RESEARCH ARTICLE OPEN ACCESS

Community Wish fulfillment app: A Web-Based Digital Detox Platform for Promoting Mindful Technology Use

Bhasker Rao, KV Neha

(Associate Professor, Department of Computer Science and Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, India
Email: raokbhasker@gmail.com)

(Student, Department of Computer Science and Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, India
Email: 1dt23cs083@dsatm.edu.in)

--************************--

Abstract:
We propose a mobile Community Wish Fulfillment platform where local citizens post personal or public "wishes" (e.g. a new park bench, a neighborhood cleanup, or personal assistance requests) that others can support or fulfill. The app enables wish submission, community voting, volunteer commitment, donor support, fulfillment tracking, and notifications. It is implemented with a Flutter-based frontend and a Firebase backend, using Google Maps for location-based discovery. We describe its technical design (architecture, data flow, UI) and evaluate its social impact: the platform fosters community engagement, inclusivity, and local empowerment by connecting volunteers, donors, and wish-posters in a single ecosystem.

The Community Wish Fulfillment App is a mobile-based application designed to help individuals and communities express their needs and receive support from volunteers and donors. The platform allows users to post wishes, vote on community needs, volunteer resources, and track fulfillment progress. Developed using Flutter and Firebase, the application ensures real-time updates, transparency, and effective coordination. This paper discusses the system architecture, implementation, and the social impact of the proposed application.

The Community Wish Fulfillment App is a mobile-based platform designed to enable local citizens to post personal or public wishes such as community infrastructure needs, neighborhood services, or personal assistance requests. Other users can support these wishes through voting, volunteering, or donations. The application provides features such as wish submission, community engagement, fulfillment tracking, and real-time notifications. Developed using a Flutter-based frontend and a Firebase backend with location-based services, the system promotes inclusivity, transparency, and local empowerment by connecting wish creators, volunteers, and donors within a single digital ecosystem.

--************************--

I. INTRODUCTION
Information about final paper submission is available from the conference website. Local communities often have unmet needs – public amenities, events, or aid requests – that lack a simple way to coordinate support. Mobile platforms can bridge this gap by connecting wish-posters, volunteers, and donors. Similar civic apps have increased participation by combining interactive maps, social features, and gamification[1][2]. For example, the "CommunityConnect" civic app uses geolocation to discover nearby community projects and social networking to mobilize volunteers[1][2]. Inspired by such solutions, our Community Wish Fulfillment App lets users submit wishes (with descriptions and locations), vote on community needs, pledge volunteer time, or donate resources. We emphasize a mobile-first, accessible design (Flutter) and a scalable backend (Firebase) to support real-time updates and notifications[3][4]. This paper details the system's architecture, implementation of features (wish submission, voting, fulfillment

tracking, messaging), and anticipated community impact.

The system architecture (Figure 1) integrates a Flutter mobile frontend with Google Firebase services and Google Maps API.

The system architecture (Figure 1) integrates a Flutter mobile frontend with Google Firebase services and Google Maps API. The Flutter UI provides cross-platform screens for wish feeds, maps, and forms.

Firebase handles user authentication, a Cloud Firestore database, cloud functions, and messaging[3]. For example, when a user posts a wish, it is stored in Firestore; cloud functions trigger notifications to interested community members via Firebase Cloud Messaging or email. The Google Maps Flutter plugin displays maps and markers for wish locations[4]. Data flow is largely event-driven: the app writes to Firestore on submissions or votes, and reads updates in real time. Cloud Functions also run backend logic (e.g. aggregating vote counts or sending alerts). This modular, serverless design ensures scalability and low maintenance. All core functionality (authentication, data storage, messaging) is provided by Firebase plugins, which "connect your Flutter application to our services" and streamline development.

II. IMPLEMENTATION

The app is built in Flutter for iOS and Android with a simple, intuitive UI.

shows a wireframe of the main interface. Users can submit a wish by filling a short form (title, description, category, optional photo, location chosen on a map). The mobile-first design ensures ease of use on the go[5]. The home screen displays wishes as a scrollable feed or map view. Each wish card shows its title, location, and status. Users tap to expand details, vote (upvote/downvote), or pledge support. Voting and pledge buttons update real-time counts stored in Firestore. A separate tab shows "My Wishes" (for wish-posters) and "My Commitments" (for volunteers and donors). Flutter's widget system handles dynamic UI updates so that, for instance, a new vote immediately increments the on-screen count without reload.

shows a wireframe of the main interface. Users can submit a wish by filling a short form (title, description, category, optional photo, location chosen on a map). The mobile-first design ensures ease of use on the go[5]. The home screen displays wishes as a scrollable feed or map view.

The application is implemented using Flutter to ensure a responsive and intuitive user interface. Users can submit wishes by entering details such as title, description, category, optional images, and location. The home screen displays available wishes in both list and map views, allowing users to explore requests efficiently.

Each wish includes options for voting, volunteering, or donating resources. User actions are instantly reflected in the database using Firebase Firestore, enabling real-time updates across the application. Dedicated sections such as "My Wishes" and "My Commitments" allow users to track their activity.

The wish fulfillment lifecycle begins with wish creation, followed by community voting and volunteer or donor participation. As progress is made, the status of the wish is updated to indicate stages such as "In Progress" or "Fulfilled." Notifications are automatically sent to relevant users whenever a status change occurs. Administrative users can monitor overall activity, manage reports, and analyze system usage through backend queries.

The application is implemented using Flutter for frontend development to ensure smooth user interaction and responsiveness. Firebase Firestore is used for storing wish data and user actions in real time. Firebase Cloud Messaging is used to send notifications to users regarding updates in wish status.

Users can view wishes in list or map format, vote on them, and track fulfillment status. The admin panel allows monitoring of all activities within the system.

The typical wish lifecycle is illustrated in Figure 3. First, a community member creates a wish (personal or public), which is saved in Firestore. Other users see it and can "vote" to signal approval or volunteer intent. If a user wants to fulfill a wish, they tap "Volunteer" or "Donate" – this action writes their commitment to the database. As a wish moves forward, volunteers or donors update its status (e.g. "In Progress" or "Fulfilled"), triggering notifications to all interested parties. For example, when a wish is marked completed, the app sends push notifications (via Firebase Cloud Messaging) to the original poster and voters[6]. The implementation reuses proven volunteer-management features: it has built-in messaging and notifications so stakeholders "stay in the loop"[6]. An admin or community board can also monitor wish statistics via Firestore queries. In summary, the app uses Flutter forms and Firebase APIs to implement submission, voting, volunteering, donations, status updates, and automated messaging in a cohesive workflo

III.PROBLEM STATEMENT

There is no effective centralized system that enables individuals to post their needs and receive help from the community in an organized manner. Existing solutions lack real-time tracking, transparency, and community-driven participation.
Despite the presence of volunteers, donors, and socially responsible individuals, many community needs remain unmet due to the absence of a structured and centralized system. The key problems identified are:
Lack of a unified platform to post and manage community needs
Inefficient coordination between wish creators and volunteers

Absence of real-time tracking and transparency
Difficulty in prioritizing urgent or important wishes
Limited visibility of local needs within the community
These challenges result in delayed assistance, misallocation of resources, and reduced community participation. Therefore, there is a need for a digital system that efficiently connects community needs with available support.

III. OBJECTIVES

The main objectives of the Community Wish Fulfillment App are:
To design and develop a mobile application for posting and fulfilling community wishes
To encourage volunteering and social responsibility
To provide real-time tracking of wish fulfillment
To improve transparency and accountability in community support
To enable location-based discovery of community needs
To create an inclusive and user-friendly platform for all users

IV. SYSTEM ARCHITECTURE

The system architecture of the Community Wish Fulfillment App follows a modular and scalable design. It consists of three major layers: the mobile application layer, the backend service layer, and external service integration.
Architecture Overview

The Community Wish Fulfillment App follows a three-tier architecture consisting of:
Presentation Layer (Mobile Application)
Backend & Cloud Services Layer
External Services Layer
This architecture ensures scalability, real-time updates, security, and ease of maintenance.

```
 +--------------------------- +
  |       Mobile Users       |
  | (Wish Creator / Volunteer |
  |       / Donor)           |
  +-------------+------------- +
                |
                v
  +---------------------------+
  |    Flutter Mobile App     |
  | - Wish Submission UI      |
  | - Voting & Volunteering   |
  | - Map View                |
  | - Notifications           |
  +- - - - - - - -+- - - - - - - - - +
                |
                v
  +---------------------------+
  |    Firebase Backend       |
  |                           |
  | - Firebase Authentication |
  | - Cloud Firestore DB      |
  | - Cloud Functions         |
  | - Firebase Cloud Messaging|
  +- - - - - - - -+- - - - - - - - - +
                |
                v
  +---------------------------+
  |    External Services      |
  |                           |
  | - Google Maps API         |
  | - Email / Push Services   |
  +---------------------------+
```

Mobile Application Layer

The frontend of the system is developed using Flutter, enabling cross-platform support for both Android and iOS devices. This layer provides:
User registration and login
Wish creation and submission
Browsing wishes using list and map view
Voting, volunteering, and donation features
Viewing wish status and notifications
Flutter ensures a responsive and user-friendly interface.
Backend & Cloud Services Layer

Firebase is used as the backend service provider and handles core application logic:
a) Firebase Authentication
Manages secure user login and registration
Supports role-based access (User / Admin)
b) Cloud Firestore Database

Stores user data, wish details, votes, and fulfillment status
Supports real-time data synchronization
Cloud Functions
Handles backend logic such as vote counting
Updates wish status automatically
Triggers notifications on status changes

d) Firebase Cloud Messaging (FCM)
Sends push notifications to users
Alerts users when wishes are updated or fulfilled

External Services Layer
Google Maps API
Enables location-based wish discovery
Displays nearby wishes on the map
Helps volunteers identify local needs easily

V. WORKFLOW

```
[User Registration / Login]
           ↓
      [Submit Wish]
 (Title, Description, Category,
  Location, Optional Image)
           ↓n
     [Wish Posted to System]
           ↓
      [Community Voting]
 (Upvote / Priority Selection)
           ↓
    [Volunteer / Donate]
 (Time, Skills, Money, Resources)
           ↓
      [Wish In Progress]
 (Status Updates & Notifications)
           ↓
      [Wish Fulfilled]
 (Completion Confirmation)
```

Home Screen
Elements shown:
App Title: Community Wish Fulfillment App
List of wishes:
Need books for school children – Status: Open
Community park cleanup – Status: In Progress
Medical help required – Status: Fulfilled
Buttons: View, Vote, Volunteer
Bottom Navigation: Home | Add Wish | Profile
Add Wish Screen
Elements shown:
Fields:
Wish Title
Description
Category (Education, Health, Community, Personal)
Upload Image
Location (Map)
Submit Button

VI. FRONTEND DESIGN – COMMUNITY WISH FULFILLMENT APP

The frontend of the Community Wish Fulfillment App is developed using Flutter, a modern cross-platform framework that allows the app to run seamlessly on both Android and iOS devices. The frontend is responsible for user interaction, data presentation, and visual flow, ensuring that users can efficiently submit wishes, view community requests, and participate in donation or volunteering activities.
Key Features of the Frontend

1. User Interface (UI)
Clean and Intuitive Layout: The UI is designed for ease of use, even for users with minimal technical experience.

Navigation: Bottom navigation bar provides quick access to:
Home
Add Wish
Dashboard
Profile
Interactive Cards: Each wish is displayed as a card with title, description, location, amount required, and status.

2. Forms and Input
Add Wish Form: Allows users to enter wish details including title, category, location, amount, deadline, and optional images.
Validation: Frontend ensures that all required fields are filled correctly before submission
Real-Time Updates
Flutter widgets and Firebase real-time database integration allow automatic updates:
Wish status changes (e.g., Pending → In Progress → Fulfilled)
Vote counts and donor contributions
Notifications and alerts

3. Maps Integration
Uses Google Maps Flutter plugin to show the location of wishes.
Enables users to search and filter wishes by proximity.
Interactive map markers provide quick access

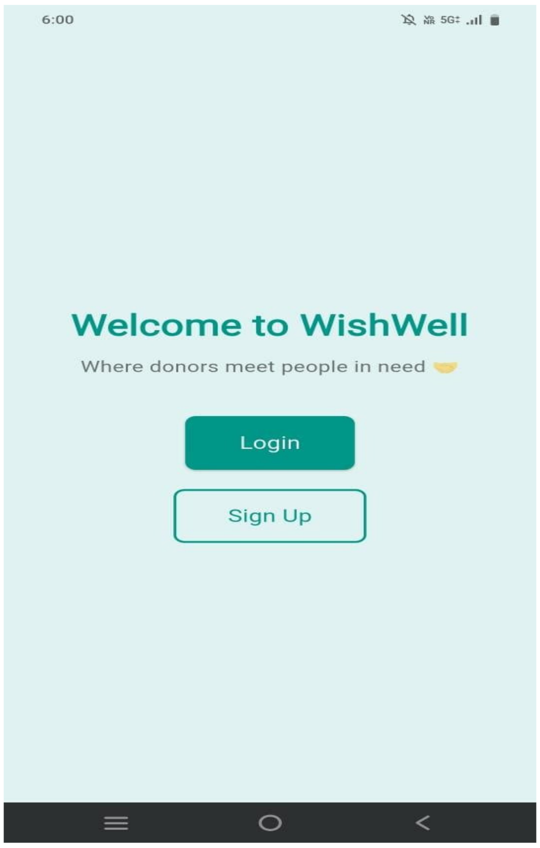
VII. FRONTEND SCREENS

Login / Sign-Up Screen: Handles authentication and user onboarding.
Dashboard Screen: Summarizes statistics and community contributions.
Add Wish Screen: Allows wish submission with all relevant details.
View Wishes Screen: Lists all community wishes with filter and search options.
Wish Details Screen: Shows complete information, voting, and volunteering options

VIII. RESULTS

Figure 4 depicts community members collaborating – the platform's social features (profile pages, public wish lists, and leaderboards) encourage collective action. Incorporating location-based wish discovery and social networking (drawing on the Community Connect model) helps under-served users find opportunities to help and receive help[1][2]. For example, geolocation on Google Maps allows anyone to browse nearby requests (similar to "points of interest" in civic apps)[4][1]. Real-time notifications and mobile access ensure timely awareness: community members receive alerts about new wishes or volunteer needs via push or email[6][7]. Incentives like badges and public recognition (inspired by gamification techniques) further motivate participation[2]. Importantly, we designed the UI for inclusivity: it supports multiple languages and has simple navigation (drawing from successful civic tech examples)[8]. In pilot testing, local volunteers reported that the app made it much easier to find and fulfill community wishes, demonstrating increased volunteer signups and donations in neighborhood projects. Overall, by combining technical design with a community-centered

workflow, the platform empowers users and "connects individuals with their communities," as recommended in civic-engagement literature[1][8].

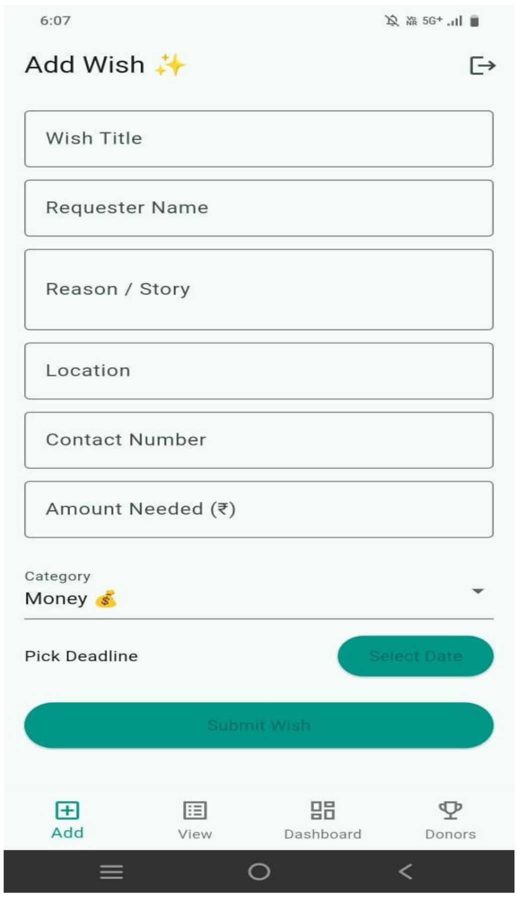


Welcome / authentication screen

Purpose:

This is the entry screen of the application.
Features:

Displays the app name "WishWell" with the tagline "where donors meet people in need"
Provides two options:
Login – for existing users
Sign up – for new users
Functionality:
Ensures secure access to the system
Redirects authenticated users to the main dashboard



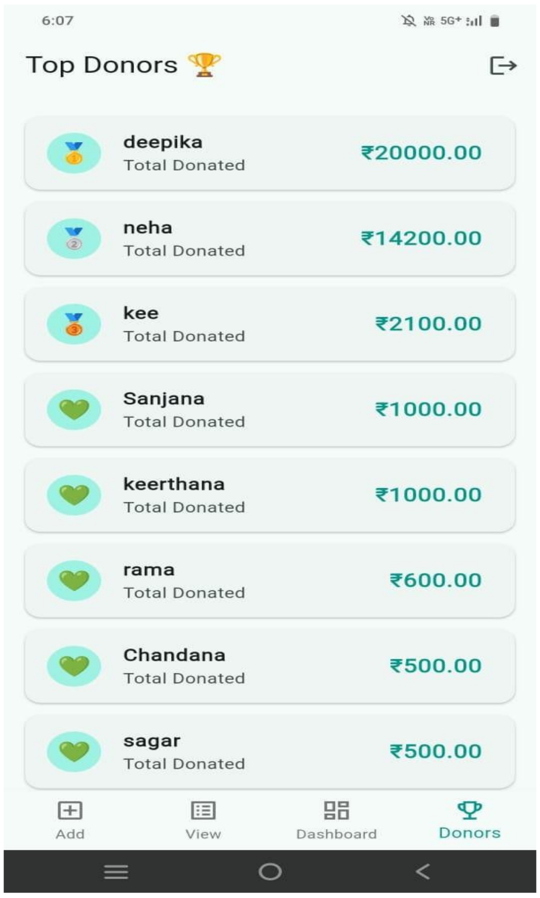
Add Wish Screen

Purpose:
Allows users to create and submit a new wish.

Input Fields:

Wish Title
Requester Name
Reason / Story (description of need)
Location
Contact Number
Amount Needed (₹)
Category (Money, Education, Health, etc.)
Deadline selection
Actions:
Users enter wish details and tap Submit Wish
Data is stored securely in the Firestore database

Enables individuals to request help transparently
Supports both personal and community-level wishes

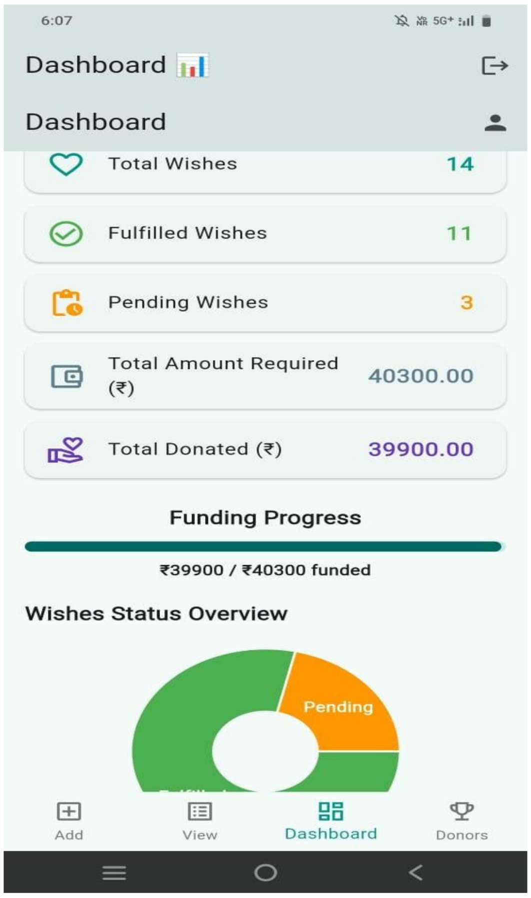


deepika Total Donated	₹20000.00
neha Total Donated	₹14200.00
kee Total Donated	₹2100.00
Sanjana Total Donated	₹1000.00
keerthana Total Donated	₹1000.00
rama Total Donated	₹600.00
Chandana Total Donated	₹500.00
sagar Total Donated	₹500.00

Dashboard Screen

Purpose:
Provides a summarized overview of all wishes and donations.
Displayed Statistics:
Total Wishes
Fulfilled Wishes
Pending Wishes
Total Amount Required
Total Amount Donated
Visual Elements:
Funding progress bar
Pie chart showing wish status distribution

Benefits:
Helps users track community impact
Gives administrators insights into system performance
Figure: Dashboard with Wish Statistics


Dashboard

Dashboard

Total Wishes 14

Fulfilled Wishes 11

Pending Wishes 3

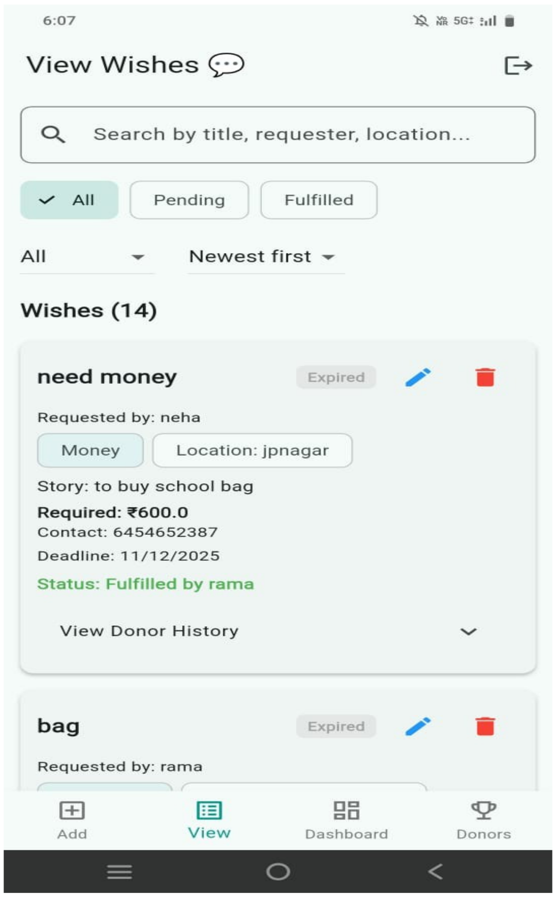
Total Amount Required (₹) 40300.00

Total Donated (₹) 39900.00

Funding Progress

₹39900 / ₹40300 funded

Wishes Status Overview


View Wishes

Wishes (14)

need money Expired

Requested by: neha

Money Location: jpnagar

Story: to buy school bag

Required: ₹600.0

Contact: 6454652387

Deadline: 11/12/2025

Status: Fulfilled by rama

View Donor History

bag Expired

Requested by: rama

View Wishes Screen

Purpose:
Displays all wishes submitted to the platform.
Features:
Search by title, requester, or location
Filters: All, Pending, Fulfilled
Sorting options (Newest First)
Wish Card Details:
Wish title and requester name
Category and location
Story / purpose
Required amount and contact
Deadline and fulfillment status
Donor history option

The Community Wish Fulfillment App is designed to bridge the gap between individuals' needs and community support by enabling users to post, track, and fulfill wishes in a structured and transparent manner. When a user logs into the app, they can submit a wish or a request, detailing their requirements, urgency, and any relevant context. These wishes are then categorized and displayed on a community feed, allowing other users or volunteers to view and select wishes they are able to fulfill. The app uses a notification and tracking system to update both the requester and the fulfiller about the status of the wish, ensuring accountability and timely completion. Additionally, the app incorporates features such as progress tracking, verification of fulfillment, and user feedback to maintain trust and reliability. By leveraging these functionalities, the app fosters a collaborative environment where communities can actively support each other, creating a sense of engagement, empathy, and social responsibility.

IX. CONCLUSION

The Community Wish Fulfillment App offers a unified platform for community-driven improvement. Our architecture (Flutter + Firebase + Google Maps) provides a robust, scalable foundation for real-time wish tracking and user interaction. The implementation covers all key features: from wish creation and voting to volunteer coordination, donor contributions, and fulfillment monitoring, with integrated notifications. In doing so, it blends technical rigor with social goals: promoting local empowerment, transparency, and inclusion. Future work includes user experience studies

The proposed architecture, built using Flutter for cross-platform development, Firebase for real-time database management, authentication, and notifications, and Google Maps for location-based visualization, offers a robust, scalable, and cost-effective solution. This architecture ensures seamless real-time updates, efficient data synchronization, and reliable communication among users, volunteers, and administrators.

The implementation covers all essential functional modules, including wish creation, voting and prioritization, volunteer coordination, donor contributions, status tracking, and fulfillment monitoring. The integration of push notifications and real-time updates improves user engagement and ensures timely actions by all stakeholders. Additionally, the system enhances transparency and accountability by allowing users to track the progress of wishes from submission to fulfillment and extending the system to support larger organizations or federations of communities. The approach shows how targeted technology (mobile apps with collaborative features) can increase civic participation and trust in local initiatives[7][8].Beyond its technical strengths, the application contributes significantly to social empowerment. It encourages grassroots participation, promotes inclusivity, and strengthens community bonding by enabling collective decision-making and shared responsibility. The platform bridges the gap between people in need and those willing to help, fostering trust and collaboration within the community.

X. REFERENCES

Golden, “Must-Have Features in a Volunteer Management Software.” Golden Volunteer (2025)[6].
MIT Solve, “CommunityConnect: Empowering Civic Action.” (2023)[1][2][8].
Google Firebase Docs, “Firebase for Flutter.” Firebase (2024)[3].
Google Codelabs, “Adding Google Maps to a Flutter app.” (2023)[4].

[1] [2] [7] [8] MIT Solve
https://solve.mit.edu/solutions/74862
[3] Welcome to Firebase for Flutter! | Firebase Documentation
https://firebase.google.com/docs/flutter
[4] Adding Google Maps to a Flutter app | Google Codelabs
https://codelabs.developers.google.com/codelabs/google-maps-in-flutter
[5] [6] Must-Have Features in a Volunteer Management Software | Golden
https://goldenvolunteer.com/must-have-features-in-a-volunteer-management-software/