CNG 495 – Cloud Computing

Fall 2025

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

Capstone Project Proposal

Team Members:

Ece Kocabay	**SID: 2591618**
Noyan Saat	**SID: 2453504**
Ali Saadettin Yaylagül	**SID: 2453637**

Course Code: CNG 495 — **Semester-Year:** Fall 2025 — **Type:** Capstone Project Proposal

Date of Submission: October 20, 2025

Contents

1 Topic

Client/Server Cloud Application Topic: A cross-platform (mobile/web) emergency assistance and coordination platform developed using a single Flutter codebase for the client and a serverless, event-driven AWS backend.

2 Project Description

HelpConnect is a cross-platform emergency assistance application designed to operate seamlessly on both mobile (iOS/Android) and web from a single, unified Flutter codebase. It connects individuals experiencing urgent but manageable emergencies with nearby volunteers and donors.

The client application interacts with the AWS backend through REST APIs exposed by **AWS API Gateway**, which routes requests to serverless **AWS Lambda** functions for business logic execution. User data and requests are stored in **Amazon DynamoDB**, while user-uploaded media is saved in **Amazon S3**. User authentication is securely managed by **AWS Cognito**, and real-time push notifications are sent via **Amazon SNS**.

A key feature of the architecture is its event-driven, serverless paradigm. New help requests written to DynamoDB will asynchronously trigger a separate Lambda function via **DynamoDB Streams** to handle the computationally intensive task of matching volunteers. This ensures the user receives an immediate response while the heavy processing occurs in the background. The project also includes a web-based **Admin Dashboard** for monitoring system activity and managing data.

3 Cloud Delivery Models

HelpConnect utilizes a mix of **Software as a Service (SaaS)**, **Platform as a Service (PaaS)**, and **Infrastructure as a Service (IaaS)** layers.

3.1 Software as a Service (SaaS)

Amazon Cognito is used for user authentication and management. It provides a complete identity solution as a managed service, removing the need to build and secure a custom authentication system.

3.2 Platform as a Service (PaaS)

AWS Lambda, **Amazon DynamoDB**, and **Amazon SNS** are the core PaaS components. AWS manages the underlying servers, scaling, and fault tolerance, allowing our team to focus exclusively on the application logic and data models.

3.3 Infrastructure as a Service (IaaS)

Amazon S3 serves as the object storage backbone. While S3 is a managed service, it abstracts the underlying hardware (IaaS), where AWS maintains the physical servers and networking, and our team manages data durability and access configurations.

4 Diagrams and Figures

4.1 Data Flow Diagram: Context Level (Level 0)

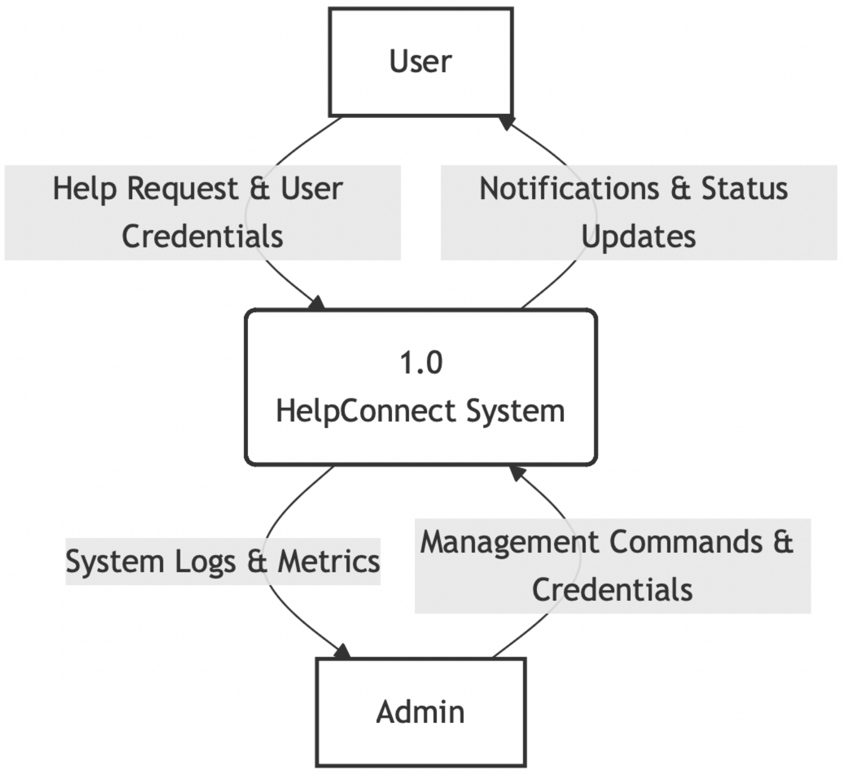
User
Help Request & User Credentials
Notifications & Status Updates
1.0 HelpConnect System
System Logs & Metrics
Management Commands & Credentials
Admin

Figure 1: Context Level (Level 0) Data Flow Diagram for the HelpConnect System.

Explanation: This diagram presents the highest-level view of the system. The entire HelpConnect application is shown as a single process ("1.0 HelpConnect System"). It illustrates the system's boundaries and its primary interactions with external entities: the 'User' and the 'Admin'. The labeled arrows represent the main categories of data that flow into and out of the system. For example, a 'User' provides a "Help Request" and "Credentials," and in return receives "Notifications."

4.2 Data Flow Diagram: Level 1

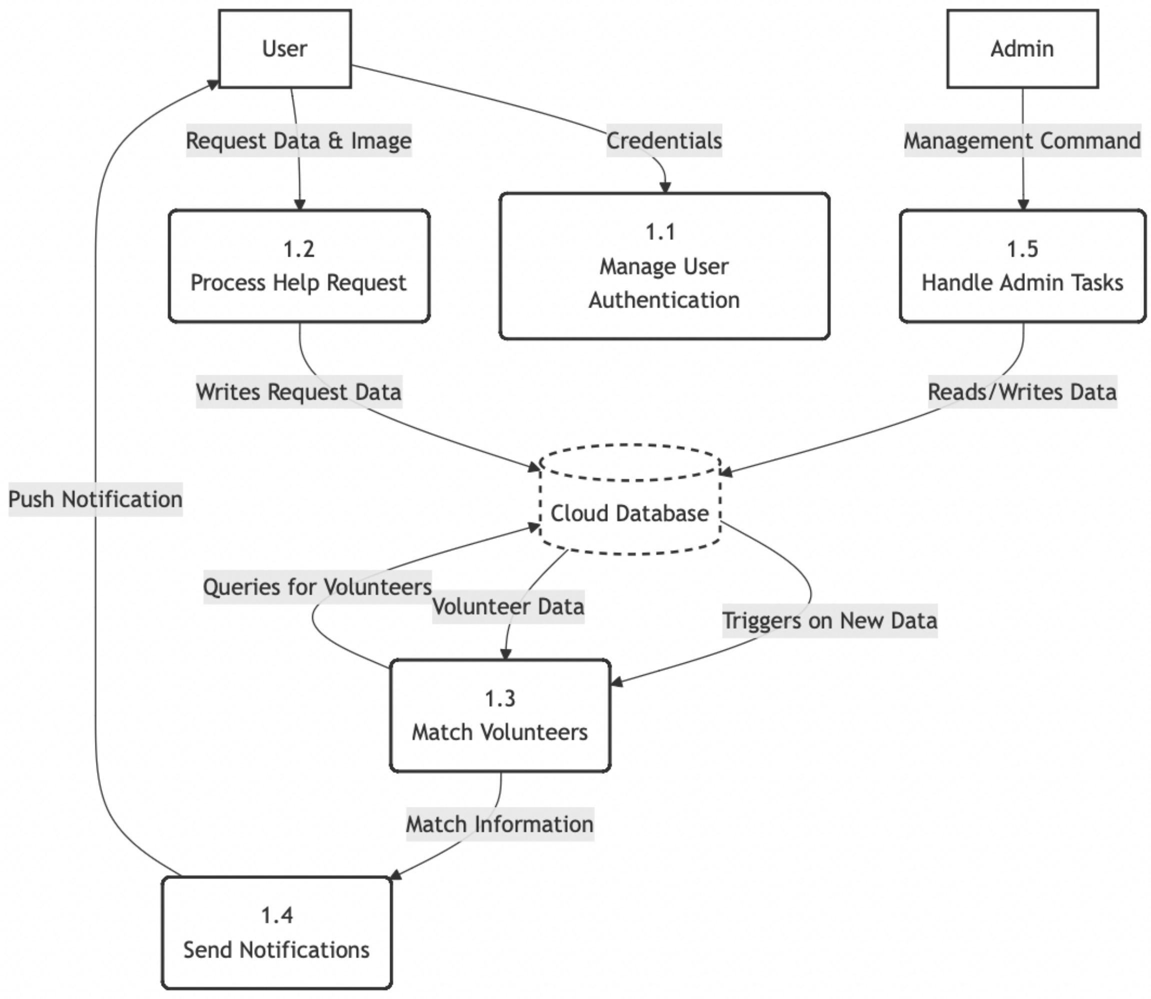

Figure 2: Level 1 Data Flow Diagram showing the major sub-processes.

Explanation: This Level 1 DFD shows the single process from the Context Level diagram into its major internal functions. It shows how data moves between these sub-processes and the central ‘Cloud Database‘ data store. Key processes include ‘Manage User Authentication‘ (1.1), ‘Process Help Request‘ (1.2), and ‘Match Volunteers‘ (1.3). A critical feature illustrated here is the asynchronous flow: the ‘Cloud Database‘ sends a ”Triggers on New Data” flow to the ‘Match Volunteers‘ process, clearly showing the event-driven nature of our architecture.

4.3 Activity Diagram: Submitting a Help Request

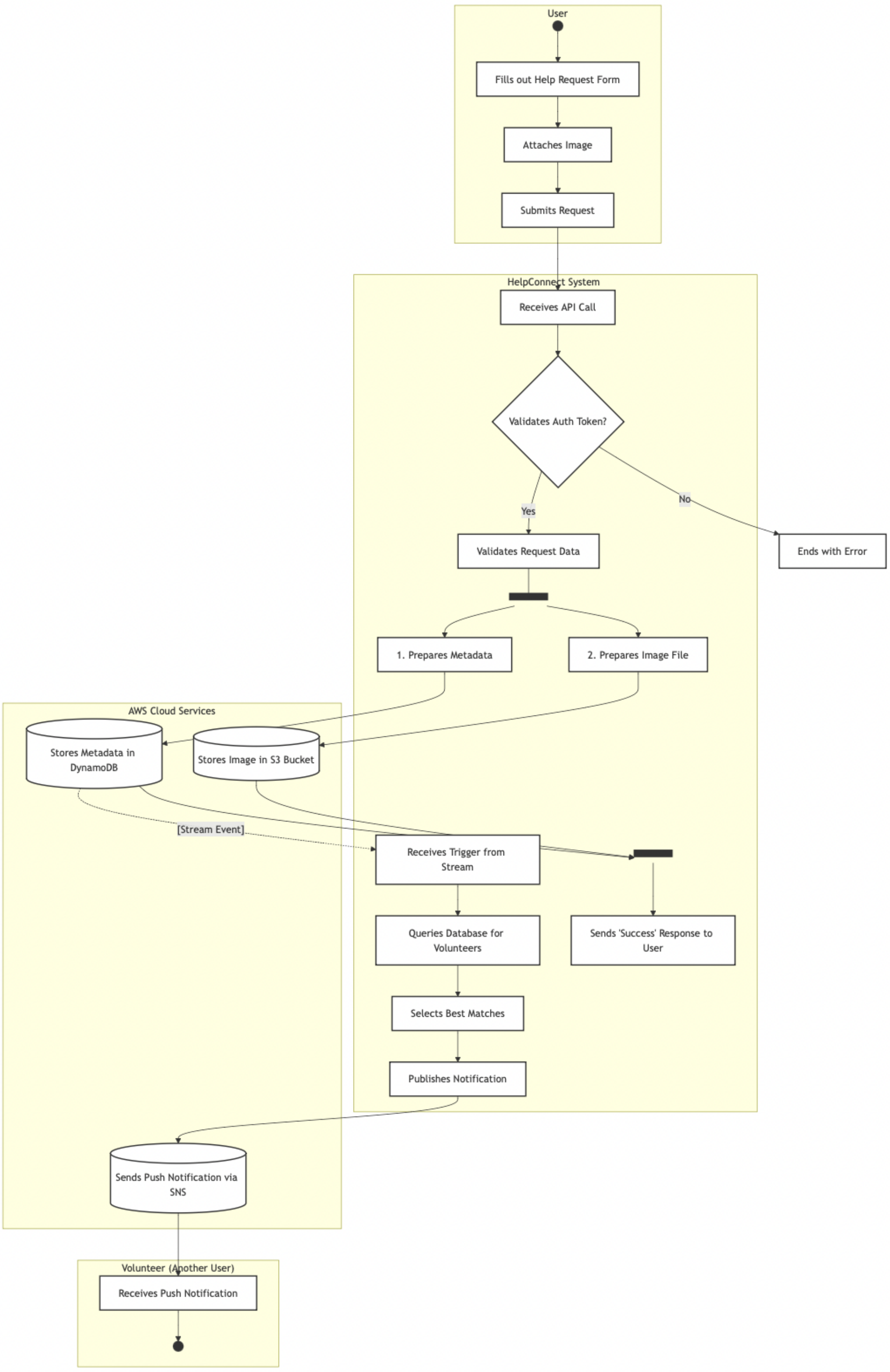

Figure 3: Activity Diagram illustrating the workflow for a user submitting a help request.

Explanation: This diagram details the step-by-step workflow for the primary use case of submitting a help request. The swimlanes clearly define responsibility for each action among the ‘User‘, the ‘HelpConnect System‘ (our application logic), the underlying ‘AWS Cloud Services‘, and a ‘Volunteer‘. The diagram correctly uses formal UML symbols, including:

- **Decision Node:** To validate the user’s auth token.
- **Fork/Join Bars:** To show that storing metadata and the image file are parallel actions that must both complete before a success response is sent to the user.
- **Asynchronous Trigger:** A dotted arrow labeled ”[Stream Event]” shows that the matching logic is triggered in the background by a database event, decoupled from the initial user submission.

5 Computation

The HelpConnect system will perform several key computational tasks to function, primarily within the AWS Lambda functions. These tasks involve processing data to facilitate the matching and notification processes.

- **Authentication and Authorization:** For every API request, the system will compute the validity of the user’s session by validating the Cognito-issued JWT (JSON Web Token). This ensures that all actions are performed by authenticated and authorized users.
- **Geospatial Matching Algorithm:** This is the core computation of the application. When a new help request is created, the system will:
  a. Retrieve the geographic coordinates (latitude and longitude) of the new request.
  b. Query the DynamoDB database for active volunteers or offers within a predefined radius.
  c. Compute a ”match score” for each potential volunteer based on multiple factors, including distance, urgency level of the request, and matching categories.
  d. Rank the results to produce a list of the most suitable candidates to notify.
- **Data Validation:** Before any data is written to the database, the backend will perform computational checks to validate the integrity of the incoming data from the client. This

includes checking data types, ensuring required fields are present, and sanitizing inputs to prevent errors or security vulnerabilities.

- **Notification Targeting:** After the matching algorithm identifies the best candidates, the system computes the specific notification payload (the message content) and determines the exact list of user device endpoints to send the alert to via Amazon SNS.

6 Data Types

The HelpConnect system will process and store several distinct types of data to function effectively. The primary categories are text, numeric, and binary data, which are handled by different AWS services.

- **Text and Structured Data (JSON):** This is the most common form of data, transmitted as JSON between the Flutter client and the backend API. It includes:
  - **User Information:** User ID (from Cognito), name, email, and user-provided profile details.
  - **Request Metadata:** A textual description of the help request, its category (e.g., ”Medical,” ”Technical”), urgency level, and current status (e.g., ”Open,” ”Fulfilled”).
  - **System Metadata:** Unique IDs for requests, timestamps, and log entries.
- **Numeric Data:** These values are typically part of the JSON objects but are crucial for computation:
  - **Geographic Coordinates:** Latitude and longitude values used for the geospatial matching algorithm.
  - **Timestamps:** Unix timestamps to record when requests are created, updated, or fulfilled.
- **Binary Data:** This refers to non-text files uploaded by users:
  - **Images:** Photos (e.g., JPEG, PNG) uploaded by users to provide visual context for their help requests. This binary data is stored directly as objects in Amazon S3.

7 Expected Contribution for Each Member

- **Noyan Saat (Frontend Lead):** Develops the Flutter-based UI for both mobile and web, ensuring an adaptive and responsive user experience. Builds the authentication flow, user forms, and the admin dashboard. Integrates the client with the AWS backend via REST APIs.
- **Ece Kocabay (Backend Developer):** Implements the serverless business logic in AWS Lambda. Develops the REST API endpoints and the event-driven matching algorithm triggered by DynamoDB Streams. Designs the DynamoDB data model and configures SNS notification logic.
- **Ali Saadettin Yaylagül (Cloud Engineer):** Manages the end-to-end AWS cloud infrastructure setup (Cognito, S3, DynamoDB, IAM policies). Configures API Gateway routes and Lambda triggers. Deploys the Flutter Web admin panel via S3 and CloudFront. Ensures the architecture is secure and scalable.

8 Project Milestones

Week	**Milestones**
Oct 20-26	Finalize proposal; set up Flutter project and GitHub repository; define DynamoDB schema and initial AWS IAM roles.
Oct 27-Nov 2	Develop Cognito authentication flow; implement core API endpoints (e.g., create request) with Lambda and API Gateway.
Nov 3-Nov 16	Implement the event-driven matching logic with DynamoDB Streams and the SNS notification system. Build the basic admin panel UI.
Nov 17-Nov 30	End-to-end testing, debugging, and performance optimization. Write and submit the Progress Report.
Dec 1-Dec 23	Finalize UI/UX, add monitoring metrics and backup scripts. Prepare final demo, presentation, and submit Final Report.

9 References

- Amazon Web Services (AWS) Documentation – Lambda, DynamoDB, S3, Cognito, SNS, CloudFront: `https://docs.aws.amazon.com`
- Flutter Official Documentation: `https://docs.flutter.dev`
- Unified Modeling Language (UML) Specification: `https://www.omg.org/spec/UML/`
- CNG 495 Capstone Project Instructions (Fall 2025–2026)