The Islamic University of Gaza

Faculty of Information

Technology

الجامعة الإسلامية بغزة

كلية تكنولوجيا المعلومات

Gaza Madad Flow

A Web Automation Model for Automating Humanitarian Aid Registration in Gaza

غزة مدد فلو

نموذج أتمتة لتسجيل المساعدات الإنسانية في غزة عبر الويب

A Graduation Project Presented to the Faculty of Information Technology, the Islamic University of Gaza In Fulfillment for the Degree of Bachelor in Information Technology

By

Maryam Refaa Skaik - 220211310

Rania Raid Kashkask - 220210245

Aya Nabil Alharazin - 220210713

Misk Saad Ashour - 220211137

Alaa Shareef Yousef - 220211783

Supervised by

Eng. Mohammed El-Agha

October, 2025

Abstract

In emergencies, technology helps people respond quickly and organized. It makes work easier by reducing manual tasks, sorting data faster, and ensuring aid goes to the right people on time.

During the October 7 war in Gaza, many people needed urgent help, but there were problems like power outages and unstable internet. In such cases, traditional manual help registration becomes slow and difficult.

Most aid registration systems in the world are either used only by governments and big organizations or are simple online forms. These systems don't easily send data to multiple platforms and are not good for places with few resources.

The goal is to create Gaza Madad Flow, a simple web tool that allows users to register once and have their data automatically sent to many aid platforms. We use tools like Laravel framework, PostgreSQL, Google Forms, Google Sheets, n8n and render to build and test this process, making sure it works well in real situations.

In this project, we use a method called Incremental Prototyping. We started with a small-scale version that collected data using a simple online form to test whether the idea would work. After validating this approach, we gradually improved the system in stages, introducing a custom web interface to apply our own validation rules and make data entry more reliable. This step-by-step process allowed us to identify problems early, refine features gradually, and develop a version ready for real-world use.

We present testing results that show automation reduces time, manual duplication, and data loss, even under Gaza's weak internet and power conditions. The project demonstrates how web automation and integration tools offer a practical, scalable, and humanitarian technology solution for crisis environments.

ملخص الدراسة

في حالات الطوارئ، تساعد التكنولوجيا الناس على الاستجابة بسرعة وتنظيم العمل .فهي تسهل العمليات من خلال تقليل المهام اليدوية، وتسريع فرز البيانات، وضمان وصول المساعدات إلى المستحقين في الوقت المناسب.

خلال حرب 7أكتوبر في غزة، كان هناك الكثير من الأشخاص الذين يحتاجون إلى مساعدات عاجلة، ولكن واجهتهم مشاكل مثل انقطاع الكهرباء وضعف الإنترنت .في مثل هذه الحالات، يصبح تسجيل المساعدات اليدوي التقليدي بطيئًا وصعبًا.

معظم أنظمة تسجيل المساعدات حول العالم إما مخصصة فقط للحكومات والمنظمات الكبيرة، أو مجرد نماذج إلكترونية بسيطة .هذه الأنظمة لا تسمح بسهولة بإرسال البيانات إلى منصات متعددة، كما أنها غير مناسبة للأماكن ذات الموارد المحدودة.

الهدف من Gaza Madad Flowهو توفير أداة ويب بسيطة وموثوقة تتيح للمستخدمين التسجيل مرة واحدة فقط، حيث تُرسل بياناتهم تلقائيًا إلى عدة منصات للمساعدات .اعتمدنا في بناء واختبار هذه العملية على أدوات مثل (Laravel framework, PostgreSQL, Google Forms, Google Sheets, n8n and Render) مع التأكد من فعاليتها في الظروف الواقعية.

في هذا المشروع، استخدمنا منهجية تُسمى Incremental Prototypingبدأنا بنسخة صغيرة لجمع البيانات باستخدام نموذج إلكتروني بسيط لاختبار الفكرة .بعد التحقق من جدوى هذا النهج، قمنا بتحسين النظام تدريجيًا على مراحل، من خلال تطوير واجهة ويب مخصّصة تتيح تطبيق قواعد تحقق خاصة بنا وجعل إدخال البيانات أكثر موثوقية .هذا النهج التدريجي ساعدنا على اكتشاف المشكلات مبكرًا، تحسين الميزات بشكل متواصل، وتطوير نسخة جاهزة للاستخدام الفعلي.

أظهرت نتائج الاختبارات أن الأتمتة تقلل الوقت والازدواجية اليدوية وفقدان البيانات، حتى في ظل ظروف الإنترنت والكهرباء الضعيفة في غزة .يوضح المشروع كيف يمكن لأدوات الأتمتة والربط بين المنصات أن توفر حلولًا عملية وقابلة للتوسع وذات طابع إنساني في بيئات الأزمات.

Dedication

بِسْمِ اللهِ الرَّحْمنِ الرَّحِيمِ

"وَقُلِ ٱعْمَلُواْ فَسَيَرَى ٱللَّهُ عَمَلَكُمْ وَرَسُولُهُ وَٱلْمُؤْمِنُونَ"

بمدادٍ من محبة وامتنان، وبحروفٍ تفيض نغمةً من الوفاء والتقدير، نخطُّ في هذه السطور نبراسًا من العرفان لكل من كان له بصمة في إنضاج هذا العمل وتحقيقه إلى واقعٍ ينبض بالحياة.

نُعَبِّر عن بالغ الشكر والعرفان للجامعة الإسلامية – كلية تكنولوجيا المعلومات، هذا الصرح العلمي الشامخ، الذي كان لنا حاضنة للفكر وضياءً ينير دروب العلم، ومساحةً آمنة لصقل الطاقات، رغم عصف الأزمات ووطأة العدوان، فجزاها الله خير الجزاء على ما أولتنا من عناية واهتمام.

ولا تفوتنا فرصةُ الثناء والتقدير لمهندس محمد الأغا، مشرف المشروع، على توجيهاته السديدة ودعمه المستمر، وعلى صبره الحاني وملاحظاته البنّاءة التي أسهمت في رفع جودة العمل، وجعلت من هذا المشروع رسالةً علميةً تتلألأ بين سطور الإنجاز.

كما نوجه التحية إلى جميع أعضاء الهيئة الأكاديمية الذين لم يبخلوا بعلمهم، وبتوجيهاتهم السديدة، فهم أنوارٌ نهتدي بها في دروب المعرفة، وسراجٌ ينير لنا السبيل نحو الإبداع والتميز.

ولأن لكل إنجازٍ قصة وطن، فإننا نُهدي هذه الثمرة من العطاء إلى شهدائنا الأبرار، الذين ارتقوا فداءً للكرامة والحق، وإلى جرحانا الصامدين، الذين كتبوا بالعزيمة والتضحية دروسًا في الصمود، وإلى أسرانا الأبطال في سجون الاحتلال، الذين صاغوا من عذابات السجون عهدًا لا يذبل، وشهادةً على شموخ الإنسان وإرادته.

وإلى أهلنا في غزة، الذين صبروا واحتسبوا، فكانوا نبراسًا لثباتنا، وحافزًا لاستمرارنا، رغم شدّة المحن وقلّة الإمكانيات، فكل الشكر والتقدير لصمودهم العظيم.

ولا ننسى أن نعبر عن امتناننا العميق لعائلاتنا وزملائنا، الذين كانوا سندًا ورفيقًا لنا في أوقات التعب، وشركاء في إنجازنا، فهم عمادُ عزيمتنا وسبب فخرنا.

وفي ختام هذا البيان، نسأل الله أن يتقبل هذا العمل خالصًا لوجهه الكريم، وأن يكون لبنةً في بناء مجتمعٍ أقوى، ووطنٍ أسمى، وأن يُسهم في رفعة العلم والمعرفة، فنجعل منه مشكاةً تنير الدروب، وسراجًا يُضيء دروب المستقبل المشرق.

Acknowledgment

إلى أهلنا الأكارم، يا من حملتم في قلوبكم عبق الحنان، وذرتم من رحيق الحب والوفاء ما لا يَنْضَبُ، كنتم شموسًا تشرق في سماء حياتنا، وأمانًا يحمي دروبنا من عواصف الزمان، بكم تعلمنا كيف تكون الإرادة صلبة، والصبر متينًا، وبدعائكم ثبتت أقدامنا على طريق العلا، فلكم منا أسمى التحايا وأطيب الأمنيات، أنتم الجذور الراسخة التي تنبت في عمق الصلابة مجدًا وسؤددًا.

ولأحبائنا الذين غادرونا، أهل الذكرى الطاهرة، أنتم في القلب نبض لا ينتهي، تسكنون أروقة الذاكرة كأنكم بيننا، تهمسون بألحان الرجاء، وتفيض محبتكم كالندى الذي يروي ظمأ الروح، فذكراكم زاد الفؤاد عطاءً، وإشراقة في الليالي الحالكة، فلكم من الدعاء أسمى آياته، وأطيب أمنياته أن تظلوا في عليين، في ضيافة الرحمن، محفوظين في القلب والوجدان.

أما شهداؤنا الأبرار، أنتم سُفر النبل والبطولة، دماؤكم الزكية أمدَّت أرواحنا بقوة العز والإقدام، خططتم بملاحمكم المجيدة أنوارًا لا تنطفئ، ونجومًا لا تغيب، وبصمودكم الباسل، نمضي نحو غدٍ مشرق، في أُفقٍ محفوف برؤاكم، وفي القلبِ لكم وُدٌّ لا تَغيبُ شموسُه تنير دروب الأجيال القادمة.

وإلى الوطن الغالي غزة، نبض القلب والروح، رغم الأهوال، تظلّين رمزًا للأمل والكرامة، نجدد لقاؤنا وهواؤنا بمحبتك، ونعهد أن نكون على درب البناء والعطاء، نعمل بصبرٍ وإخلاص، لنصنع مستقبلًا يليق بحُبك وكرامتك، ليظل درْبُنا مرفوع الراية، وسليل المجد يعانق السماء.

وكل من وقف معنا، من أساتذة، ومشرفين، وأهلٍ وزملاء، لكم من القلب أسمى آيات الشكر والوفاء، أنتم من ساندتنا وساهمتم في رسالتنا، فبكم كانت مسيرتنا أزهى وأبهى، فلكم الدعوات أن ينير الله دروبكم، ويبارك سعيكم، ويكتب لكم الخير في الدنيا والآخرة.

نسأل الله العلي العظيم، أن يتقبل منا هذا الجهد، وأن يجعله نورًا يضيء دروبنا في خدمة الإنسانية، وأن يكون بداية لسلسلة من الأعمال التي ترفع راية الوطن عالية، وتبقى منارةً للعز والكرامة بين الأمم.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
API	Abstract Programming Interface
MVP	Minimum Viable Product
NOGs	Non-Governmental Organizations
n8n	Node for Workflow Automation
PoC	Proof of Concept
RPA	Robotic Process Automation
SDLC	Software Development Life Cycle

Chapter 1
Introduction

Chapter 1
Introduction

1.1 Overview

Humanitarian aid plays an important role in helping people who suffer because of conflicts, natural disasters, and famines [1], [2]. In areas affected by war and food shortages, providing emergency help, such as food, water, shelter, and medical care, is necessary to save lives and keep human dignity [1], [2].

To meet the increasing needs, humanitarian organizations use computerized systems that make aid delivery more efficient, transparent, and effective [3]. These digital tools help with processes like registering people in need, distributing supplies, and monitoring aid over time [3]. As technology improves, it creates better ways for many groups to work together and reduce the time it takes to respond to emergencies [3].

Automation also becomes very important in today's humanitarian work [4]. It makes tasks easier and helps deliver aid faster to areas affected by conflict or famine [4]. Modern tools allow organizations to manage aid more accurately from signing up beneficiaries to checking if they qualify and delivering supplies [4]. Automation creates a reliable system that works well even in difficult conditions [4].

Along with automation, AI changes how aid reaches those in need [5]. In crisis zones where time and resources are limited, AI makes aid work faster, smarter, and more efficient [5]. AI-powered tools help predict disasters or hunger early by analyzing large amounts of data, so organizations can act sooner [5]. AI also automates repetitive tasks, like data entry and report creation, which lowers errors and allows staff to focus on important decisions [5]. AI is becoming a big help in humanitarian aid, especially in war zones and poor communities [5].

Automating aid registration is especially important [6]. Instead of using manual forms or asking people to fill out the same information many times, automated systems gather beneficiary data once and send it through backend processes that verify and sort the information [6]. This speeds up aid delivery, reduces errors, and

makes sure people get help quickly [6]. These systems often use online forms, central databases, and connect with other platforms through APIs [6]. By automating registration, aid becomes faster, more scalable, and less prone to mistakes [6].

Many existing systems aim to automate parts of humanitarian work, such as registering beneficiaries and distributing supplies. Different platforms focus on different things: some directly connect individuals with aid providers, some manage logistics, and others help organizations coordinate aid during emergencies. Together, these systems improve how quickly and effectively aid reaches people in need. However, there are still challenges, especially in places like Gaza, where people suffer from war in 2023. Problems like communication disruptions and weak internet make it difficult for people to register for aid and get help.

Gaza Madad Flow solves this urgent problem. It offers a fast, simple, and reliable way for people in Gaza to register for aid after the war in October 2023. The main goal is to let individuals register once for help and have their data automatically shared with all the important humanitarian platforms and programs. This removes the need to go through many different registration procedures, which are often hard to access.

Gaza Madad Flow responds directly to the serious damage to infrastructure and communication in Gaza [7]. Because telecommunications networks have been destroyed, internet access is often lost, making it hard for people to request aid online [7]. At the same time, about 90% of the population has been displaced and needs urgent humanitarian support [7]. The lack of access to registration has become a critical issue [7].

Gaza Madad Flow uses automation and artificial intelligence to speed up and simplify data processing. AI-powered tools help people submit their information easily. Meanwhile, back-end automation shares this data with multiple aid agency databases at the same time. By reducing manual work and combining registration steps, the system improves access to aid, speeds up service delivery, and makes sure no eligible person misses help. Overall, this approach increases the reach and speed of emergency humanitarian operations.

1.2 Problem Statement

Following the October 2023 war, the primary mechanism for displaced families in Gaza to receive humanitarian aid—digital registration—breaks fundamentally, creating an insurmountable barrier for the most vulnerable.

This core problem comes from a catastrophic convergence of three factors:

- Total infrastructure collapse: A severe lack of electricity and near-total internet failure make it impossible for families to reliably charge a device or access online services.
- Chaotic system design: The aid ecosystem presents a confusing flood of disparate links for separate programs, forcing users to repeatedly enter data in an environment that does not function well.
- Prevalent security threats: The digital space fills with fraudulent, official-looking links designed to scam desperate people, making it very hard to distinguish legitimate aid from malicious traps.

This impossible situation means families, despite their profound need, cannot complete registration and thus miss life-saving assistance.

1.3 Main Objectives

In Gaza Madad Flow, we aim to develop and deploy a web automation model to help people in Gaza Strip, who are affected by the October 2023 war, register for all humanitarian aid through a custom web interface with proper input validation. The model automatically, with minimal user input enrolls their data across multiple aid platforms and forms without the need for repetitive manual registration processes. We aim to reduce user effort, overcome digital blocks (such as lack of internet, electricity, and devices), and ensure access to aid as quickly as possible.

Specific Objectives

1. Research available aid platforms to evaluate their current services and performance in the Gaza Strip, and identify their limitations, problems, and registration mechanisms.

2. Define the project requirements and design specifications for a simple, scalable web automation model with low connectivity.
3. Develop and deploy a web-based automation model with a custom user interface (Laravel-based) that enables one-time validated data entry, eliminating the need for repetitive manual registrations across multiple platforms.
4. Test and validate the automation model using real aid forms, to ensure it is accurate, reliable, and fast.

1.4 Scope and Limitations

1.4.1 Scope

- **Geographical Scope:** Gaza Madad Flow focuses only on residents of the Gaza Strip.
- **Target Group:** Gaza Madad Flow targets people in Gaza who are affected by the war in 2023.
- **Automation Model:** Gaza Madad Flow developed as a Minimum Viable Product (MVP), a functional web-based automation system with a custom Laravel interface and full backend validation. It allows one-time registration, automatically processes and organizes the data, and distributes it to multiple aid platforms using n8n workflows.
- **Application:** Gaza Madad Flow is tested with real and simulated data to verify performance, reliability, and accuracy. It is deployed on Render for cloud-based accessibility and live testing. While it is currently connected to selected platforms for demonstration, it is designed to be scalable and ready for integration with actual humanitarian systems in the future.

1.4.2 Limitations

- **Restricted Number of Aid Platforms for Testing:** The automation model is tested on a small number of aid platforms for demonstration purposes.
- **Resource Constraint:** The project is developed under limited technical resources. Since no dedicated hosting environment is available, the system is deployed using Render's free tier service.

- **Alternative Job Scheduling:** While Render provides a built-in cron job feature, it is only available as a paid service. To address this, the team uses UptimeRobot to periodically trigger a route that executes scheduled background jobs.

- **Limitation in Automated Data Mapping with LLMs:** The project aims to integrate a Large Language Model (LLM) within the n8n workflow to automatically map registration data into formats required by different humanitarian platforms. However, this requires paid API usage and frequent request-limit errors reduce feasibility. As a result, automated mapping with LLMs is not fully implemented, and data mapping is handled manually.

1.5 Importance of the project

Gaza Madad Flow is important because it helps solve big problems people and organizations face during emergencies, especially in places like Gaza that experience conflict and poverty. Here are the main reasons why Gaza Madad Flow matters:

- **Humanitarian Impact:** Gaza Madad Flow makes it easy for people in need to get help from aid groups. It helps vulnerable people in Gaza access basic needs like food, money, and aid more quickly. This improves their lives.

- **Saves Time:** Registering for aid often takes a lot of time and repeating the same information many times. Gaza Madad Flow allows people to register once, and then their information is shared automatically with many aid groups. This helps aid arrive faster in emergencies.

- **Uses Modern Technology:** Gaza Madad Flow uses modern web technologies, like Laravel, PostgreSQL, and APIs. These make the system simple and reliable. Using technology this way shows how innovations can solve humanitarian problems and create new solutions.

- **Scalable and Flexible:** Although Gaza Madad Flow starts in Gaza, it can grow to help other regions with similar issues. Its flexible design allows adding more aid services and customizing it for different areas worldwide.

- **Supports Collaboration:** Gaza Madad Flow provides a foundation for aid organizations to work together better. It reduces duplicate efforts and helps use resources more effectively. This makes giving aid more organized and efficient.

- **Educational and Social Value:** As a graduation project, it shows how students can use what they learn to fix real-world problems. It also adds to the field of technology for social good by offering a practical example.

In short, Gaza Madad Flow uses technology to create a better, faster, and more effective way to share aid during wars. If it succeeds, it can motivate similar efforts in other parts of the world, proving that technology can help solve important social issues.

1.6 Tools and Equipment

1.6.1 Tools

In Gaza Madad Flow , we use several tools to help us work better and faster. These tools support different parts of the project, from automation and coding to communication and task management. Below is a list of the main tools we use and how we use them.

1. Laravel Framework

Laravel is a free and open-source PHP web framework used for building modern web applications [8]. It follows the Model–View–Controller (MVC) architecture and provides features such as routing, authentication, database management, and input validation [8].

In Gaza Madad Flow, Laravel is used to develop the custom user interface and backend of the system [8]. It enables strong input validation, efficient database operations, and integration with services like Google Sheets through APIs [8]. Laravel's structured design ensures the system is maintainable and scalable [8].

2. MySQL Database

MySQL is a free and open-source relational database management system

(RDBMS) based on Structured Query Language (SQL) [9]. It is widely used for storing and managing structured data because of its reliability and scalability [9].

In the first version of Gaza Madad Flow, MySQL served as the main database for storing citizens' registration records, ensuring data integrity and fast retrieval [9].

3. PostgreSQL Database

PostgreSQL is a free and open-source object-relational database management system (ORDBMS) based on Structured Query Language (SQL) [10]. It is widely used for storing and managing structured data due to its reliability, scalability, and strong data integrity features [10].

Gaza Madad Flow was first developed using MySQL for local prototyping and later adopted PostgreSQL as the main production database, providing reliable, consistent, and efficient data storage and retrieval [10].

4. HTML

HTML (HyperText Markup Language) is the standard markup language for creating web pages. It defines the structure of content displayed by web browsers [11].

In Gaza Madad Flow, HTML is used to build the layout of the registration form and other pages, serving as the skeleton of the application [11].

5. CSS

CSS (Cascading Style Sheets) defines the style and formatting of web content, including layout, colors, and fonts [12].

In Gaza Madad Flow, CSS is used to style the homepage, registration form, and confirmation page, ensuring a consistent and user-friendly interface [12].

6. Tailwind CSS

Tailwind CSS is a utility-first CSS framework that simplifies responsive design through prebuilt utility classes [13].

In Gaza Madad Flow, Tailwind CSS is the main styling tool for the registration form, making it modern and responsive while reducing manual CSS work [13].

7. JavaScript

JavaScript is a dynamic programming language that enables interactive features on web pages [14].

In Gaza Madad Flow, JavaScript validates user inputs, shows error messages, manages multi-step form navigation, and dynamically displays form sections, making registration smoother [14].

8. n8n Platform

n8n is an open-source workflow automation tool that combines low-code features with support for complex processes [15].

In Gaza Madad Flow, n8n is used to [15]:

- Send user data automatically to multiple aid platforms.
- Save data into Google Sheets for tracking and backup.
- Filter requests by location or aid type.
- Detect and avoid duplicate submissions.
- Reduce manual work by connecting all systems in one workflow.

Deployment Note:

In Gaza Madad Flow, the team used n8n Cloud instead of a self-hosted installation. This choice ensured higher reliability, faster deployment, and easier maintenance under Gaza's limited connectivity and resource constraints. The cloud version handles workflow execution, authentication, and data transfers securely without requiring local server configuration.

9. Google Sheets

Google Sheets is a cloud-based spreadsheet tool for collaborative data management [16].

In Gaza Madad Flow, it is used for backup, reporting, and easier monitoring by non-technical aid workers [16]. The sheet is updated automatically via n8n whenever new data is submitted [16].

10. Google Forms

Google Forms is an online survey tool integrated with Google Sheets [17].

In Gaza Madad Flow, it is used in the early prototype to collect user input and test the registration process before building the custom interface [17].

11. Render

Render is a cloud hosting platform for web applications, databases, and services [18]. It supports automatic deployments from GitHub, SSL certificates, and monitoring [18].

In Gaza Madad Flow, Render hosts the Laravel application [17]. The free tier is used to make the system accessible online for development and testing [17].

12. UptimeRobot

UptimeRobot is a monitoring service that checks website availability and triggers actions if needed [19].

In Gaza Madad Flow, UptimeRobot is configured to ping a dedicated route every hour to run background jobs (e.g., transferring registration data from PostgreSQL to Google Sheets) [19]. This replaces Render's paid cron service [19].

13. GitHub

GitHub is a version control platform based on Git [20].

In Gaza Madad Flow, GitHub manages code versions, supports collaboration, and connects with Render for automated deployments [20].

14. Visual Studio Code

Visual Studio Code is a lightweight code editor with extensions for multiple programming languages [21].

In Gaza Madad Flow, it is the main environment for developing and debugging the Laravel application [21].

15. Microsoft Word

Microsoft Word is a word processing tool for documentation [22].

In Gaza Madad Flow, it is used to prepare requirements and reports, with options to export into professional formats [22].

16. WhatsApp

WhatsApp is a messaging platform used for fast communication [23].

In Gaza Madad Flow, it facilitates coordination between team members and the supervisor for updates and task management [23].

Why We Use Google Sheets in Gaza Madad Flow ?

Google Sheets serves as a **central bridge** between the PostgreSQL database and the n8n Cloud workflows due to Gaza's unstable connectivity and Render's free-tier limitations.

- **Central Data Backup:** Automatically stores submissions to prevent loss during outages.
- **Accessible Monitoring:** Allows non-technical aid workers to view records easily.

- **Automation Bridge:** Connects PostgreSQL to n8n for automated submissions.
- **Real-Time Updating:** Reflects new data as soon as it synchronizes hourly.
- **Prototype Testing Support:** Used during early prototypes (1 & 2) before full Laravel integration.
- **Low-Resource Friendly:** Works with weak internet and minimal devices.
- **Collaboration:** Enables transparent multi-user access.
- **Integration:** Connects seamlessly through Google APIs.
- **Data Reporting:** Exports structured data for analysis.

1.6.2 Equipment

In Gaza Madad Flow, we use our personal laptops to handle all development, testing, and documentation tasks. These devices ensure smooth workflow.

Device 1: HP Pavilion Gaming Laptop 15-dk0xxx, Windows 10 Pro 64-bit, Intel® Core™ i7-9750H @ 2.60GHz (6 Cores / 12 Threads), 16.0 GB RAM.
Device 2: HP ProBook 450 G2, Windows 10 Pro 64-bit, Intel® Core™ i7-5500U @ 2.40GHz (2 Cores / 4 Threads), 16.0 GB RAM.
Device 3: HP EliteBook 850 G7, Windows 10 Pro 64-bit, Intel® Core™ i7-10510U @ 1.80GHz (4 Cores / 8 Threads), 16.0 GB RAM.

Note: In addition to laptops for development, mobile phones and routers were used for communication and stable internet access during the project.

Chapter 2
Related Works

Chapter 2
Related Works

2.1 List of the Related Works

2.1.1 Almotqadmon

Almotqadmon is a digital platform based in Palestine, primarily focusing on the Gaza Strip [24]. It supports individuals by connecting them with institutions offering humanitarian assistance, job opportunities, scholarships, and training programs—especially targeting youth and recent graduates [24]. The platform does not provide the services directly but facilitates access by publishing verified registration links and announcements from relevant aid providers and institutions [24].

Almotqadmon offers many features, including [24]:

- Connection of users with humanitarian aid providers through automated processes.
- Distribution of humanitarian aid—such as food, financial support, and emergency assistance—via electronic transfers.
- Update of job listings from trusted sources.
- Verification of scholarship opportunities for local and international studies.
- Provision of professional training programs in various fields.
- Implementation of community-based initiatives and events aimed at improving living conditions.
- User registration through forms that collect essential information.
- Content delivery that is fast and organized using automation tools.

Figure 2.1 displays a screenshot from the Almotqadmon user dashboard, illustrating how users access various services such as aid registration, job listings, and training opportunities.

Figure 2.1 : A screenshot of Almotqadmon user dashboard [24].

2.1.2 Reach4Help

Reach4Help is an open-source web platform that connects vulnerable people to local organizations offering basic aid services [25]. Created initially to manage the COVID-19 pandemic, the platform expands its scope to reach conflict zones, disaster-affected areas, and underdeveloped communities worldwide—Palestine, Ukraine, Canada, and India are just a few examples [25]. It focuses on registration and coordination rather than direct aid disbursement or financial transfers [25].

The platform allows for registering needs via online application forms and matching users with available resources such as mental health centers, shelters, food relief, and emergency commodities [25]. It works with a global network of volunteers and integrates with communication platforms such as WhatsApp, Telegram, and Messenger to make it easier to use [25].

Reach4Help offers many features, including [25]:

- Over 10,000 validated providers of assistance across the world are highlighted with an interactive map.
- Users are registered to request services based on their location and needs.

- Seamless integration with messenger platforms (e.g., WhatsApp, Telegram) is achieved.
- A multilingual, accessibility-focused interface is designed.
- A user-first and privacy-first approach that protects user safety and privacy is ensured.
- A community-governance model that promotes openness and responsiveness is implemented.

Figure 2.2 displays a screenshot of the Reach4Help's interactive map, highlighting local aid service providers in the Gaza region and how users can locate nearby assistance.

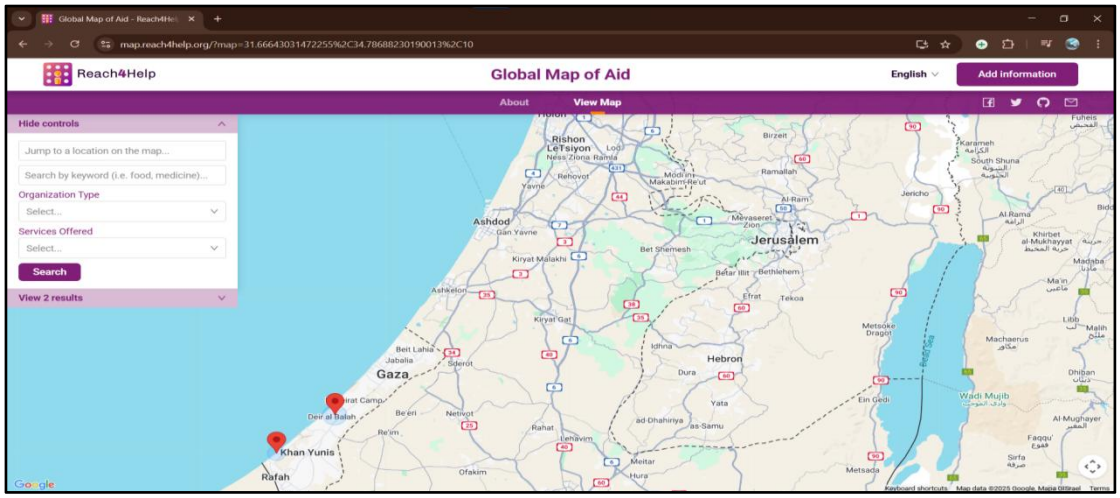


Figure 2.2 :A screenshot of the Reach4Help's interactive map [25].

2.1.3 DAEM

DAEM Platform is a digital aid distribution application developed by the Lebanese government in collaboration with the World Bank [26], [27]. It is a web-based application designed specifically to support vulnerable and low-income Lebanese citizens affected by the country's ongoing economic crisis [26], [27].

The platform primarily operates within Lebanon and functions as a centralized system for registering for and delivering financial assistance [26], [27].

DAEM offers many features, including [26], [27]**:**

- Fully digital registration and application processes are enabled.
- Automated eligibility checks are conducted.
- SMS notifications for application status updates are sent.

- Integration with national databases for information verification is achieved.

Figure 2.3 shows a screenshot of DAEM platform's digital registration form, which allows users to apply for financial assistance through Lebanon's national aid system.

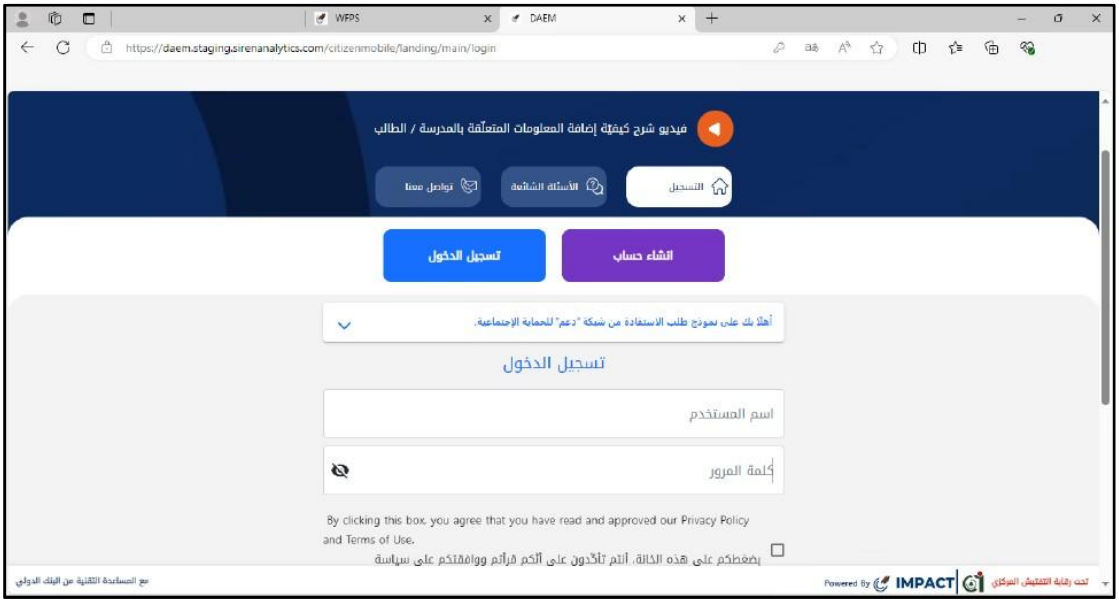


Figure 2.3 : A screenshot of DAEM platform's digital registration form [26], [27].

2.1.4 UiPath

UiPath is a web and desktop enterprise automation platform used globally to support business and humanitarian work [28]. UiPath tools assist local and international NGOs during crises, such as the Ukrainian conflict, by automating refugee registration, intake operations, and logistics workflows [28]. It is not an aid agency but a digital foundation that helps organizations coordinate more effectively by reducing redundant work and producing higher quality data [28].

UiPath offers many features, including [28]:

- Refugee intake and beneficiary registration procedures for NGOs are automated.
- Integration with logistics systems to increase visibility and monitoring of donated items is achieved.
- Centralized data collection and reduction of human error are enabled with RPA tools.
- NGOs are provided with free access during crisis response situations.

- The role of a bridge between humanitarian institutions and affected individuals is fulfilled without direct aid processing.

Figure 2.4 shows a screenshot of UiPath's collaboration with the Romanian Red Cross, illustrating the platform's real-world application in automating humanitarian aid delivery processes.

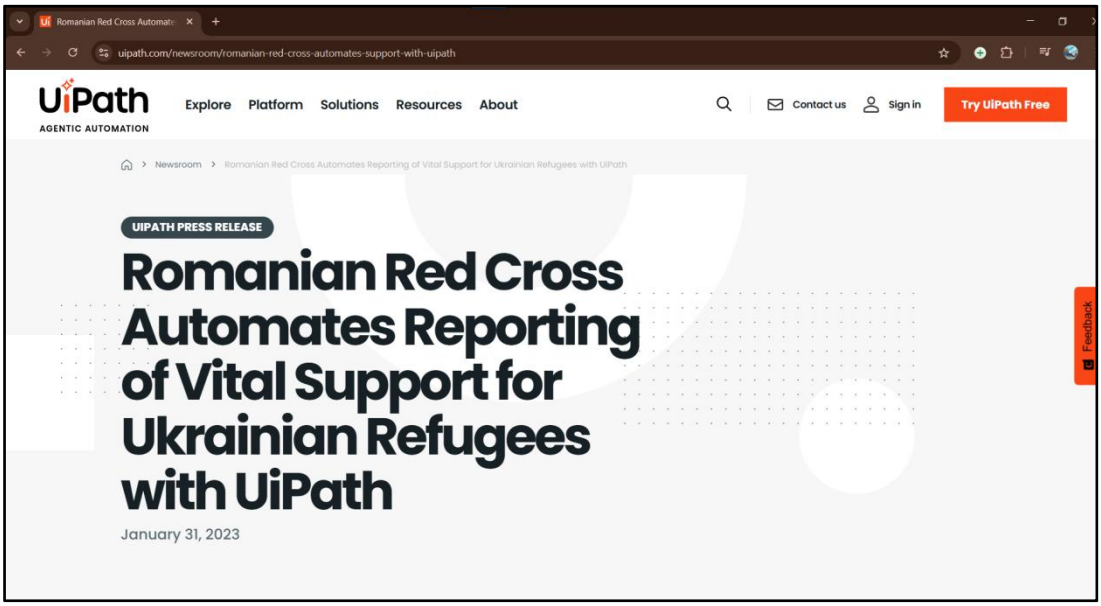


Figure 2.4 : A screenshot from UiPath's collaboration with the Romanian Red Cross [28].

2.1.5 AidKit

AidKit is a digital platform created to help organizations deliver emergency and disaster relief efficiently [29]. Designed with a mobile-first approach, it is also offline-capable, ensuring accessibility in remote or disaster-affected areas [29]. The platform is available on both web and mobile devices [29].

Initially developed during the COVID-19 pandemic to rapidly distribute cash aid, AidKit now supports governments, non-profits, and community organizations worldwide to launch and manage aid programs [29]. It specializes in providing direct cash assistance to individuals and communities affected by disasters and emergencies [29].

AidKit relies on registration and transfers for aid distribution, combining data collection and direct payment processes to ensure effective delivery [29].

AidKit offers many features, including [29]:

- Direct cash assistance is provided.
- Eligibility is verified and fraud is detected.
- Fast data collection and real-time reporting are enabled.
- Payments are automated.
- A geo-targeting system is used.

Figure 2.5 shows a screenshot of AidKit's emergency response interface illustrating user interaction via the demo request form.

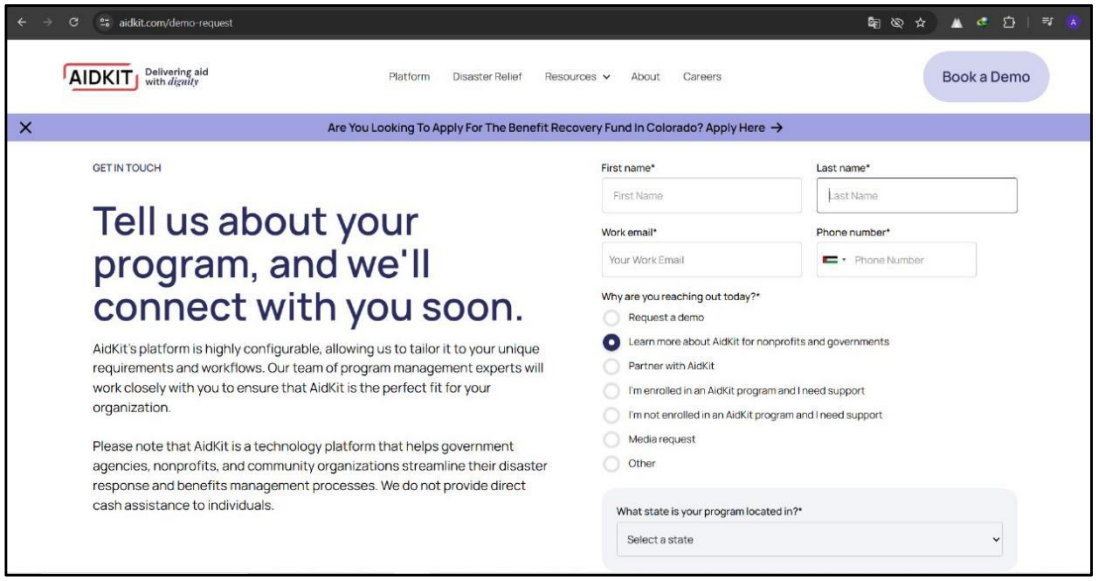


Figure 2.5 : A screenshot of AidKit's emergency response interface [29].

2.2 Review of the Related Works

Table 2.1 provides the main differences between Gaza Madad Flow and the related works in terms of purpose, scope, target users, use of automation, access model, and technology stack.

Table 2.1 : Differences Between Existing Aid Platforms and Gaza Madad Flow

Platform	Almotqadmon	Reach4Help	DAEM	UiPath	AidKit	Gaza Madad Flow
Purpose	Connect users with jobs, scholarships, aid; acts as a mediator	Connect people with local volunteers/orgs for basic services	Distribute financial/food support to Lebanese citizens	Automate digital tasks in business and institutional settings	Help orgs manage & distribute aid efficiently	Register people once, auto-submit to multiple aid platforms
Scope	Palestine (Gaza & West Bank), social & economic development	Global (38+ countries), especially during emergencies	National (Lebanon only)	Global (multiple sectors, e.g. finance, logistics)	Local to regional deployments	Local (focused on Gaza emergency response)
Target Users	Job seekers, students, trainees, aid seekers	People needing food, medicine, or shelter	Low-income Lebanese families	Organizations, NGOs, businesses, tech staff	Governments, NGOs, aid orgs	People affected by war, especially Gaza residents
Use Of Automation	Automated listings, updates, categorized forms	Auto-match users to nearest relevant org based on location	Manual verification & eligibility checking	Backend task automation (data input, extraction, email handling)	Manage applications, verify eligibility, process payments	Auto-submit user data to multiple aid platforms from one form
Access Model	Free to users	Free and open-source	Free for eligible Lebanese citizens	Paid enterprise licenses	Paid subscriptions (org-based)	Free for individual users and aid orgs
Technology Stack	Form-based CMS (exact tech not public)	React.js, Node.js, Firebase, MongoDB (open-source)	Centralized gov portals (IMPACT); tech not publicly detailed	UiPath Studio, Orchestrator, Robots (cloud/desktop)	Web + mobile apps, offline support	Laravel (PHP), PostgreSQL, API integration with Google Forms

Chapter 3
Methodology

Chapter 3
Methodology

3.1 General Overview of The Methodology

3.1.1 Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) are help Software projects to ensure quality and manage complexity [30]. SDLC outlines a sequence of key phases commonly planning, analysis, design, development, testing, implementation, and maintenance [30].

By progressing through these stages methodically, teams can translate initial ideas into a stable final product [31]. This life-cycle is important in software engineering because it provides a disciplined framework for organizing work and controlling the development process [31]. A well-defined SDLC helps in planning and design, testing and iterative improvement, which in turn reduces project risks and ensures the software meets user requirements [31].

In practice, using SDLC principles means problems are caught early and the end result is a high-quality, maintainable system that aligns with stakeholders' needs. Given these benefits, our team grounded its approach in SDLC concepts, selecting a specific development methodology suited to our project's context. Figure 3.1 presents the Software Development Life Cycle (SDLC).

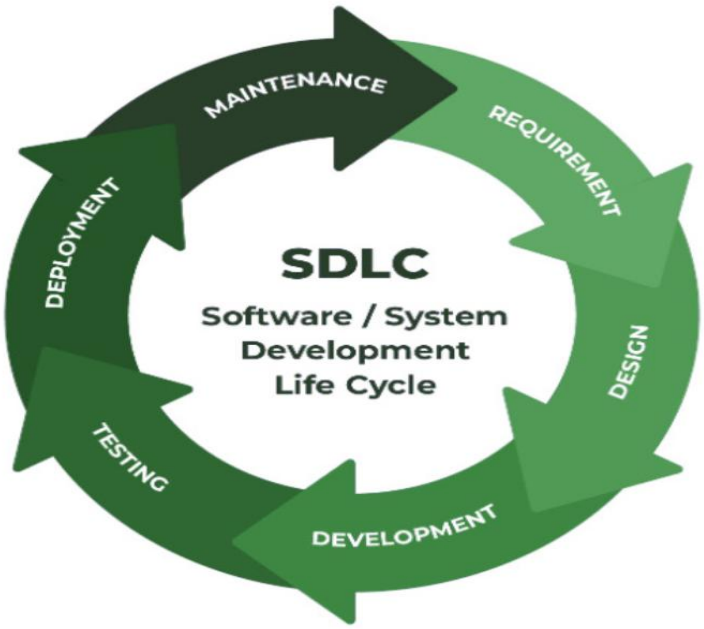


Figure 3.1 : Software Development Life Cycle [32].

3.1.2 Prototyping Development Methodology

After evaluating various SDLC models, we choose the Prototyping development methodology for this project [33]. The Prototyping Model is one of the most commonly used Software Development Life Cycle (SDLC) models, especially when customers do not know the exact requirements in advance [33]. In this model, a prototype of the final product is developed, tested, and refined repeatedly based on customer feedback until an acceptable version is achieved [33]. This prototype then serves as the foundation for developing the final product [33].

The process typically starts by interviewing customers and creating a high-level model [33]. This document guides the development of the initial prototype, which includes only the basic functionality required by the customer [33]. Once the customer identifies problems or areas for improvement, the prototype is refined accordingly [33]. This cycle of feedback and improvement continues until the customer approves the prototype and considers it satisfactory [33].

3.1.3 Types of Prototyping

1. **Rapid throwaway prototyping:** Involves quickly creating a basic model to capture user requirements and gather early feedback [33]. This prototype is developed only to explore ideas and clarify functionality and is then discarded [33]. It is not included in the final product [33]. This approach is useful when initial requirements are unclear and need to be shaped through quick user interaction [33].

2. **Evolutionary prototyping:** Takes a different approach by developing an initial prototype that is continuously refined and expanded based on ongoing user feedback [33]. The prototype evolves gradually into the final system [33]. This method saves time and effort compared to rebuilding the system from scratch for every iteration [33].

3. **Extreme Prototyping:** Mainly used in web development, extreme prototyping breaks the process into three phases: first building basic static pages, then simulating the business logic through a services layer, and finally implementing the full back-end services [33]. It aims to quickly produce a working web-based system [33].
4. **Incremental Prototyping:** Is a methodology where the final system is divided into multiple small, manageable components or modules [33]. Each component is developed as an independent prototype, tested, and refined before being integrated into the full system [33]. This method reduces complexity by focusing on specific development tasks and allows parallel work across different modules [33]. It also facilitates early testing and quick identification of issues [33]. Additionally, it provides flexibility, as individual parts can be modified without impacting the entire system [33].

After evaluating the different types of prototyping models, our team decided to adopt Incremental Prototyping as the development methodology for this project.

3.1.4 Phases of Prototyping Model

Prototyping Model has six phases

1. Requirements Gathering: In the first phase, we gather and analyze the system requirements [33]. We discuss the goals and expectations of the project with our supervisor to clearly define what the system should achieve [33]. These discussions help us identify the main features and constraints of the system [33].

2. Quick design: In this phase, we create a quick and basic design of the system [33]. This design provides an initial layout that shows the system's general structure and interface [33]. It is not detailed, but it gives our supervisor a clear idea of how the system will function in its early form [33].

3. Build a Prototype: After completing the quick design, we develop a basic working prototype that demonstrates the essential functions of the system [33]. This version allows our supervisor to interact with the system and understand how the final product may work [33].

4. Initial user evaluation: We then present the prototype to our supervisor for evaluation [33]. During this phase, the supervisor provides feedback about the system's strengths, weaknesses, and possible improvements [33]. This feedback guides the next refinement phase [33].

5. Refining prototype: Based on the supervisor's feedback, we modify and improve the prototype [33]. We repeat this cycle of review and revision until the supervisor approves the model and confirms that it meets the project objectives [33].

6. Implement Product and Maintain: Once the prototype is finalized, we implement the complete system based on the approved design [33]. After implementation, we perform final testing and prepare the system for demonstration [33]. We also plan for basic maintenance to ensure continued reliability.

Figure 3.2 presents the phases of prototyping model [33].

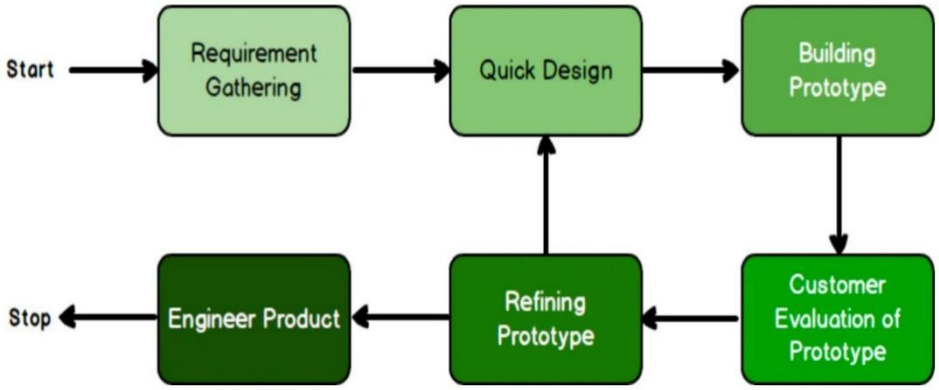


Figure 3.2 : Phases of Prototyping Model [34].

3.1.5 Advantages of Prototyping Model

Prototyping provides several advantages that make it ideal for early-stage software development and refinement [33]:

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Early feedback from customers and stakeholders can help guide the development process and ensure that the final product meets their needs and expectations.
- Prototyping can be used to test and validate design decisions, allowing for adjustments to be made before significant resources are invested in development.
- Prototyping can help reduce the risk of project failure by identifying potential issues and addressing them early in the process.

- Prototyping can facilitate communication and collaboration among team members and stakeholders, improving overall project efficiency and effectiveness.

3.2 Reason for Choosing Prototype

We choose the prototyping model for several important reasons that fit our project:

1. The main objective of using prototyping is to explore and validate the feasibility of automating humanitarian aid registration in a scalable system, allowing iterative refinement of features and performance improvements.
2. Many technical and functional requirements are unclear at the start. Prototyping enables step-by-step development and improvement through continuous feedback.
3. Prototyping provides flexibility to experiment with different technical approaches, helping identify performance improvements and maintain scalability for future enhancements.
4. Incremental testing of each prototype allows early detection of issues, such as duplicate records, API limitations, and data validation errors, reducing risks and increasing confidence in the final system.
5. The approach facilitates frequent demonstrations to the supervisor, ensuring continuous alignment with project goals and refining the design through constructive feedback.
6. Modular prototyping makes the system easier to extend, allowing integration of new aid platforms or AI-based validation in future iterations without redesigning the solution.

3.3 Task Management

3.3.1 Requirements Analysis

The development starts with understanding the project needs. The team meets with the supervisor to clarify the idea and identify the basic features for the prototype. The main goal is to create a simple web tool that allows users to register once and automatically send their information to different aid platforms, which is crucial in the Gaza Strip for rapid aid delivery during emergencies.

In the initial phase:

- Citizens submit their data through a Google Form and receive a confirmation message.
- The form collects essential fields: name, ID, contact information, and type of aid.
- Submitted data is automatically saved in Google Sheets.
- The prototype does not include a full web interface, focusing instead on validating core automation workflows.

This early approach allows the team to test automation workflows and verify the system's feasibility without building a full web application.

As the project evolves:

- The Google Form is replaced with a custom Laravel Blade web interface to improve accessibility, usability, and reliability.
- All submissions are stored in a PostgreSQL database, centralizing citizen records and ensuring data integrity.
- A scheduled hourly batch job synchronizes data from PostgreSQL to Google Sheets, reducing API calls and improving performance.
- n8n workflows handle automated validation, formatting, and submission to multiple aid platforms.

This evolutionary approach allows the team to start with a simple, testable prototype and gradually develop a robust, scalable system while continuously refining requirements based on feedback and testing.

3.4 Prototyping Phases

Based on the identified requirements, we create an initial design to visualize the system's workflow and functionalities. During the early phase:

- We draw a Google Form to illustrate how citizen data will be collected.
- We plan a Google Sheet to automatically store submissions for centralized access and processing.
- We sketch a prototype workflow to demonstrate automation using n8n:

- Detect new entries in the Google Sheet.
- Clean, organize, and map collected data.
- Submit processed data to humanitarian aid platforms using POST requests, ensuring automated, reliable, and efficient data transmission.

As the project evolves, we enhance the design to improve usability, performance, and scalability:

- We replace the Google Form with a custom Laravel Blade interface, providing a more reliable and user-friendly experience.
- We store submitted data in a PostgreSQL database, centralizing citizen records and ensuring data integrity.
- We introduce a scheduled hourly batch job to sync data from PostgreSQL to Google Sheets in bulk, reducing API calls and improving performance.
- We maintain n8n workflows for cleaning, mapping, and submission to multiple aid platforms.

This quick design plan enables us to understand the main system functions early and provides a clear roadmap for review with our supervisor before developing the first working prototype.

3.4.1 Prototype 1: Data Collection from Google Forms

In the first prototype, we focus on collecting and storing user data from a Google Form. The form has basic fields like name, ID, contact information, and aid type. When users submit the form, the data automatically saves to a connected Google Sheet.

This phase ensures that data collection and organization work correctly before developing subsequent prototypes.

3.4.2 Prototype 2: Automated Submission to a Single Aid Platform

The second prototype expands the system's functions. Here, n8n reads data from the Google Sheet and automatically fills out a sample aid registration form on an external website (aid platform).

The process involves:

- Accessing the registration form
- Mapping and entering user data into the right fields
- Submitting the form automatically

This prototype shows how the system simulates human interaction with aid websites and acts as an automation layer in the background. We test it on one aid platform for controlled testing.

3.4.3 Prototype 3: Multi-Platform Integration and Real-Case Testing

In the third prototype, we test the system with real citizen data on multiple aid platforms. Although the initial focus is on a single registration page, the design supports many platforms in the future. Goals include:

- Verifying that the system functions correctly across different aid platforms.
- Ensuring n8n handles various form structures dynamically.
- Testing accuracy and reliability with real user data.

3.4.4 Prototype 4: Laravel + PostgreSQL + Hourly Batch Workflow

The fourth prototype focuses on user-friendliness, reliability, and efficiency:

- **Custom Laravel Blade Interface:** Citizens submit data through a web interface optimized for accessibility and reduced errors.
- **Centralized PostgreSQL Database:** All submissions are stored securely, ensuring integrity and efficient retrieval.

  The system was initially developed locally using MySQL for prototyping and later migrated to PostgreSQL for cloud deployment on Render.
- **Scheduled Hourly Batch Synchronization:** Data transfers in bulk from PostgreSQL to Google Sheets, minimizing API calls and improving performance.
- **n8n Automation Workflows:** Handle data cleaning, mapping, and automated submission via POST requests to multiple platforms.

This prototype demonstrates a robust and scalable system capable of handling real-world citizen data efficiently while maintaining high reliability and performance.

3.4.5 Prototype 5: Hosting and Deployment on Render

To ensure continuous operation, we deploy the system on Render, a cloud-based hosting platform. Benefits include:

- **24/7 Cloud Operation:** Citizens can access the system at any time.
- **Reliable Batch Execution:** Hourly synchronization jobs and n8n workflows run consistently.
- **Scalability and Performance:** The system handles increasing volumes of data without performance degradation.
- **Monitoring & Maintenance:** Integrated logging and monitoring tools support proactive maintenance and issue resolution.

Deploying on Render confirms that the Laravel + PostgreSQL backend and automated n8n workflows operate reliably outside the development environment.

3.5 Prototyping Testing

Since the project evolves into a Minimum Viable Product (MVP), we focus testing on verifying that the system processes real citizen data accurately and that automated workflows operate reliably. This ensures that all components—the Laravel Blade interface, PostgreSQL database, hourly batch synchronization, n8n workflows, and cloud deployment on Render—work together seamlessly under realistic conditions.

We perform testing by submitting real or simulated citizen data through the Laravel Blade interface and observing how the system processes it through automated workflows. The main objectives of testing include:

- Ensuring all fields in the custom user interface are captured correctly and completely.
- Confirming that new entries in the PostgreSQL database are transferred to Google Sheets accurately during the hourly batch synchronization.
- Validating that n8n workflows successfully submit data to multiple aid platforms via automated POST requests.
- Handling errors gracefully, such as failed submissions, duplicate entries, or invalid data, without disrupting the system.

- Monitoring workflow performance, response times, and reliability under real-case conditions.
- Verifying deployment functionality to ensure the system works correctly in a live cloud environment, including real-time data processing, workflow execution, and user accessibility.

This approach ensures that the automated aid registration model functions as expected, providing accurate, reliable, and efficient service for humanitarian scenarios.

Prototype Task Management

We manage each test case as a separate task on our team's Trello board. For each task, we include:

- The input scenario
- Expected results vs. what actually happens
- Test status (Pass or Fail)
- Notes to improve the system
- Assigned team member for this task

When a test fails, we analyze the problem and make necessary adjustments. This repeated testing helps us ensure that the automation stays stable, flexible, and ready for future use.

3.6 Timeline

3.6.1 From 1 July to 15 July 2025 – Requirements Gathering

- Start the development by understanding what the system needs to do.
- Meet with the supervisor to clarify the project idea and align expectations.
- Clarify the overall and technical objectives of the project.
- Analyze the basic needs of users in Gaza.
- Set the main goal: create a simple web tool that allows users to register once and automatically send their data to multiple aid platforms.
- Focus on solving a real need in the Gaza Strip, where aid must be delivered quickly during emergencies.

- Decide that the user fills out a form and receives a confirmation message.

3.6.2 From 16 July to 31 July 2025 – Quick Design

- Review all gathered requirements to identify core features and prioritize tasks.
- Create a simple design to visualize how the system works.
- Draw a mockup of the Google Form that collects user data.
- connect Google Sheet to google form that automatically saves user data.
- Use the n8n automation tool to Detect new data entries in the Google Sheet then Format and prepare the data after that Send the data to aid platforms via forms or APIs
- Share the design with the supervisor to confirm the approach before building the first version.

3.6.3 From 1 August to 31 August 2025 – Prototype 1 Development (Data Collection Phase)

- Focus on building the first prototype of the system.
- Create a Google Form with essential fields (e.g., name, ID, contact info, aid type).
- Link the Google Form to a Google Sheet to store responses automatically.
- Use the n8n platform to detect new submissions in the Google Sheet.
- Set up automation to monitor and extract form data as it arrives.
- Ensure the collected data is clean, organized, and ready for the next automation steps.
- Test the full data collection process with sample entries.
- Verify that all submissions are stored correctly and without duplication.

3.6.4 From 1 September to 30 September 2025 – Prototype 2 Development (Single Platform Automation)

- Expand the system to simulate automated data entry into a single humanitarian aid platform.
- Use n8n to read user data from the connected Google Sheet.
- Access a sample registration form on an external aid platform.
- Map user data to the corresponding fields in the registration form.

- Automatically fill in the form using the stored data.
- Submit the form programmatically without human interaction.
- Simulate how the system behaves like a real user submitting aid requests.
- Test the full automation flow on one selected aid platform.
- Validate that the data is submitted accurately and the process runs smoothly.
- Use this prototype to verify the effectiveness of automation as a background service.
- Present the working prototype to the supervisor for feedback and approval.

3.6.5 From 1 October to 31 October 2025 – Prototype 3 Development (Multi-Platform Integration)

- Test the system using real user data across multiple aid platforms.
- Begin with one registration page as a base for scaling to other platforms.
- Ensure the system architecture supports adding more platforms in the future.
- Check if the system performs correctly on different aid websites.
- Verify that **n8n** can dynamically handle various form structures.
- Test the accuracy and reliability of automated submissions using real input.
- Confirm the system's ability to adapt and grow while maintaining consistent logic.
- hare the prototype results with the supervisor for evaluation and suggestions.

3.6.6 From 1 November to 30 November 2025 - Prototype 4 Development (Laravel + PostgreSQL + Hourly Batch Workflow)

- Develop a custom Laravel Blade interface for citizens to submit their data. The interface includes client-side validation (HTML5 and JavaScript) to reduce user errors, ensure required fields are filled, and provide instant feedback for incorrect input.
- Implement a centralized PostgreSQL database to securely store submissions. Laravel's Eloquent ORM enforces schema rules, while server-side validation ensures data integrity before insertion. Sensitive data is stored with appropriate encryption and indexing for performance.
- Set up scheduled hourly batch synchronization using Laravel's Scheduler and Queue system to transfer data from PostgreSQL to Google Sheets in bulk. This

reduces the number of API calls, improves efficiency, and guarantees synchronization even under unstable network conditions.

- Configure n8n automation workflows to handle data cleaning, mapping, and automated submission to multiple aid platforms.
- Test the end-to-end workflow to ensure reliability, performance, and accuracy.

3.6.7 From 1 December 2025 to 30 December 2025 – Deployment, Cloud Testing & Maintenance

Deployment and Cloud Setup

- Prepare the deployment plan and assign responsibilities.
- Set up the target environment for deployment using **Render cloud platform** to ensure **continuous 24/7 availability**.
- Export and package the automation workflows from n8n for deployment.
- Configure **Google Forms and Google Sheets** (or custom Laravel interface) to work with the deployed version.
- Connect automation tools to real or demo aid platforms for live testing.
- Run **system validation** to ensure data flows correctly end-to-end.
- Confirm that the **full system** (Laravel interface, PostgreSQL backend, hourly batch synchronization, and n8n automation workflows) operates reliably in a **live cloud environment**.

Operational Testing & Monitoring

- Validate that **hourly batch syncs** and **n8n workflows** execute consistently without manual intervention.
- Test system **scalability and performance** under increasing volumes of user data.
- Monitor logs, system metrics, and performance dashboards using Render's integrated tools to ensure **stability** and detect issues early.
- Detect and correct errors or failed automation tasks.

Maintenance & Continuous Improvement

- Refine and optimize workflows for improved speed, reliability, and resource efficiency.

- Address any issues raised by users promptly to ensure smooth operation.
- Collect supervisor feedback and implement improvements.
- Keep the system ready for **future updates and integration with additional aid platforms**.
- Present the fully deployed system to the supervisor for **final review** and make last adjustments based on feedback.

Table 3.1 presents the project timeline from 1 July to 31 December 2025. It shows the phases of the project along with their corresponding time frames.

Table 3.1 : Project Timeline from July to December 2025

Requirements Gathering	1 July - 15 July 2025
Quick Design	16 July - 31 July 2025
Data Collection Phase	1 August - 31 August 2025
Single Platform Automation	1 Septemper - 30 Septemper 2025
Multi-Platform Integration	1 October - 31 October 2025
Laravel + PostgreSQL + Hourly Batch Workflow	1 November - 30 November 2025
Deployment, Cloud Testing & Maintenance	1 December - 30 December 2025

Chapter 4

Specification

Chapter 4

Specification

4.1 Background

Right now, the Gaza Strip faces a difficult situation. After the war in October 2023, many people lose their homes, and buildings are destroyed. Today, thousands of families are moved away from their homes, and hunger puts lives at risk. Because of this, the need for quick, fair, and reliable help is very important.

At present, each organization uses its own registration form and method. People have to fill out their information many times for different groups. These forms are different in language and what they ask for. There is no shared system or main database between aid groups.

This process is very hard, especially for families without good internet or electricity. Many people do not register in time or miss chances to get help. Also, when organizations work separately, they might do the same work twice or miss people who need help.

Our project, Gaza Madad Flow, tries to fix this problem. It gives a custom user interface (UI), where people can register only once. After that, their information automatically goes to different aid groups. This saves time, prevents mistakes, and helps more people get help. During war, hunger, and displacement, this system makes aid faster, fairer, and easier to deliver.

4.2 Project Description

This project shows an AI-powered, no-code automation model built with the n8n platform and a Laravel backend. It helps improve the process of registering for humanitarian aid in Gaza Strip by making it simpler and automating how people send their data to aid organizations.

Gaza Madad Flow collects information from applicants through a custom Laravel Blade web form. The system checks the submitted data for completeness, formatting, and duplicates before saving it securely in a PostgreSQL database. The validated data

is then synchronized to Google Sheets, which acts as the source for the automation workflow.

Using n8n, Gaza Madad Flow automatically fetches the records, cleans and arranges the information, and prepares it to match each aid platform's requirements. Instead of filling forms through a browser, the system now submits the data directly to aid platforms via POST requests. This reduces manual work, speeds up registration, and makes access to aid faster, more reliable, and fairer, especially during emergencies when time and resources are limited.

The collected information includes personal details, family size, displacement information, health status, and types of damage. Gaza Madad Flow is now a Minimum Viable Product (MVP), meaning it is fully functional, user-facing, scalable, and capable of supporting real humanitarian aid registration.

What the Automation Model Does:

- Uses an AI-powered, no-code tool (n8n) to automate workflow execution.
- Fetches new citizen records from Google Sheets, checking that only non-registered applicants are processed.
- Cleans and organizes the data to fit the formats required by aid platforms.
- Extracts each platform's structure through GET requests.
- Maps the collected citizen data to the correct fields of each platform.
- Submits the data automatically via POST requests.
- Marks successful submissions in Google Sheets to prevent duplicate registrations.

4.2.1 Actors in the System

Citizen(User)

The Citizen is the primary actor in the Gaza Madad Flow system. This actor represents individuals in Gaza who register for humanitarian assistance.

The citizen interacts with the system through a Laravel-based web interface, filling in required details such as personal, family, and housing information.

After completing the form, the citizen submits it, which triggers validation, data

storage, and synchronization with Google Sheets for automated processing by the AI Agent.

Main Goal:

Register once and have the data automatically shared across multiple aid platforms without redundancy or repeated effort.

Related Use Cases:

- Fill Registration Form
- Submit Application

AI Agent

The AI Agent is an automated workflow engine powered by n8n, responsible for all backend operations.

It retrieves new citizen data from Google Sheets, validates and maps the information, and submits it to humanitarian platforms using HTTP POST requests.

Finally, it updates the sheet with submission results and timestamps.

Main Goal:

Process citizen records automatically—without human intervention—to ensure efficient data distribution across multiple aid platforms.

Related Use Cases:

- Fetch Citizens Data
- Retrieve Aid Links
- Extract Form Structure
- Clean, Prepare, and Map Citizen’s Data
- Submit Aid Application
- Update Submission Status

Use Case Descriptions

UC-1: Fill Registration Form

Actors	Citizen
Description	Citizen fills in all required information (personal, family, housing) using the Gaza Madad Flow registration form.
Preconditions	The citizen has access to the Gaza Madad Flow web interface.
Postconditions	The form is completed and ready for submission.
Trigger	The citizen opens the registration form to start a new application.
Main Flow	1. Citizen opens the Gaza Madad Flow registration page. 2. Citizen fills in all required fields. 3. The system checks for missing or invalid data. 4. Citizen reviews and confirms entered information.
Alternative Flow	3a. If validation fails (e.g., empty required fields, invalid ID, or incorrect phone format) → system displays an error message requesting correction.

UC-2: Submit Application

Actors	Citizen
Description	Citizen submits the completed form, triggering validation and data storage in MySQL.

Preconditions	The citizen has completed all mandatory fields correctly.
Postconditions	The data is validated, stored in MySQL, and queued for the AI Agent to process.
Trigger	Citizen clicks the “Submit” button on the form.
Main Flow	1. Citizen submits the completed application. 2. The system validates form data. 3. Valid data is stored in MySQL and synced to Google Sheets. 4. A success message appears.
Alternative Flow	2a. If duplicate ID or spouse ID exists → system displays an error message and rejects submission.

UC-3: Fetch Citizens Data

Actors	AI Agent
Description	The AI Agent retrieves new citizen entries from Google Sheets to begin automated processing.
Preconditions	Citizen data exists in Google Sheets and is marked as unsent.
Postconditions	New records are loaded into the automation workflow.
Trigger	The AI Agent runs automatically every hour using an **“**On Add or Update” trigger.
Main Flow	1. AI Agent connects to Google Sheets. 2. Reads unsent citizen rows. 3. Stores them in workflow memory for processing.

UC-4: Retrieve Aid Links

Actors	AI Agent
Description	The AI Agent retrieves all available humanitarian aid registration links from the connected Google Sheet for data submission.
Preconditions	Aid platform links exist in the Google Sheet.
Postconditions	All platform links are available for form extraction.
Trigger	The AI Agent starts its hourly workflow after fetching citizen data.
Main Flow	1. AI Agent reads the “Aid Links” sheet. 2. Retrieves all URLs of active aid registration forms. 3. Stores the links for the next phase (form extraction).

UC-5: Extract Form Structure

Actors	AI Agent
Description	The AI Agent extracts the HTML structure and field metadata from each humanitarian aid platform’s form.
Preconditions	Aid platform links are retrieved successfully.
Postconditions	Form field structures are extracted and ready for data mapping.
Trigger	The AI Agent begins the extraction process for each aid link.
Main Flow	1. The AI Agent opens each link. 2. Extracts form fields, labels, and CSRF tokens. 3. Saves the structure for mapping.

UC-6: Clean, Prepare, and Map Citizen's Data

Actors	AI Agent
Description	The AI Agent cleans, standardizes, and maps each citizen's information to the corresponding fields of the aid platform forms.
Preconditions	Citizen data and form structures have been successfully retrieved.
Postconditions	Mapped datasets are ready for submission.
Trigger	The AI Agent starts the mapping phase after extracting form structures.
Main Flow	1. AI Agent removes empty or invalid fields. 2. Standardizes text, numbers, and dates. 3. Matches each field in the citizen record to its equivalent form input on the target platform. 4. Stores the mapped results for submission.

UC-7: Submit Aid Application

Actors	AI Agent
Description	The AI Agent automatically submits the mapped citizen data to multiple humanitarian aid platforms using HTTP POST requests.
Preconditions	Mapped data and target form fields are ready.
Postconditions	Citizen data is sent to all targeted aid platforms.

Trigger	The AI Agent executes the submission workflow.
Main Flow	1. The AI Agent sends POST requests with citizen data to each platform. 2. Waits for success or error responses. 3. Logs the submission results.

UC-8: Update Submission Status

Actors	AI Agent
Description	The AI Agent updates the Google Sheet with the submission status and timestamp for each processed citizen record.
Preconditions	Submission responses are available for all attempted platforms.
Postconditions	Sheet is updated; failed cases are flagged for retry.
Trigger	Completion of the submission process.
Main Flow	1. The AI Agent updates each row in Google Sheets with the latest status. 2. Marks successful submissions as completed. 3. Flag failed submissions for retry.

Figure 4.1 shows Gaza Madad Flow Use Case Diagram

The diagram below illustrates the interaction between the two main actors Citizen and AI Agent and the major system use cases.

The citizen performs data entry and submission through the web interface, while the

AI Agent handles all backend automation, including retrieving, cleaning, mapping, and submitting data to humanitarian aid platforms, and updating submission statuses.

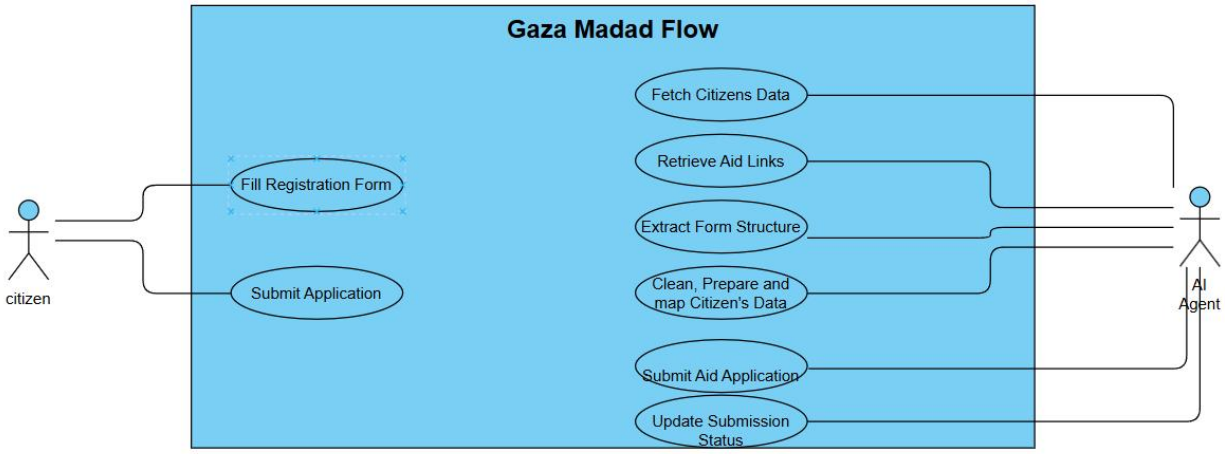


Figure 4.1: Gaza Madad Flow Use Case Diagram

4.3 Main Functional Requirements

The Gaza Madad Flow system provides essential functionality to users who seek humanitarian aid. These features are designed to ensure ease of use, minimize effort, and increase access to multiple aid platforms through a unified and automated registration experience. The following are the core functional requirements.

1. Unified Data Entry (Single Form Submission) : Citizens enter their information through a Laravel Blade web interface, which collects all required details including personal information, family size, displacement status, health status, and damages.

2. Automatic Data Synchronization: The system synchronizes validated data from PostgreSQL to Google Sheets, which serves as the main source for the automation workflows.

3. Automatic Multi-Platform Registration: The n8n automation model fetches only non-registered records, cleans and maps them, then submits the data to multiple aid platforms via POST requests. Successful submissions update the ScrappedAt column to prevent duplicate registrations.

4. Form Validation and Error Feedback :The system checks for missing or invalid data (e.g., empty fields, wrong ID format) before submission and provides clear error messages to help users correct them.

4.4 Main Non-Functional Requirements

1. Support for Low Connectivity: The form is designed to work in simple, lightweight formats (e.g., Google Forms) to be accessible even in areas with limited internet, power, or device capability. This ensures access for displaced or vulnerable individuals with limited resources.

2. Data Privacy and Safety: Users' data is not shared publicly. The system ensures that only authorized aid platforms receive the submitted information. This respects the privacy of vulnerable users.

3. Secure Data Backup and Storage: All submissions are securely stored in a structured SQL database to ensure data integrity, reliability, and ease of access. The database provides a consistent backup mechanism, allowing submitted data to be safely retained and recovered when needed for future verification or auditing.

4. Deployment and Accessibility: The system is deployed on the Render cloud platform to provide scalability, availability, and real-time accessibility for users under real-world conditions.

4.5 The Workflow

The Gaza Madad Flow automation model follows a clear process to register people for humanitarian aid. The steps below show the sequence from receiving data to sending it for aid registration:

1. Creating the Registration Form: The first step designs the registration form. The form collects important data such as personal details, family size, displacement status, and types of damage. The form is made based on the needs of different aid platforms, with fields chosen for each platform’s data needs.

2. Storing Data in Google Sheets: After the form is submitted, the data is automatically stored in a Google Sheet. This central sheet acts as the main database, keeping all applicant information and serving as the core part of the automation. The Google Sheet connects with the n8n platform, which starts the next steps in the workflow.

3. Triggering the Model: The workflow starts by the “on Add” event in the citizens’ sheet. When new data is added to the sheet, the automation model processes the data every hour to keep submissions up to date. This step makes sure the system stays in sync with new applicants.

4. Retrieving Aid Registration Links: After the trigger, the system gets the “Aid Links” from their respective sheet. These links point to the right aid registration platforms. The system stores these links to ensure the correct platform is used for each applicant.

5. Looping Through Aid Links: The system goes through the aid links, handling each one according to its specific requirements. For each aid platform:

    - Scraping the Aid Page: The system fetches the aid platform’s page and gets the HTML content to learn the form structure and needed fields.

    - Mapping Fields: The system matches the fields of the aid registration form to the data fields from the citizen’s record in the Google Sheet.

6. Processing Citizen Data: For each citizen in the Google Sheet, the system goes through their data and processes it according to the form requirements of the aid platform:

    - Data Handling: The system changes and formats data so the values match what the aid form expects.

    - Form Submission: The system automatically submits the completed form. The submission uses the right method—an HTTP request, a background request, or the Browserless service for browser-based submissions.

7. Handling the Submission Result: After the form is submitted, the system gets the result and checks that the submission is successful. If needed, the system takes corrective actions based on the outcome.

This workflow fully automates the process of registering for humanitarian aid, from data entry to submission and result checking, improving both efficiency and accuracy.

4.6 Sheet Details and Data

We analyze several aid registration links to see how often certain fields are required. To do this, we choose to keep only the most needed and common fields. This makes the form standard so it fits most aid registration links and helps reduce the time users spend filling in the same data on different sites. Our data collection covers different types of help, such as medical support for patients, emergency shelter like tents and coverings for people displaced by losing their homes, and financial aid for families with or without a monthly income.

The sheet also includes the following data fields:

1. Head of Household Information: This includes the full name and ID number. This information helps verify identity and makes sure each person is only recorded once. The ID number is important because it is a unique ID and helps prevent double registration across different platforms.

2. Contact Information: This includes a phone number so we can communicate directly with the household if needed. This is important for follow-up or urgent messages about the assistance.

3. Income Source: This helps us understand the family's financial situation and decide if they need help. Knowing if the family has a regular income helps us measure their eligibility for financial aid or other support.

4. Family Composition: This collects the number of family members and their ages. Knowing the size of the family and the ages of the members helps us decide what kind of aid they need, such as food, shelter, or medical help for children and the elderly.

5. Housing Information: This gives an overview of the family's housing conditions. It helps us see if the family loses their home or needs emergency shelter, and what kind of support they require.

6. Family Needs: This section helps identify the family's urgent needs, like food, medicine, or financial help. It helps us prioritize aid distribution.

7. Family Losses During the War: This includes losses of food, medical support, financial aid, and other support. Knowing how much the family loses during the conflict helps us customize the aid to fit their specific situation.

8. Bank Account Information (if available): This allows the aid organization to transfer funds securely and efficiently. It helps send cash directly to the family and ensures they receive help in the easiest way, especially when in-kind aid is not possible.

The sheet is easy to read and export, and it can connect with systems that send data to external registration links. Based on the selected data fields, we develop a single form for users to fill out once. The collected data is stored in the sheet and can automatically help apply to different aid platforms.

Source of Fields

We collected and designed the fields for the Google Sheet by studying and analyzing different humanitarian aid registration platforms and their forms. We identified the most common and important fields used by these systems to create one unified structure that fits all aid organizations.

After analyzing and comparing them, we organized the data into six main sections for clarity and completeness:

1. Personal Information
2. Contact Information
3. Family Formation
4. Housing and Displacement
5. Family Needs & Income

6. Bank Details

Each field was selected carefully to match real beneficiary data and stay compatible with most aid systems. These fields were then used in the custom Laravel UI form, allowing citizens to enter their data once, which is saved in the database and synchronized with Google Sheets for automation and sharing with multiple platforms.

This made the Gaza Madad Flow form simple, complete, and ready for integration, reducing repeated work and making data entry easier.

Chapter 5
Design and Implementation

Chapter 5

Design and Implementation

5.1 Database Table Structure (Citizens Table)

The Gaza Madad Flow system stores all citizen registration data in a single table named citizens.

This table is the central component of the database and contains all personal, family, housing, health, and financial details needed for humanitarian aid registration.

It is also directly linked to the automation workflows that export data to Google Sheets and connected aid platforms.

Table 5.1 shows the detailed structure of the citizens table used in Gaza Madad Flow. It lists all fields stored in the database along with their meanings in both English and Arabic.

Table 5.1: Structure and Description of the Citizens Table

Field Name	English Description	الوصف بالعربية
id	Unique identifier for each citizen record.	المعرّف الفريد لكل سجل مواطن.
first_name, second_name, third_name, family_name	Full name divided into four parts for accurate identification.	الاسم الكامل مقسّم إلى أربعة أجزاء لتحديد الهوية بدقة.
document	The type of official document presented by the citizen like National ID, Passport, Birth Certificate.	نوع المستند الرسمي الذي يقدمه المواطن مثل بطاقة الهوية، جواز السفر، شهادة الميلاد.
identity_type	Category of identification used like personal ID, spouse ID.	نوع الهوية المستخدمة مثل هوية شخصية أو هوية الزوج/الزوجة.
identity_number	The citizen's identification or document number.	رقم الهوية أو المستند الخاص بالمواطن.
date_of_birth	Date of birth of the citizen.	تاريخ ميلاد المواطن.
gender	Gender of the citizen.	جنس المواطن.
marital_status	Current marital status.	الحالة الاجتماعية الحالية.
spouse_first_name, spouse_second_name, spouse_third_name, spouse_family_name	Full name of the spouse.	الاسم الكامل للزوج أو الزوجة.
spouse_identity_type	Type of identification of the spouse.	نوع هوية الزوج أو الزوجة.

Field Name	English Description	الوصف بالعربية
spouse_identity_number	Identification number of the spouse.	رقم هوية الزوج أو الزوجة.
is_war_victim	Indicates if the person is a war victim.	يوضح ما إذا كان المواطن متضررًا من الحرب.
is_disabled	Indicates if the citizen has a disability.	يوضح ما إذا كان المواطن من ذوي الإعاقة.
disability_type	Specifies the type of disability.	يحدد نوع الإعاقة.
has_chronic_disease	Indicates if the person has chronic diseases.	يوضح ما إذا كان المواطن يعاني من أمراض مزمنة.
phone_1, phone_2	Primary and secondary phone numbers.	رقم الهاتف الأساسي والثانوي.
governorate, city, housing_complex, neighborhood, street, address, nearest_landmark	Complete address details.	تفاصيل العنوان الكامل للمواطن.
family_members_count	Total number of family members.	العدد الإجمالي لأفراد الأسرة.
male_count, female_count	Number of males and females in the family.	عدد الذكور والإناث في الأسرة.
children_under_2, children_under_3, children_under_5	Number of children under these age groups.	عدد الأطفال دون هذه الفئات العمرية.
children_aged_5_16_count, children_aged_6_18_count	Number of school-age children.	عدد الأطفال في سن الدراسة.
number_of_school_students, number_of_university_ students	Number of students at school and university levels.	عدد الطلاب في المدارس والجامعات.
number_of_infants	Number of infants in the family.	عدد الرضّع في الأسرة.
number_of_people_with_disabilities	Number of people with disabilities.	عدد ذوي الإعاقة في الأسرة.
number_of_people_with_ chronic_diseases	Number of people with chronic diseases.	عدد المصابين بأمراض مزمنة.
number_of_elderly_over_60	Number of elderly people above 60.	عدد كبار السن فوق 60 سنة.
number_of_pregnant_women	Number of pregnant women.	عدد النساء الحوامل.
number_of_breastfeeding_ women	Number of breastfeeding women.	عدد النساء المرضعات.
number_of_injured_due_to_war	Number of family members injured by war.	عدد المصابين من أفراد الأسرة بسبب الحرب.

Field Name	English Description	الوصف بالعربية
is_caring_for_non_family_members	Whether the family cares for non-family members.	هل ترعى الأسرة أفرادًا من خارجها.
number_of_children_cared_for_not_in_family_under_18	Number of non-family children under 18 cared for.	عدد الأطفال دون 18 سنة من خارج الأسرة الذين تتم رعايتهم.
reason_for_caring_for_children	Reason for caring for non-family children.	سبب رعاية الأطفال من خارج الأسرة.
is_family_member_lost_during_war	Indicates loss of family members during the war.	يوضح ما إذا فُقد أحد أفراد الأسرة خلال الحرب.
relationship_to_family_members_lost_during_war	Relationship to lost family members.	صلة القرابة بالأفراد المفقودين.
housing_ownership	Type of housing ownership owned, rented, shelter.	نوع ملكية السكن مملوك، مستأجر، مأوى.
type_of_housing	Classification of the housing house, tent, apartment	نوع السكن منزل، خيمة، شقة.
is_war_damage	Indicates if the house was damaged by war.	يوضح ما إذا كان المنزل قد تضرر بسبب الحرب.
extent_of_housing_damage_due_to_war	Degree of housing damage caused by war.	مدى الضرر في السكن نتيجة الحرب.
is_displaced_due_to_war_and_changed_housing_location	Indicates if the family was displaced.	يوضح ما إذا تم تهجير الأسرة بسبب الحرب.
displaced_governorate, displaced_housing_complex, displaced_street, displaced_address, displaced_place_of_displacement	New address details after displacement.	تفاصيل العنوان الجديد بعد النزوح.
urgent_basic_needs_for_family	Urgent needs identified by the family.	الاحتياجات الأساسية العاجلة للأسرة.
secondary_needs_for_family	Secondary or less urgent family needs.	الاحتياجات الثانوية للأسرة.
sources_of_family_income	Sources of income for the family.	مصادر دخل الأسرة.
monthly_income_shekels	Family's monthly income in shekels.	الدخل الشهري للأسرة بالشيكل.
is_unable_to_use_land_or_properties_due_to_war	Whether the family lost access to land or property due to war.	هل فقدت الأسرة القدرة على استخدام الأراضي أو الممتلكات بسبب الحرب.
has_bank_account	Indicates if the family has a bank account.	يوضح ما إذا كانت الأسرة تمتلك حسابًا بنكيًا.
account_holder_name, bank_name_branch, account_holder_id_number, account_number	Banking information for aid transfer.	بيانات الحساب البنكي للتحويلات المالية.

Field Name	English Description	الوصف بالعربية
agree_to_share_data_for_assistance	Consent to share data with humanitarian organizations.	الموافقة على مشاركة البيانات مع الجهات الإنسانية.
exported_at	Timestamp that marks when a record was successfully synchronized to Google Sheets. It remains null until the synchronization process is completed. Once marked with a date and time, it indicates that the citizen's data has been exported and should not be resent again.	الحقل الذي يحدد وقت وتاريخ مزامنة السجل إلى Google Sheets. يبقى فارغًا حتى تكتمل عملية المزامنة، ويتم تعبئته بعد نجاح تصدير البيانات للدلالة على أن السجل تم مزامنته ولا يجب إعادة تصديره مرة أخرى.

5.2 Model Design

The model starts with the user interface (UI) form, where citizens fill in their information for aid registration. This form is a web-based data entry system deployed on Render, allowing the public to fill the form online.

When a citizen submits the form, the data is stored in a remote database. The main table in the database is the citizens table, which contains personal information such as name, ID, contact details, and aid-related data. The table also includes a column named exported_at, which remains null for all new records.

To transfer the collected data into the automation system, a scheduled background process is developed. This process checks the database for new records where the exported_at field is null. It then exports these records to a Google Sheet used by the n8n workflow. Once each record is successfully exported, the system updates the exported_at column with the current timestamp to prevent reprocessing of the same entry.

This automated export job is triggered by an external monitor using UptimeRobot, which runs a specific route from the web server every hour. The route activates a job that fetches new citizens' data, writes it to the Google Sheet, and updates the exported_at field.

When the data reaches Google Sheets, it becomes the entry point for the n8n workflow. The workflow uses a Google Sheets Trigger node that detects any new

rows added to the sheet. The trigger also runs every hour, but not at exactly the same time as UptimeRobot. While UptimeRobot executes the export task each hour based on the minute it was originally configured, the n8n trigger activates slightly later—typically at the next full hour. This sequence ensures that the workflow starts after UptimeRobot has completed its data export, allowing all newly added records to be processed correctly.

This model ensures smooth synchronization between the web form, the database, and the workflow. It eliminates manual data transfers, keeps citizen records organized, and allows continuous data flow from form submission to automated aid application processing.

Figure 5.1.1 shows the model design of the Gaza Madad Flow system. It shows how citizen data moves from the user interface to the automation workflow. When a citizen submits the form, the data is stored in the remote database. Every hour, the backend job, triggered by UptimeRobot, exports new records to Google Sheets. The Google Sheets trigger in n8n then detects these entries and starts the Gaza Madad Flow workflow to process and submit the data automatically.

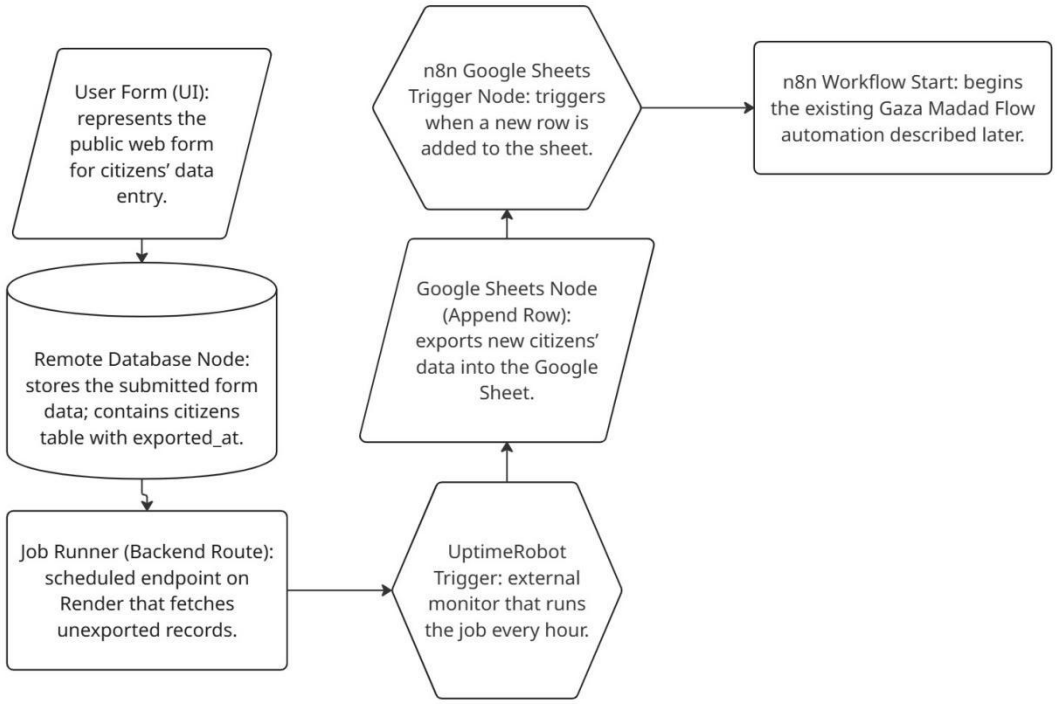


Figure 5.1: Model Design of the Gaza Madad Flow system

5.3 Workflow Design

Figure 5.2 shows the overall workflow of the Gaza Madad Flow system. The process starts with a trigger on the citizens' Google Sheet whenever new data is added. Then the system reads the list of aid links from the Aid Links Sheet and loops through them one by one. For each aid link, the workflow scrapes the page content and extracts the required fields. These fields are then mapped to the citizens' information, and the system loops over each citizen's data to prepare the inputs. A condition checks whether the record is new (by checking the Scrapped At field). If it is new, the system submits the form using the appropriate method (HTTP request or browserless). Finally, the sheet updates with the date of processing, completing the workflow.

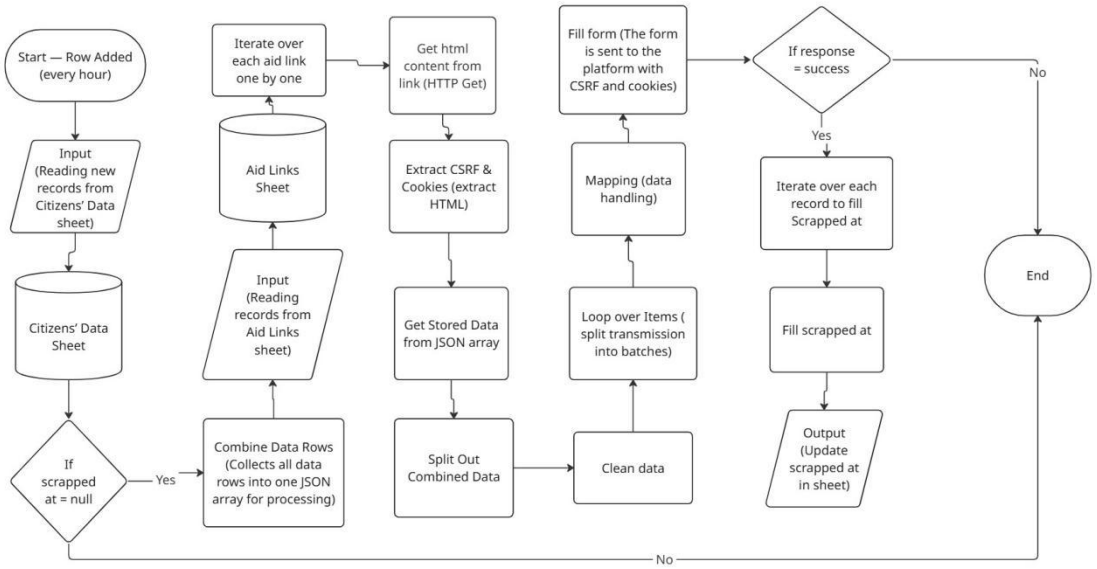


Figure 5.2: Workflow diagram of the Gaza Madad Flow system

5.4 Workflow Implementation

5.4.1 Technical Specifications

Google Sheets:

Google Sheets is a free, web-based spreadsheet app from Google inside Google Drive [35]. It lets users create, update, and change spreadsheets and share data online in real time [35]. Google Sheets is part of the Google Docs Editors suite, which also includes Google Docs, Slides, and Forms [35].

In our project, Google Sheets serves as the main database. It stores aid links used in the workflow and citizen data for aid requests. Google Sheets is connected to n8n, acting as a central data store. The citizen data collected through Google Forms is saved in Google Sheets automatically, making it easy to access, collaborate in real time, and update data. This helps the system monitor, validate, and process data for each person who seeks aid.

n8n:

n8n is an open-source workflow automation tool that lets users automate tasks and connect different services and apps without a lot of code [36]. It has a visual interface to design workflows that link to many APIs and services, making complex processes smoother [36].

In our project, n8n automates fetching aid links from Google Sheets, processing citizen data, and sending data to different aid platforms. Automating these tasks makes things faster and reduces human error.

5.4.2 Workflow Technical Implementation

Google Sheets Node:

The Google Sheets node automates work in Google Sheets and connects Google Sheets with other apps [37]. n8n supports many Google Sheets features, such as creating, updating, deleting, appending, removing, and getting documents [37].

In our workflow, we use the Google Sheets node three times. First, as a trigger to fetch citizens' data when a new record is added. Second, as a Get Rows operation to get the list of aid links. Finally, to update the "Scrapped At" column in the sheet after the workflow finishes.

Edit Fields (Set) Node:

The Set node adds or changes data in the workflow [37]. It can create new fields or overwrite existing ones, which is important in flows that rely on data from earlier nodes [37].

In our workflow, we use the Set node three times. First, in the "Clean Data" step, it converts empty values to numbers (for example, missing counts → 0). Second, in the "mapping" step, it arranges citizens' data into the correct format and field names

expected by the aid form, including fields like full name, province codes, and marital status. Finally, in the “fill scrapped at” step, it adds a timestamp to the “Scrapped At” column once the entry is submitted successfully.

Loop (SplitInBatches):

The Loop node processes many items or repeats an action, going through a list of elements one by one [37].

In our workflow, we use loops in two places: first, “Loop Over Items” goes through all aid links for each citizen, and second, “Loop Over Items1” goes through each citizen’s mapped data to ensure it is submitted correctly across all aid platforms.

If Node:

The If node is a conditional node that splits a workflow into two paths based on a comparison [37].

In our workflow, first we use it to check the “Scrapped At” column to see if a row has already been processed. This prevents duplicate submissions and makes sure only new citizen data is read and sent, then used to check the form action in the extracted HTML, helping the workflow decide which aid platform (form) is currently being processed, last usage To check the response result after submission, confirming whether the submission was successful before updating the sheet.

HTTP Request Node:

The HTTP Request node makes HTTP requests to pull data from any service with a REST AP [37]. It supports methods like GET, POST, PUT, DELETE, and allows headers, query parameters, and body data [37].

In our workflow, we use it twice. First, with a GET request to fetch the HTML page content of the aid links, which is later processed by the Extract HTML node. Second, with a POST request to submit citizen data into the online form. This POST request includes extra headers, cookies, and a CSRF token to mimic a real browser session and bypass anti-bot protections, ensuring successful submissions.

Extract HTML Node:

The Extract HTML node pulls content from an HTML source using CSS selectors [37].

In our workflow, it extracts three key items from the HTML page of the aid links: the form labels (to map citizen data to input fields), the CSRF security token (needed for valid POST submissions), and the session cookies (to authenticate the request). By pulling these details, the workflow places citizen data into the correct form fields and passes the site's security checks.

Code Node (Store Data):

The Code node allows custom JavaScript execution inside the workflow. It is used to manipulate or prepare data in ways that are not directly possible using built-in nodes [37].

In this project, we use the Code node to collect all citizens' records from Google Sheets and store them together inside a single JSON array. This array is then passed to the next nodes, making it easier to split, reuse, and process the data multiple times within the workflow.

Split Out Node:

The Split Out node is used to separate combined data items into individual elements so that each one can be processed independently [37].

In our project, this node takes the JSON array created by the Code node and splits it back into single records. This enables the workflow to handle every citizen's information separately during the submission phase.

Figure 5.3 shows the overall workflow of Gaza Madad Flow implemented in n8n.

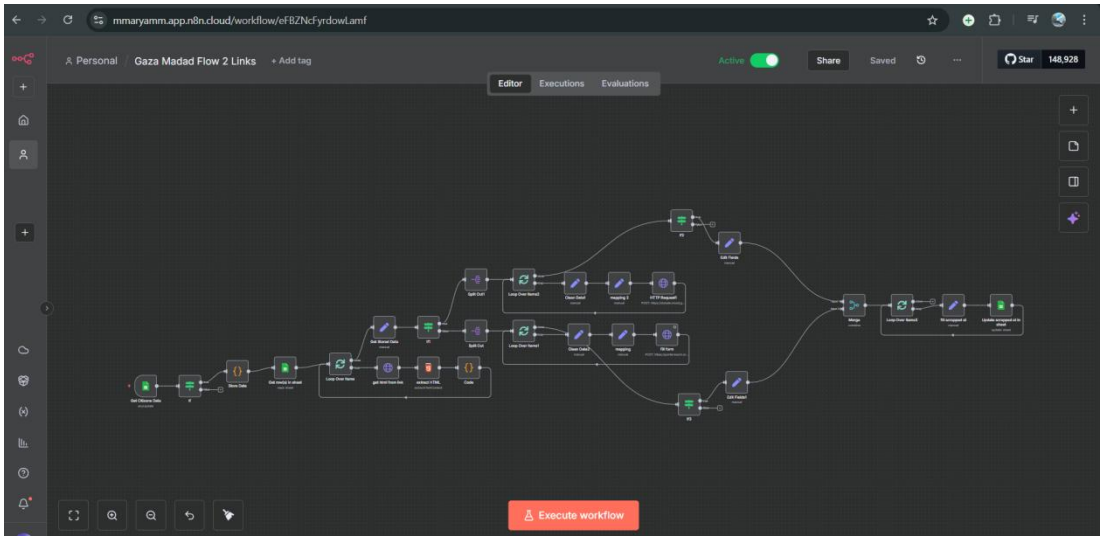


Figure 5.3: Gaza Madad Workflow Implementation in n8n

5.5 Workflow Testing

Testing Methodology

When we test the Gaza Madad Flow project, we use a clear method to ensure the system works correctly and reliably. Our main focus is end-to-end testing, which means we check the whole workflow from data collection to the final submission. To keep results realistic, we use dummy data with real-world traits so the tests reflect real situations the system may face.

Test Environment Setup

- Test Data: We prepare a realistic set of dummy data about beneficiaries and store it in Google Sheets. This keeps the test environment close to the real system while keeping data secure.

- Aid Links: We also create another Google Sheet to store "Aid Links." This sheet acts as the main data source for the n8n platform, helping the system get the needed links during testing and demonstration.

Test Scenarios

We design several main test scenarios to check the system's core features. For this, we use the Saudi Heritage Center's humanitarian aid portal (https://portal.ksach.org/), and the Future hope for affected families (https://donate.mosd.gov.ps/war-form) as a real test environment.

1. End-to-End Data Flow Test

Objective: Ensure the system reads data from Google Sheets, processes it through all steps, and sends it to the aid platform.

Procedure: We add new entries in the Google Sheet and watch how the system detects, processes, and submits the data.

2. Field Mapping Verification

Objective: Confirm that every data field goes to the correct place on the platform's registration form.

Procedure: We test values like "Full Name" and "Family Size" to see if they appear in the right form fields.

Conclusion and Demonstration

The testing phase shows that Gaza Madad Flow is accurate and efficient. Using Google Sheets as the data source proves that the system can handle practical, real-world scenarios with dummy but realistic test data.

Through portal, we demonstrate that the system can:

- Automatically scrape, map, and submit data.
- Complete the whole process without human intervention.

In conclusion, the portal helps validate our workflow during testing and will remain the official demo platform to showcase the project's functionality. Based on these tests, we confirm that the system is stable, accurate, and ready for further scaling and development.

5.6 Result

The Gaza Madad Flow system successfully automates the process of filling and submitting humanitarian aid forms across multiple online platforms. In addition to the workflow itself, a user interface (UI) was developed to make system control, monitoring, and data validation easier.

The UI allows administrators to:

- Validate and review citizens' data before it is processed by the workflow.
- View and monitor submission progress in real time.
- Manage and update the list of aid links.
- Review submission logs and check for any errors.

This interface provides a simple and accessible way to interact with the automation process, especially for users who do not work directly inside n8n. It ensures that citizen information is verified and accurate before the system performs automatic submissions, improving the overall reliability and accuracy of the Gaza Madad Flow system.

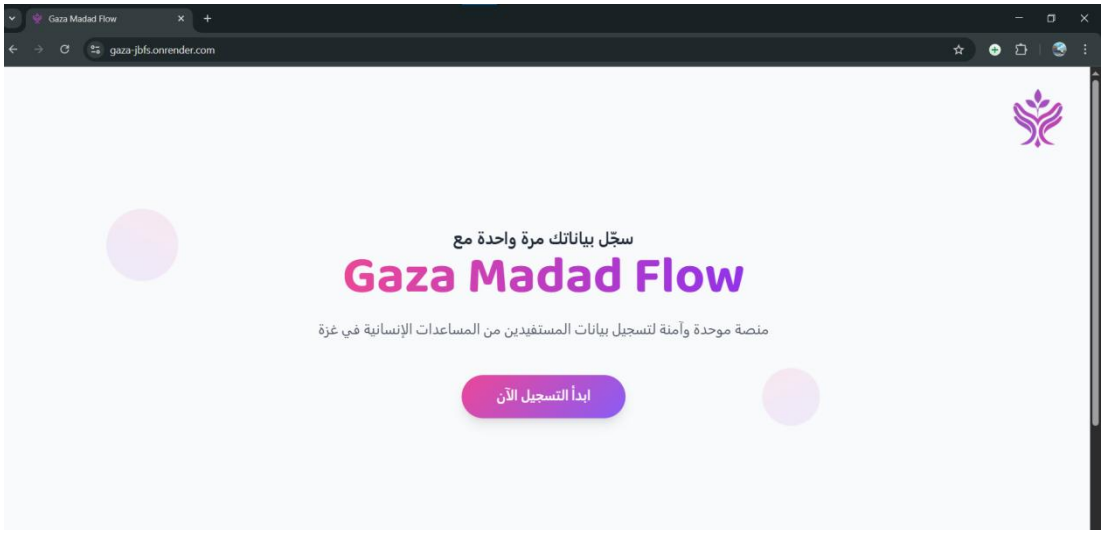


Figure 5.4.a: Gaza Madad Workflow Website Implementation

Figure 5.4.b: Gaza Madad Workflow Website Implementation

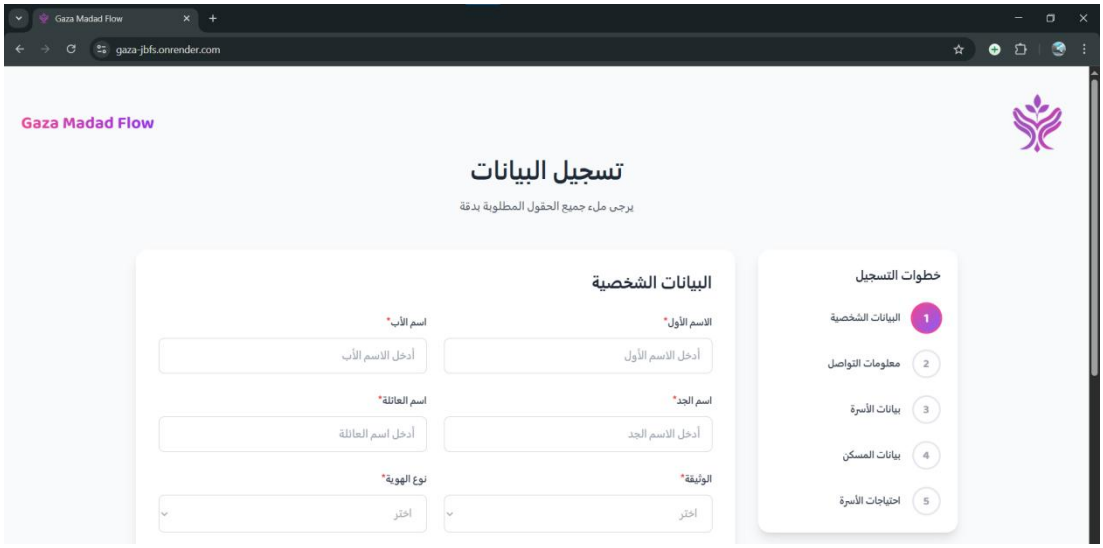
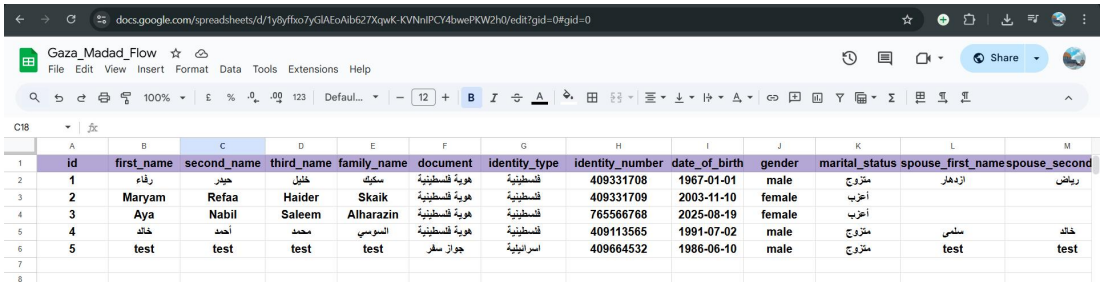


Figure 5.4.c: Gaza Madad Workflow Website Implementation

A demonstration video of the Gaza Madad Flow website and workflow operation is available on Gaza Madad Flow.

Note: In the video, we changed the UptimeRobot scheduler interval from 1 hour to 5 minutes, and the Google Sheets trigger in n8n from 1 hour to 1 minute to reduce the video duration.

Figure 5.5 shows a portion of the citizens' data stored in Google Sheets, including personal details, demographic information, family conditions, income data, and processing status (the Scrapped At column records the date when each entry was exported and processed).



id	first_name	second_name	third_name	family_name	document	identity_type	identity_number	date_of_birth	gender	marital_status	spouse_first_name	spouse_second
1	رقاء	حيدر	خليل	سكيك	هوية فلسطينية	فلسطينية	409331708	1967-01-01	male	متزوج	ازدهار	رياض
2	Maryam	Refaa	Haider	Skaik	هوية فلسطينية	فلسطينية	409331709	2003-11-10	female	أعزب		
3	Aya	Nabil	Saleem	Alharazin	هوية فلسطينية	فلسطينية	765566768	2025-08-19	female	أعزب		
4	خالد	أحمد	محمد	السوسي	هوية فلسطينية	فلسطينية	409113565	1991-07-02	male	متزوج	سلمى	خالد
5	test	test	test	test	جواز سفر	اسرائيلية	409664532	1986-06-10	male	متزوج	test	test

Figure 5.5: Citizens Data Sheet used in Gaza Madad Flow

Chapter 6

Conclusion and Future Works

Chapter 6
Conclusion and Future Works

6.1 Conclusion

Humanitarian aid helps save lives and protect the dignity of people affected by wars, displacement, and crises. In this kind of situation, especially in Gaza after the October 2023 war, quick and reliable aid registration systems become very important. Because communication systems are badly damaged, power goes out, and internet is weak, traditional ways of registering for aid become slow and difficult.

Around the world, many digital platforms, like AidKit, UiPath, and Reach4Help, are created to make humanitarian work easier. These systems help register people, give out aid, and improve coordination. But most of these systems need a strong internet connection, central national systems, or access to official databases. These conditions are often missing in Gaza during conflict. So, these systems cannot work easily in emergencies, where people use limited mobile data, shared devices, and informal networks.

To close this gap, we create Gaza Madad Flow a simple, flexible, and automatic registration model made for the emergency needs of displaced and affected people in Gaza. The main idea is to let users register once using a single online form. Then, the data automatically spreads across many aid platforms and registration links. This lowers repeated work, saves time, and speeds up support during emergencies.

The system uses Google Forms, Google Sheets, and the n8n automation platform to collect, format, check, and send data to other platforms. We use a step-by-step method called Incremental Prototyping. We build and test the system gradually: first collecting data, then automating a single form, and finally testing it in real cases with multiple platforms. Each step makes the system better and adds useful insights.

Gaza Madad Flow is not just a technical idea; it is a real solution to help bridge the digital gap in aid access. By using simple automation tools, we show that even with little infrastructure, fast and effective humanitarian registration and coordination are

possible. The project shows how technology when made for the specific situation can help improve resilience, fairness, and dignity in times of crisis.

6.2 Future Works

In the future, Gaza Madad Flow will keep improving to better support aid work in Gaza Strip and other regions. We plan to add more features to reach more people, work faster, and adapt to different needs.

One goal is to connect the system with more aid platforms, including government and NGO websites. This way, data can go straight to trusted sources, speeding up help and making the information more accurate.

Another focus is making the system handle different types of forms automatically. Instead of adjusting it for each new form, it will recognize and adapt on its own. This makes it easier to expand and maintain.

We also want to use artificial intelligence to help analyze user input. AI can identify urgent cases quicker and suggest the best actions to aid workers, making the process smarter.

Supporting multiple languages and devices is another priority. This will make the system more accessible to varied populations, especially those using basic phones or sharing internet.

Finally, we plan to turn Gaza Madad Flow into a cloud-based service. It will be faster, more secure, and able to give real-time feedback. Aid organizations will find it easier to customize and use for their specific needs.

All these improvements aim to make Gaza Madad Flow a more powerful tool. It can help with quick and fair registration processes not only in Gaza Strip but also in other places facing crises and displacement.

Additionally, the system can be improved through several new practical features that emerged during testing and evaluation. One important addition is Dynamic Link Addition, where new aid registration links introduced after users have already registered will automatically receive existing user data without requiring re-registration. Another planned feature is Registration Status Tracking, which will add a “Status” field to record whether each registration attempt was successful or failed, enabling automatic retries for failed submissions.

To make the process even smoother, User-Link Mapping will dynamically link each user to multiple aid platforms, allowing the system to handle successful and failed submissions separately. Furthermore, Data Editing and Updating will let users or administrators modify registration data when updates or corrections are needed, ensuring continuous data accuracy and integrity.

These combined improvements will make Gaza Madad Flow smarter, faster, and more adaptable — ready to meet the growing humanitarian and technical challenges of the future.

Chapter 7
BIBLIOGRAPHY

Bibliography

[1] Wikipedia, “Humanitarian aid in conflict zones,” Jun. 3, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Humanitarian_aid_in_conflict_zones

[2] World Food Programme, “Conflict and hunger,” Oct. 19, 2020. [Online]. Available: https://www.wfp.org/conflict-and-hunger

[3] World Economic Forum, “Digital tools to build a better world,” Jan. 15, 2018. [Online]. Available: https://www.weforum.org/stories/2018/01/digital-tools-to-build-a-better-world

[4] Palladium Group, “From data to action: The emerging role of AI in the humanitarian sector,” Mar. 26, 2025. [Online]. Available: https://thepalladiumgroup.com/news/From-Data-to-Action-The-Emerging-Role-of-AI-in-the-Humanitarian-Sector

[5] World Food Programme, “Does artificial intelligence hold the key to ending hunger?” May 15, 2025. [Online]. Available: https://www.wfp.org/stories/does-artificial-intelligence-hold-key-ending-hunger

[6] UCIPR Ukraine, “Presentation of the concept for improving the automated humanitarian aid system,” May 13, 2025. [Online]. Available: https://ucipr.org.ua/en/news/presentation-of-the-concept-for-improving-the-automated-humanitarian-aid-system

[7] UNRWA, “UNRWA situation report 169 – Gaza Strip and West Bank, including East Jerusalem,” May 1, 2025. [Online]. Available: https://www.unrwa.org/resources/reports/unrwa-situation-report-169-situation-gaza-strip-and-west-bank-including-east-jerusalem

[8] Laravel, “Installation,” Laravel Documentation, 2025. [Online]. Available: https://laravel.com/docs/12.x/installation

[9] MySQL, “What is MySQL?” [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

[10] PostgreSQL Global Development Group, “About PostgreSQL.” [Online]. Available: https://www.postgresql.org/about/

[11] Mozilla, “HTML: HyperText Markup Language,” MDN Web Docs, Jul. 9, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML

[12] E. Spivak, “What is Cascading Style Sheets (CSS)?” Wix Encyclopedia, Aug. 28, 2023. [Online]. Available:

https://www.wix.com/encyclopedia/definition/cascading-style-sheets-css
[13] Builder.io, “Tailwind CSS.” [Online]. Available: https://www.builder.io/glossary/tailwind-css
[14] M. Megida, “What is JavaScript? A definition of the JS programming language,” freeCodeCamp, Mar. 29, 2021. [Online]. Available: https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/
[15] n8n, “Workflow automation tool.” [Online]. Available: https://n8n.io/
[16] Wikipedia, “Google Sheets,” Sep. 20, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Google_Sheets
[17] Google, “Google Forms – Survey tool.” [Online]. Available: https://docs.google.com/forms/u/0/
[18] App Generator, “Getting Started with Render.” [Online]. Available: https://app-generator.dev/docs/deployment/render/index.html
[19] Uptime Robot s.r.o., “UptimeRobot: Free website monitoring service.” [Online]. Available: https://uptimerobot.com/
[20] GitHub, “Version control platform.” [Online]. Available: https://github.com/
[21] Microsoft, “Visual Studio Code – Code editor.” [Online]. Available: https://code.visualstudio.com/
[22] Microsoft, “Microsoft Word – Office suite.” [Online]. Available: https://www.microsoft.com/en-us/microsoft-365/word
[23] WhatsApp, “WhatsApp – Messaging platform.” [Online]. Available: https://www.whatsapp.com/
[24] Almotqadmon, “Almotqadmon platform.” [Online]. Available: https://www.motqdmon.com/?m=1
[25] Reach4Help, “A global aid connector.” [Online]. Available: https://reach4help.org/
[26] Government of Lebanon, “DAEM – Digital aid platform by Lebanese government,” 2021. [Online]. Available: https://daem.impact.gov.lb
[27] Government of Lebanon, “IMPACT platform,” 2021. [Online]. Available: https://impact.gov.lb/home
[28] UiPath, “Automation for humanitarian aid,” 2025. [Online]. Available: https://www.uipath.com/
[29] AidKit, “Emergency and disaster relief platform,” Jun. 26, 2025. [Online].

Available: http://aidkit.com/emergency-disaster-response
[30] CloudDefense.AI, “Software development life cycle (SDLC),” Sep. 14, 2023. [Online]. Available: https://www.clouddefense.ai/system-development-life-cycle/
[31] OpenText, “What is SDLC?” 2023. [Online]. Available: https://www.opentext.com/what-is/sdlc
[32] M. Tahir, T. Fatima, N. Khan, and M. Qasim, “Software Development Life Cycle (SDLC) Method [Figure],” ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/figure/Software-Development-Life-Cycle-SDLC-Method_fig1_369906619
[33] GeeksforGeeks, “Software engineering | Prototyping model,” Apr. 12, 2025. [Online]. Available: https://www.geeksforgeeks.org/software-engineering-prototyping-model/
[34] M. Khan, “Software development model: Prototyping,” Medium, Apr. 30, 2020. [Online]. Available: https://medium.com/design-bootcamp/software-development-model-prototyping-468b2ea16ae6
[35] Corporate Finance Institute, “Google Sheets.” [Online]. Available: https://corporatefinanceinstitute.com/resources/excel/google-sheets/
[36] A. Guzey, “What is n8n? Open source automation explained,” DEV Community, Jul. 7, 2025. [Online]. Available: https://dev.to/proflead/what-is-n8n-open-source-automation-explained-30i1
[37] n8n Documentation, “n8n documentation.” [Online]. Available: https://docs.n8n.io/