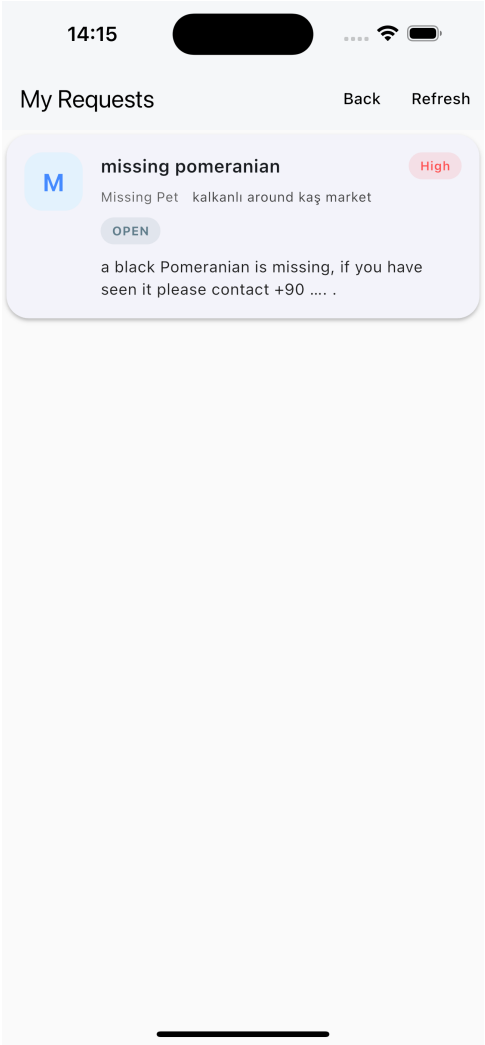

Figure 10: My Requests Screen - View all submitted requests

4.2.4 Reviewing and Accepting Offers

When volunteers offer help, you can review and accept them.

1. Open a request from **"My Requests"**
2. Scroll down to view the **Offers** section
3. Each offer shows:

- Volunteer's email
- Their note/message
- Estimated arrival time (ETA)

4. Tap **"Accept"** on the preferred offer
5. The request status changes to `IN_PROGRESS`
6. Other offers are automatically marked as not selected

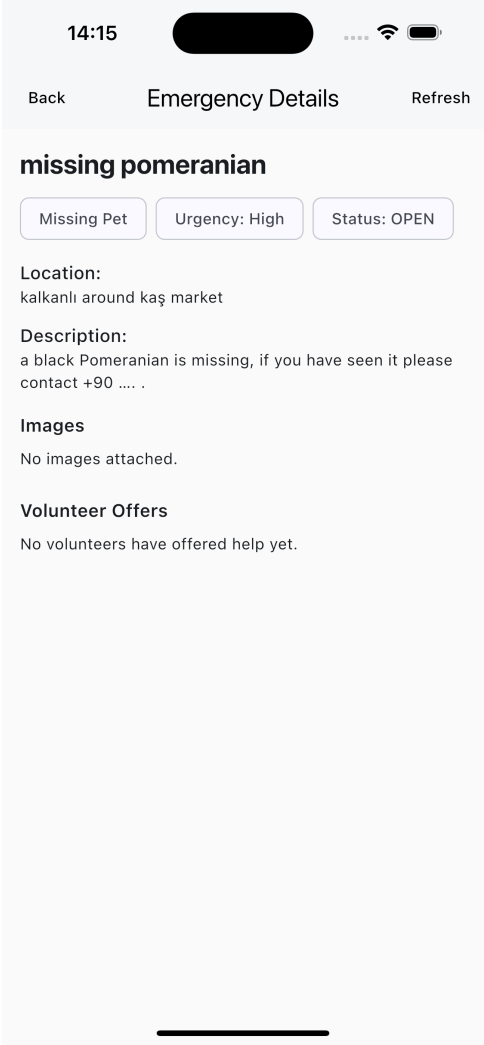

Figure 11: Request Detail Screen - Review volunteer offers

4.2.5 Closing a Request

After receiving help, close your request.

1. Open the request from **”My Requests”**
2. Tap the **”Close Request”** button
3. Confirm the closure
4. The request status changes to `CLOSED`
5. Closed requests are removed from the public listing

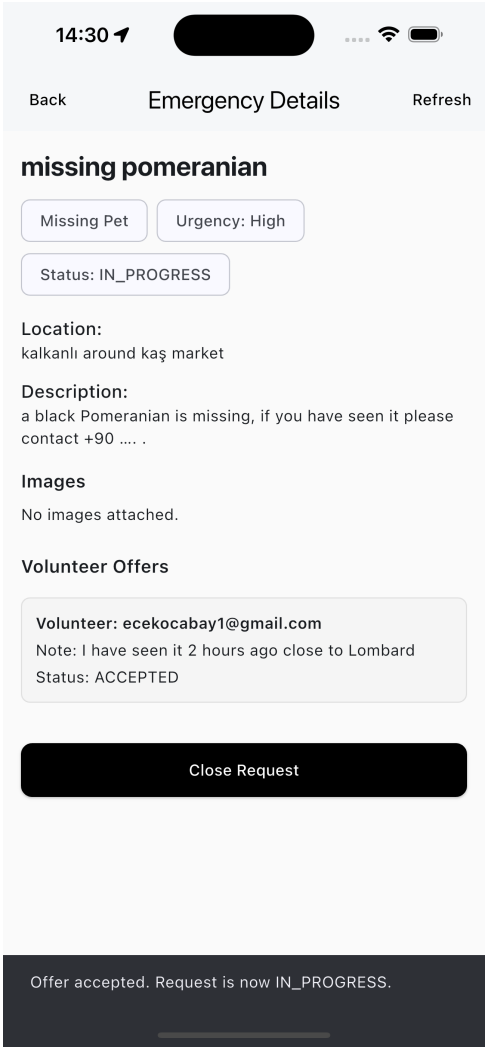

Figure 12: Close Request - Mark the request as resolved

4.3 Volunteer Tutorial

Volunteers can browse nearby help requests and offer assistance.

4.3.1 Volunteer Home Screen

The volunteer home screen shows help requests near your location.

- **Map View:** Displays nearby requests as markers
- **Radius Filter:** Adjust search radius (5km, 10km, 20km)
- **Request Cards:** Shows title, category, urgency, and distance
- **Navigation Options:** Switch Role, Refresh, My Offers, Profile

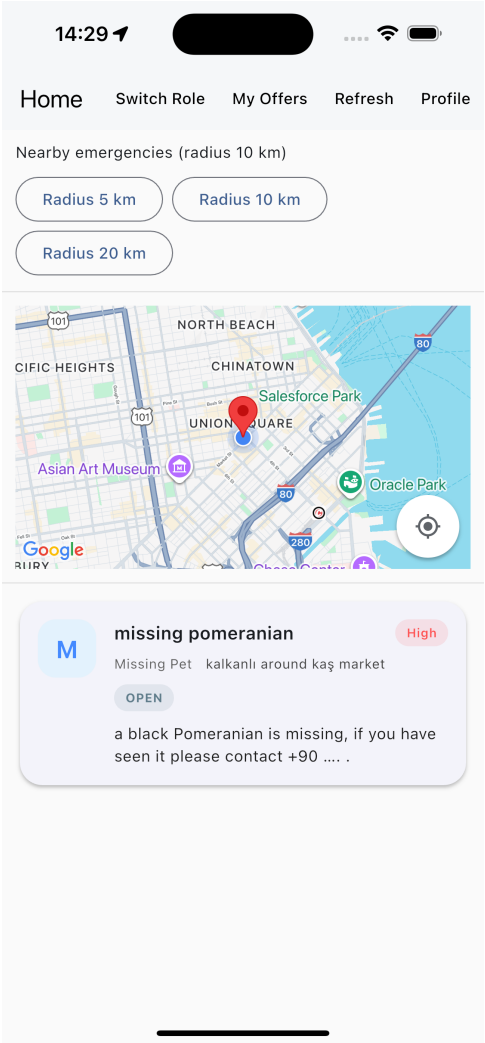

Figure 13: Volunteer Home Screen - Browse nearby help requests

4.3.2 Viewing Request Details

Get more information about a request before offering help.

1. Tap on a request card or map marker
2. View the full request details:
    - Title and description
    - Category and urgency level
    - Location on map
    - Attached images (if any)
    - Help seeker information
3. Decide if you can provide assistance

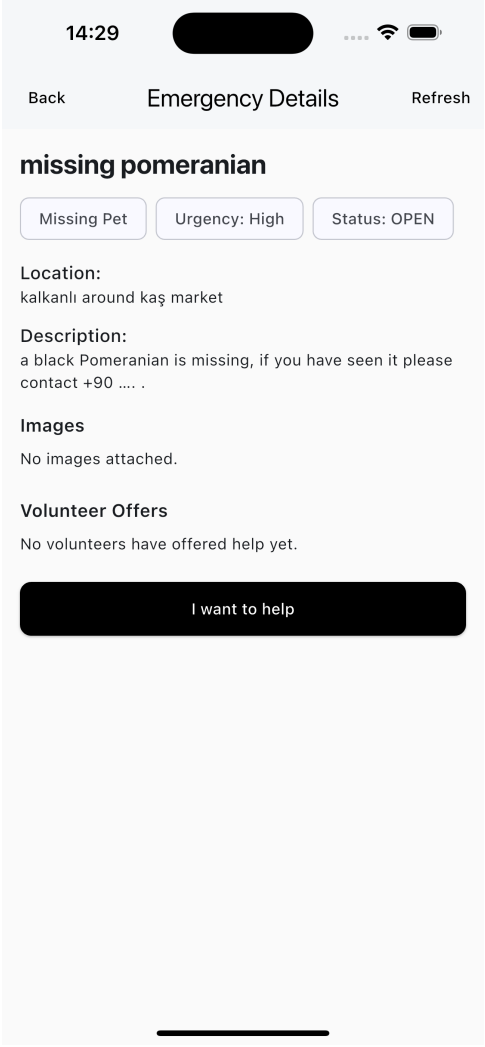

Figure 14: Request Details Screen - View full request information

4.3.3 Offering Help

Submit an offer to help with a request.

1. Open a request's detail page
2. Tap the **"Offer Help"** button
3. Enter your **Estimated Arrival Time** (in minutes)
4. Write a **Note** to the help seeker (optional)
5. Tap **"Submit Offer"**
6. Your offer is sent to the help seeker
7. The help seeker receives an email notification

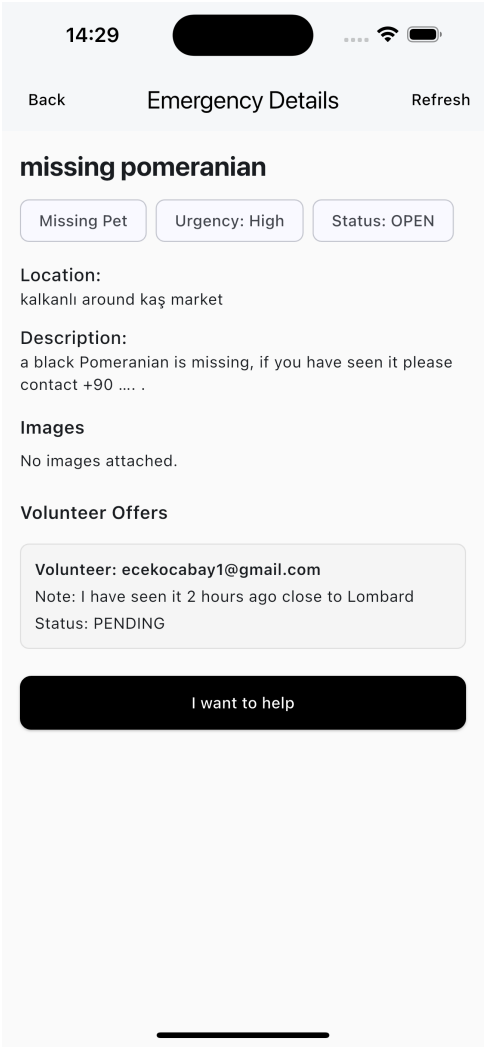

Figure 15: Offer Help Screen - Submit your offer with ETA

4.3.4 Tracking Your Offers

Monitor the status of your submitted offers.

1. Tap **"My Offers"** from the home screen
2. View all offers you have submitted
3. Offer statuses:
    - `PENDING` - Waiting for response
    - `ACCEPTED` - Help seeker accepted your offer
    - `REJECTED` - Another volunteer was selected
4. Tap an offer to view request details

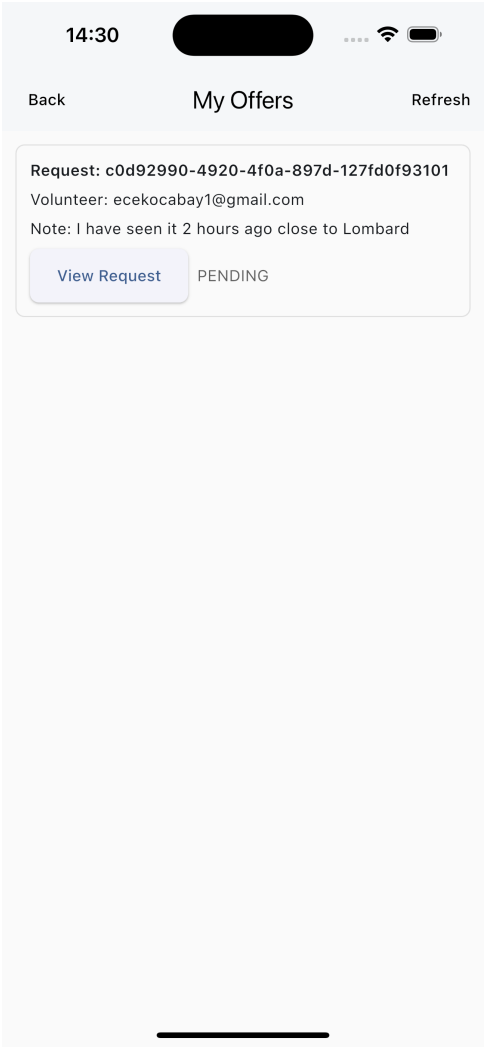

Figure 16: My Offers Screen - Track your submitted offers

4.4 Admin Dashboard Tutorial

Admin users have access to system management functions. Only users in the ”Admin” Cognito group can access these features.

4.4.1 Accessing Admin Dashboard

1. Login with an Admin account (must be member of ”Admin” Cognito group)
2. On the Role Selection screen, the **”Admin”** button appears only for authorized users
3. Tap **”Admin”** to open the dashboard
4. If not authorized, a red warning banner displays: ”You are NOT an admin”

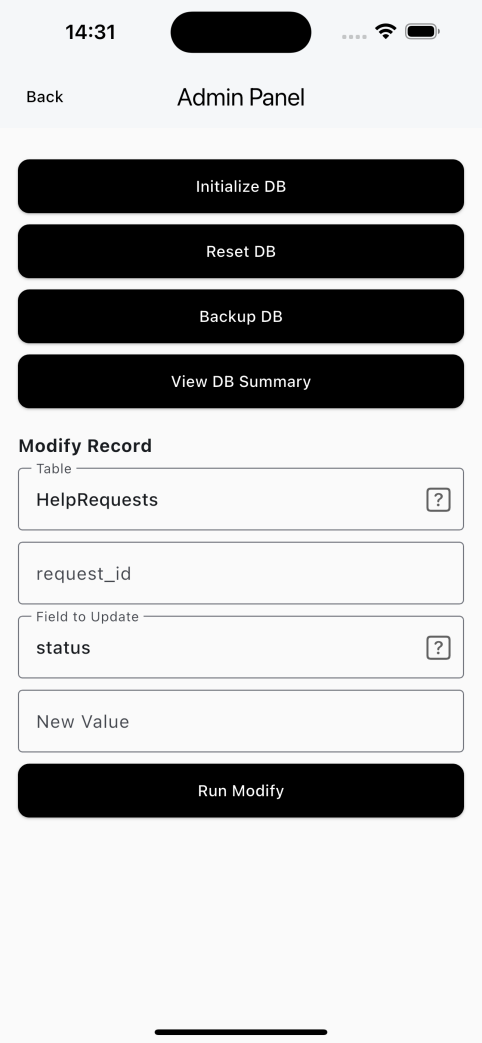

Figure 17: Admin Dashboard - System management interface

4.4.2 Initialize Database

Create or reinitialize the DynamoDB tables.

Safety Considerations:

- This action is idempotent - running it multiple times will not duplicate tables.
- Existing data is preserved if tables already exist.
- Only use this for initial setup or recovery scenarios.

Steps:

1. Tap **"Initialize DB"**
2. Wait for the operation to complete (typically 2–5 seconds)
3. Success message displays: "Tables created" or "Tables already exist"
4. Verify by using "View DB Summary" to confirm table structure

4.4.3 Reset Database

Clear all data from the database tables.

Safety Warnings:

- **CAUTION:** This action permanently deletes ALL records from all tables.
- This action **CANNOT be undone** - consider backing up first.
- Table structures are preserved; only data is removed.
- Use only for testing environments or complete system reset.

Steps:

1. **Recommended:** First tap "Backup DB" to save current data
2. Tap **"Reset DB"**
3. All records in HelpRequests, HelpOffers, and NotificationSettings are deleted
4. Success message displays record counts deleted from each table
5. Verify by using "View DB Summary" to confirm empty tables

4.4.4 Backup Database

Export all database tables to Amazon S3.

Backup Details:

- All tables are exported as JSON files
- Files are organized by timestamp for easy retrieval
- Backups are stored indefinitely until manually deleted from S3

Steps:

1. Tap **"Backup DB"**
2. The system scans all tables and serializes records to JSON
3. Files are uploaded to S3 bucket: `helpconnect-backups/`
4. Directory structure: `YYYYMMDD-HHMMSS/TableName.json`
5. Success message shows:
    - S3 bucket and prefix location
    - Record counts for each table
    - Individual file paths

4.4.5 View Database Summary

Inspect current database contents.

1. Tap **"View DB Summary"**
2. View record counts for each table
3. Sample records are displayed for verification
4. Useful for verifying data after Initialize, Reset, or Modify operations

4.4.6 Modify Records

Edit specific records for moderation purposes.

Record Key Selection:

The key fields required depend on the selected table:

- **HelpRequests:** Requires `request_id` (UUID of the request)
- **HelpOffers:** Requires both `request_id` AND `offer_id` (composite key)
- **NotificationSettings:** Requires `user_id` (Cognito sub/UUID of the user)

Allowed Fields by Table:

- **HelpRequests:** status, title, description, urgency, category, location
- **HelpOffers:** status, note, estimated_arrival_minutes
- **NotificationSettings:** notify_enabled, email

Steps:

1. Select the **Table** from the dropdown (HelpRequests, HelpOffers, NotificationSettings)
2. The form dynamically updates to show required key fields
3. Enter the record **Key** value(s):
    - For HelpRequests: paste the `request_id` UUID
    - For HelpOffers: paste both `request_id` and `offer_id`
    - For NotificationSettings: paste the `user_id`
4. Select the **Field to Update** from the dropdown
5. Enter the **New Value**:
    - For status: use OPEN, IN_PROGRESS, or CLOSED
    - For notify_enabled: use true or false
    - For numeric fields: enter integer values
6. Tap **"Run Modify"**
7. Success message displays the updated record with all fields

4.5 Profile and Settings

4.5.1 Profile Screen

Users can view and manage their account settings.

1. Tap **"Profile"** from the home screen
2. View your account information
3. Manage notification preferences
4. Tap **"Logout"** to sign out

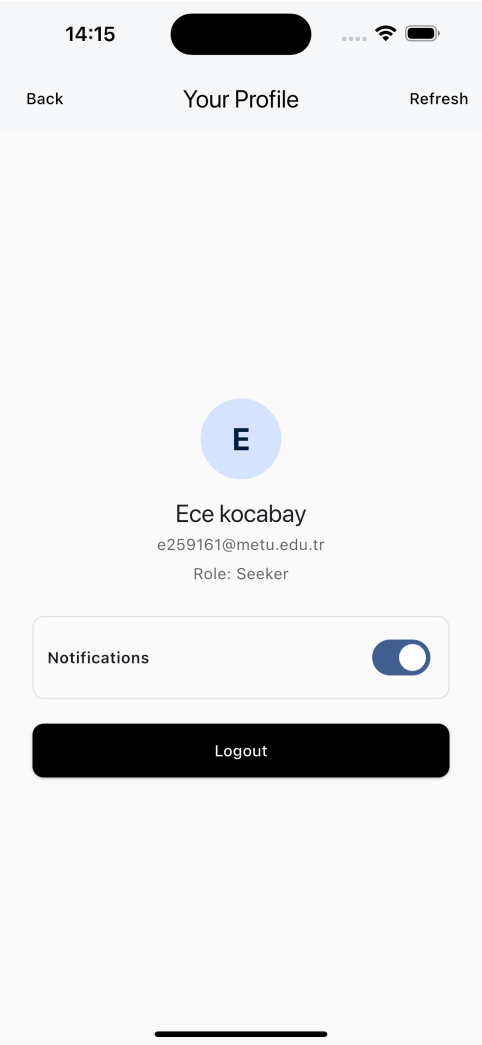

Figure 18: Profile Screen - View account and settings

5 Deliverables

The project includes the following components:

- **Flutter Frontend:** Dart code for iOS, Android, and Web (29 files, ≈4,200 LOC).
- **AWS Lambda Backend:** Python scripts for serverless business logic (24 functions, ≈1,000 LOC).
- **Database Schema:** DynamoDB table definitions with GSI configurations.
- **GitHub Repository:** All code, reports, and README files.
- **Final Report:** This document with technical documentation and user manual.