The islamic University of Gaza

Faculty of IT

الجامعة الإسلامية بغزة

كلية تكنولوجيا المعلومات

Web-Based Aid Management System for Organizations in Gaza

نظام إدارة المساعدات عبر الإنترنت للمنظمات في غزة

By

عبدالله أشرف محمد جرادة 120202594

محمد حبيب سعدي شلدان 120200515

أحمد يوسف أحمد لافي 120202117

محمد هيثم سليم جرادة 120202071

Supervised by

م. شادي سمارة

2025/05

Abstract

This project will present a web-based platform developed to serve institutions operating in the Gaza Strip. These institutions have faced challenges in managing import and export operations, registering beneficiaries, and obtaining accurate data to facilitate decision-making. Such processes were often carried out manually or through multiple, disconnected tools, which led to slower workflows and reduced accuracy of results. For example, a beneficiary's data might be entered in a paper record or Excel sheet and later updated in another tool, which could result in discrepancies between records or the loss of important details such as the delivery date or the value of assistance. This inconsistency in data could lead to inaccurate decisions, such as providing aid to someone who had already received it or failing to include an actual beneficiary in final reports.

In addition, traditional data entry methods can cause errors in beneficiaries' phone numbers, whether due to typographical mistakes or the failure to update numbers in a timely manner. This problem sometimes makes it difficult to contact targeted individuals to inform them about the date of receiving assistance or attending a specific event, which may cause them to miss the opportunity to benefit from the service on time and negatively affect the effectiveness of the programs provided.

The main goal of the project is to build an interactive digital system that helps institutions register beneficiaries, organize service data, and monitor import and export activities more effectively. The system is also equipped with real-time statistics and reports that enable management to make data-driven decisions.

Our team adopted the and delivering services in implementing the project, which relies on dividing the work into short phases (sprints). The system was developed incrementally, with each phase reviewed regularly. This approach provided flexibility in making adjustments and allowed quick implementation of improvements based on feedback. Laravel was used for the backend, while HTML, CSS, and JavaScript were used for the frontend. The project consists of a user interface and an admin dashboard.

The project is expected to achieve its objectives successfully, enhancing the efficiency of institutional work and reducing dependence on traditional methods. It will also provide an accurate database that enables service improvement and supports making reliable decisions. In the future, we recommend adding multilingual support and providing a mobile-compatible version to increase usability and expand the user base.

ملخص الدراسة

سيقدّم هذا المشروع منصة إلكترونية تم تطويرها لخدمة المؤسسات العاملة في قطاع غزة، حيث واجهت هذه المؤسسات صعوبات في إدارة عمليات الصادرات والواردات، وتسجيل المستفيدين من خدماتها، والحصول على بيانات دقيقة تسهّل عملية اتخاذ القرار. كانت هذه العمليات تُنفذ غالبًا بشكل يدوي أو باستخدام أدوات متعددة ومنفصلة، مما أدى إلى بطء في العمل وقلة الدقة في النتائج. فعلى سبيل المثال، قد يتم إدخال بيانات أحد المستفيدين في سجل ورقي أو ملف Excel ثم يتم تحديثه في وقت لاحق على أداة أخرى، الأمر الذي قد يتسبب في اختلاف الأرقام بين السجلات أو فقدان بعض المعلومات الهامة مثل تاريخ الاستلام أو قيمة المساعدة. هذا التباين في البيانات قد يؤدي إلى قرارات غير دقيقة، مثل صرف المساعدات لشخص استلمها مسبقًا أو عدم احتساب مستفيد فعلي ضمن التقارير النهائية.

إضافة إلى ذلك، قد تتسبب الطرق التقليدية في تسجيل البيانات بأخطاء في أرقام الهواتف الخاصة بالمستفيدين، سواء بسبب خطأ كتابي أو عدم تحديث الرقم في الوقت المناسب. هذه المشكلة تؤدي أحيانًا إلى صعوبة التواصل مع الأشخاص المستهدفين لإبلاغهم بموعد استلام المساعدة أو حضور فعالية معينة، مما قد يضيع عليهم فرصة الاستفادة من الخدمة في الوقت المحدد ويؤثر سلبًا على فعالية البرامج المقدمة

هدف المشروع بشكل رئيسي إلى بناء نظام رقمي تفاعلي يساعد المؤسسات على تسجيل المستفيدين، وتنظيم بيانات الخدمات المقدمة، ومتابعة عمليات الصادرات والواردات بطريقة أكثر فاعلية. كما زُوّد النظام بإحصائيات وتقارير لحظية تُمكّن الإدارة من اتخاذ قرارات مبنية على بيانات دقيقة.

اعتمدنا كفريق العمل في تنفيذ المشروع على منهجية Agile التي تعتمد على تقسيم العمل إلى مراحل قصيرة (Sprints)، حيث جرى تطوير النظام بشكل تراكمي مع مراجعة كل مرحلة بشكل مستمر. هذا الأسلوب أتاح مرونة في التعديل وسرعة في تنفيذ التحسينات بناءً على الملاحظات. تم استخدام Laravel في البرمجة الخلفية، ولغات HTML وCSS وJavaScript في الواجهة الأمامية. يتكوّن المشروع من واجهة مخصصة للمستخدمين ولوحة تحكم للإدارة.

سيحقق المشروع أهدافه بنجاح، وسيساهم في رفع كفاءة العمل داخل المؤسسات، والحد من الاعتماد على الطرق التقليدية. كما سيتيح قاعدة بيانات دقيقة تُمكّن من تحسين الخدمات واتخاذ قرارات مبنية على معلومات موثوقة. نوصي في المستقبل بإضافة دعم للغات متعددة وتوفير نسخة متوافقة مع الهواتف المحمولة لزيادة سهولة الاستخدام وتوسيع قاعدة المستخدمين.

Dedication

To our parents and families who provided us with the tremendous support and assistance we needed during this great journey.

To

Our supervisor

Engineer:- Shadi Samara

For his valuable guidance and advice throughout the project, and for his patience, tolerance, and support, especially in this difficult situation due to the war.

To

To everyone who helped us complete this project, to our martyrs, our wounded, and our people in the beloved Gaza Strip.

Acknowledgment

Elhamdulillah , who has blessed us and enabled us to achieve this accomplishment despite the war and the difficulties we faced during this aggression against our beloved Gaza.

Thank you to the IHH Foundation for contributing to the success of our idea and for supporting us during these circumstances.

Thank you also to the Deanship of our College, the College of Information Technology, for their wonderful cooperation over the past years and during the war. We ask Allah to protect and support our teachers.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

IDE	Integrated Development Environment
PHP	Personal Home Page
RDBMS	Relational Database Management System
ERD	Entity-Relationship Diagram
GUI	Graphical User Interface
AI	Artificial Intelligence
GPT	Generative Pre-trained Transformer
SRS	System Requirements Specification
UI	User Interface
CRUD	Create, Read, Update, Delete
UAT	User Acceptance Testing
URL	Uniform Resource Locator
FK	Foreign Key
NGOs	Non-Governmental Organizations
UML	Unified Modeling Language
MVC	Model–View–Controller
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
AJAX	Asynchronous JavaScript and XML
UAT	User Acceptance Testing
env	Environment file
API	Application Programming Interface
PHPUnit	PHP Unit Testing Framework
DevTool	Developer Tools
PDF	Portable Document Format

Chapter 1
Introduction

1.1 Introduction

The World Wide Web has emerged as one of the most significant technological transformations of the modern era, radically reshaping how individuals, businesses, and organizations operate [1] platform for managing. It facilitates instant access to information, enables comprehensive communication, and supports the automation of complex processes. For modern organizations, web-based platforms offer an efficient means to consolidate and process data, manage workflows, and deliver services to beneficiaries with greater accuracy and efficiency [2].

In line with this global transformation, our graduation project focuses on developing an electronic platform specifically designed to support organizations operating in the Gaza Strip [3]. The platform aims to standardize operational procedures, streamline workflows, and improve the accuracy of beneficiary data for better decision-making. Key functionalities include tracking and recording imports and exports, managing warehouse inventories, supporting large-scale distribution operations, and generating comprehensive reports to assist in strategic planning and monitoring progress.

Despite the availability of digital tools, many organizations in Gaza still face significant challenges in managing their core operations effectively. This is particularly evident in tracking imports and exports along with their sources and destinations, maintaining accurate and up-to-date beneficiary records, and producing reliable statistics to guide strategic planning. In many cases, [4]. For example, beneficiary information might initially be recorded on paper or in spreadsheets and later updated in a separate system, resulting in data mismatches, missing details, or duplicated records [5]. Such issues can lead to errors in decision-making, such as providing aid to individuals who have already received it [6], or failing to notify eligible beneficiaries.

Additionally, reliance on traditional data management methods often results in errors in contact information, particularly phone numbers [7]—either due to typographical mistakes or outdated records. These errors hinder effective communication with beneficiaries, making it difficult to inform them of aid distribution dates or scheduled events, which in turn reduces the overall effectiveness of organizational programs.

To address these challenges, this project proposes the development of an interactive online platform that integrates and simplifies data management processes. The system enables efficient beneficiary registration, accurate monitoring of warehouse imports and exports, and provides real-time statistical reporting through dedicated user interfaces [8], empowering organizations with timely and reliable information for improved decision-making and service delivery..

1.2 Statement of the problem

During the ongoing war, organizations in the Gaza Strip face significant challenges in managing critical operations such as tracking imports and exports, maintaining

warehouse inventories, and accurately recording the distribution of aid. In many cases, these processes are still conducted manually through paper records, spreadsheets, or separate, unlinked databases. This approach slows workflows, fragments operational processes, increases the risk of data loss, and limits access to accurate and timely information.

For example, beneficiary or shipment data may be recorded in one system and later updated in another, resulting in inconsistencies such as missing delivery dates, inaccurate allocation records, or duplicated entries. These discrepancies can lead to resource mismanagement, for instance, providing assistance to individuals who have already received it or excluding eligible recipients from aid distribution.

In addition, errors in contact information particularly, outdated phone numbers or incomplete records, can severely hinder communication. This makes it difficult to notify beneficiaries, partners, or other stakeholders about aid distribution schedules or planned events, reducing participation and missing opportunities to deliver timely assistance.

Without a centralized, integrated platform to manage these operations, organizations will continue to face operational inefficiencies, inconsistent records, reduced program effectiveness and reduced transparency. A unified web-based system is therefore essential to streamline workflows, ensure data accuracy, and provide real-time reporting to support informed decision-making during both emergency and recovery phases.

1.3 Objectives

The project aimed to design and implement an interactive web-based platform that addresses the challenges identified in the problem statement. By integrating all operations into a single system, the platform seeks to improve data accuracy, reduce delays, and enhance communication with beneficiaries.

1.3.1 Main Objective

To develop an integrated web-based platform that enables organizations in the Gaza Strip to efficiently manage imports and exports, maintain accurate warehouse inventories, register beneficiaries, and generate real-time reports to support effective decision-making.

1.3.2 Sub Objectives

The following sub-objectives were defined to ensure the successful achievement of the main project goal. Each sub-objective directly contributes to the realization of the overall aim:

- Encourage organizations to adopt and actively use the proposed platform.

- Develop a secure, intuitive, and user-friendly interface for beneficiary registration and data management to ensure accuracy and efficiency.
- Centralize the management of imports, exports, and aid distribution to streamline institutional operations.
- Improve data integrity by minimizing manual entry errors, preventing duplication, and applying automated validation.
- Provide real-time statistics and reporting tools to support timely and evidence-based decision-making.
- Implement an administrative dashboard for activity monitoring, operational oversight, and report generation.
- Offer advanced search and filtering capabilities for fast and precise data retrieval.
- Ensure fair and timely distribution of assistance to all targeted beneficiaries.
- Replace fragmented tools and manual workflows with a unified digital system to enhance operational efficiency.
- Support future scalability through multilingual support, mobile compatibility, and API integration for seamless system interoperability.

1.4 Scope and Limitations

1.4.1 Scope

The scope of this project covers the design and development of a web-based platform tailored for organizations operating in the Gaza Strip to manage imports, exports, warehouse inventories, and beneficiary records. The platform includes the following key features:

- Secure registration and management of beneficiary information.
- Centralized tracking of imports, exports, and aid distribution.
- Real-time statistics and reporting for informed decision-making.
- Administrative dashboard for monitoring operations and generating reports.
- Advanced search and filtering capabilities for quick data retrieval.
- Role-based access control to ensure system security and data privacy.
- Integration with APIs to enable smooth communication between system components.

1.4.2 Limitations

- The current version of the system has the following limitations:
- Designed primarily for desktop web browsers; mobile application support is not yet implemented.
- Language support is limited to a single primary language.

- Requires internet connectivity for all operations; offline functionality is not supported.
- The system does not integrate with external governmental or customs databases at this stage.
- Data accuracy depends on the correctness and timeliness of the information entered by authorized users.

1.5 Importance of the project

This project holds significant importance as it addresses critical operational challenges faced by organizations in the Gaza Strip, particularly during times of crisis. By transitioning from manual and fragmented data management methods to an integrated web-based platform, the system will:

- Enhance efficiency by streamlining the management of imports, exports, warehouse inventories, and beneficiary records within a unified interface.
- Improve decision-making through the availability of accurate, real-time statistics and reports, enabling organizations to respond swiftly to emerging needs.
- Increase transparency and accountability by maintaining consistent, traceable, and verifiable records of aid distribution and resource allocation.
- Optimize resource allocation by reducing duplication, minimizing errors, and ensuring equitable distribution of assistance to all targeted beneficiaries.
- In essence, the platform will not only improve the internal operations of organizations but also contribute to the broader goal of delivering timely, accurate, and fair services to those in need.

Chapter 2
Related Works

2.1 Rahma Around the world

A similar system, *"Rahma Around the World"*, was developed to manage aid distribution and beneficiary registration. While it focuses mainly on recording beneficiaries and tracking distributed aid, the proposed system provides a more advanced and comprehensive solution. It includes additional features such as **project management**, **warehouse tracking**, and **user role control**, as well as **import/export management** and **Excel report generation**. These enhancements make the system more flexible, data-driven, and suitable for large humanitarian organizations [9].

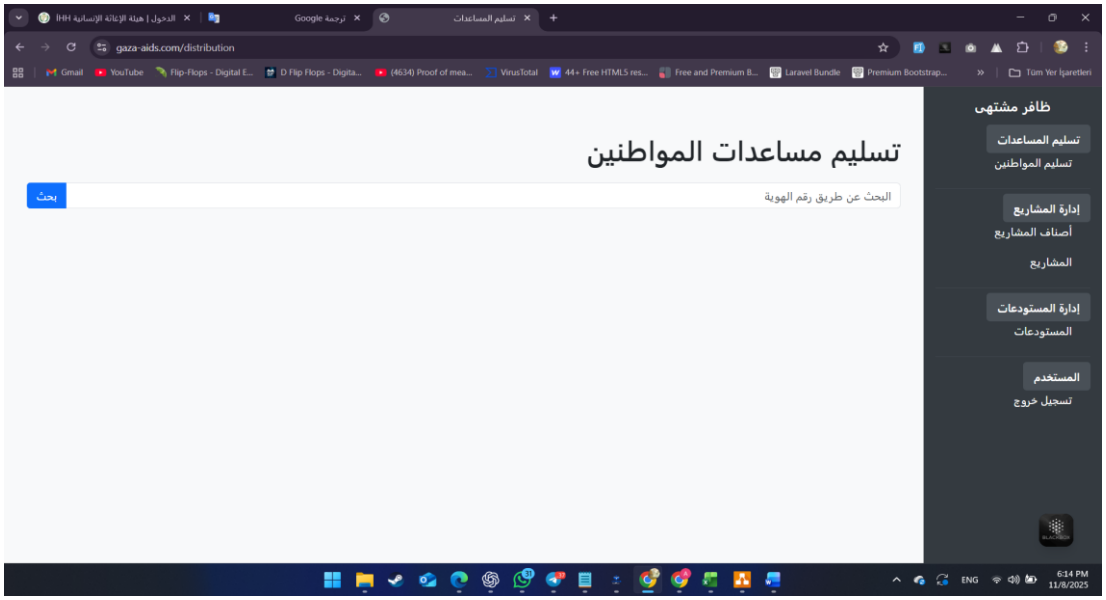

Figure 0-1 Rahma Around the world

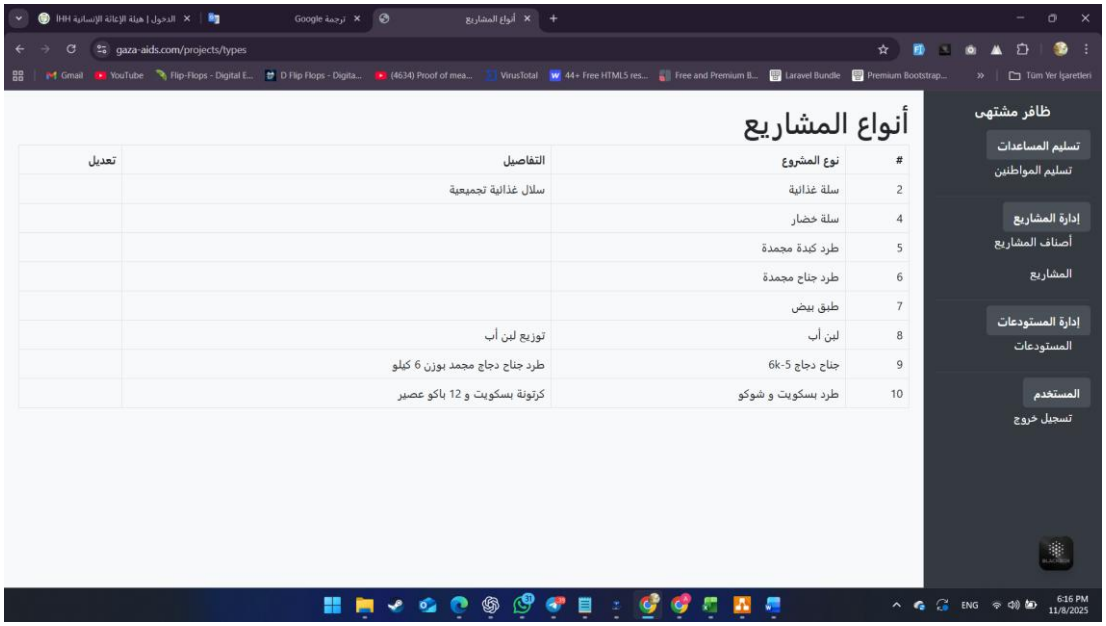

Figure 0-2 Rahma Around the world

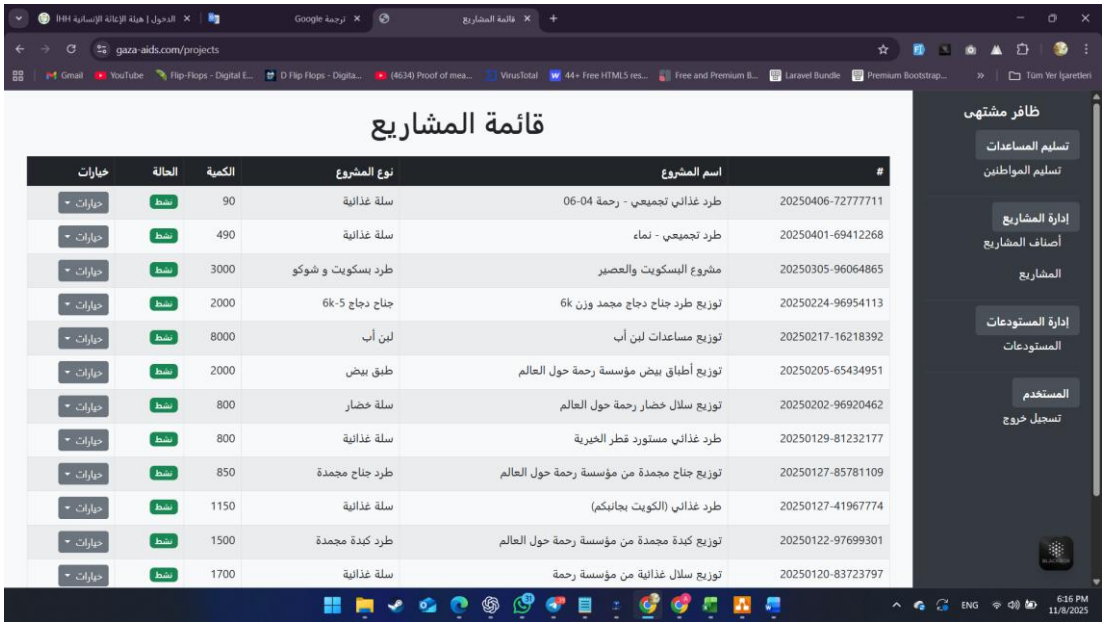

Figure 03- Rahma Around the world

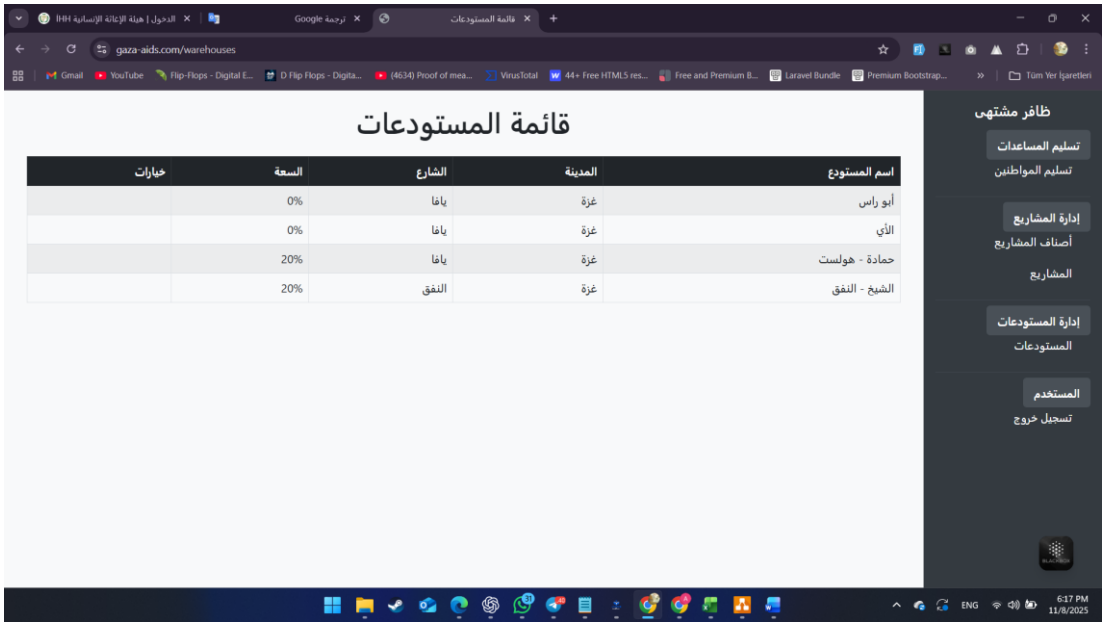

Figure 04- Rahma Around the world

2.2 Some similar sites

CharityTracker: https://www.charitytracker.com

📝A case management system for institutions to register beneficiaries, track aid, and generate reports.

✅Suitable for easily tracking beneficiaries and documenting deliveries. [10]

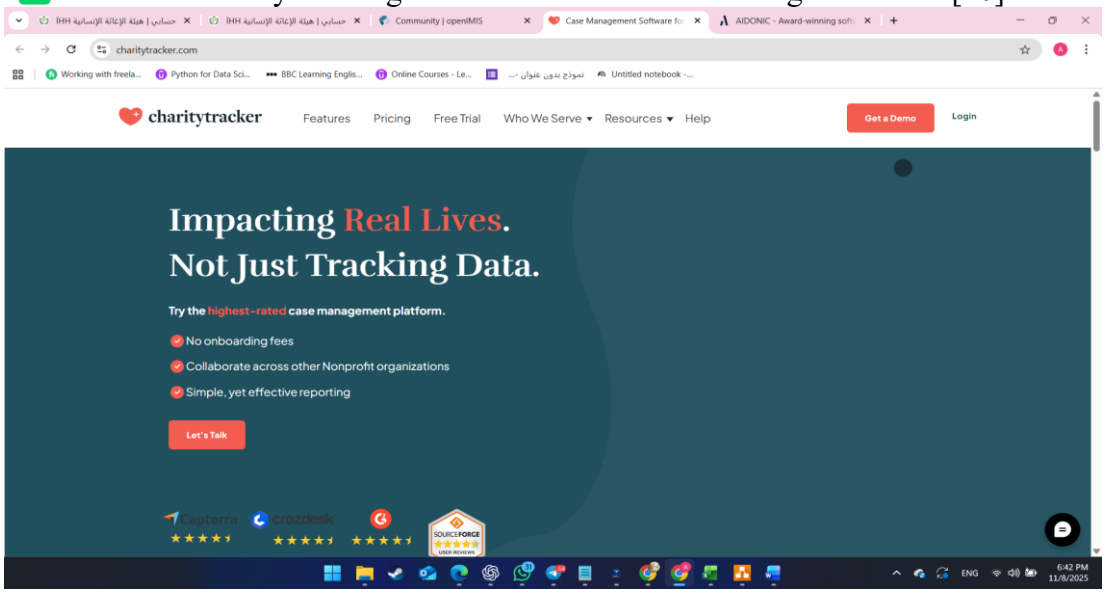

Figure 0-1 CharityTracker

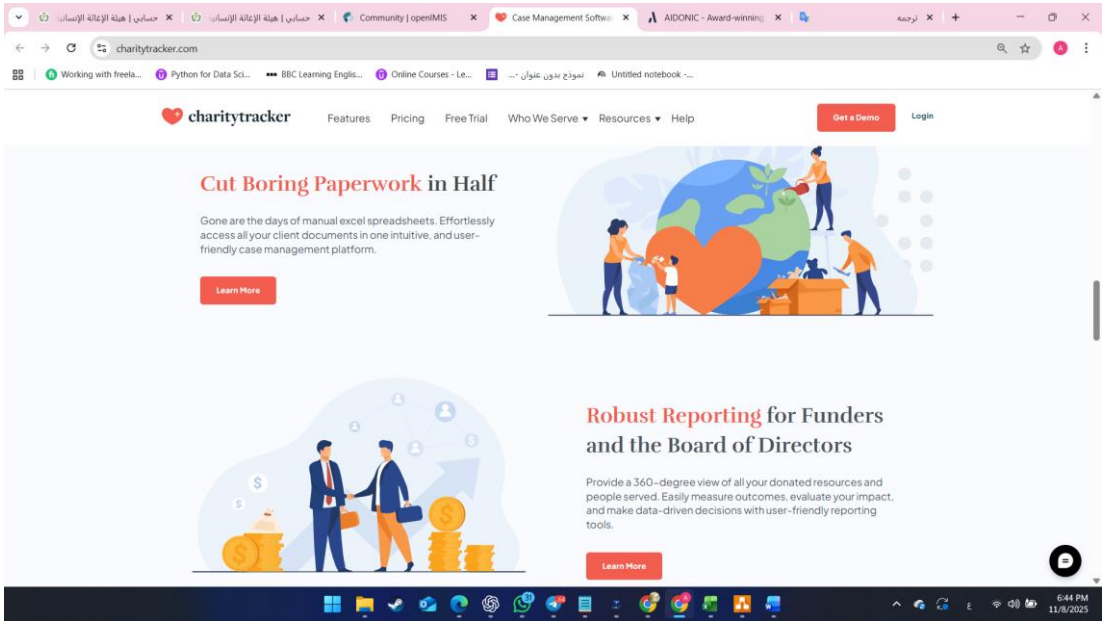

Figure 02- CharityTracker

openIMIS: https://openimis.org

📝An open-source program for managing beneficiary data and financial or in-kind benefits.

✅Ideal for organizations that wish to customize the system to suit their technical needs. [11]

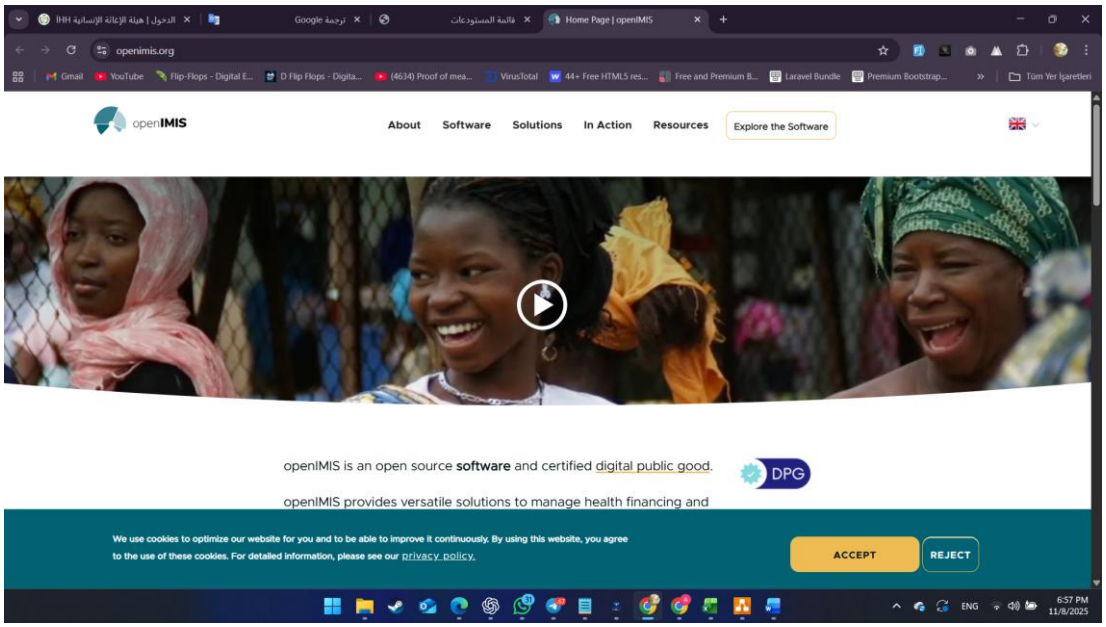

Figure 0-3 openIMIS

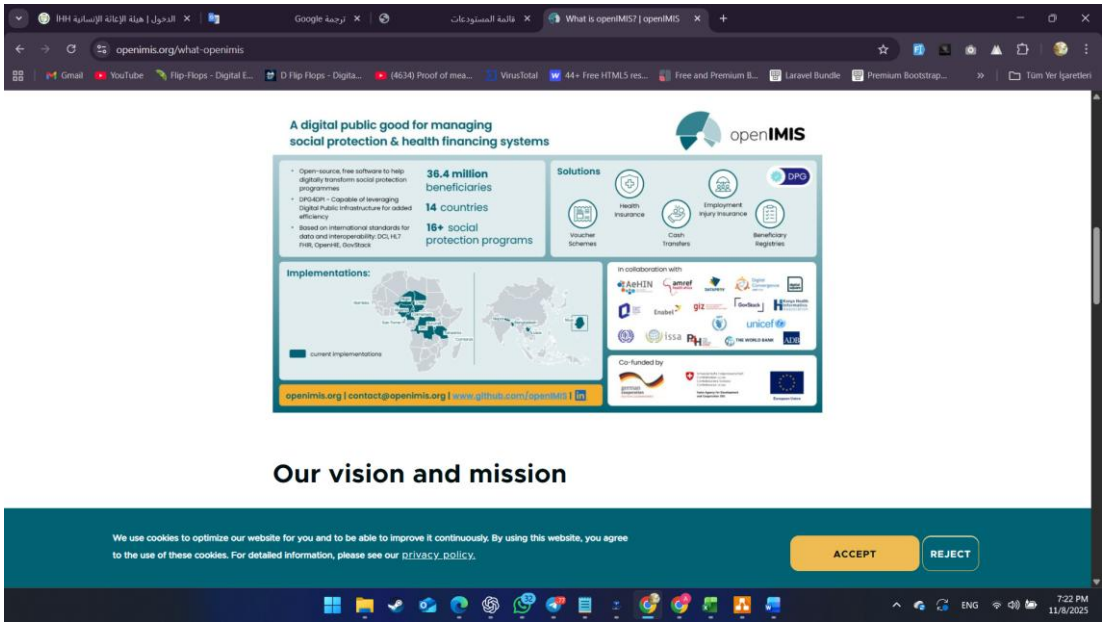

Figure 04- openIMIS

AIDONIC: https://www.aidonic.io

📝 An advanced digital platform for managing distributions and aid (including e-vouchers and payments).

✅ Ideal if you want a modern, transparent digital system with real-time tracking [12].

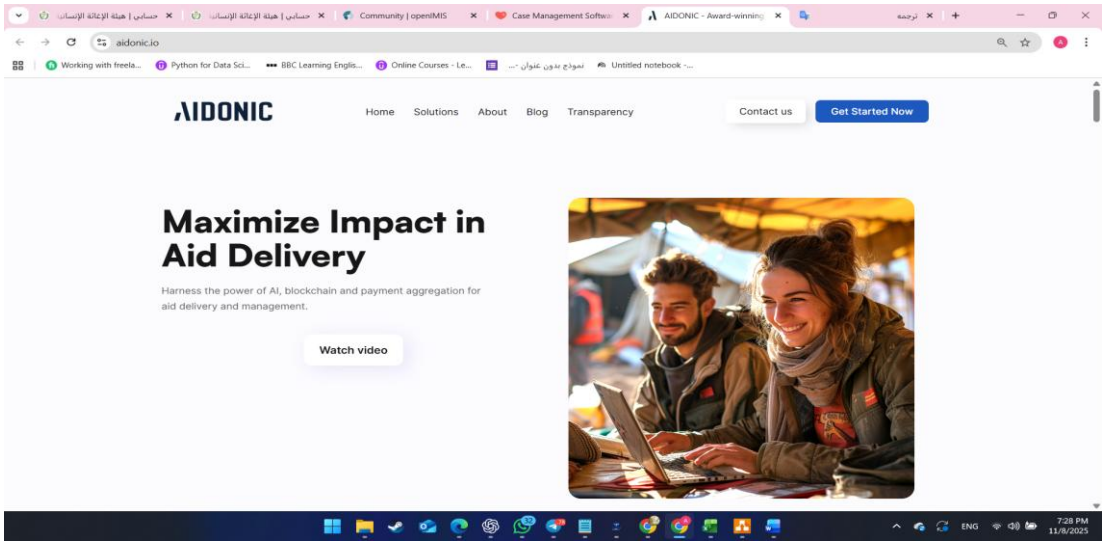

Figure 0-5 AIDONIC

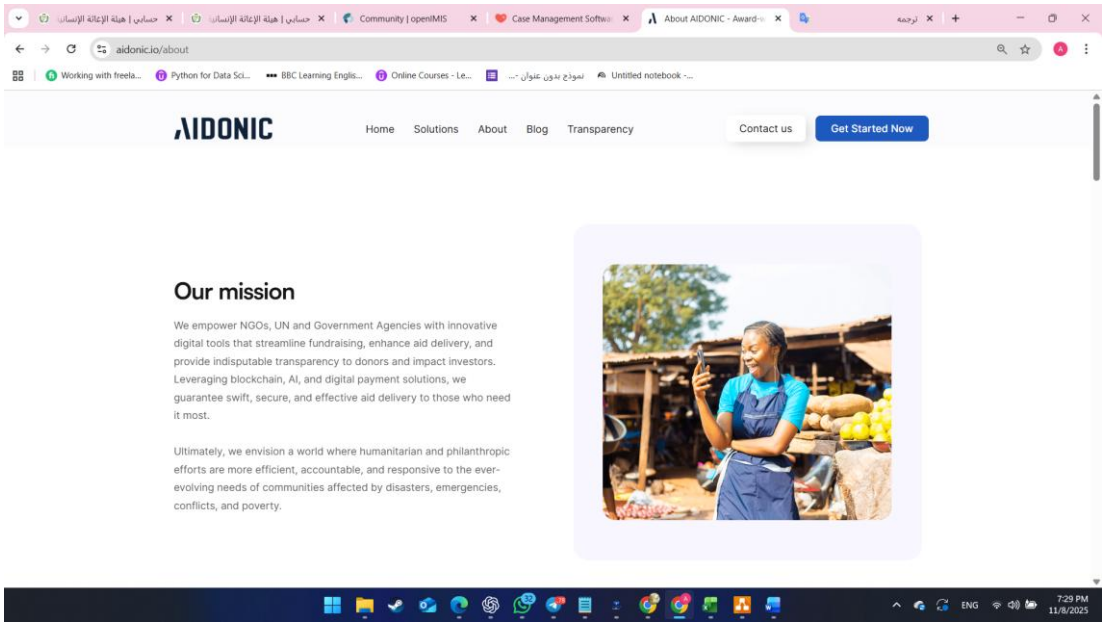

Figure 0-6 AIDONIC

LMMS (Last Mile Mobile Solutions): https://www.wvi.org/disaster-management/last-mile-mobile-solution-lmms

📝 A field-based system for registering beneficiaries and tracking parcel distribution down to the "last mile."

✅ Excellent for projects with field teams delivering aid directly [13].

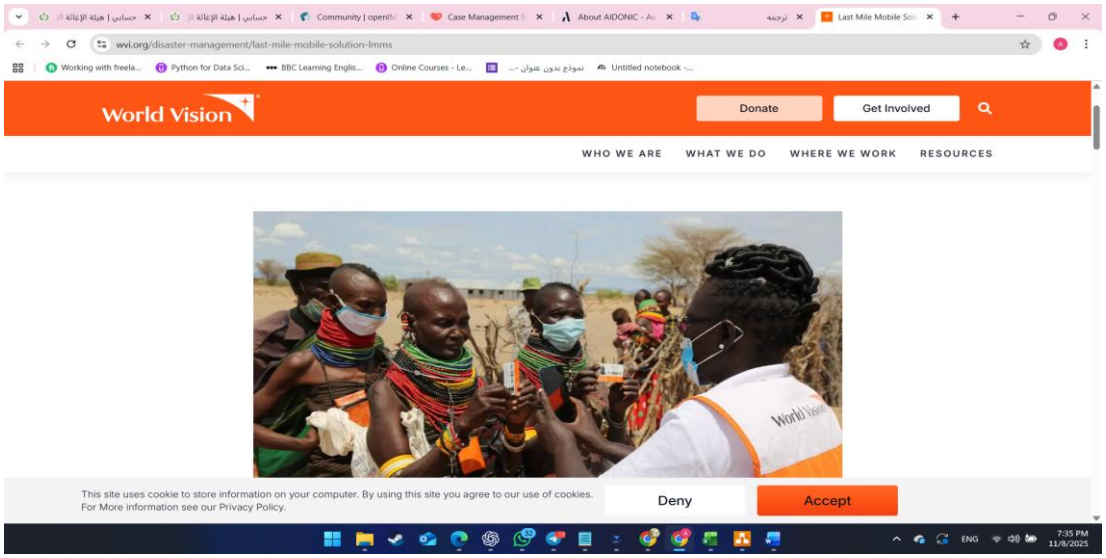

Figure 0-7 LMMS (Last Mile Mobile Solutions)

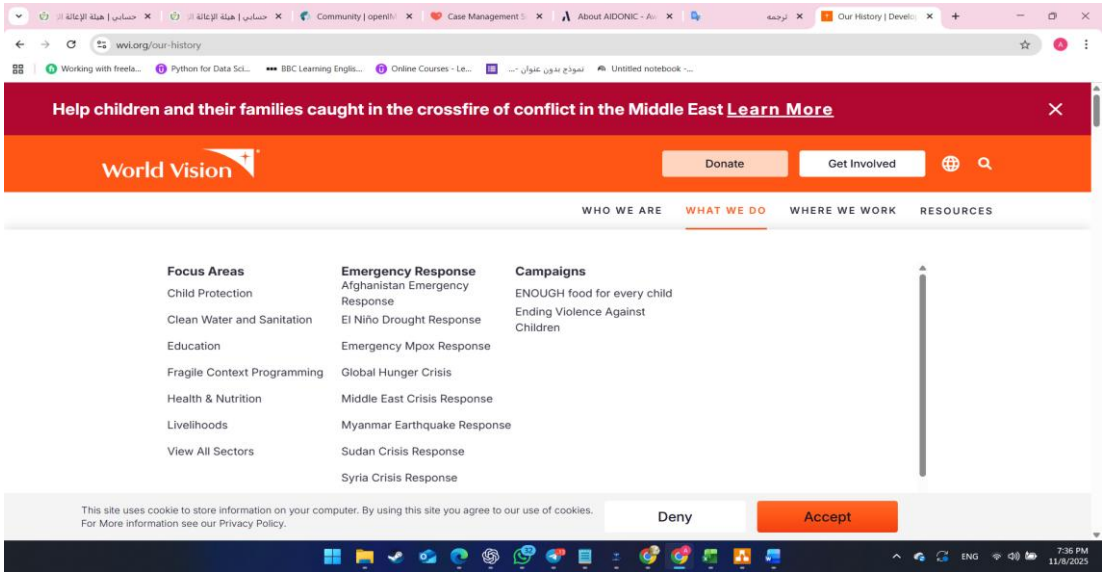

Figure 0-8 LMMS (Last Mile Mobile Solutions)

Chapter 3
Methodology

3.1 Introduction

In this chapter, we explain how we planned, developed, and tested our project, a web-based platform that enables organizations in the Gaza Strip to efficiently manage imports, exports, warehouse inventories, and beneficiary records. To manage our work step by step, we used the Agile methodology, specifically the Scrum approach [14]. This allowed us to remain organized, divide the work into short iterative phases (called sprints), and continuously improve the system based on feedback from stakeholders.

3.1.1 Chosen Methodology: Agile (Scrum)

We selected Agile Scrum because it allowed us to break the project into smaller, manageable parts and focus on one feature at a time [15]. This approach enabled early testing, quick modifications when necessary, and gradual delivery of a fully functional system.

The Scrum model we followed included:

- **Sprints:** Each sprint lasted approximately two weeks and focused on specific features, such as beneficiary registration, import/export tracking, or real-time reporting.

- **Daily Meetings:** We held regular short meetings to discuss progress, identify obstacles, and align tasks.

- **Sprint Planning:** Before each sprint, we defined the tasks to be implemented and assigned responsibilities.

- **Sprint Review & Feedback:** At the end of each sprint, we demonstrated the completed features to stakeholders and gathered feedback for improvement.

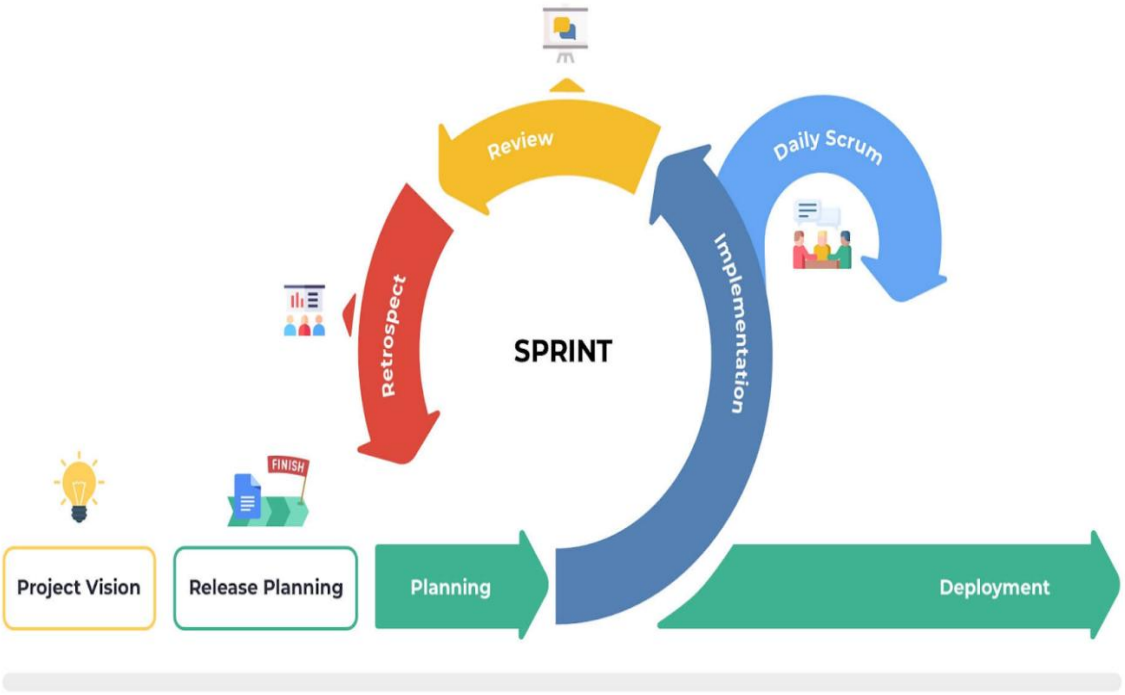

Figure 3.11- Agile (Scrum)

3.2 Tools and equipment

3.2.1 Tools

1. Laravel: is a free and open-source PHP-based web framework for building web applications [16]
We use it to write the backend logic of our website

2. PhpStorm: is an Integrated Development Environment (IDE) for PHP developers built to maximize developer productivity [17]. The IDE desktop application helps you write, edit, analyze, refactor, test, and debug PHP code on Windows, macOS, and Linux. We use it to write PHP code.

3. Visual Studio Code: is an integrated development environment developed by Microsoft for Windows, Linux, macOS and web browsers [18]. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded version control with Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality. [10] We use it to write frontend code.

4. Draw.io Desktop: is a stand-alone, offline application available for Windows, macOS and Linux [19]. This is ideal if you want the enhanced security of a desktop application with no reliance on an internet connection.
We use it to make Database diagrams.

5. MySQL: is an open-source relational database management system (RDBMS) [20]. We use it to manage our database.

6. **MySQL Workbench:** is an official graphical user interface (GUI) tool provided by **Oracle** for managing, designing, and developing **MySQL** databases [21]. It offers features such as SQL query writing and execution, entity-relationship (ERD) diagram design, user and privilege management, and database import/export. It simplifies working with databases through a visual interface instead of relying solely on the command line.

7. Microsoft Office: Microsoft Word 2021 / create project documentation.

8. Chatgpt AI model:

3.2.2 equipment

The equipment we use to build Gaza Aid
Management is:
Laptop:
1- HP NootBook NPNUTVJ
2- HP pa
3-
4-

3.3 Project Sprints and Tasks

The development process was divided into **five structured sprints** using the Scrum methodology. Each sprint focused on a specific set of tasks, deliverables, and objectives that contributed incrementally to the completion of the system.

3.3.1 Sprint 1 – Requirements & Analysis (2 weeks)

- **Objective**: Build a clear understanding of the system's needs and design its foundation.
- **Main Tasks**:
  - Conduct requirement gathering from target institutions.
  - Analyze current challenges in imports/exports and beneficiary management.
  - Design the **Entity Relationship Diagram (ERD)** for the database.
  - Prepare the **System Requirements Specification (SRS)** document.
- **Deliverables**: Requirements documentation and ERD diagram.

3.3.2 Sprint 2 – Beneficiary Module & Basic UI (3 weeks)

- **Objective**: Provide institutions with a secure and user-friendly way to register beneficiaries.
- **Main Tasks**:
  - Develop the beneficiary registration module (CRUD operations).
  - Implement data validation to minimize errors and duplication.
  - Build the initial **user interface** (login, dashboard, beneficiary forms).
  - Connect the module to the database.
- **Deliverables**: Functional beneficiary registration system with a basic user interface.

3.3.3 Sprint 3 – Imports/Exports Tracking (3 weeks)

- **Objective**: Enable institutions to accurately manage imports, exports, and warehouse inventories.
- **Main Tasks**:
  - Develop the imports and exports management module.
  - Link transaction records with warehouse inventories.
  - Implement search and filtering functions for imports/exports data.
  - Ensure secure storage and retrieval of records.
- **Deliverables**: Imports/exports tracking module fully integrated with the database.

3.3.4 Sprint 4 – Admin Dashboard & Reports (2 weeks)

Objective: Provide administrators with tools for monitoring and decision-making.

- **Main Tasks**:
  - Develop the administrative dashboard.
  - Implement real-time statistical reporting and visualizations (charts, summaries).
  - Add role-based access control for administrators and staff.
  - Integrate notifications for system activities.
- **Deliverables**: Functional admin dashboard with live reports and monitoring features.

3.3.5 Sprint 5 – Testing & Deployment (2 weeks)

- **Objective**: Ensure the system is reliable, bug, and ready for use.
- **Main Tasks**:
  - Perform **unit testing, integration testing, and user acceptance testing**.
  - Fix bugs and optimize performance.
  - Prepare user manual and final documentation.
  - Deploy the system for institutional use.
- **Deliverables**: Fully tested, optimized, and deployed web-based platform.

Table 3-1 Show the Sprints and Tasks

.Sprint No	Duration	Main Tasks Completed	Output/Deliverables
Sprint 1	weeks 2	Requirements gathering, system analysis, and initial database design	Requirements document, ER diagram
Sprint 2	weeks 3	Developing beneficiary registration module and basic user interface	Functional beneficiary registration form, basic UI layout
Sprint 3	weeks 3	Implementing imports/exports tracking module and integrating with the database	Imports/exports management functionality
Sprint 4	weeks 2	Building the admin dashboard with statistical reports	Admin dashboard, real-time reporting feature
Sprint 5	weeks 2	System testing, bug fixing, and preparing deployment	Tested and deployed web platform

Table 3.31- Sprints and Tasks

Figure 3:3shows Timeline Sprint

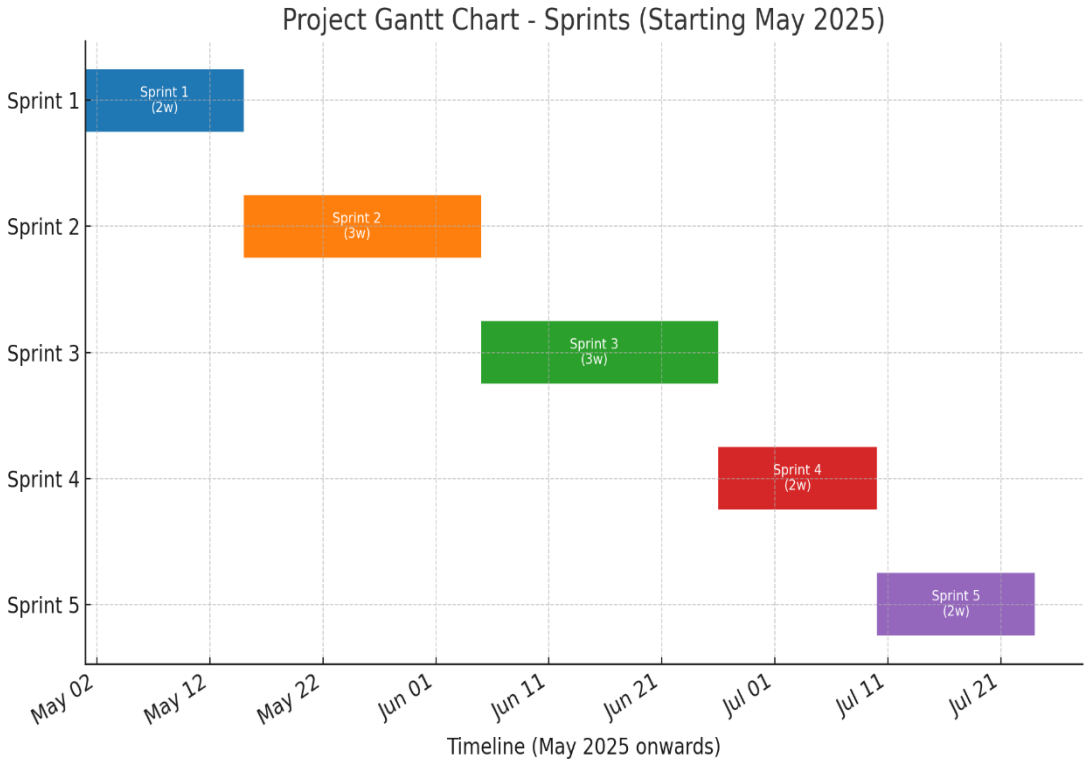

Figure 3.31- Timeline Sprint

Chapter 4
System Analysis

4.1 Introduction

System analysis is a critical step in the development process, as it defines the functional and non-functional requirements of the platform and provides a clear blueprint for its design and implementation. This chapter presents the requirements, use cases, and database model for the proposed web-based platform that supports organizations in the Gaza Strip in managing imports, exports, warehouse inventories, and beneficiary data.

4.2 System Requirements

4.2.1 Functional Requirements

The functional requirements describe what the system must do to meet the needs of its users:

1. Allow secure login and role-based access (Admin, Data Entry Officer, Viewer).
2. Enable beneficiary registration, update, search, and deletion.
3. Track and manage imports and exports records.
4. Maintain warehouse inventory linked to import/export transactions.
5. Generate real-time statistical reports and charts.
6. Provide advanced search and filtering features.
7. Support notification and alert system for administrators.
8. Enable administrators to manage user accounts and permissions.

4.2.2 Non-Functional Requirements

These requirements ensure the quality and performance of the system:

1. **Usability**: The interface should be intuitive and user-friendly.
2. **Performance**: System should handle concurrent data entry and reporting efficiently.
3. **Security**: All sensitive data must be encrypted, and access should be role-based.
4. **Scalability**: The system should support future enhancements such as mobile versions and multilingual support.
5. **Reliability**: Ensure minimal downtime and accurate data processing.
6. **Maintainability**: Code and database should be structured for easy updates.

4.3 Use Case Analysis

- **Primary Actors**: Administrator, Data Entry Officer, Beneficiary.
- **Main Use Cases**:
    - Login / Authentication.
    - Register Beneficiary.
    - Manage Imports/Exports.
    - Generate Reports.
    - Manage Users & Permissions.

The following table summarizes the key use cases of the proposed system:

No.	Use Case	Actor	Basic Flow	Alternative Flow
1	Login / Authentication	Administrator, Data Entry Officer, Viewer	User enters credentials → System verifies → Grants access based on role.	Invalid credentials → Show error; account locked → Notify admin.
2	Register Beneficiary	Data Entry Officer	Open form → Enter beneficiary details → Validate → Save record.	Duplicate detected → Prompt merge/update; Missing fields → Show validation errors.
3	Manage Imports/Exports	Administrator, Data Entry Officer	Create shipment record → Link to warehouse → Update status → Save.	Invalid data/IDs → Show error; Insufficient permissions → Access denied.
4	Warehouse Inventory	Administrator, Data Entry Officer	View inventory → Adjust quantities based on transactions → Save updates.	Negative stock → Reject update; Concurrency conflict → Retry/lock record.
5	Search & Filtering	All roles	Enter criteria → System filters across modules → Display results.	No results found → Suggest broader filters.
6	Generate Reports	Administrator, Viewer	Select time range/criteria → Generate charts/tables → Export PDF/XLS.	Large dataset → Paginate/async export; No permission → Access denied.
7	Manage Users & Roles	Administrator	Create user → Assign role/permissions → Activate/Deactivate.	Duplicate email → Prompt update; Role conflict → Validation warning.
8	Notifications	All roles	Trigger event (new aid, low stock) → System sends alert (in-app/SMS/email).	Invalid contact data → Log error; User muted channel → Skip notify.

Table 4.31-Use Case in our project

Table 4.3-2 Use Case 1: Login / Authentication

Use Case ID	**UC-01**
Use Case Name	Login / Authentication
Actors	Administrator, Data Entry Officer, Viewer
Description	This use case allows users to log in to the system using their credentials. The system verifies the provided data and grants access according to the user's role.
Preconditions	The user must have a valid account in the system.
Main Flow	1. User enters username and password. 2. System verifies credentials. 3. System grants access based on role privileges.
Alternative Flow	- Invalid credentials → Display error message.- Account locked → Notify administrator.
Postconditions	User successfully logged into the system with proper role permissions.

Table 4.3-3 Use Case 2: Register Beneficiary

Use Case ID	**UC-02**
Use Case Name	**Register Beneficiary**
Actors	**Data Entry Officer**
Description	**This use case allows the data entry officer to register new beneficiaries and manage their records within the system.**
Preconditions	**The user must be logged in and have permission to add beneficiaries.**
Main Flow	**1. User opens the beneficiary registration form. 2. Enters beneficiary details. 3. Validates input data.4. Saves the record successfully.**
Alternative Flow	**- Duplicate entry detected → Prompt to merge/update. - Missing required fields → Show validation errors.**
Postconditions	**New beneficiary record is stored in the system database.**

Table 4.3-4 Use Case 3: Manage Imports/Exports

Use Case ID	**UC-03**
Use Case Name	Manage Imports/Exports
Actors	Administrator, Data Entry Officer
Description	Enables users to log, review, and manage import and export operations related to aid and warehouse transactions.
Preconditions	User must be authenticated and have sufficient privileges.
Main Flow	1. Create shipment record. 2. Link to the corresponding warehouse. 3. Update shipment status. 4. Save transaction.
Alternative Flow	- Invalid data or IDs → Show error. - Insufficient permissions → Access denied.
Postconditions	Import/export record successfully created and linked with warehouse data.

Table 4.3-5 Use Case 4: Warehouse Inventory

Use Case ID	**UC-04**
Use Case Name	Warehouse Inventory
Actors	Administrator, Data Entry Officer
Description	This use case handles the viewing and adjustment of inventory levels in the warehouse based on incoming and outgoing stock movements.
Preconditions	User must be authorized to access warehouse data.
Main Flow	1. User views current inventory list. 2. Adjusts quantities based on transactions. 3. Saves updated values.
Alternative Flow	- Negative stock → Reject update. - Concurrency conflict → Retry or lock record.
Postconditions	Inventory data updated and saved accurately.

Table 4.3-6 Use Case 5: Search & Filtering

Use Case ID	**UC-05**
Use Case Name	Search & Filtering
Actors	All Roles
Description	Allows users to search and filter data across modules to quickly retrieve required records.
Preconditions	User must be logged in.
Main Flow	1. Enter search criteria. 2. System filters records across modules. 3. Display relevant results.
Alternative Flow	- No results found → Suggest broader filters.
Postconditions	Filtered data displayed according to user's query.

Table 4.3-7 Use Case 6: Generate Reports

Use Case ID	**UC-06**
Use Case Name	Generate Reports
Actors	Administrator, Viewer
Description	Enables users to generate reports with charts and tables, filtered by time or type, and export them as PDF or Excel files.
Preconditions	User must have reporting privileges.
Main Flow	1. Select time range and criteria. 2. System generates chart/table report. 3. Export as PDF/Excel.
Alternative Flow	- Large dataset → Use pagination or asynchronous export. - No permission → Access denied.
Postconditions	Reports generated successfully and available for export.

Table 4.3-8 Use Case 7: Manage Users & Roles

Use Case ID	**UC-07**
Use Case Name	Manage Users & Roles
Actors	Administrator
Description	Allows administrators to create, update, and deactivate user accounts, as well as assign system roles and permissions.
Preconditions	Administrator must be logged in.
Main Flow	1. Create new user account. 2. Assign role and permissions. 3. Activate or deactivate as needed.
Alternative Flow	- Duplicate email → Prompt update. - Role conflict → Show validation warning.
Postconditions	User account updated successfully in the system.

Table 4.3-9 Use Case 8: Notifications

Use Case ID	**UC-08**
Use Case Name	**Notifications**
Actors	**All Roles**
Description	**Sends notifications or alerts to users when specific events occur, such as new aid arrival or low stock levels.**
Preconditions	**System is running and users are registered for notifications.**
Main Flow	**1. Trigger event (new aid, low stock). 2. System sends alert via in-app message, SMS, or email.**
Alternative Flow	**- Invalid contact data → Log error. - User muted channel → Skip notification.**
Postconditions	**Notifications delivered successfully to valid users.**

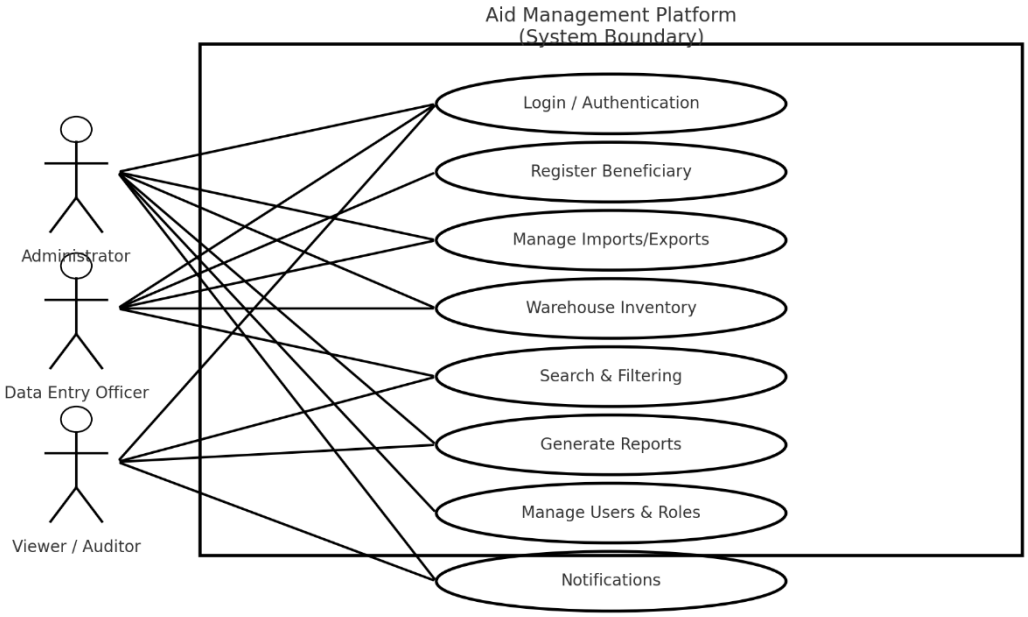

Figure 4.3-1 Use Case in our project

4.4 Database Design (ERD)

4.4.1 General Description

The Entity-Relationship Diagram (ERD) represents the logical structure of the database used in the proposed system. It illustrates how users, beneficiaries, warehouses, stock movements, deliveries, and permissions are interconnected. The design ensures efficient management of aid distribution operations, accurate inventory tracking, and flexible role-based access control.

4.4.2 Main Modules

1. **User Management (Users / Sessions)**
    - *Users*: Stores essential account information such as id, id_no, password, type_user, and name.
    - *Sessions*: Maintains login/logout session data linked to user_id, including IP address and last activity.
2. **Coupons and Beneficiaries**
    - *Type_Coupons*: Defines types of aid coupons.
    - *Beneficiaries*: Records details of beneficiaries (e.g., due_date, receipt_date) and links them to coupon types.
3. **Warehouses and Inventory Management**
    - *Warehouses*: Stores inventory quantities per coupon type.
    - *Stock_Movements*: Logs each inventory transaction (inbound/outbound).
    - *Stock_Deliveries*: Tracks the quantities delivered from stock movements.
4. **Clothing Packages**
    - *Clothing_Packages*: Manages distribution of clothing aid packages, linked with distribution_place_id and project_id (from Constants).
5. **Constants**
    - *Constants*: A hierarchical table (via parent_id) for storing reusable classification values such as distribution places or project types.
6. **Roles and Permissions**
    - *Role_Pages*: Defines the accessible system pages (name + URL).
    - *Role_Page_User*: Connects users to authorized pages.
    - *Role_Btns*: Defines available actions (buttons) on each page.
    - *Role_Btn_User*: Connects users to permitted actions at button level.
7. **Job and Queue System**
    - *Jobs, Job_Batches, Failed_Jobs*: Implements Laravel's queue system for asynchronous tasks such as background imports, notifications, and report generation.

4.4.3 Key Relationships

1. **Users → Sessions / Beneficiaries / Clothing_Packages**: A user can have multiple sessions, update beneficiaries, and manage clothing packages.
2. **Type_Coupons → Beneficiaries / Warehouses / Stock_Movements**: Coupon types are the reference entity for beneficiaries and inventory.
3. **Warehouses ↔ Stock_Movements ↔ Stock_Deliveries**: Warehouses hold stock; movements track in/out operations; deliveries record distributed quantities.
4. **Constants**: Acts as a flexible classification hierarchy across different modules.
5. **Role Management**: Permissions are implemented through mapping (pivot) tables role_page_user and role_btn_user.

4.4.4 Database Analysis

Strengths

1. Clear separation between users, inventory, beneficiaries, and permissions.
2. The **Constants** table provides flexibility for dynamic values (e.g., projects, locations).
3. Tight connection between stock movements and deliveries ensures accurate reporting.
4. Fine-grained access control with page- and button-level permissions.

Points for Improvement

1. The *Users* table lacks basic fields such as email and phone for communication.
2. The *Beneficiaries* table could include a direct foreign key to warehouses or stock movements.
3. *Clothing_Packages* links beneficiaries only by id_num; using a beneficiary_id FK would improve integrity.
4. The permission system is functional but complex; a simplified roles/permissions structure may be more efficient.
5. The *Constants* table may become overloaded if too many classifications are stored within it.

4.4.5 Practical Usage

1. Suitable for NGOs and institutions that manage aid distribution (warehouses, coupons, clothing).
2. Supports accurate real-time inventory management (inbound/outbound + deliveries).
3. Provides flexible permissions tailored to organizational hierarchies.
4. Scalable design ready for future enhancements (mobile application, multilingual support).

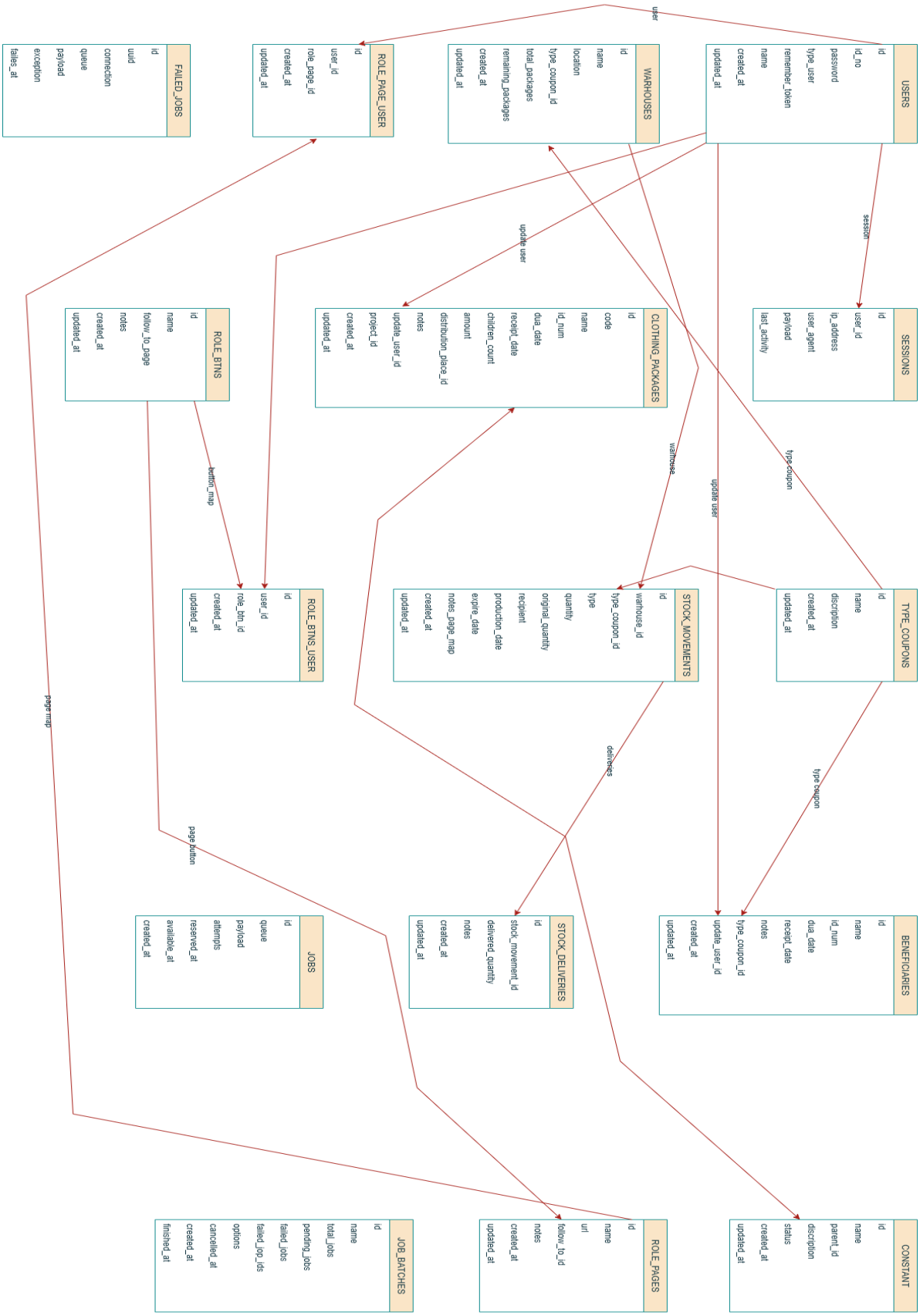

Figure 4.41- UMLDigrame

Chapter 5
Implementation

This chapter explains how the proposed system was implemented, covering the environment setup, development stages, and integration of different modules. The implementation followed the Agile methodology using Laravel for the backend and HTML, CSS, and JavaScript for the frontend.

5.1 Development Environment

- Backend Framework: Laravel (PHP) – chosen for its MVC structure, security features, and scalability.
- Frontend Tools: HTML, CSS, JavaScript, and Bootstrap 5 – for building responsive and user-friendly interfaces.
- Database: MySQL – used for structured storage and relational queries.
- IDE Tools: PhpStorm and Visual Studio Code – for backend and frontend coding respectively.
- Version Control: Git & GitHub – for collaborative code management and tracking changes.
- Design Tools: Draw.io for ERD and system diagrams, Figma for UI prototyping.

5.2 Backend Implementation

The backend of the proposed system was developed using the **Laravel framework**, a PHP-based platform that follows the **Model–View–Controller (MVC)** architecture. This design pattern separates the logic, data handling, and user interface, making the system modular, secure, and easy to maintain. The backend handles all data processing, authentication, role management, and communication between the database and the frontend.

---

1. Database Connection

The system's database was implemented using **MySQL**, integrated with Laravel through the environment configuration file (.env).

The database structure follows the **Entity Relationship Diagram (ERD)** designed earlier in Chapter 4, ensuring logical relationships among the main entities such as *Users, Beneficiaries, Warehouses, Projects, Imports/Exports,* and *Reports*.

Laravel's **migration feature** was used to automatically create and update database tables. Each migration defines a table structure, including fields, data types, and relationships (foreign keys). This ensures consistent deployment across development environments.

In addition, Laravel **Eloquent ORM** (Object Relational Mapping) was used to interact with database tables through PHP classes instead of SQL queries. This provides cleaner and more secure data handling.

---

2. Authentication & Authorization

User authentication was implemented using **Laravel Breeze**, which provides a pre-built and secure login and registration system.

- The authentication process uses encrypted passwords (via Laravel's built-in Hashing library) and session-based user management.
- The system defines multiple roles:
    - **Administrator** – Full control over all modules (users, beneficiaries, projects, and reports).
    - **Data Entry Officer** – Limited access, mainly for adding and updating records.
    - **Viewer** – Can view reports and statistics without modification privileges.

Authorization is enforced through **Middleware** that restricts access based on user roles. Each route or controller action is protected to ensure that unauthorized users cannot perform restricted operations.

---

3. Beneficiary Management Module

The beneficiary module forms the core of the system, allowing institutions to manage their beneficiaries effectively.

- CRUD operations (Create, Read, Update, Delete) were developed using Eloquent Models and Controllers.
- Each beneficiary record includes personal details, identification number, contact information, and service eligibility data.
- Validation rules were implemented to prevent duplicate entries (using unique constraints) and ensure data completeness.
- The module also supports **real-time search and filtering** based on various criteria (e.g., name, ID number, project type).
- Data pagination was applied using Laravel's paginate() function to enhance performance when displaying large datasets.

All operations are logged for audit purposes, allowing administrators to track updates and data changes.

---

4. Import/Export Module

This module allows institutions to record and manage import and export operations related to aid shipments and inventory flow.

- Staff members can log new **import** (incoming aid) or **export** (distributed aid) transactions by specifying project type, quantity, date, and warehouse location.
- Each record is automatically linked to a warehouse, ensuring real-time updates to inventory quantities.
- The **Admin** role can review, approve, or edit these records.
- The module interacts directly with the **Warehouse Management System**, ensuring consistency between imported/exported quantities and warehouse stock levels.
- Additionally, the module is integrated with the **Reporting system**, allowing statistical summaries of import/export trends over time.

---

5. Reporting Module

The reporting module provides analytical tools and data visualization features for administrators and decision-makers.

- Reports are generated dynamically using the **Laravel Charts** package, which displays visual statistics such as aid distribution per month, warehouse utilization, and beneficiary coverage.
- Users can export reports in multiple formats including **PDF** and **Excel**.
- The system supports filtering by time period, project type, warehouse, and service status.
- Laravel's query builder was optimized to handle large datasets efficiently, while ensuring quick response times.

The reporting system contributes significantly to transparency and decision-making by providing accurate real-time data on institutional operations.

5.3 Frontend Implementation

The frontend of the proposed system was designed to provide a **responsive, interactive, and user-friendly** experience for administrators and staff members. It was developed using **HTML5**, **CSS3**, **JavaScript**, and the **Metronic Bootstrap 5 Admin Template**, which served as the main foundation for the system's structure and design components.

---

1. Framework and Template Customization

The user interface was built on top of the **Metronic Bootstrap 5 Admin Template**, a professional and modern frontend framework known for its modularity, flexibility, and clean design.

While the template offered a strong starting point, extensive customization was applied to align it with the project's specific requirements:

- The **color palette**, **fonts**, and **layouts** were modified to match the project's branding and improve visual clarity.
- Unnecessary elements and demo components were removed to reduce complexity and optimize page load performance.
- Custom pages were developed from scratch, including **beneficiary management**, **project administration**, **warehouse management**, and **import/export tracking** interfaces.
  These modifications resulted in a tailored and efficient interface that fits the functional needs of the system while maintaining a professional appearance.

---

2. Core Frontend Modules

The frontend system is divided into several main modules, each providing a clear and focused user experience:

1. **Dashboard Module:**
   Displays general system information such as total beneficiaries, warehouse statuses, and recent import/export records. It serves as the control center for administrators to quickly access key data.
2. **Beneficiary Management Interface:**
   Provides responsive and user-friendly forms for adding, updating, and searching beneficiary data. The forms include validation to prevent missing or duplicated records and ensure accurate data entry.
3. **Import/Export Management Interface:**
   Allows staff to log import and export operations, specifying quantities,

warehouses, and responsible users. Admins can view, approve, or edit these entries directly from the same interface.

4. **Warehouse Management Pages:**
   Present detailed lists of warehouse stock levels and remaining quantities. When stock levels drop below a predefined threshold, the interface displays visual alerts to notify administrators.
5. **Project and User Management:**
   Includes pages that allow administrators to create projects, assign users, and manage staff permissions efficiently. Each action dynamically interacts with the backend via asynchronous requests.

---

3. JavaScript and Interactivity Enhancements

To improve user experience and interactivity, **JavaScript** was used extensively across the frontend:

- **AJAX (Asynchronous JavaScript and XML)** enables smooth data communication between the frontend and backend without page reloads. This allows operations such as data search, filtering, and record updates to happen instantly.
- The **DataTables** library was integrated to create interactive tables supporting **sorting, filtering, pagination, and export options (Excel/PDF)**, ensuring efficient navigation through large datasets.
- Input validation and real-time feedback were implemented to reduce user errors and enhance usability.
- The interface dynamically updates displayed data upon completing operations such as record creation or editing, without requiring page refresh.

---

4. Responsiveness and Accessibility

The frontend follows responsive design principles to ensure usability across various screen sizes and devices:

- The layout automatically adapts to desktops, tablets, and smartphones using Bootstrap's grid system.
- Elements such as buttons, tables, and forms are optimized for both mouse and touch interactions.
- Consistent color contrast and font sizes improve readability and accessibility for all users.

---

5. Integration with Backend

The frontend communicates with the backend through **RESTful API endpoints** developed in Laravel.

Each action performed by the user — such as registering a beneficiary, managing

warehouse data, or logging import/export records — sends asynchronous AJAX requests to the server for processing and validation.
This architecture ensures data consistency, system security, and smooth synchronization between the user interface and the backend database.

---

6. Notifications and User Feedback
A **notification system** was implemented using Laravel's built-in notification mechanism combined with JavaScript alerts.
Notifications are displayed within the interface whenever significant events occur, such as:

- Successful creation or update of a record.
- Low warehouse stock alerts.
- Errors in data entry or authorization.

These instant feedback messages help users stay informed, reduce mistakes, and ensure a smooth workflow.

Summary
The frontend implementation delivers a clean, organized, and fully responsive interface that connects seamlessly with the backend system.
By combining **Bootstrap 5**, **Metronic Template**, **AJAX**, and **DataTables**, the system provides an efficient environment for managing beneficiaries, warehouses, and aid distribution operations with high usability and strong performance.

AJAX was implemented on this page to enhance interactivity and improve user experience by enabling asynchronous data updates without reloading the entire page:

Figure 5.3-1 AJAX

```
function getBeneficiaries() {
    var s_code  = $('#s_code').val();
    var s_id_no = $('#s_id_no').val();
    var s_name  = $('#s_name').val();
    var s_place = $('#s_place').val();

    var url = "{{ route('clothing.result_search') }}";

    $.ajax({
        url: url,
        type: 'GET',
        data: {
            s_code:  s_code,
            s_id_no: s_id_no,
            s_name:  s_name,
            s_place: s_place
        },
        success: function (response) {
            $('#resultDiv').html(response);
        },
        error: function () {
            alert('حدث خطأ أثناء جلب البيانات');
        }
    });
}
```

Figure 5.3-2 Code of AJAX

```
function searchReport() {
    var place = $('#place').val();
    var due_date = $('#due_date').val();
    var project_id = $('#project_id').val();

    $.ajax({
        url: "{{ route('clothing.report_ajax') }}",
        type: 'GET',
        data: {
            place: place,
            due_date: due_date,
            project_id: project_id
        },
        success: function (data) {
            $('#resultDiv').html(data);
        },
        error: function () {
            alert('حدث خطأ أثناء جلب البيانات');
        }
    });
}
```

Figure 5.3-3 Code of AJAX

A DataTable was implemented on this page to display and manage tabular data efficiently, providing features such as sorting, searching, and pagination:

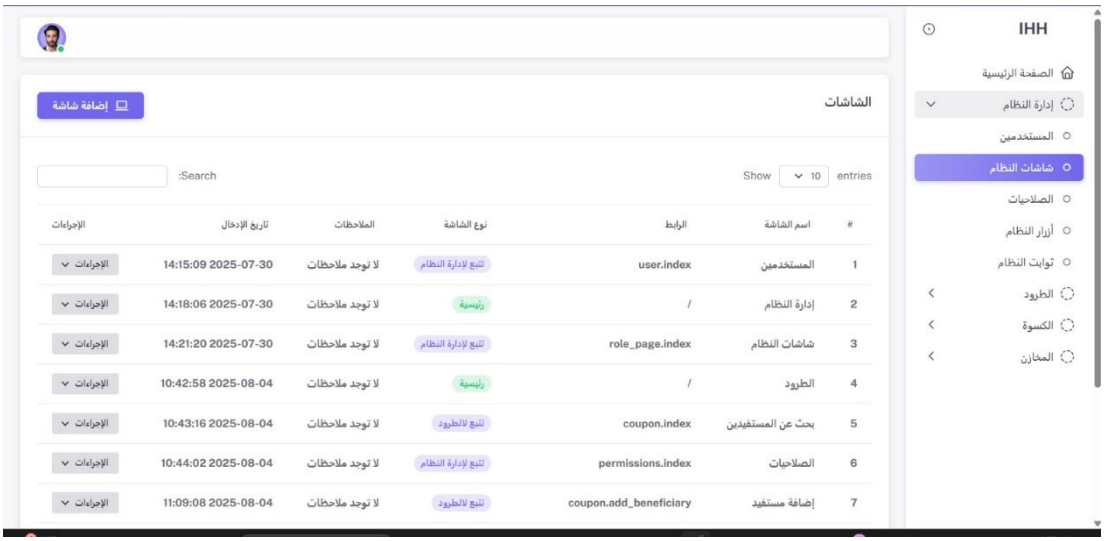

Figure 5.3-4 DataTable

5.4 Admin Dashboard

The admin dashboard serves as the control center of the system. It provides:

- Summary Cards: Total beneficiaries, total imports, total exports, and active services.
- Charts: Visualization of monthly aid distribution and import/export trends.
- User Management: Add/edit/remove system users and assign roles.

5.5 Testing

We applied multiple testing approaches to ensure system quality:

- **Unit Testing:** Tested each function (e.g., beneficiary creation, report generation) individually.
- **Integration Testing:** Verified interactions between modules (e.g., beneficiary records linked with reports).
- **User Acceptance Testing (UAT):** Shared the prototype with institutional staff for real-world feedback.
- **Bug Fixing & Refinement:** Applied necessary fixes to improve performance and usability.

5.6 Deployment

- The system was deployed on a **local Apache server** during development.
- Final deployment prepared for **shared hosting** or **institutional intranet server**.
- Database hosted on MySQL server, with daily backup scripts.

This is some pages from the system:

Figure 5.6-1 SomePages

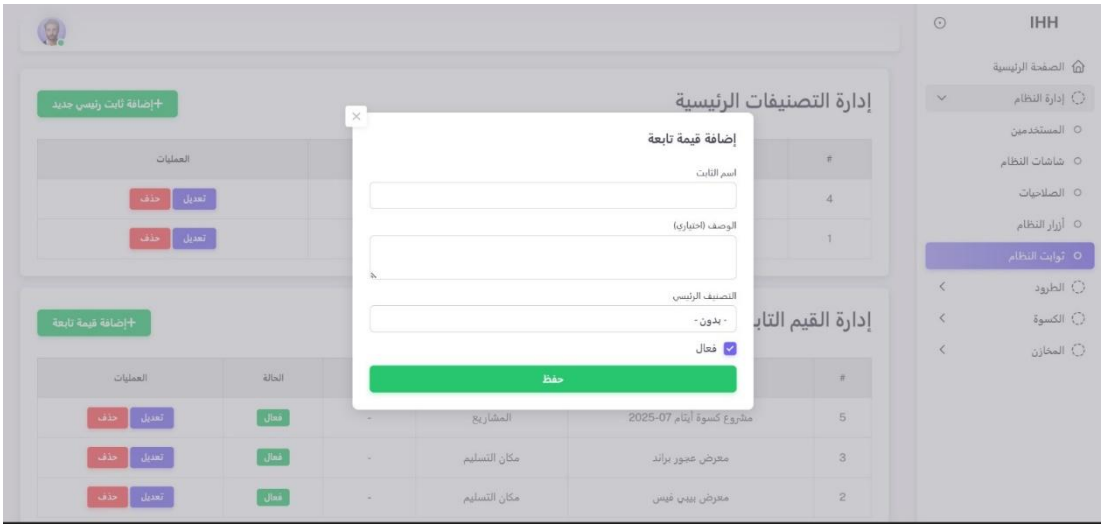

Figure 5.6-2 SomePages

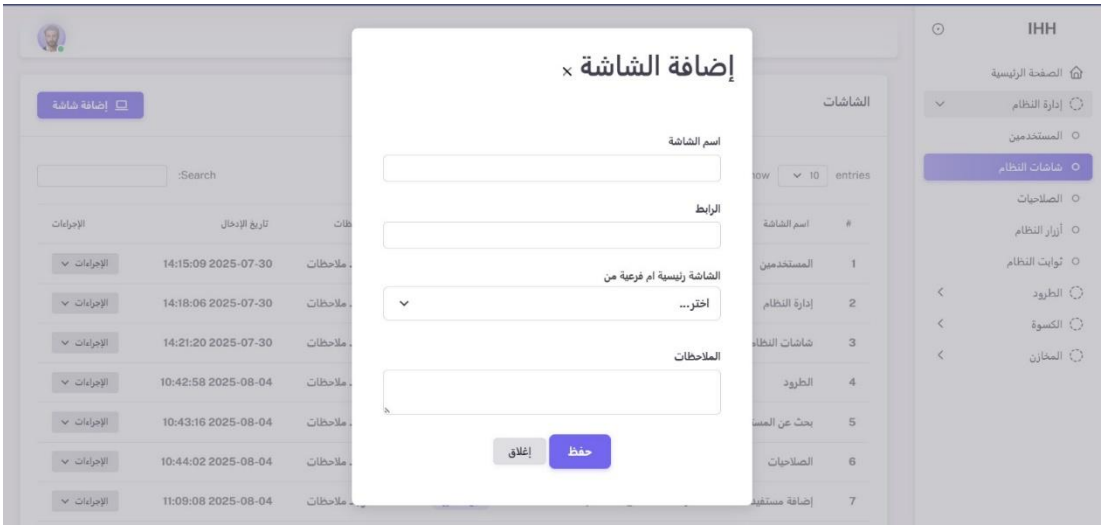

Figure 5.6-3 SomePages

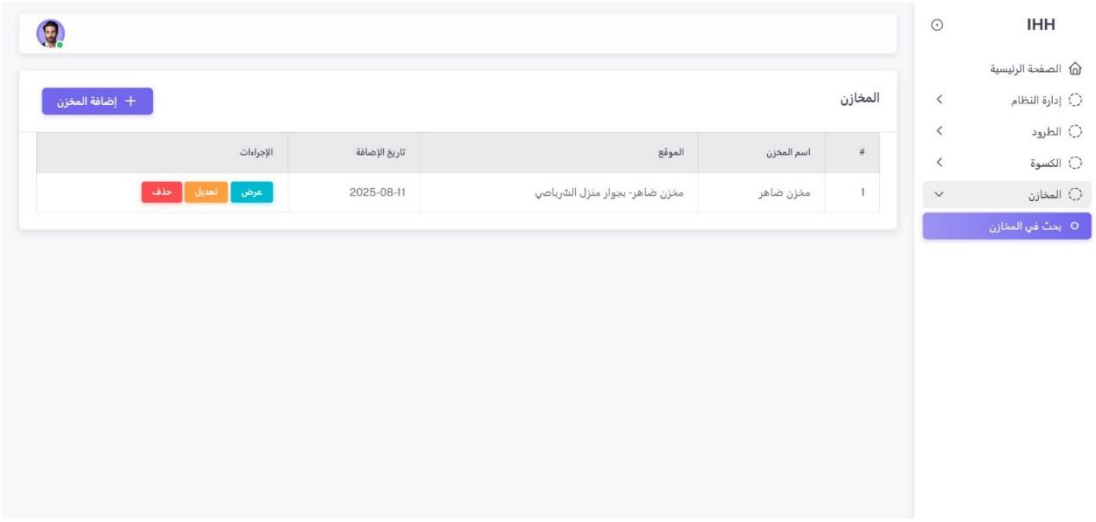

Figure 5.6-4 SomePages

Figure 5.6-5 SomePages

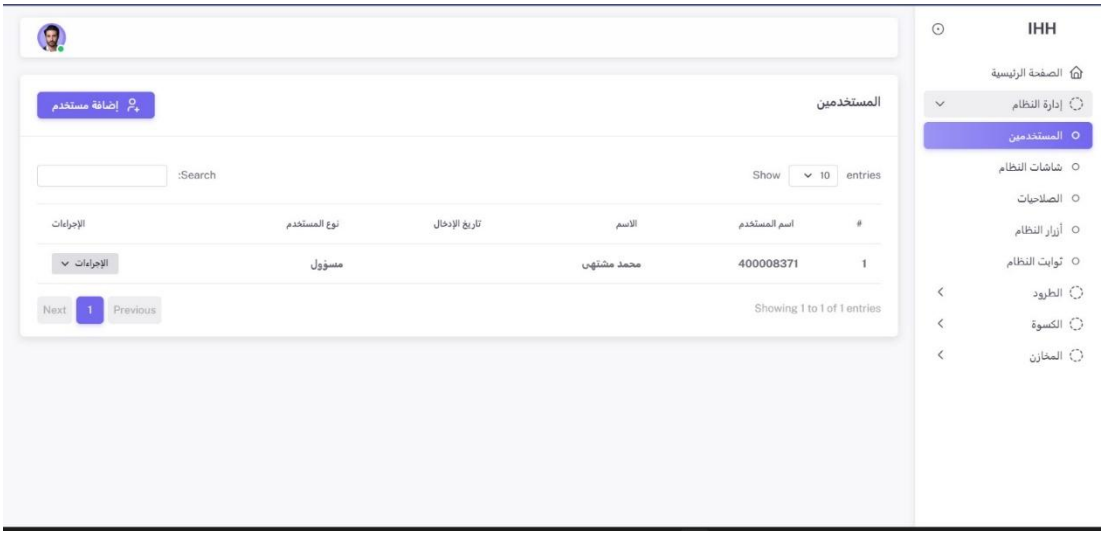

Figure 5.6-6 SomePages

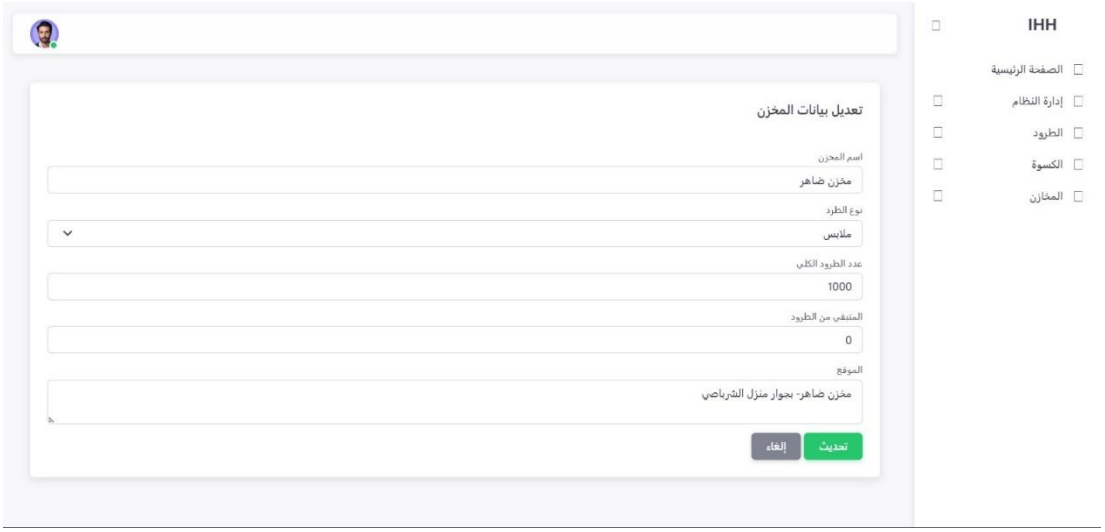

Figure 5.6-7 SomePages

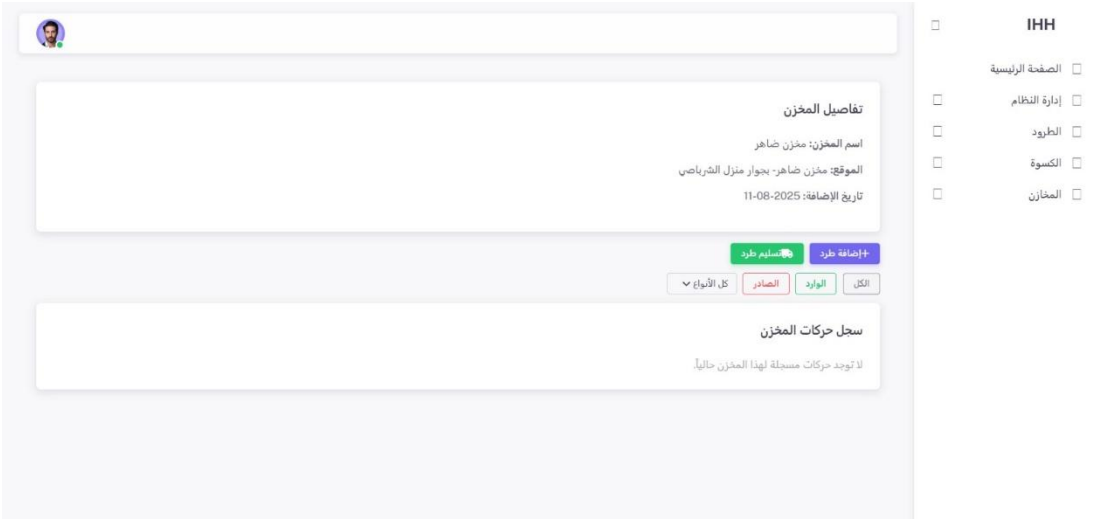

Figure 5.6-8 SomePages

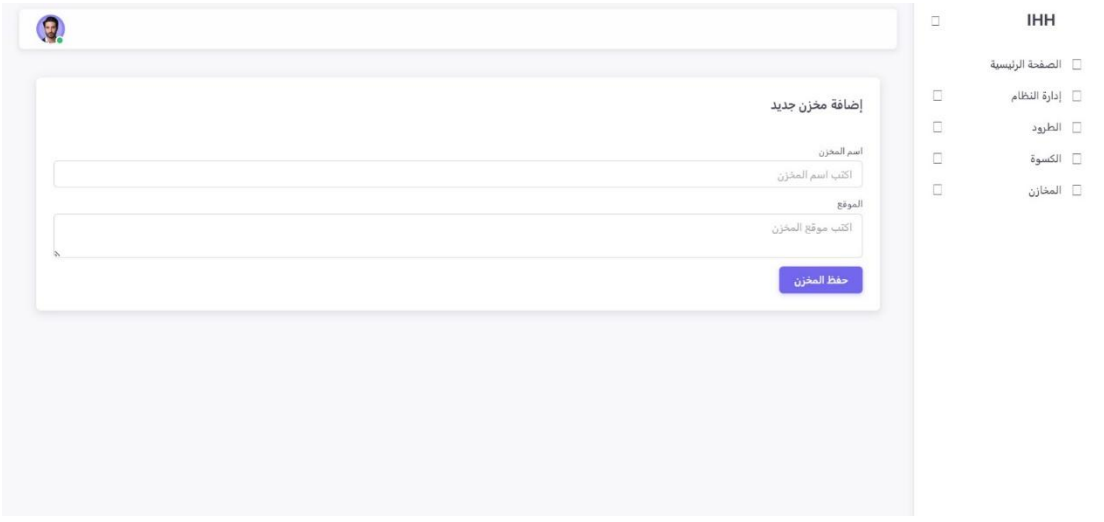

Figure 5.6-9 SomePages

Attached is a link to the project on Git Hup:
https://github.com/mohammedjr9/bakery-app [22]

Website link: https://ihh-gaza.com/ [23]

Chapter 6
Testing

Software testing is an essential phase to ensure that the developed system meets the requirements and performs as expected. The purpose of testing is to detect errors, verify that modules interact correctly, and confirm that the system is user-friendly and reliable for institutional use.

6.1 Testing Objectives

1. **Verify functional requirements are implemented correctly.**
2. **Ensure the system is free of major bugs.**
3. **Validate that administrators and staff can use the system effectively.**
4. **Evaluate performance under different conditions.**

6.2 Types of Testing Performed

6.2.1 Unit Testing:

Unit testing was applied to check individual components of the system, such as beneficiary registration, login authentication, and report generation. Each function was tested with valid and invalid inputs to confirm correct handling.

Example:

- Input: Registering a beneficiary with missing required fields.
- Expected Output: Error message prompting user to complete mandatory data.
- Result: System rejected incomplete input and displayed correct error message.

6.2.2 System Testing

Integration testing focused on verifying how modules interact. For instance, when a new beneficiary is added, their information should automatically be available in the reporting module and accessible by administrators.

Example:

- Action: Register new beneficiary → Generate a monthly report.
- Expected Output: The new record appears in the report.
- Result: Integration confirmed successfully.

6.2.3 System Testing

System testing validated the platform as a whole. This included checking:

- Proper navigation between all modules.
- Data consistency between beneficiaries, import/export logs, and reports.
- Responsiveness across different devices (desktop and mobile).

6.2.4 User Acceptance Testing (UAT)

Institutional staff tested the system in real-world scenarios to ensure usability. Feedback was collected and improvements applied (such as enhancing the beneficiary search filters and simplifying the import/export entry form)

6.2.5 Performance Testing

Stress tests were conducted by uploading multiple records (beneficiaries and logs) to evaluate how the system performs under heavy data loads. The platform was able to handle thousands of records without noticeable slowdown.

6.2.6 Images of some code testing processes

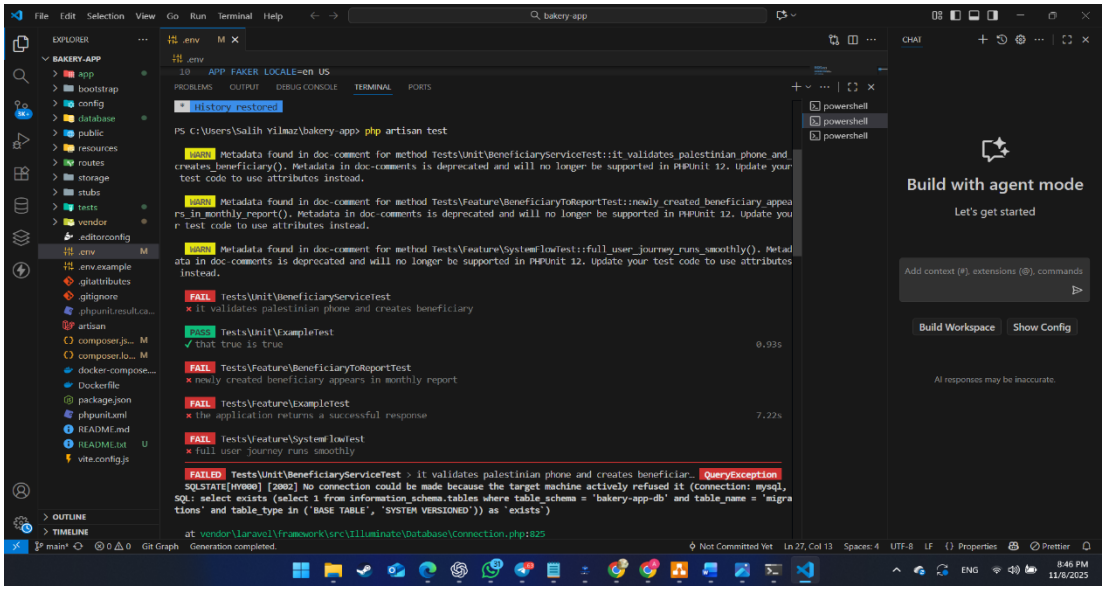

Figure 6.2-1 Fail Test

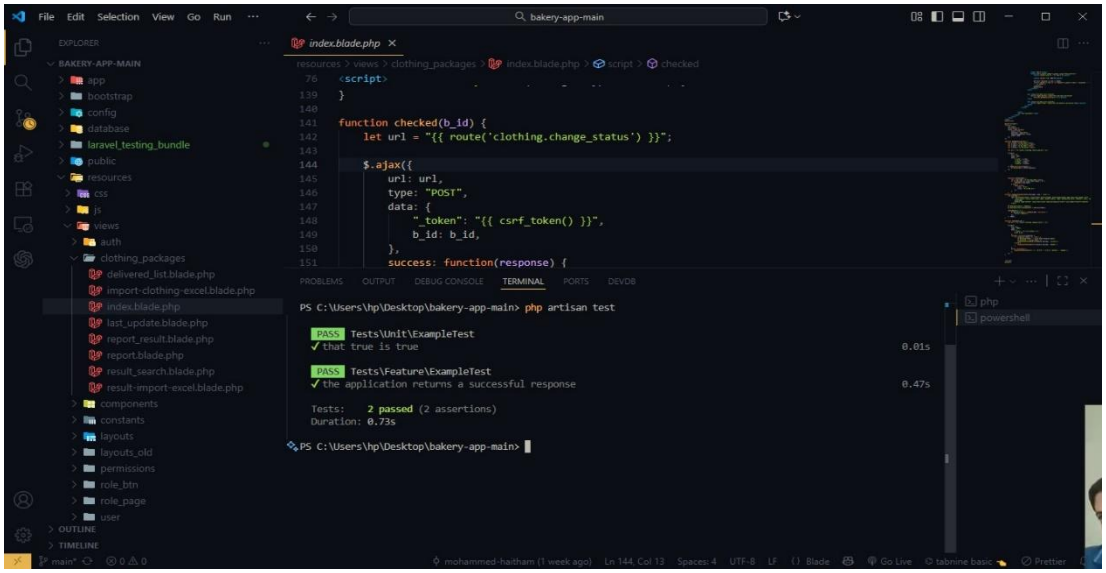

Figure 6.22- Pass Test

Dusk (Browser UI Testing):

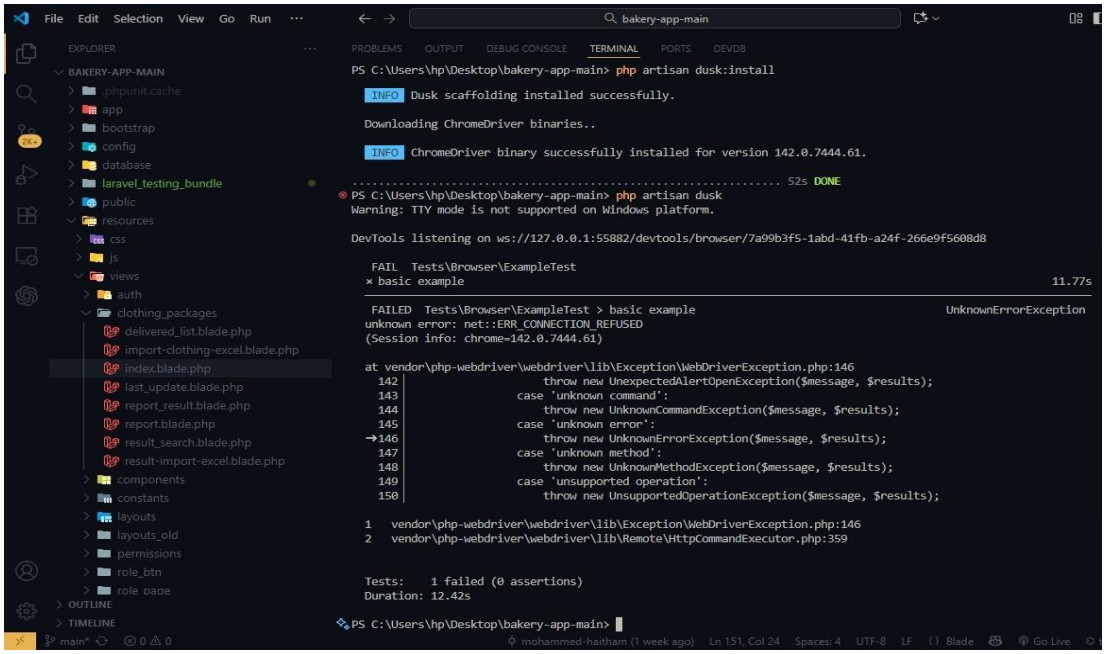

Figure 6.2-3 Fail Dusk Test

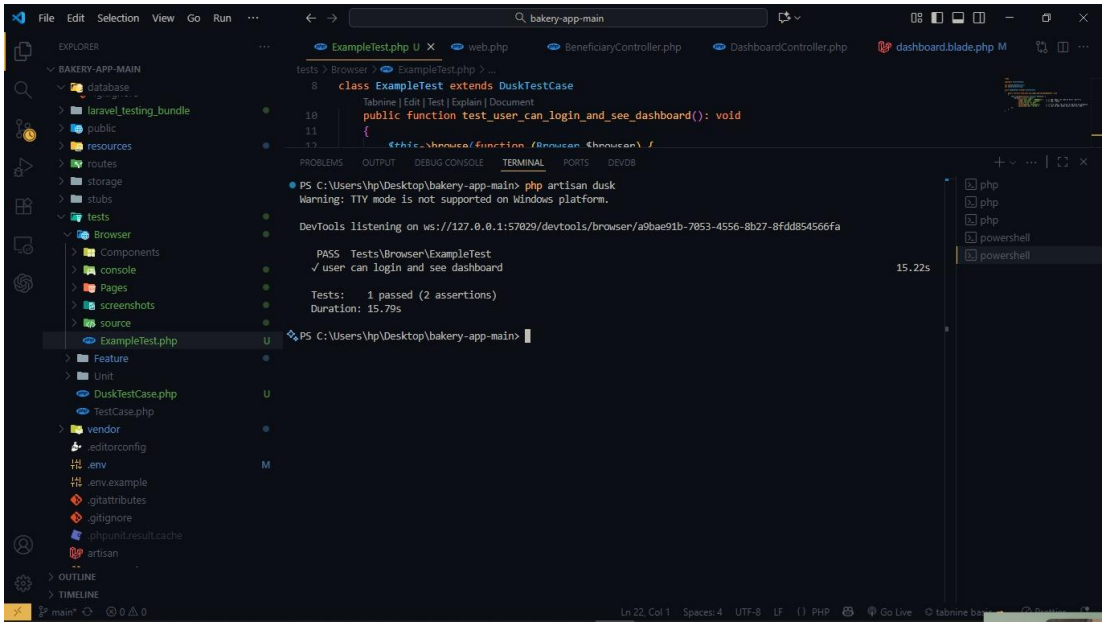

Figure 6.2-4 Pass Dusk Test

Benchmarking (Performance test):

This runs a custom performance test that often inputs experimental data (e.g., 5000 records) to test the system's speed.

```php
<?php

namespace App\Console\Commands;

use Illuminate\Console\Command;

class BenchmarkBulkImport extends Command
{
    protected $signature = 'benchmark:bulk-import {count}';
    protected $description = 'Simulate a bulk import benchmark test';

    public function handle()
    {
        $count = $this->argument('count');
        $this->info("Benchmarking bulk import for {$count} records...");

        $start = microtime(true);
        sleep(2);
        $time = microtime(true) - $start;

        $this->info("Done in {$time} seconds!");
    }
}
```

Figure 6.2-5 Code Performance test (Benchmark)

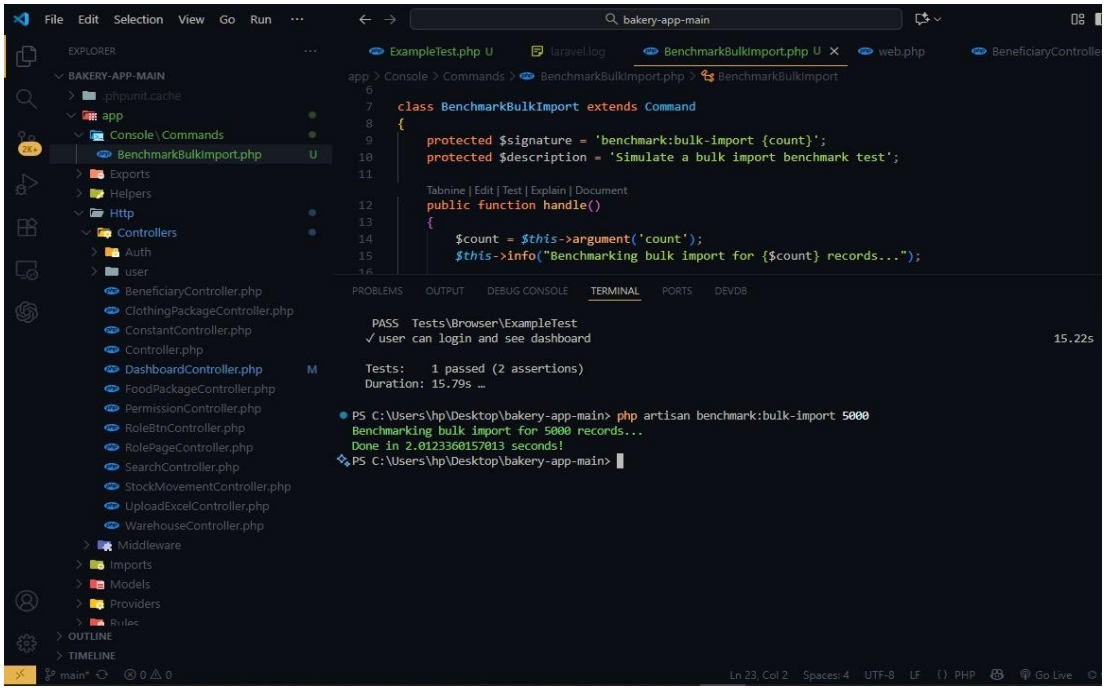

Figure 6.2-6 Result Code Performance test (Benchmark)

6.3 Testing Tools Used

- **PHPUnit (Laravel):** For automated unit testing.

- **Browser Dev Tools:** For checking frontend responsiveness and debugging JavaScript.

- **Manual Testing:** By institutional staff during UAT phase.

6.4 Test Cases Examples

Test Case ID	Test Case ID	Input	Expected Output	Result
TC-01	Beneficiary Registration	Valid data	Record saved successfully	Pass
TC-02	Beneficiary Registration	Missing required fields	Error message shown	Pass
TC-03	Login Authentication	Wrong credentials	Access denied	Pass
TC-04	Import/Export Logging	Valid data entry	Record saved successfully	Pass
TC-05	Generate Report	Select filter range	Filtered report generated	Pass

Table 6.41- Test Cases

6.5 Testing Results

- **All critical modules passed successfully.**
- **Minor UI and validation issues fixed.**
- **UAT confirmed system usability and effectiveness.**

References

[1] "World Wide Web،" 1996 .[متصل] .

[2] K. C .& .L. J. P. Laudon ،"Managing the Digital Firm تأليف "،*Firm, Management Information Systems* .2022 ،

[3] G. B. D .& .M. A. Westerman ،"Turning Technology into Business Transformation تأليف "،*Leading Digital* ،Harvard Business Review Press ، .2014

[4] E. S. R. D. D .& .K. D. Turban ،"Decision Support and Business Intelligence Systems تأليف "،*Decision Support and Business Intelligence Systems* ،Pearson Education .2011 ،

[5] R .& .N. S. B. Elmasri ،"Fundamentals of Database Systems تأليف "، *Fundamentals of Database Systems* ،Pearson Education .2016 ،

[6] C. J. Date ،"An Introduction to Database Systems تأليف "،*An Introduction to Database Systems* ،Pearson Education. .2004 ،

[7] T. C. Redman ،"The Impact of Poor Data Quality on the Typical Enterprise،" 1998.

[8] D. J. Power ،"Decision Support Systems: Concepts and Resources for Managers،" Greenwood Publishing Group.2002 ،

[9] .[متصل] "مؤسسة رحمة حول العالم،"Available: https://gaza-aids.com/distribution.

[10] "CharityTracker .[متصل] "،Available: https://www.charitytracker.com.

[11] "openIMIS .[متصل] "،Available: https://openimis.org.

[12] "AIDONIC .[متصل] "،Available: https://www.aidonic.io.

[13] "LMMS .[متصل] "،Available: https://www.wvi.org/disaster-management/last-mile-mobile-solution-lmms.

[14] "https://www.agilealliance.org/agile101."/

[15] "https://scrumguides.org."/

[16] "Laravel .[متصل] "،Available: https://laravel.com./

[17] "php storm .[متصل] "،Available: https://www.jetbrains.com/phpstorm./

[18] "VS.code .[متصل] "،Available: https://code.visualstudio.com./

[19] "Draw.io .[متصل] "،Available: https://app.diagrams.net./

[20] "MY SQL .[متصل] "،Available: https://www.mysql.com./

[21] "work bench .[متصل] "،Available: https://www.mysql.com/products/workbench./

[22] "Code in github .[متصل] "،Available: https://github.com/mohammedjr9/bakery-app.

[23] "IHH-GAZA .[متصل] "،Available: https://ihh-gaza.com./