**University of Palestine**
**Software Engineering and Artificial Intelligence**
**Software Engineering Department**

# E-Gaza Platform: Digital Renaissance Gateway to Rebuild Communication and Services Between Citizens and Governments in Gaza

**Submitted By** :

| | |
|---|---|
| **Muntaha Zaher Mustafa Shabat** | **120201580** |
| **Aya Adnan Khalil Al-Massri** | **120201426** |
| **Muath Majdi Fawzi Kullab** | **120191506** |

**Supervised By:**

Dr.Abdalhmid Zughbor

# Abstract

The e-Gaza Platform was developed as a response to the urgent challenges faced by citizens and government institutions in Gaza after the destruction of many traditional service centers. The absence of a centralized digital system has created serious difficulties for citizens in accessing essential services such as submitting complaints, receiving official announcements, and interacting effectively with government institutions. To address this gap, the e-Gaza Platform provides a unified and user-friendly solution where citizens can register, log in, manage their profiles, submit complaints, and receive notifications, while government institutions and employees can publish announcements, assign employees to track complaints, and manage service workflows.The project adopted the Agile Scrum methodology, dividing the work into iterative sprints that allowed flexibility, continuous improvement, and adaptation to evolving requirements. This ensured that the development process remained responsive to user feedback and project constraints.From a technical perspective, the backend was developed using Laravel RESTful API, with PostgreSQL (Supabase) as the database. The frontend user interfaces were first designed using Figma and then implemented with HTML, CSS, and JavaScript. The backend was deployed on Render using Docker .Testing was carried out iteratively throughout the project. Insomnia was used to validate the API endpoints, checking accessibility, request/response formats, authentication, and error handling. Additionally, manual black-box testing was performed to ensure that user-facing features such as login, complaint submission, and announcements behaved correctly and delivered the expected results.The outcome is a functional prototype of a digital governance system tailored to the needs of Gaza. Although the platform is not yet deployed for public use, it demonstrates the potential of technology to improve transparency, accessibility, and efficiency in government services. This prototype lays the foundation for a future system that can be expanded and adopted to support governance in Gaza during times of crisis and beyond.

# ملخص

تم تطوير منصة e-Gaza استجابةً للتحديات الكبيرة التي يواجهها المواطنون والمؤسسات الحكومية في غزة بعد تدمير العديد من مراكز الخدمات التقليدية. لقد أدى غياب نظام مركزي رقمي إلى صعوبات حقيقية أمام المواطنين في الوصول إلى الخدمات الأساسية مثل تقديم الشكاوى، متابعة الإعلانات الرسمية، والتواصل الفعال مع المؤسسات الحكومية. لمعالجة هذه الفجوة، تقدم المنصة حلاً موحداً وسهل الاستخدام، حيث يمكن للمواطنين التسجيل، تسجيل الدخول، إدارة ملفاتهم الشخصية، تقديم الشكاوى، واستلام الإشعارات، بينما تتمكن المؤسسات الحكومية والموظفون من نشر الإعلانات، تكليف الموظفين بمتابعة الشكاوى، وإدارة سير العمل للخدمات .اعتمد المشروع على منهجية Agile Scrum ،حيث تم تقسيم العمل إلى Sprints تتكرر باستمرار سمحت بالمرونة والتحسين المستمر والتكيف مع المتطلبات المتغيرة، مما ساعد في جعل عملية التطوير أكثر استجابة لاحتياجات المستخدمين والظروف المحيطة. من الناحية التقنية، تم تطوير الواجهة الخلفية باستخدام Laravel RESTful API مع قاعدة بيانات PostgreSQL عبر Supabase. أما الواجهات الأمامية فقد صُممت أولاً باستخدام Figma ثم طُبقت باستخدام HTML و CSS و JavaScript. كما نُشرت برمجة الواجهة الخلفية على Render باستخدام Docker. أُجريت الاختبارات بشكل متكرر طوال فترة المشروع. حيث استُخدم برنامج Insomnia لاختبار API Endpoint للتحقق من الوصول إليها، وصحة إدخال وإخراج البيانات، وآلية المصادقة، والتعامل مع الأخطاء. كما تم إجراء اختبارات manual من نوع Black Box Testing للتأكد من أن الميزات الظاهرة للمستخدم مثل تسجيل الدخول، تقديم الشكاوى، والإعلانات تعمل بشكل صحيح وتؤدي النتائج المتوقعة . والنتيجة النهائية هي نموذج أولي وظيفي لنظام رقمي مصمم خصيصاً لتلبية احتياجات غزة. ورغم أن المنصة لم تُطلق بعد للاستخدام العام، إلا أنها أظهرت إمكانيات التكنولوجيا في تحسين الشفافية وسهولة الوصول والكفاءة في تقديم الخدمات الحكومية. كما يشكل هذا النموذج قاعدة يمكن البناء عليها مستقبلاً لتوسيع المنصة وتبنيها رسمياً في غزة سواء في أوقات الأزمات أو في المستقبل.

# TABLE OF CONTENTS

## Chapter 1

## INTRODUCTION



## Chapter 2

## LITERATURE REVIEW



## Chapter 3

## METHODOLOGY

## Chapter 4

## ANALYSIS AND DESIGN



## Chapter 5

## IMPLEMENTATION

## Chapter 6

## TESTING

# TABLE OF FIGURES

# TABLE OF TABLES

# TABLE OF ABBREVIATIONS

| | Abbreviation | Meaning |
|---|---|---|
| **1** | RESTful | Representational State Transfer |
| **2** | API | Application Programming Interface |
| **3** | UML | Unified Modeling Language |
| **4** | UI | User Interface |
| **5** | UX | User Experience |
| **6** | DB | Database |
| **7** | SQL | Structured Query Language |
| **8** | HTTP | Hypertext Transfer Protocol |
| **9** | SSL | Secure Sockets Layer |
| **10** | CORS | Cross-Origin Resource Sharing |
| **11** | HTML | Hyper Text Markup Language |
| **12** | CSS | Cascade Style Sheet |
| **13** | JS | JavaScript |
| **14** | VS Code | Visual Studio Code |
| **15** | UAE | United Arab Emirates |
| **16** | UMANG | Unified Mobile Application for New-age Governance |

## Chapter 1

# INTRODUCTION

## 1.1 Introduction

This chapter introduces the project idea, its background, the problem it aims to solve, and the objectives behind its development. It also discusses the research motivation, significance, scope, and limitations of the proposed solution. The objective of this chapter is to provide a clear understanding of the context of the e-Gaza Platform and highlight its role in rebuilding communication between citizens and government institutions in Gaza.

## 1.2 Background

Gaza has faced continuous challenges due to the destruction of essential infrastructure such as municipalities, ministries, and other service providers. This has left citizens with limited access to fundamental services like electricity, water, healthcare, education, and public administration [1][2] .The breakdown of communication between citizens and government institutions has widened the gap, creating delays in receiving services, difficulties in submitting complaints, and a lack of timely updates regarding critical announcements [3].

In this context, the e-Gaza Platform is proposed as a transformative web-based solution. It provides a centralized point of interaction between citizens and government institutions, enabling citizens to:Register and securely access services,submit complaints and requests, pay bills digitally by integration with pay tools, stay informed about official announcements.For government institutions, the platform offers tools to:Manage citizen requests effectivelyu publish and update announcements , assign employees to handle complaints and services and build transparency and accountability in their interactions with citizens.

This project reflects the global shift toward digital governance, where governments increasingly rely on online systems to connect with their citizens and provide accessible, transparent, and efficient services.

## 1.3 Problem Statement

The destruction of physical infrastructure in Gaza has resulted in the collapse of communication channels between citizens and government institutions. Citizens face difficulties in accessing even the most basic services, including paying bills, receiving government updates, or submitting complaints.

Currently:

- Physical offices are damaged or overloaded.
- Citizens waste time and effort in manual processes.
- Institutions cannot track citizen requests effectively.
- There is no centralized platform for managing communication and services.

This situation leads to frustration, inefficiency, lack of accountability, and a growing trust gap between citizens and government institutions. A digital solution is urgently needed to reconnect citizens with essential services.

## 1.4Problem Questions

From the problem statement, the following research questions are derived:

1. How can we design and implement a centralized, user-friendly digital platform to connect citizens with government institutions in Gaza?
2. How can the platform facilitate seamless access to services such as bill payments, complaint submission, and government updates?
3. How can government institutions efficiently track, assign, and resolve citizens' requests and complaints in real time?
4. How can the platform foster transparency, accountability, and trust between citizens and government?
5. How can the system remain functional and accessible despite Gaza's unstable infrastructure and limited resources?

## 1.5 Research Motivation

The motivation behind this project stems from the urgent need to restore communication channels and improve access to services for Gaza's citizens. The absence of a centralized digital hub creates significant challenges for both citizens and institutions. By implementing this platform, the project aims to:

- Provide citizens with quick, reliable, and digital access to services.
- Support government institutions with efficient tools to manage citizen interactions.
- Reduce time, effort, and financial costs associated with manual systems.
- Rebuild trust between citizens and institutions by increasing transparency.

This project is not just a technical development; it represents a crucial step toward digital transformation in Gaza and toward creating a sustainable, connected future for public service delivery.

## 1.6 Research Objectives

### 1.6.1 Main Objective

To digitally transform government services in Gaza by bridging the gap between citizens and institutions.

### 1.6.2 Other Objectives

- To design and develop a web-based platform that enables citizens to register, submit complaints, pay bills, and receive official government updates.
- To provide government institutions with a centralized dashboard to manage requests, announcements, and employees.
- To integrate local payment methods to support bill payments in an accessible and user-friendly way.
- To ensure scalability and security for future expansions, updates, and feedback-driven improvements.
- To enable transparent, real-time communication between citizens and government, thereby improving efficiency and accountability [4].

## 1.7 Significance of Research

This research contributes to addressing some of the most pressing challenges in Gaza by:

- Bridging the gap between citizens and government institutions.
- Providing a transparent and efficient digital service delivery model.
- Supporting institutional accountability and improving citizen trust.
- Introducing interaction features (notifications, following, friendships) not commonly found in traditional e-government platforms.
- Offering a scalable and cloud-deployed solution that remains functional despite infrastructure challenges.

Moreover, the project has the potential to serve as a model for digital transformation in other regions facing similar crises.

## 1.8 Scope and Limitations

### 1.8.1 Scope

- Platform type: Web application accessible via desktops and mobile browsers.
- Target users: Citizens, government institutions and employees.
- Core features: Authentication, complaints, bill management, announcements, dashboards, and citizen–institute interactions.

### 1.8.2 Limitations

- The system is limited to Gaza and its local institutions

- A mobile application (iOS/Android) is not included in the initial version.
- Advanced features such as inter-institutional communication and the admin dashboard were designed but not fully implemented due to time constraints.
- Internet connectivity is required, which may be unstable in Gaza.

## 1.9 Conclusion

In conclusion, the e-Gaza Platform represents a critical solution to the communication and service delivery challenges faced in Gaza. By leveraging technology, it aims to reconnect citizens with government institutions, improve transparency, and reduce inefficiencies in public services. This chapter provided an overview of the background, identified the problems and research questions, and outlined the motivation, objectives, and contributions of the project, preparing the ground for the detailed literature review and methodology in the following chapter.

## Chapter 2

# LITERATURE REVIEW

## 2.1 Introduction

This chapter reviews the tools and technologies used in the development of the e-Gaza platform, as well as similar e-government projects. The focus is on presenting the backend and frontend technologies, design tools, testing tools, and deployment environments that were essential in building the system. Furthermore, the chapter highlights related projects and compares them with e-Gaza to emphasize the strengths and contributions of our work.

## 3.2 Used Tools and Technologies

The development of the e-Gaza platform relied on a combination of backend, frontend, design, testing, and deployment technologies. These were carefully selected to ensure scalability, usability, and reliability.

### 3.2.1 Analysis Tools

- Google Forms: used to gather feedback from citizens and government institutions. Google Forms is a versatile tool for creating surveys and collecting data, making it ideal for understanding user needs and expectations.

Figure 2.1: Google Form Logo

- Interviews and Focus Group: will complement the previous point, to gain deeper insights into user requirements.

### 3.2.2 Design Tools

- Figma: Used for UI/UX design, creating wireframes and prototypes that guided frontend implementation.

Figure 2.2: Figma Logo

- Lucidchart: Used to design use case and class diagrams

Figure 2.3: Lucidchart Logo

- WebSequenceDiagrams: Used to create sequence diagrams for visualizing workflows.

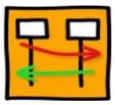

Figure 2.4: WebSequenceDiagrams Logo

### 3.2.3 Development Tools

- Laravel (PHP Framework):Used to build RESTful APIs with secure authentication and modular structure. Laravel's MVC architecture helped in maintaining clean and scalable code.

Figure 2.5: Laravel Logo

- Supabase (PostgreSQL): A cloud-hosted database that ensured data security and real-time access

Figure 2.6: Supabase Logo

- VS Code:A lightweight yet powerful code editor used for writing and debugging the backend and frontend code.

Figure 2.7: VS Code Logo

- Laragon: A portable development environment used to run PHP and MySQL locally for backend testing before deployment.

Figure 2.8: Laragon Logo

- HTML, CSS, and JS:Used to design and style the core frontend structure and implement interactive features.


Figure 2.9: HTML,CSS,JS Logo

### 3.2.4 Testing Tools

- Insomnia: Used for manual API testing to verify backend endpoints during development.

Figure 2.10: Insomnia Logo

- Web Browsers (Chrome, Firefox): Used to test UI/UX functionality and cross-browser compatibility.

Figure 2.11: Web Browsers Logo

- Mailtrap: Used to simulate outgoing emails in a safe environment. It was used to verify Laravel email configuration (such as verification emails and password reset emails) during development.

Figure 2.12: Mailtrap Logo

### 3.2.5 Version Control and Deployment Tools

- Git and GitHub: Enabled collaboration and version control, ensuring that changes were tracked efficiently.


Figure 2.13: Git and GitHub Logo

- Docker: Used to containerize the backend application for deployment on Render, ensuring consistent environments.


Figure 2.14:Docker Logo

- Render: Cloud platform used to deploy the backend APIs.

Figure 2.15: Render Logo

- Netlify: A cloud platform used to host and deploy the frontend of the project (HTML, CSS, JS). It provides fast, secure, and free hosting for static websites.


Figure 2.16: Netlify Logo

## 3.3 Similar Projects

To evaluate existing solutions, we reviewed three similar platforms and compared them to e-Gaza.

- **Smart Dubai (UAE)**

Smart Dubai is a project designed to transform Dubai into a smart city by offering all government services online, including bill payments, permits, licenses, healthcare services, and more[5].


Figure 2.17: Smart Dubai Logo

Key Strengths:

1. All government services are available online.
2. Flexible and scalable platform.
3. Efficient and quick interaction with citizens.
4. Online payment features.
5. High data protection and security

Key Weaknesses:

1. Limited to users within Dubai.
2. Relies on advanced technologies that may not be available to everyone.
3. Some transactions may be slow at times.
4. Needs more frequent updates to stay aligned with technological developments.

- **India's UMANG (Unified Mobile Application for New-age Governance)**

UMANG is a mobile app that provides access to over 1,200 central and state government services in India [6].


Figure 2.18: India's UMANG Logo

Key Strengths:

1. Supports multiple languages.
2. Integration with Aadhaar for secure authentication.
3. Real-time updates on service requests.
4. Centralized access to 1,200+ government services.

Key Weaknesses:

1. Limited to urban areas; rural access is poor.
2. No integration with local payment systems.
3. Complex user interface for non-tech-savvy users.
4. Inconsistent service availability ,some services are unreliable or slow.

- **Kenya's eCitizen**

A one-stop portal for accessing government services, including passport applications and business permits[7].


Figure 2.19: Kenya's eCitizen Logo

Key Strengths:

1. Comprehensive Service Offering.
2. User-Friendly Interface.
3. Integration of Multiple Services.
4. Supports business and citizen needs.

Key Weaknesses:

1. Digital Literacy Barriers.
2. Limited Offline Functionality
3. Technical Issues such as frequent downtime, and slow response times.
4. Slow response times.
5. Frequent downtime , Unreliable service affects user experience

The table below compares the features of the three projects with e-Gaza Platform

Table 2.1: Comparison Between e-Gaza Platform With Similar Projects

| Feature | Dubai (UAE) | India's UMANG | Kenya's eCitizen | e-Gaza |
|---|---|---|---|---|
| **Service Range** | High (All government services online) | High (Over 1,200 services) | Moderate (Various government services) | Moderate |
| **Web and Mobile Support** | Web and Mobile | Web and Mobile | Web and Mobile | Web Only |
| **Real-time Complaint Tracking** | Yes | Yes | Yes | Yes (With automated response and resolution tracking) |
| **User-friendly Interface** | High | Moderate | Moderate | High (Prioritizes user experience with intuitive design) |
| **Scalability** | High (Smart city infrastructure) | Moderate | Moderate | High (Cloud-based infrastructure for scalability and flexibility) |
| **Accessibility** | High | Moderate | Moderate | High (Focus on accessibility for all citizens, including those with disabilities) |
| **Integration with Other Systems** | High | Moderate | Moderate | High (Integration with other government databases and services) |

## 3.4Conclusion

This chapter presented the tools and technologies used in developing the e-Gaza platform, including backend frameworks, frontend technologies analysis, design tools, and deployment environments. It also reviewed similar projects such as Smart Dubai (UAE), India's UMANG, and Kenya's eCitizen, identifying their strengths and weaknesses. The comparison highlighted how e-Gaza is distinguished by being lightweight, free, socially interactive, and tailored for Gaza's unique crisis context. These insights guided our design and confirmed the originality and value of our contribution.

## Chapter 3

# METHODOLOGY

## 3.1 Introduction

This chapter describes the methodology we followed to develop and implement the e-Gaza Platform. Since our requirements were not fully clear from the beginning and many details appeared gradually during development, we chose the Agile methodology (Scrum framework). This approach gave us the flexibility to build the system iteratively, deliver working increments, and adapt to changes that happened during the project.

## 3.2Chosen Methodology: Agile Scrum

**Scrum** is an Agile framework that divides work into short development cycles called sprints. Each sprint delivers a functional increment of the system that can be tested and reviewed [8] .

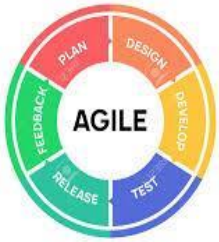


Figure 3.1: Agile Methodology

## 3.3 Justifications

We selected Scrum because:

1. The project requirements were not fully ready at the start.
2. We needed flexibility to add or modify features based on feedback.
3. Gaza's unstable conditions (electricity and internet cuts) required us to work in small increments and keep a working version online after each sprint.
4. Scrum encourages continuous testing and feedback, which matched our way of working.

These reasons have made us choose to Scrum rather than pther methodology such as Waterfall, which needs a fixed requirments from the start and Kanban, who lacks clarity of roles and sprint structure [9][10].

## 3.4Scrum Roles in The Project

We adapted Scrum roles to match our small team:

Table 3.1: Scrum Roles

| Role's Name | Participants | What participant done? |
|---|---|---|
| Product Owner | Muntaha Shabat | Proposed the project idea, collected requirements, managed the product backlog, and ensured the project was aligned with user needs. |
| Scrum Master | Muntaha Shabat | Helped organize sprints, solved technical blockers, and facilitated progress. |
| Development Team | Muntaha Shabat<br>Aya Almassri<br>Mohad Kullab | 1. Muntaha Shabat: System Analyst,UI/UX Designer and Backend developer , Functional Tester(API Tester).<br>2. Aya Almassri: Frontend developer<br>3. Mohad Kullab: Shared Tester role with Muntaha |

## 3.5Scrum Artifacts and Events in The Project

In our project, Scrum artifacts and events worked together to ensure continuous progress and delivery [11]

1. Product Backlog and Sprint Planning: At the beginning of the project, we gathered requirements and stored them in the Product Backlog such as Authentication system feature have multiple sub features like registration and login , Announcements Management feature and Complaints Management feature. During Sprint Planning, we selected the most important items to move into the Sprint Backlog.
2. Sprint Backlog and Weekly Scrum Meetings: The Sprint Backlog guided our weekly meetings , where the team discussed progress, and adjusted tasks to keep the sprint on track.
3. Increment and Sprint Review:At the end of each sprint, a working increment such as authentication in Sprint 1 or Service Request Management in Sprint 2 was delivered and tested using Insomnia and manual testing. Sprint Reviews allowed us to validate these increments.

## 3.6 Mapping Scrum to The Project Phases

Although Scrum is iterative, our project naturally followed these steps [11]:

1. Analysis: Requirements collected from deep search , brainstorming and discussions with family and neighbors (acting as stakeholders). Documented as user stories in the backlog.
2. Design: We created UML diagrams (use case, class, sequence) using Lucidchart and WebSequenceDiagrams. The UI/UX was designed using Figma.
3. Implementation:
   - Backend: Laravel RESTful APIs, deployed with Docker on Render.
   - Frontend: HTML , CSS and JS.
   - Testing:

     Functional API testing using Insomnia and Manual black box testing after deployment each sprint .
4. Deployment: Each sprint increment was deployed to the cloud, so a live version was always accessible.

## 3.7 Conclusion

In conclusion, adopting Agile Scrum was the best fit for our project. It allowed us to handle evolving requirements, develop iteratively, and ensure that every sprint delivered usable results. Despite facing technical and environmental challenges, we were able to build and deploy the e-Gaza Platform successfully by following Scrum practices.

Chapter 4

# ANALYSIS AND DESIGN

## 4.1 Introduction

This chapter presents the analysis and design phase of the e-Gaza Platform , detailing how the functional and non-functional requirements were interpreted and structured into use cases, user stories, and an overview of the planned design components such as use case , sequence and class diagrams.This chapter organized as sprints. Each sprint represents a set of related features implemented during a particular development cycle. In addition to the core sprints, an enhancement sprint is included to improve system efficiency and scalability .

## 4.2 Analysis

This section outlines the detailed functional requirements for each sprint. It includes use case descriptions, user stories, and analysis of system behavior based on user roles.This system involves multiple stakeholders:

- **Citizens :** Individuals who use the platform to access government services, interact with institutions, and communicate with others.
- **Government Institutions :** Official entities that publish announcements, handle complaints and service requests.
- **Government Employees:** Authorized staff who represent institutions and support their operations.
- **Admin :** oversee the platform, manage users, and ensure smooth operation**.**

### 4.2.1 Sprint 1: User Authentication System

This sprint focuses on implementing secure registration and login ,password recovery, and account verification functionality for citizens, government institutes, and government employees.

#### 4.2.1.1 Functional Requirements

- User should be able to select type to register
- User should be able to register with personal or institutional information
- User should be able to verify their email using a verification code.
- User should be able to log in using national ID and password after verification.
- User should be able to reset password using their registered email
- User should be able to change password
- The system should be able to register users (citizen, employee, or institute) with different required fields.
- System should be able to send email verification code upon registration.
- System should be able to activate/deactivate accounts if needed
- System should be able to allow login only for active accounts

- System should be able to prevent login via national ID and password.
- System should be able to prevent login if email is not verified
- System should be able to allow forgotten password workflow through email with reset link.
- System should be able to encrypt and store passwords securely
- System should be able to validate duplicate national IDs with type or email entries across user types.

#### 4.2.1.2 Use Case Descriptions

Covers detailed description about use cases

Table 4.1: UC1 Register New User

| | |
|---|---|
| UC-01 | Register New User |
| Actors | Citizen, Government Institute Representative, Government Employee |
| Preconditions | User does not have an existing account |
| Main Flow | 1. System displays user type selection screen<br>2. User selects type (citizen/institute/employee)<br>3. System displays appropriate registration form<br>4. User completes form and submits<br>5. System validates data and sends SMS verification code<br>6. User enters received code<br>7. System verifies code and creates account<br>8. System displays success message |
| Alternative Flows | -Invalid data: System displays error messages<br>- Verification timeout: System allows resend of code<br>- Existing user: System informs user account exists |
| Postconditions | User account created and verified. |

Table 4.2: UC2 User Login

| | |
|---|---|
| UC-02 | User Login |
| Actors | All registered users |
| Preconditions | User has a registered account |
| Main Flow | 1. System displays initial login screen (ID + user type)<br>2. User enters credentials<br>3. System validates and displays password screen<br>4. User enters password<br>5. System authenticates and grants access<br>6. System displays appropriate dashboard |
| Alternative Flows | - Invalid credentials: System displays error<br>- Forgotten password: User initiates password recovery |
| Postconditions | User logged in. |

Table 4.3: UC3 Password Recovery

| UC-03 | Password Recovery |
|---|---|
| Actors | All registered users |
| Preconditions | Users are authenticated<br>User has registered email address |
| Main Flow | 1. User selects "Forgot Password"<br>2. System prompts for ID and user type<br>3. User provides information<br>4. System sends email verification code<br>5. User enters code<br>6. System verifies and allows password reset<br>7. User enters new password<br>8. System confirms password change |
| Postconditions | Password updated. |

Table 4.4: UC4 Change Password

| UC-04 | Change Password |
|---|---|
| Actor | All registered users |
| Precondition | Users are authenticated |
| Main Scenario | 1. Users goes to change password<br>2. Enters current + new password<br>3. system updates |
| Alternative Scenario | - Wrong current password |
| Postcondition | Password updated |

#### 4.2.1.3 User Stories

Covers all the authentication-related user goals and expectations.

1. As a new user, I want to select my user type (citizen, government institute, or government employee) so that I can register with the appropriate form.
2. As a citizen, I want to input my personal details (name, ID, mobile number, etc.) so I can create an account to access government services.
3. As a government institute representative, I want to input our institute details (institution name, registration number, authorized representative details) so we can register our organization.

4. As a government employee, I want to input my employee details (employee ID, department, position) along with personal details so I can register for system access.
5. As a user registering for the system, I want to receive and enter a verification code sent to my email address so my account can be activated securely.
6. As a registered user, I want to first enter my ID and user type, then be prompted for my password so I can securely access my account.
7. As a user who forgot my password, I want to request a password reset via email verification so I can regain access to my account.
8. As a logged-in user, I want to change my password after successful login so I can maintain account security.

### 4.2.2 Sprint 2: End-to-End Citizen-Institute Engagement

This sprint focuses on implementing full interaction capabilities between the citizen and the platform. It enables the citizen to engage with government institutes, submit service requests, complaints, view announcements, manage social features like friendships and messaging, and receive notifications. It also introduces management capabilities for institutes regarding announcements, service requests, and complaints, allowing smoother citizen–institute communication.

#### 4.2.2.1 Functional Requirements

- Citizen should be able to view and update their profile (photo, name, address, etc.
- Citizen should be able to add/edit connected social profiles (Facebook, Instagram, WhatsApp)
- Citizen should be able to search for other citizens
- Citizen should be able to send and accept friend requestt
- Citizen should be able to view a list of friends
- Citizen should be able to unfriend an existing friend
- Citizen should be able to message friends
- Citizen should be able to follow/unfollow a government institute
- Citizen should be able to view a list of followed institute.
- Citizen should be able to view announcements published by followed institutes
- Citizen should be able to open and read full content of each announcement
- Citizen should be able to make like and comment on announcements
- Citizen should be able to view a list of followed institutes, filtered by their region
- Citizen should be able to select a service type for the chosen institute
- Citizen should be able to submit a service request /complaint by writing a description and selecting a date

- Citizen should be able to view a list of submitted requests /complaints with status: (Pending, Completed, Rejected)
- Citizen should be able to view a list of government-related bills (electricity, water, taxes, etc.)Each bill shows: type, government entity, amount, due date, status (Paid/Unpaid)
- Citizen should be able to click “Pay Now” for unpaid bill
- Citizen should receive notifications related to actions like:
    1. Request status is updated
    2. A friend request is accepted
    3. A new announcement is published by a followed institute
    4. A new pill has to paied

- Institute should be able to approve or reject announcements created by employees
- Institute should be able to publish approved announcements
- Government Institute should be able to view list of announcements created by employees
- Institute should be able to edit or delete published announcements
- Institute should be able to add service types or bill type related to its functions
- Institute should be able to assign employees to each service type and bill type
- Institute should be able to edit or delete service types or bill type
- Institute should be able to view incoming service requests and complaints, filtered by status
- Institute should be able to assign employees to handle specific requests or complaints
- Institute should receives notifications related to actions like:

    1. A new service request or complaint is submitted
    2. A citizen follows their profile
    3. An employee creates an announcement

- System should be able to allow citizens to edit their profile information.
- System should be able to allow uploading of profile pictures.
- System should be able to store and validate profile updates.
- System should be able to generate a personalized feed for citizens.
- System should be able to deliver posts to the dashboards of followers.
- System should be able to allow likes and comments on posts.
- System should be able to allow citizens to send friend requests.
- System should be able to allow users to accept or reject friend requests.
- System should be able to update friendship status accordingly.
- System should be able to list accepted friends for each citizen.

- System should be able to allow citizens to unfriend others.
- System should be able to send notifications for citizens and institute according to action
- System should be able to allow citizens to follow/unfollow government institutes..
- System should be able to allow institutes to publish posts.
- System should allow users to mark notifications as read.
- System should be able to show unread and recent notifications.

#### 4.2.2.2 Use Case Descriptions

Table 4.5: UC1 Submit Service Request

| UC-01 | Submit Service Request |
|---|---|
| Actor | Citizen |
| Preconditions | Citizen must be logged in and must follow at least one institute. |
| Main Flow | 1. Citizen navigates to service request form<br>2. Chooses institute and service type<br>3. Enters request details<br>4. Submits the request |
| Postconditions | Service request is saved and assigned a "Pending" status; notification is sent to the institute. |

Table 4.6: UC2 Manage Announcement

| UC-02 | Manage Announcement |
|---|---|
| Actor | Institute Admin |
| Preconditions | Announcement must be created by an employee. |
| Main Flow | 1.Admin views pending announcements<br>2. Approves or rejects<br>3. Publishes announcement<br>4.Edit or delete announcement after posted |
| Postconditions | Approved announcements are visible to followers rejected ones are discarded |
| Exceptions | No pending announcements Unauthorized access attempt |

Table 4.7: UC3 Add Complaint

| UC-03 | Add Complaint |
|---|---|
| Actor | Citizen |
| Preconditions | Must be logged in and must follow the institute. |

| | |
|---|---|
| Main Flow | 1. Citizen chooses an institute<br>2. Fills in complaint form<br>3. Submits complaint |
| Postconditions | Complaint is saved and assigned a "Pending" status; notification sent to institute. |

Table 4.8: UC4 Pay Bill

| | |
|---|---|
| UC-04 | Pay Bill |
| Actor | Citizen |
| Precondition | Citizen must be logged in. |
| Main Flow | 1. Citizen navigates to Bills & Payments tab.<br>2. System displays list of bills with status.<br>3. Citizen clicks on “Pay Now” for an unpaid bill.<br>4. System processes the payment.<br>5. System updates bill status to “Paid” |
| Alternative Flow | - If the bill is already marked as "Paid", the "Pay Now" button is disabled. |
| Postconditions | The bill status is updated, and the payment is recorded. |

Table 4.9: UC5 Add/Edit Connected Profiles

| | |
|---|---|
| UC-05 | Add/Edit Connected Profiles |
| Actor | Citizen |
| Precondition | Citizen is authenticated |
| Main Scenario | 1. Citizen opens connected profiles tab<br>2. Add/edits<br>3. Submit and confirm<br>4. system saves |
| Alternative Flow | - System rejects invalid URLs |
| Postcondition | Connected profile updated |

Table 4.10: UC6 View friends list

| | |
|---|---|
| UC-06 | View friends list |
| Actor | Citizen |
| Precondition | Citizen is authenticated |
| Main Scenario | 1. clicks friends tab<br>2. system shows list of friends |
| Alternative Flow | - No friends<br>- show empty message |
| Postcondition | Friends list displayed |

Table 4.11: UC7 Manage Service request/bill/complaints

| | |
|---|---|
| UC-07 | Manage Service request/bill/complaints |
| Actor | Institute Admin |
| Preconditions | Institute is authenticated. |
| Main Flow | 1.Admin of institute add service types or bill type for each institute function<br>2. Admin assign list of employee to this type of service or bill<br>3. Admin assign specific employee to specific incoming request or complaints<br>4.System notify employee which assigned |
| Postconditions | Each type of service or bills have list of employee<br>Each service request or complaint is handeled by specific employee |

#### 4.2.2.3 User Stories

1. As a citizen I want to update my profile so that my personal info is accurate
2. As a citizen I want to connect my social accounts so people can reach me easily
3. As a citizen I want to add/remove friends so I can build a social network
4. As a citizen I want to message my friends so I can communicate privately
5. As a citizen I want to follow/unfollow institutes so I can stay updated
6. As a citizen I want to view announcements so I stay informed about government news
7. As a citizen I want to make like and comment on announcements so I can express and interact
8. As a citizen I want to submit service requests so I can request help from institutions
9. As a citizen I want to track my requests so I know the status of my servic
10. As a citizen I want to submit complaints so I can report problems
11. As a citizen I want to get notifications so I stay updated about activities
12. As a citizen I want to view my bills and pay them easily so that I can manage and settle my obligations with government institutions efficiently
13. As a Government Institute, I want to approve or reject announcements submitted by employees so that only appropriate content is published.
14. As a Government Institute, I want to publish approved announcements to keep citizens informed.
15. As an Institute I assign employees to requests so I can respond to citizens efficiently
16. As an Institute I handle complaints so I can solve it and address public concerns
17. As an Institute I get notifications so I can respond quickly to actions

### 4.2.3 Sprint 3: Institute and Employee Functional Completion

This sprint finalizes  profile management for government institutes and employees, ensuring control over announcements, service handling, and communication.

#### 4.2.3.1 Functional Requirements

- Government Institute should be able to
- Government Institute should be able to view Institute Dashboard with total number of: Employees , Followers , Service Requests (by status) ,Complaints (by status) ,Announcements (by status)
- Government Institute should be able to view and edit institute profile info (logo, name, address)
- Government Institute should be able to add/edit connected links (Facebook, Website)
- Government Institute should be able to view a list of followers (citizens who followed the institute
- Government Employee should be able to view and update their profile (photo, contact info)
- Government Employee should be able to create new announcement and submit it to institute for approval
- Government Employee should be able to view details of assigned service requests or complaints
- Government Employee should be able to change status for assigned service requests or complaints
- Government Employee should be able to assigne bill to specific citizen

#### 4.2.3.2 Use Case Description

Table 4.12: UC1 View Institute Dashboard

| UC-01 | View Institute Dashboard |
|---|---|
| Actor | Government Institute |
| Precondition | Institute is authenticated |
| Main Scenario | 1. Institute logs in<br>2. navigates to dashboard<br>3. System fetches & shows data |
| Alternative Flow | - System returns error if data fetch fails |
| Postcondition | Dashboard data shown |

Table 4.13: UC2 Edit  Profile

| UC-02 | Edit  Profile |
|---|---|
| Actor | Government Institute and Government Employee |
| Precondition | Institute and employee are authenticated |
| Main Scenario | 1. Institute and employee navigates to profile |

| | |
|---|---|
| | 2. edits fields<br>3. submits<br>4. system saves changes |
| Alternative Flow | - Validation fails (e.g., invalid link) |
| Postcondition | Profile updated |

Table 4.14: UC3 Add/Edit Connected Profiles

| UC-03 | Add/Edit Connected Profiles |
|---|---|
| Actor | Government Institute |
| Precondition | Institute is authenticated |
| Main Scenario | 1. Institute opens connected profiles tab<br>2. Add/edits<br>3. Submit and confirm<br>4. system saves |
| Alternative Flow | - System rejects invalid URLs |
| Postcondition | Connected profile updated |

Table 4.15: UC4 View Followers

| UC-04 | View Followers |
|---|---|
| Actor | Government Institute |
| Precondition | Institute is authenticated |
| Main Scenario | 1. clicks followers tab<br>2. system shows list of followers |
| Alternative Flow | - No followers<br>- show empty message |
| Postcondition | Followers list displayed |

Table 4.16: UC5 Create Announcement

| UC-05 | Create Announcement |
|---|---|
| Actor | Government Employee |
| Precondition | Employee is authenticated |
| Main Scenario | 1. Employee fills announcement form<br>2. submits<br>3. system stores with status "pending" |
| Postcondition | Announcement created in pending status |

Table 4.17: UC6 Edit/Delete Pending Announcement

| UC-06 | Edit/Delete Pending Announcement |
|---|---|
| Actor | Government Employee |

| | |
|---|---|
| Precondition | Employee is authenticated<br>Announcement still pending |
| Main Scenario | 1. Employee edits/deletes<br>2. system verifies status<br>3. updates or deletes |
| Postcondition | Announcement updated or deleted |

Table 4.18: UC7 View Assigned Service Requests/Complaint

| | |
|---|---|
| UC-07 | View Assigned Service Requests /Complaint |
| Actor | Government Employee |
| Precondition | Employee is authenticated<br>Institute Assigned employee to service requests or complaints |
| Main Scenario | 1. Employee opens Assigned request or complaints tab<br>2. views filtered list by status |
| Postcondition | Assigned Requests or complaints list appear |

Table 4.19: UC8 Update Request or Complaint Status

| | |
|---|---|
| UC-08 | Update Request or Complaint Status |
| Actor | Government Employee |
| Precondition | Employee is authenticated<br>Employees assigned to service requests or complaints |
| Main Scenario | 1. Employee opens assigned service requests or complaints tab<br>2. selects Specific requests or complaints<br>3. View requests or complaints details to solve<br>4. Change status |
| Postcondition | Service requests or complaints status are updated |

#### 4.2.3.3 User stories

1. As a Government Institute, I want to view a summary dashboard of service requests, employees, complaints, and announcements so that I can monitor platform activity.
2. As a Government Institute, I want t o edit my profile and add connected links (e.g., website, WhatsApp) to keep my contact information current.
3. As a Government Institute, I want to view the list of citizens following my profile to understand engagement.

4. As a Government Employee, I want to create announcements and submit them for institute approval so that I can contribute updates or news.
5. As a Government Employee, I want to edit or delete pending announcements that haven't been approved yet in case of mistakes before approval.
6. As a Government Employee, I want to view the service requests and complaints assigned to me so that I can follow up efficiently
7. As a Government Employee, I want to update the status of assigned service requests and complaints to reflect progress.
8. As a Government Employee, I want to edit my personal profile to keep my contact and personal info updated.
9. As a Government Employee, I want to send bill to citizen so he can pay easily with local payment tools in Gaza.

### 4.2.4 Sprint 4: Institutional Communication and Admin Panel (Future Enhancement )

This sprint is proposed as an enhancement to improve institutional coordination and streamline document issuance. It will be implemented only if time permit.

This sprint enables direct communication between government institutions .In many governmental workflows, services often depend on coordinated actions between multiple institutions. Traditionally, these interactions are manual, slow, and require the citizen to physically carry documents between ministries, which leads to inefficiency, delays, and frustration , especially in high pressure environments.Also, add an Admin Panel to manage users, institutions, and platform health.

**Real World Example :**

For instance, rather than requiring a citizen to collect a birth notification from the health authority and deliver it to the interior ministry for a birth certificate, the health institution could send this request directly to the interior ministry within the platform. This simplifies the process for both government staff and citizens

#### 4.2.4.1 Functional Requirements

- Government Institute Should be able to send structured service requests to other government institutions (e.g., birth registration → Interior Ministry).
- Government Institute should be able to receive and process requests from other institutions.
- Government Institute should be able to track sent requests with status ("pending", "accepted", "rejected").
- Government Institute should be able to send status updates or responses to requests.
- Government Institute should be able to view an Inbox and Outbox for requests.

- Government Employee should be able to log in and act on behalf of their institution.
- Government Employee should able to draft and send official service requests.
- Government Employee should be able to update the status of incoming requests.
- Government Employees should be able to view list of sent/received requests.
- Admin should be able to log in securely as a superuser.
- Admin should be able to create/edit/suspend users and institutions.
- Admin should be able to approve or deny new institution registration requests.
- Admin should be able to monitor inter-ministry communication logs.
- Admin should be able to view and export platform analytics (e.g., most requested services, number of users, etc.).
- Citizen should be able to view status of automated requests (e.g., birth certificate).
- Citizen should be able to receive a notification when a certificate is ready.
- Citizen should be able to download the certificate from the dashboard once issued.
- System should be able to send structured requests between institutions (with status tracking).
- System should be able to log and archive all inter-institutional communications.
- System should be able to notify relevant users (e.g., citizen or receiving institution) on request status change.
- System should be able to manage and display institution inbox/outbox.
- System should be able to assign and verify institution roles and permissions.
- System should be able to allow admins to manage users and institutions.
- System should be able to generate usage reports for the admin dashboard.

#### 4.2.4.2 Use Case Descriptions

Table 4.20: UC1 Send Institutional Request

| UC-01 | Send Institutional Request |
|---|---|
| Actor | Government Employee for sender institute |
| Precondition | Employee is authenticated |
| Main Scenario | 1. Employee fills out request form according to request type.<br>2. Sends it to another institute |
| Alternate Flow | - Request rejected due to validation issues<br>- Notify sender |
| Postcondition | Request appears in the recever institute's inbox |

Table 4.21: UC2 Received Institutional Request

| | |
|---|---|
| UC-02 | Received Institutional Request |
| Actor | Government Employee for recever institute |
| Precondition | Employee is authenticated,Request available in inbox |
| Main Scenario | 1. Employee opens request<br>2. Reviews<br>3. Accepts or Rejects |
| Alternate Flow | - Request rejected due to validation issues<br>- Notify sender |
| Postcondition | Citizen receives update |

Table 4.22: UC3 Manage Platform

| | |
|---|---|
| UC-03 | Manage Platform |
| Actor | Admin |
| Precondition | Logged in as admin |
| Main Scenario | Views users, institutions, requests, reports → takes action |
| Postcondition | Changes applied system-wide |

#### 4.2.4.3 User Stories

1. As a government institute, I want to send official service requests to other ministries (e.g., birth notification), so the process can be handled automatically.
2. As a government institute ,I want to view received requests in an inbox, so I can process them accordingly.
3. As a government institute ,I want to track the status of outgoing requests, so I can ensure they are processed.
4. As an admin, I want to manage users and institutions, so I can ensure system compliance and correct usage.
5. As an admin, I want to view all inter-institutional requests, so I can monitor platform interactions.
6. As an admin, I want to see system usage statistics, so I can assess platform effectiveness.
7. As an admin, I want to approve/reject new government institutions, so only verified ministries access the system.
8. As a citizen, I want to receive notifications when a certificate is issued, so I don't have to check manually or move between ministries.

### 4.2.5 Non Functional Requirements

- **Performance:** The system should support up to 10,000 concurrent users without performance degradation.
- **Scalability:** The system should be scalable to support future additions like new institutions or services.
- **Availability:** The system should be available 24/7 with a downtime of less than 1% per year.
- **Security :** All user data and communication must be encrypted using HTTPS and SSL.
- **Authentication:** Secure login via national ID and password and email verification is required.
- **Maintainability:** The system codebase should follow Laravel standards and be modular for easy maintenance.
- **Responsiveness:** Pages and actions should respond within 2 seconds under normal network conditions.
- **Backup and Recovery**: The system should have daily automated backups with a recovery time objective under 30 minutes.
- **Auditability**: Admin should be able to view logs of actions performed by any institution.
- **Usability:** The interface must follow user-friendly UI/UX guidelines, and be easy to use even for non-technical users.
- **Browser Support:** Compatible with Chrome, Firefox, and Edge (latest two versions).
- **Mobile Support** : The platform should be responsive and mobile-friendly.

## 4.3 Design

This section presents the visual and architectural design elements used to implement the described functionalities. These designs support developers in understanding the structure, behavior, and interactions of system components .

This section includes diagrams for each sprint such as:

- Use Case Diagrams : To visualize interactions between actors and system features across each sprints [12].
- Sequence Diagrams: To demonstrate the dynamic behavior and and step-by-step interactions betweena the system[13].
- Class Diagrams: To represent the main system entities , attributes and their relationships[14].

### 4.3.1 **Sprint 1 Diagrams**

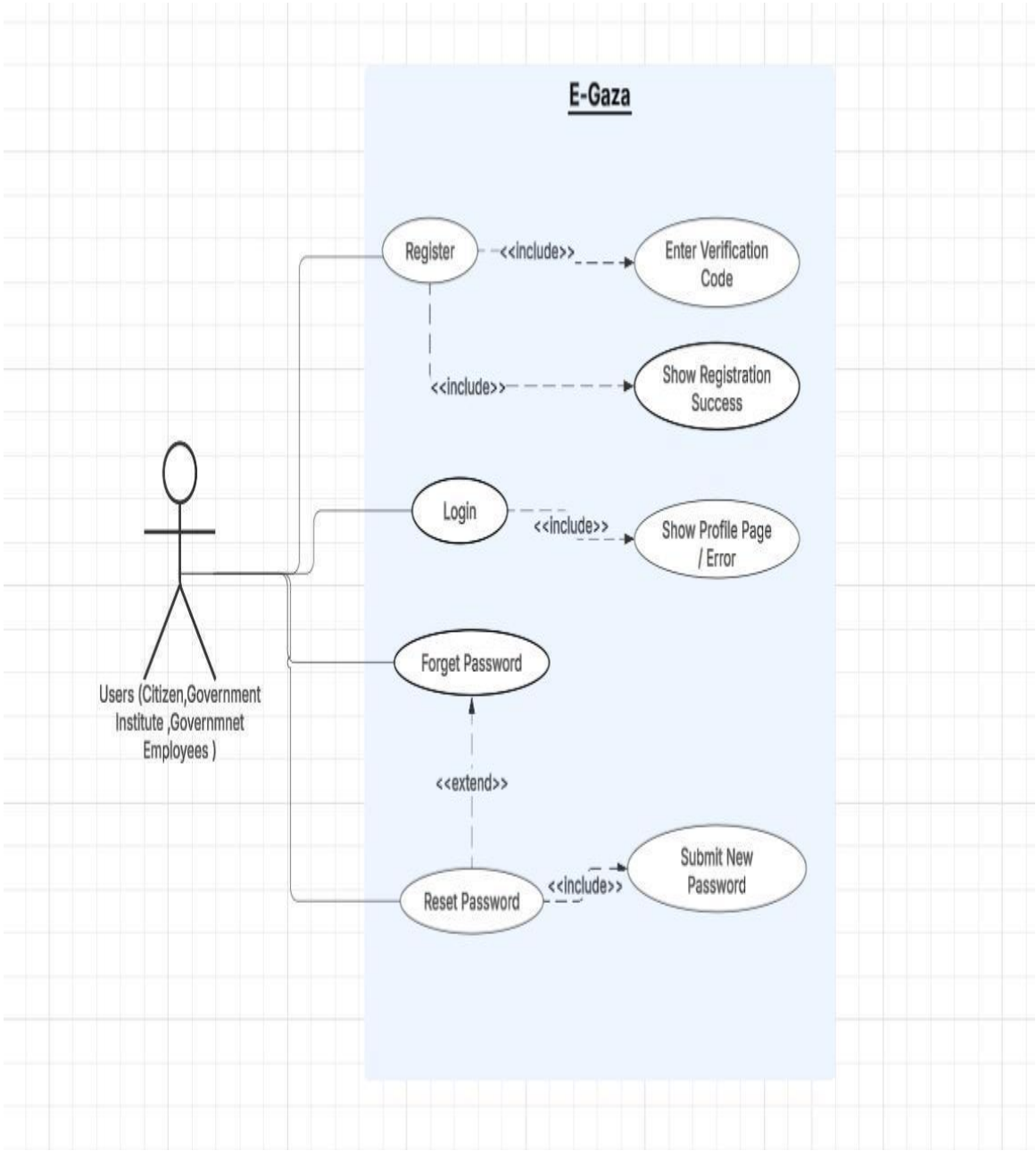


Figure 4.1 : Use Case  Digram For Sprint 1

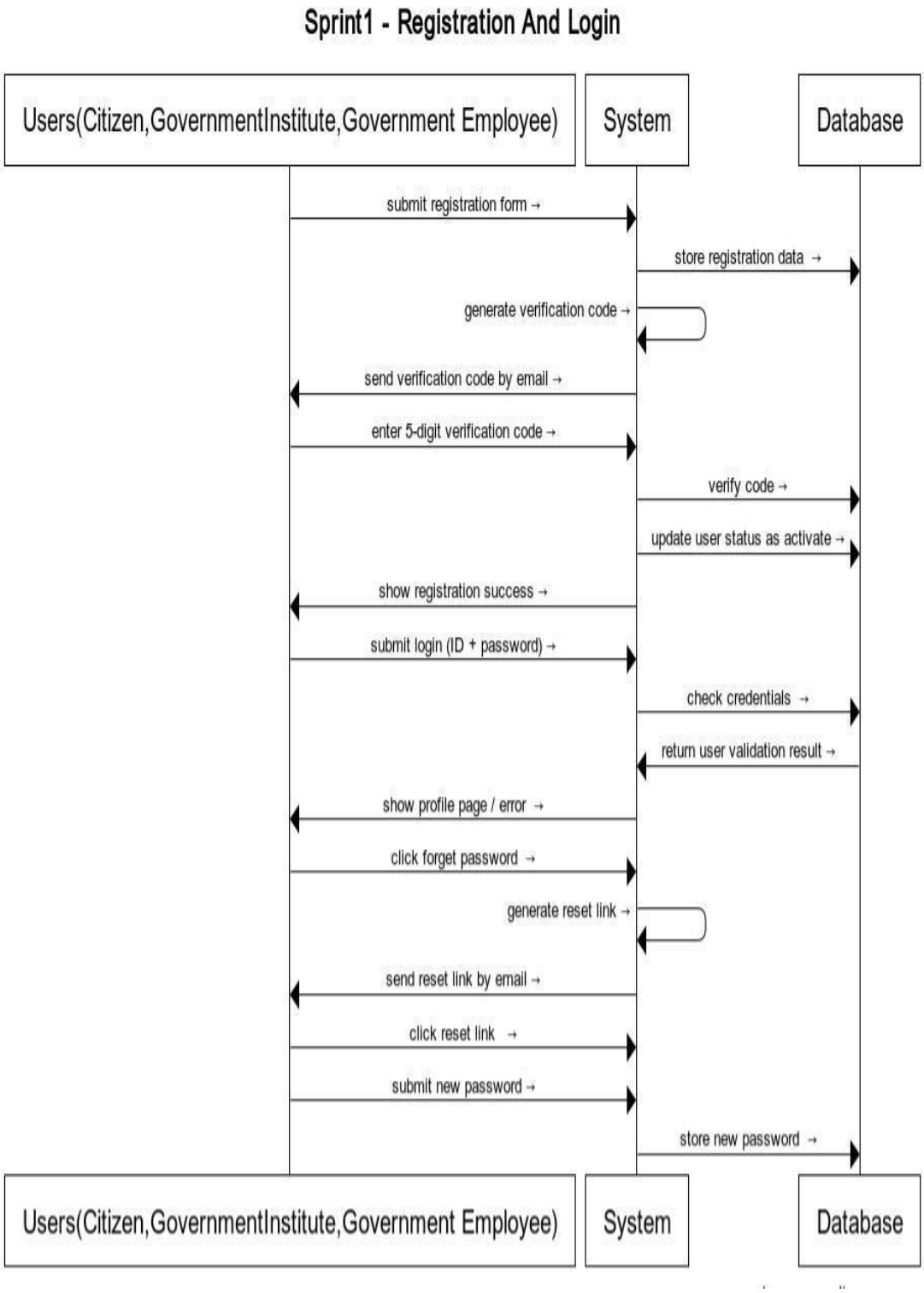


Figure 4.2: Sequence Digram For Sprint 1

### 4.3.2 Sprint 2 Diagrams

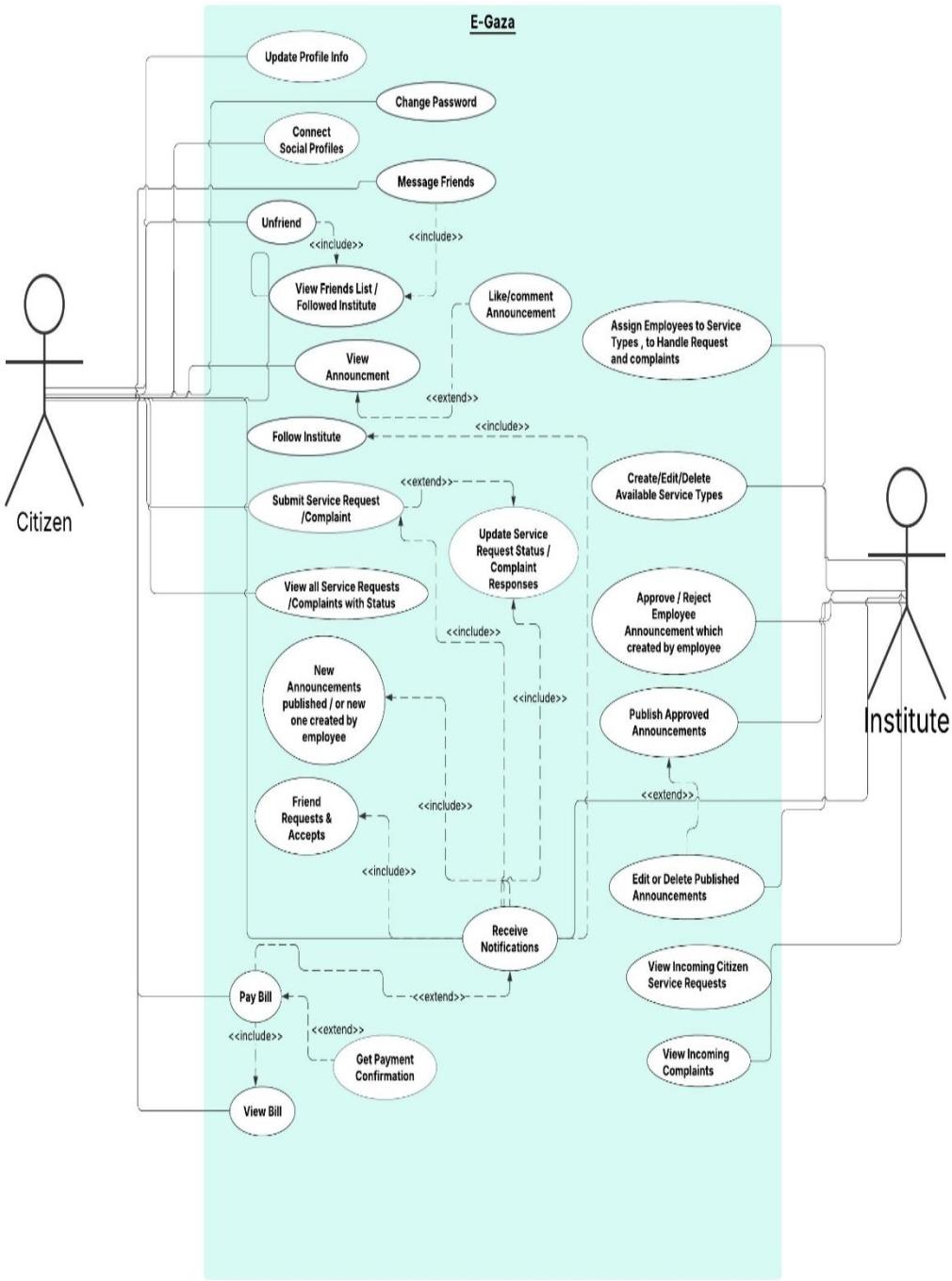


Figure 4.3: Use Case Digram For Sprint 2

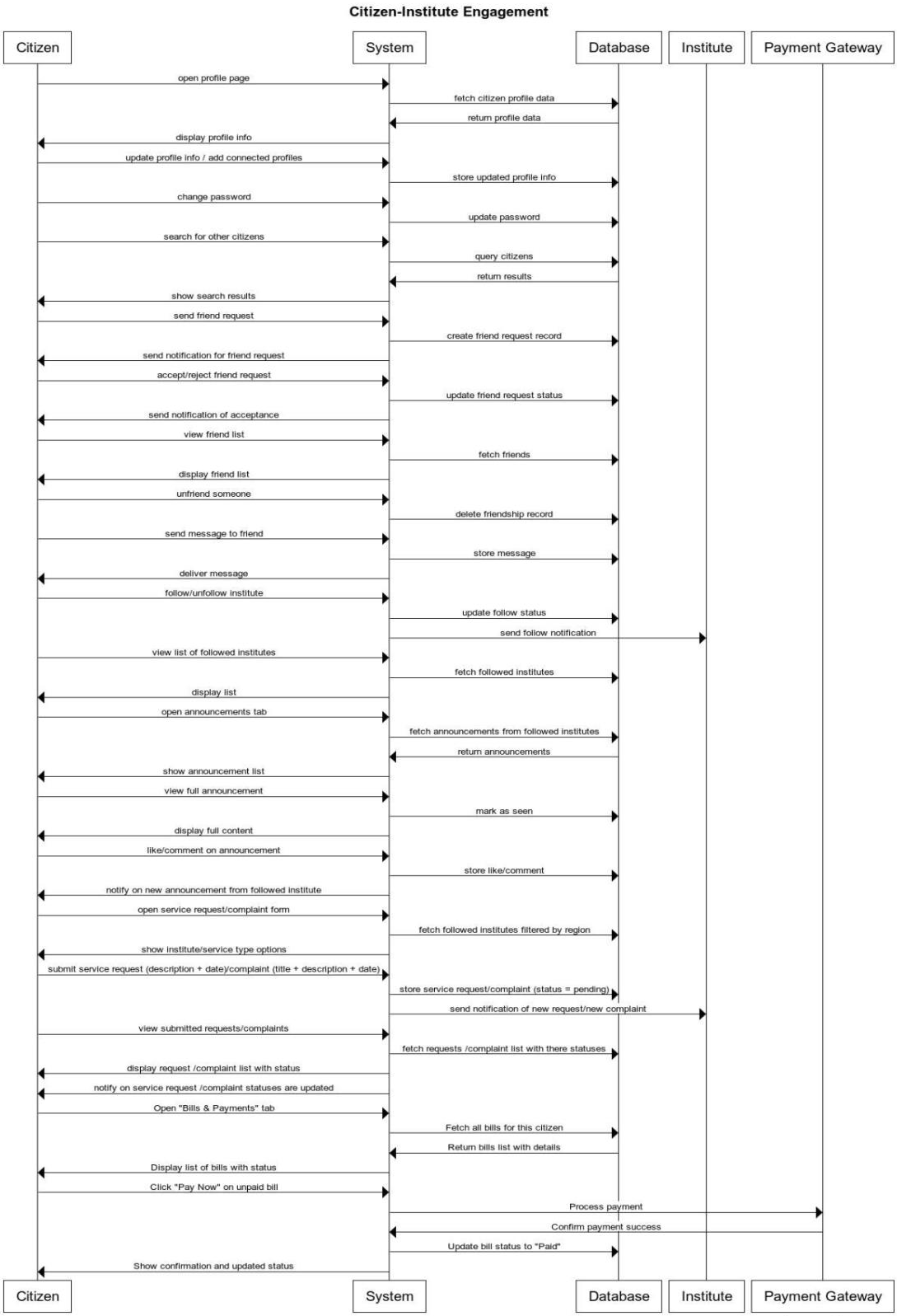


Figure 4.4: Sequence  Digram For Sprint 2

### 4.3.3 Sprint 3 Diagrams

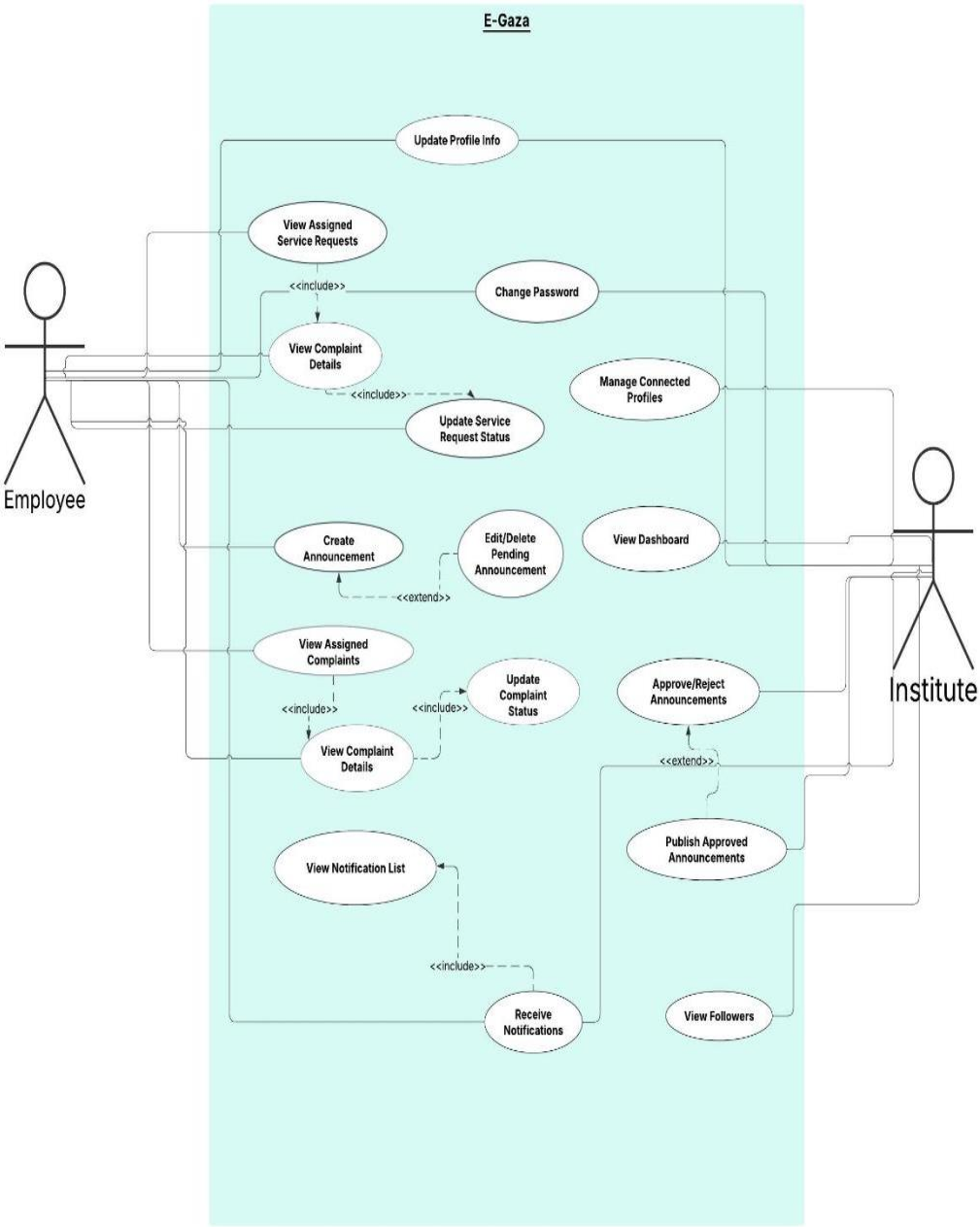


Figure 4.5: Use Case Digram For Sprint 3

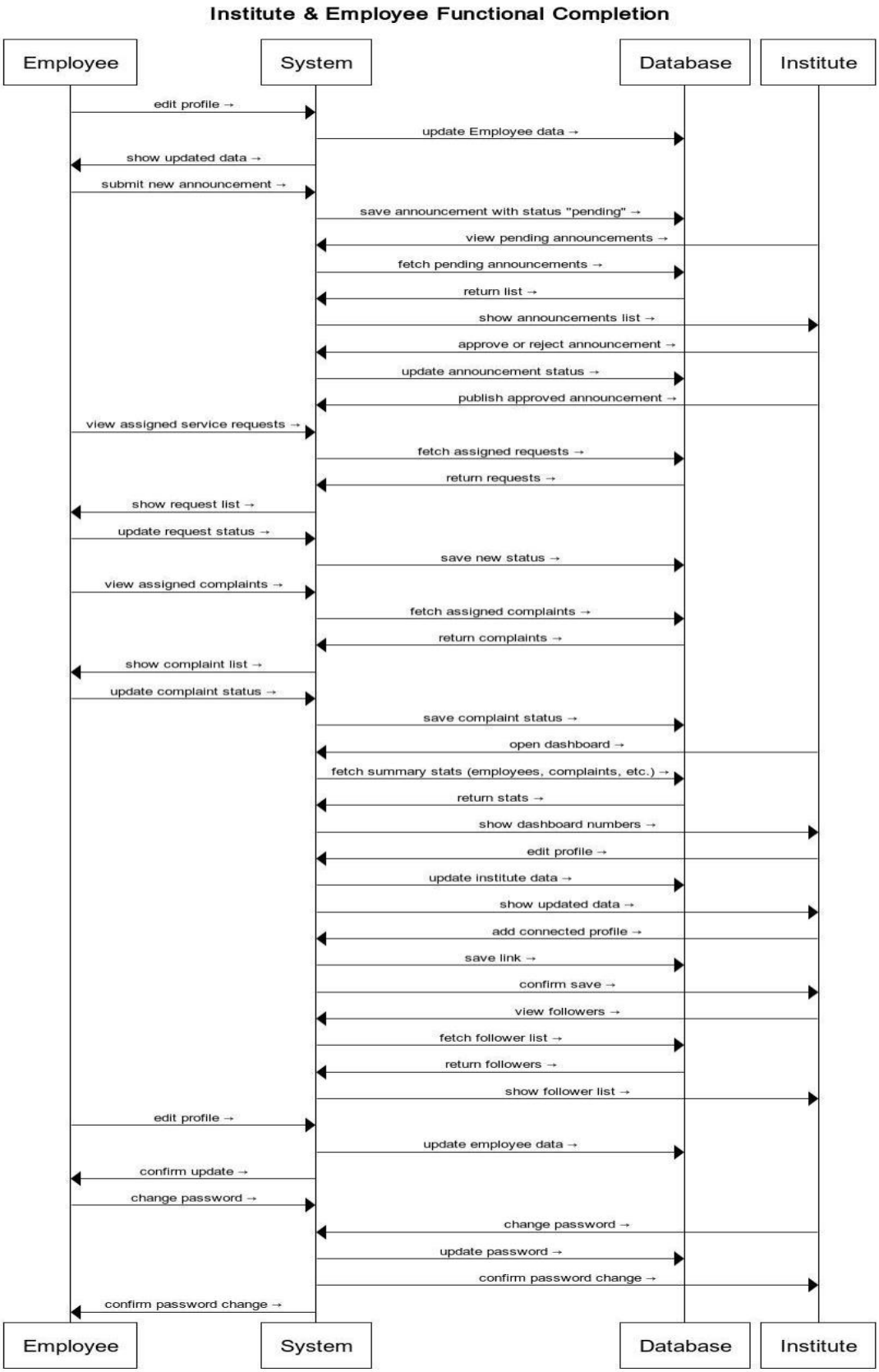


Figure 4.6: Sequence Digram For Sprint 3

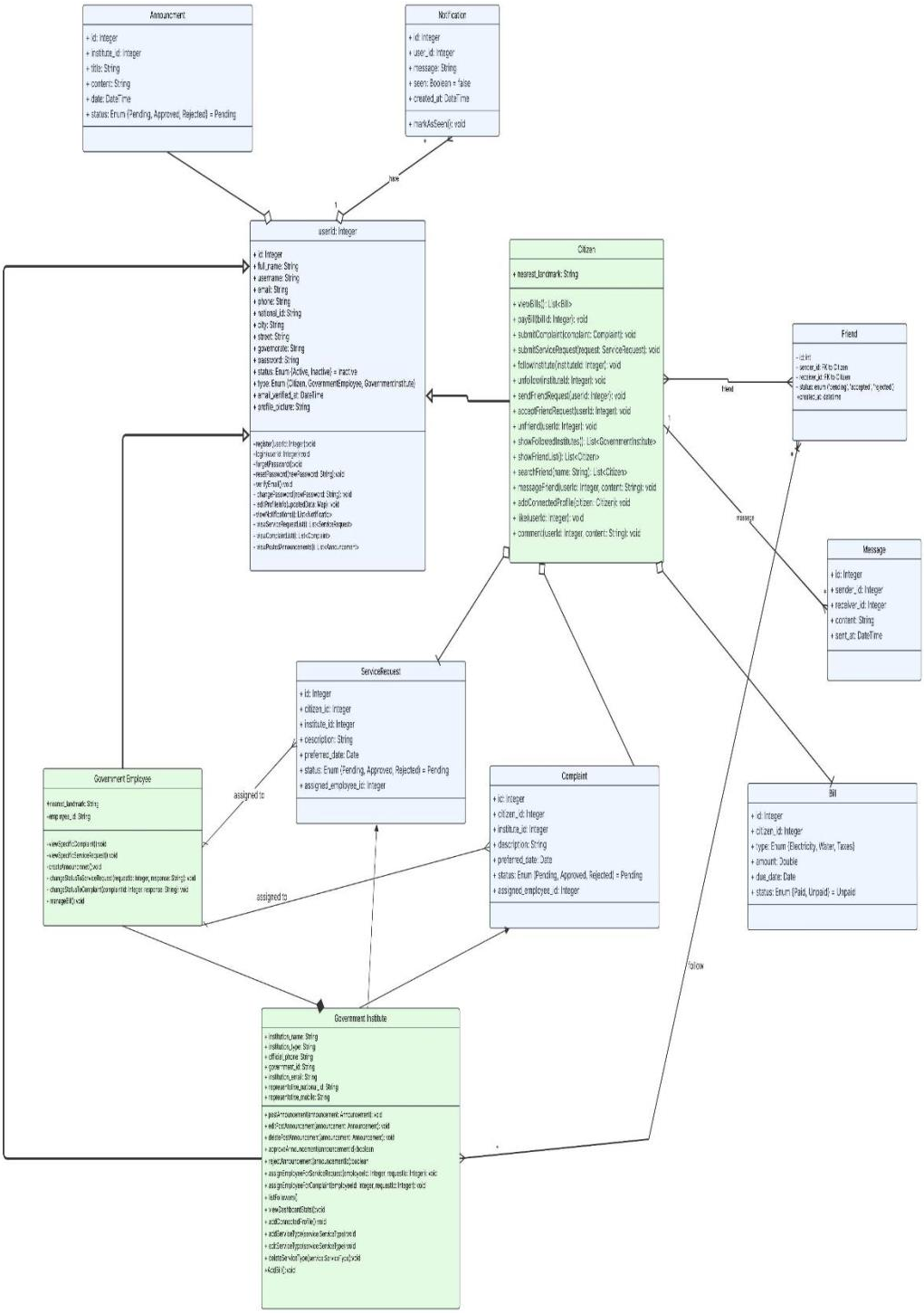


Figure 4.7: Class Digram For All Sprints

### 4.3.4 Sprint 4 Diagrams

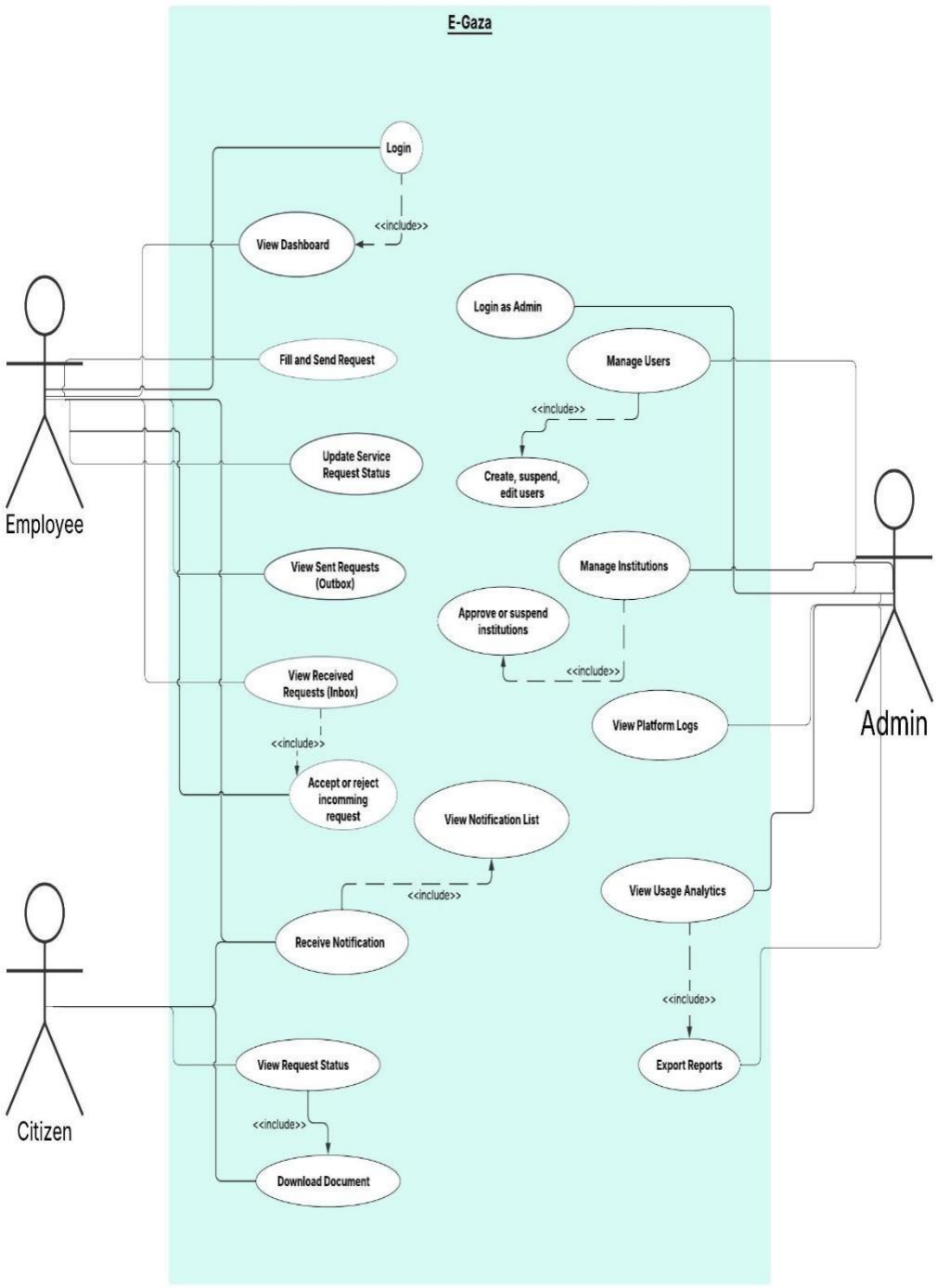


Figure 4.8: Use Case Digram For Sprint 4

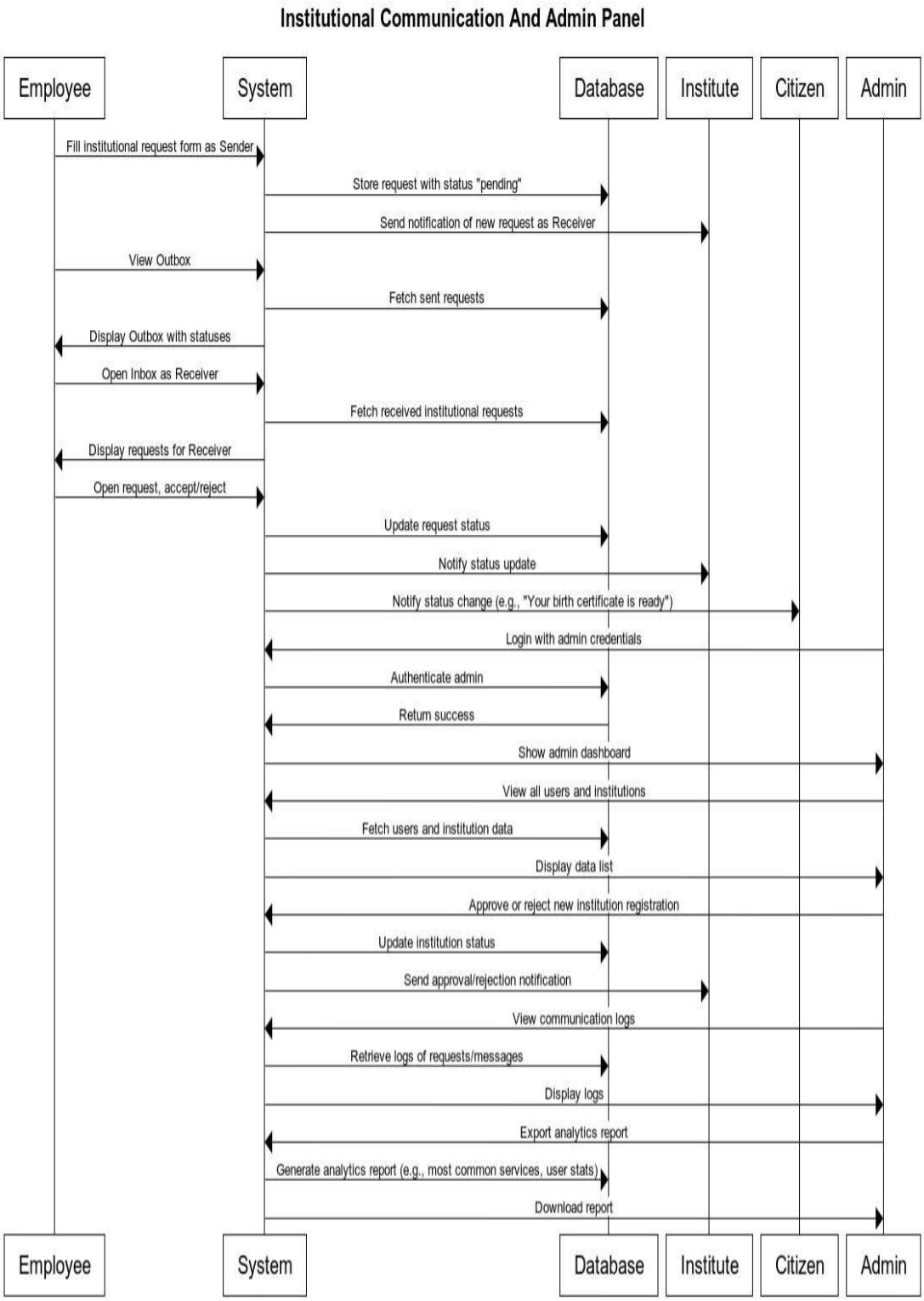


Figure 4.9: Sequence Digram For Sprint 4

#### 4.3.5 UI / UX Design

The UI/UX design was created using Figma and organized according to the Agile Scrum methodology. The design file was divided into four main boards ready for development . Inside each board, multiple screens were designed to reflect the features implemented in each sprint:

**Figma Design :**

https://www.figma.com/design/59TXpQqfkWT7KbdCrnCo0x/Graduation-Project?node-id=0-1&p=f&t=ViqZaYmOgVmP9t5w-0

## 4.4 Conclusion

In this chapter presented the analysis and design of the e-Gaza platform by identifying user needs, organizing features into sprints. Sprints 1, 2, and 3 focused on implementing core functionalities such as user registration, login, profile management, dashboard interactions, and communication features .Additionally, Sprint 4 was proposed as a future enhancement to enable institutional communication and provide the admin with full system management capabilities .The design artifacts included in this chapter will serve as a reference for developers and stakeholders, ensuring consistency, clarity, and alignment with the system’s objectives.

## Chapter 5

# IMPLEMENTATION

## 5.1 Introduction

This chapter describes the practical implementation of the e-Gaza platform, which was carried out in sprints, each focusing on delivering a specific set of functionalities. The project was developed as a web application, with the backend developed using Laravel RESTful API and the frontend implemented separately using modern web technologies. This chapter also includes screenshots of the main interfaces of the system with deployment links.

## 5.2 Technologies and Tools Used

- Backend Framework: Laravel (PHP)
- API Design: RESTful API
- Database: Supabase (PostgreSQL)
- Frontend Framework: HTML, CSS, JavaScript
- UI/UX Design: Figma
- Version Control: Git and GitHub
- API Testing Tool: Insomnia
- Backend Deployment: Render and Docker
- Frontend Deployment: Netlify
- Environment Config: Docker, .env (API keys, DB creds, mail)

## 5.3 Sprint-Based lmplementation And Time Duration

The implementation was divided into focused sprints, each delivering usable features. A summary is presented below:

Table 5.1: Sprint-Based lmplementation And Time Duration

| Sprint | Duration | Main Deliverables |
|---|---|---|
| Sprint 1: User Authentication System | 3 weeks | • Role-based registration for citizens, government institutes, and employees<br>• Email verification for account activation<br>• Secure login using national ID and password<br>• Password reset and change features |

| | | |
|---|---|---|
| | | • Account activation/deactivation and login restrictions |
| Sprint 2:End-to-End Citizen-Institute Engagement | 5 weeks | • Citizen and Institution Profile management<br>• Friend requests, messaging, and following government institutes<br>• Complaint and service request System<br>• Announcement viewing, likes, comments, and notifications<br>• Bill viewing and simulated payment |
| Sprint 3: Institute and Employee Functional Completion | 4 weeks | • Institute dashboard with stats (followers, requests, etc.)<br>• Profile editing and social links management<br>• Announcement creation and approval workflow<br>• Service request and Complaints tracking System<br>• Pay assignment |
| Sprint 4: Institutional Communication and Admin Panel (Future Enhancement) | Planned only | This sprint was Analysed but not implemented due to time limitations. |

## 5.4API Testing and Deployment During Implementation

### 5.4.1 API Testing

At first, every API endpoint was tested locally using Insomnia to check:

- If the endpoints were working correctly.

- If the request and response formats were valid.
- If the authentication with tokens was functioning as expected.
- How the system handled errors and validation rules.

These tests were done iteratively after finishing each feature in the sprints to make sure the system behaved as planned.

### 5.4.2 Deployment

After testing locally, The backend was deployed on Render using Docker. The frontend was deployed on Netlify for hosting static files (HTML, CSS, JS). After deployment, all API endpoints were tested again to ensure compatibility between the backend and frontend.

Deployment and Repository Links:

- Backend GitHub Repository: https://github.com/GraduationProject-eGaza/eGaza-backend
- Backend Deployment (Render): https://egaza-backend.onrender.com
- Frontend GitHub Repository: https://github.com/GraduationProject-eGaza/eGaza-frontend

## 5.5 Interfaces

Below some screenshots for the main user interfaces:

**Registration Screen**

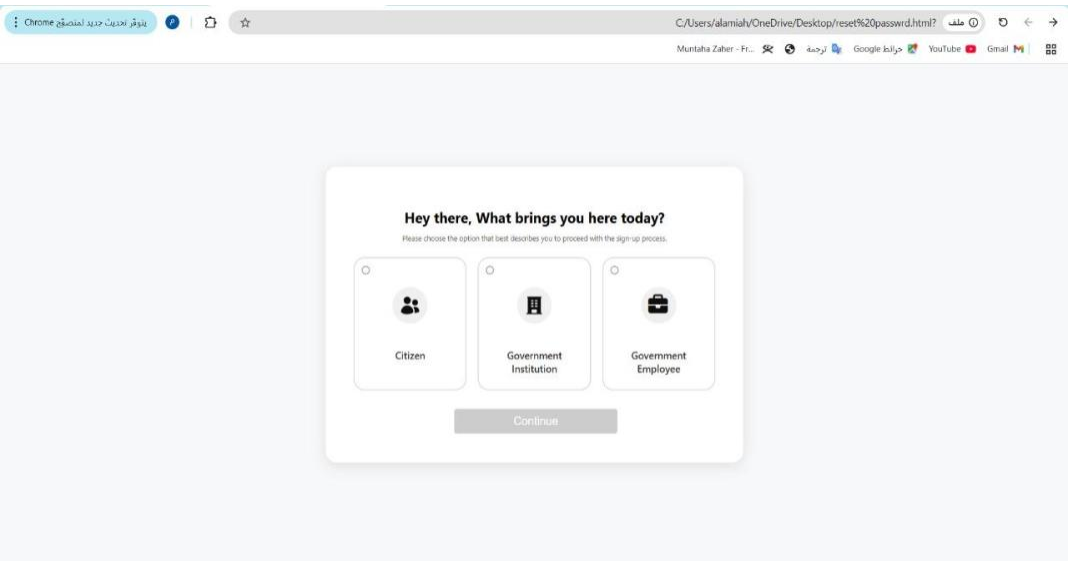


Figure 5.1: Register as Citizen, Government Institutes and Employee

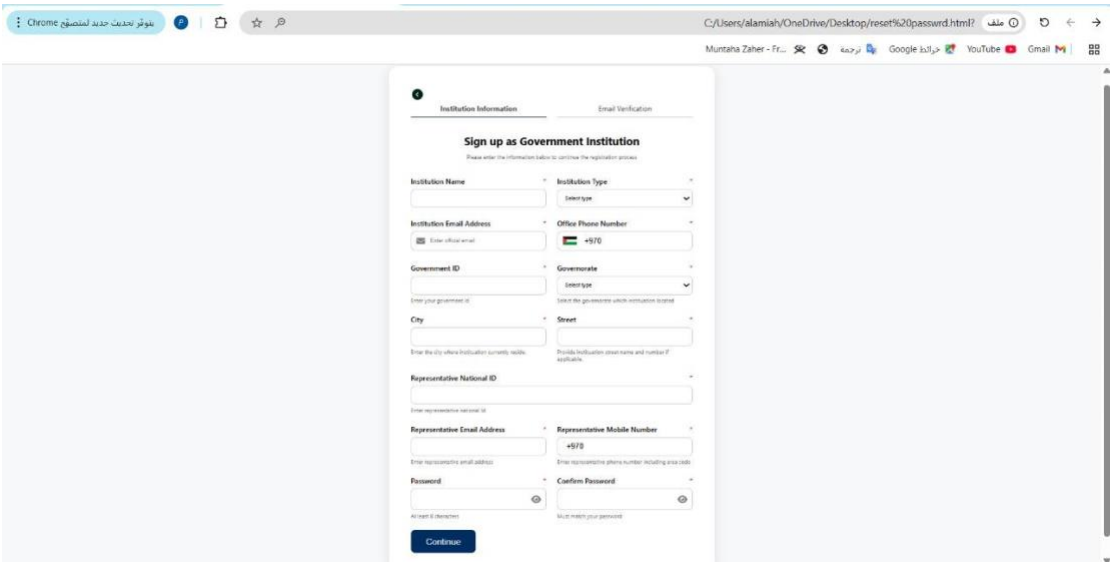


Figure 5.2: Registration Form

## Citizen Dashboard

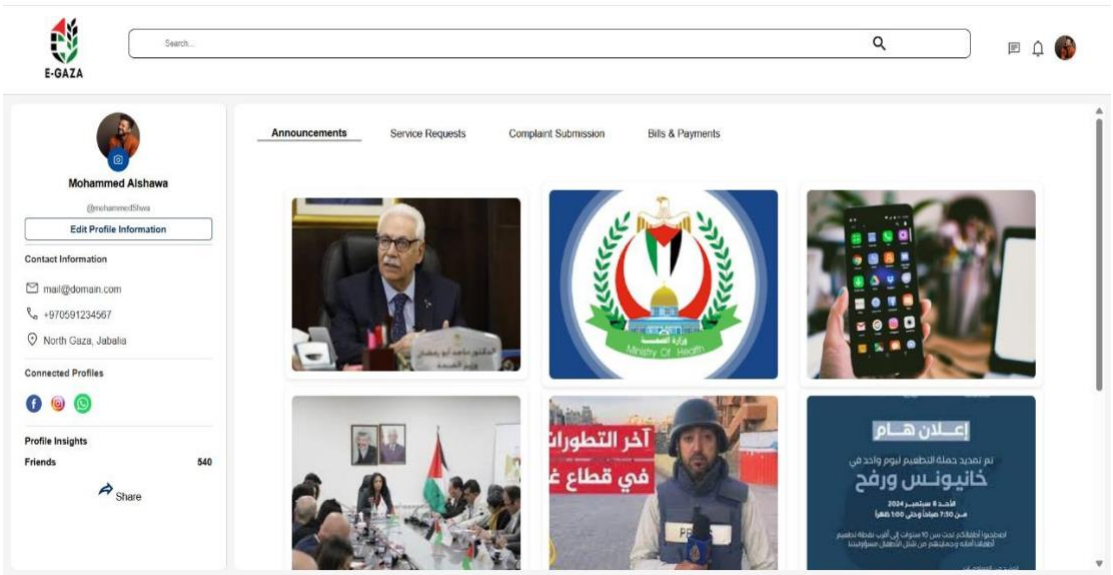


Figure 5.3: Citizen Dashboard

## Complaint Submission Form

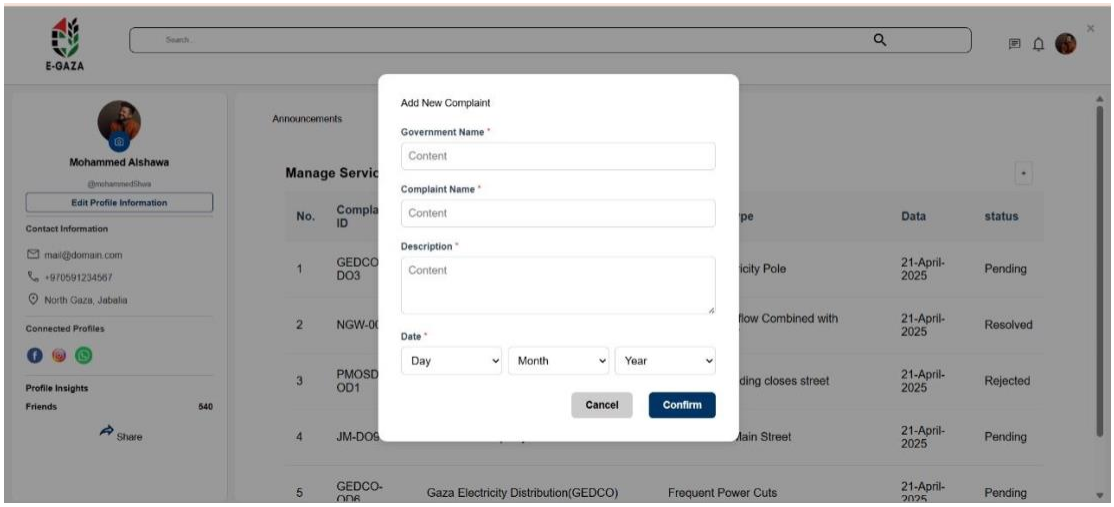


Figure 5.4: Complaint Submission From

## Institute Dashboard

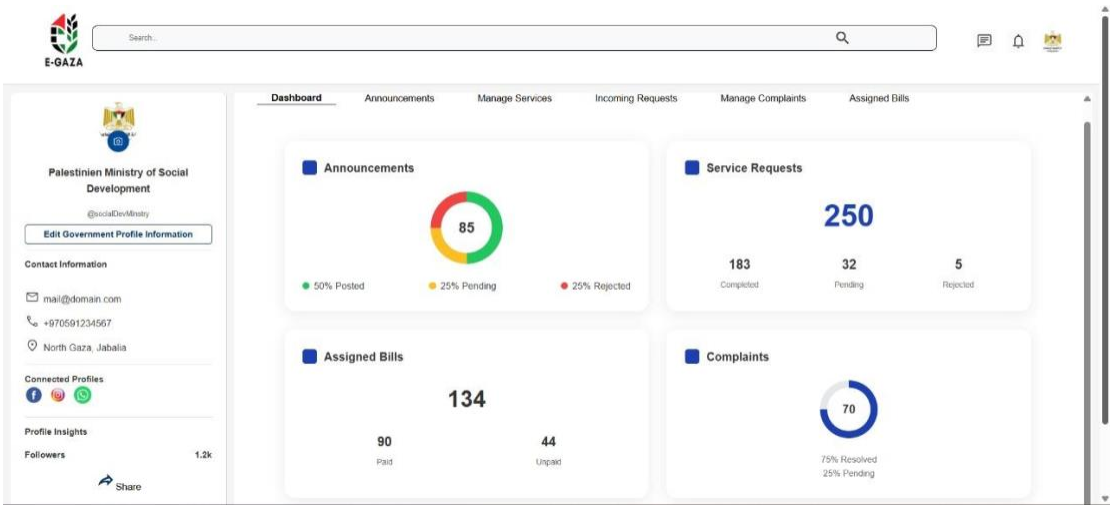


Figure 5.5: Institute Dashboard

**Add  to Service Type by Institute**

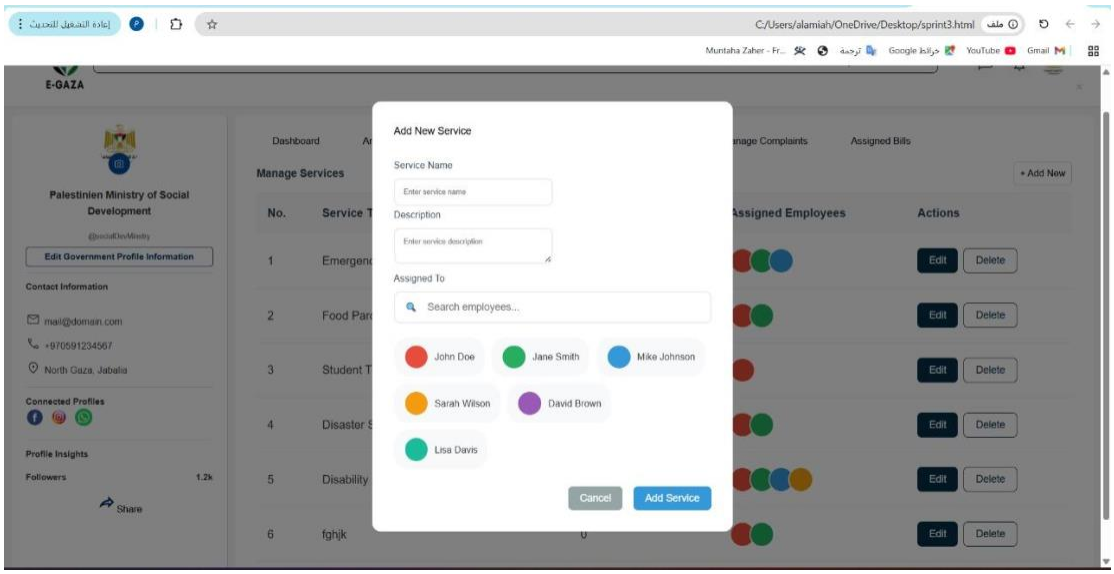


Figure 5.6:Add  to Service Type by Institute

## 5.6 Conclusion

This chapter covered the practical implementation of the e-Gaza system. Using modern tools  RESTFUL API Laravel, Docker, Supabase, and Render, the system was developed and deployed in a modular and scalable way. By following Scrum, each sprint delivered critical features that match the requirements outlined in earlier phases. The screenshots and descriptions demonstrate how users interact with the system and how government services can be streamlined in Gaza through digital means.

## Chapter 6

# TESTING

## 6.1 Introduction

This chapter presents the testing phase of the e-Gaza Platform. Testing was carried out to ensure that the system met its functional requirements, worked as expected, and provided a smooth user experience. Since the project followed the Agile Scrum methodology, testing was performed iteratively after each sprint to validate the implemented features.We chose manual testing using the black box approach, because our focus was on validating the system's input/output behavior.

## 6.2 Testing Approach

The testing strategy used in this project was Manual Black Box Testing, since the focus was on validating the functionality of the system from the user's perspective, without going into the internal code logic[16][17].

**Why Manual Testing was Selected :**

- The project scope and time were limited.
- Requirements were evolving, and manual black box testing provided flexibility to verify actual behavior against requirements.
- The system was web-based, allowing manual verification of all main features

Black Box Testing: Chosen because the interest was in ensuring correct outputs for different inputs (success and error cases), regardless of the internal implementation.
API Testing: Focusing on verifying the functionality of backend RESTful APIs before integration with the frontend.

## 6.3 Tools Used

- Insomnia: For testing backend APIs (authentication, profiles, complaints, announcements, etc.).
- Browser Testing: For verifying frontend integration with backend APIs .

## 6.4 Test Cases and Test Scenarios

Testing was made after implementation each sprint , below the main features such as : Registration , Login (from sprint 1), Complaints System (from sprint 2) and Announcements System (from sprint 3)with their respective test cases

**Feature 1: Registration (from Sprint 1)**

Table 6.1:Test Case For Registration Feature

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC1.1 | Register with valid data | User enters full details (ID, email, etc) | Account created, verification email sent. | Pass |
| TC1.2 | Register with missing fields | Citizen leaves email blank | Error shown: “Email required.” | Pass |
| TC1.3 | Register with duplicate ID with different Type User | Citizen who registered with National ID want to register as Employee | Account created, verification email sent. | Pass |
| TC1.4 | Register with duplicate ID and User Type | Citizen with his National ID try to register again | Error shown: “[Citizen ] with ID already registered. | Pass |

**Feature 2: Login (from Sprint 1)**

Table 6.2:Test Cases For Login Feature

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC2.1 | Login with correct credentials | Correct ID + password | Redirect to dashboard. | Pass |
| TC2.2 | Login with incorrect password | Wrong password | Error: “Invalid credentials.” | Pass |
| TC2.3 | Login without verifying emai | User not activate his account | Valid user but email not verified | Pass |

**Feature 3: Complaint System (from Sprint 2)**

**Feature 3.1: Citizen Submits Complaint**

Table 6.3:Test Cases For Citizen Submits Complaint

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC3.1.1 | Citizen submits complaint with valid details | Citizen enters full details to | Complaint saved successfully with status = Pending. | Pass |

| | | | | |
|---|---|---|---|---|
| | | submit complaint | | |
| TC3.1.2 | Citizen submits complaint without required fields . | Empty subject/details | Validation error, complaint not saved. | Pass |
| TC3.1.3 | Unauthorized user tries to submit complaint. | Token for Employee put in header | Access denied | Pass |

### Feature 3.2: Institute Assigns Employee

Table 6.4 :Test Cases For Institute Assigns Employee

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC3.2.1 | Institute admin assigns complaint to valid employee. | Institute try to choose employee id from list | Complaint assigned, employee notified. | Pass |
| TC3.2.2 | Institute tries to assign complaint to a non-employee (e.g., citizen Id). | Put incorrect id which for non employee or employee not worked in the same institute | Error: Invalid assignment. | Pass |
| TC3.2.3 | Institute tries to assign complaint which already assigned | Institute choose complaint id for institute which already assigned to employee | Error: Complaint already assigned | Pass |

### Feature 3.3: Employee Works and Resolves Complaint

Table 6.5:Test Cases For Employee Works and Resolves Complaint

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC3.3.1 | Employee updates complaint progress | Employee review citizen's complaint and work to solve without change status | Progress saved, complaint status remains pending until marked resolved | Pass |
| TC3.3.2 | Employee marks complaint as resolved. | Employee choose from list resolved instead of pending | Complaint status updated to Resolved, citizen notified | Pass |

| | | | | |
|---|---|---|---|---|
| TC3.3.3 | Employee tries to resolve a complaint not assigned to them | Put incorrect complaint id in request | Access denied. | Pass |
| TC3.3.4 | Unauthorized user (citizen) tries to resolve complaint and change status | Token for citizen put in header | Access denied | Pass |

**Feature 4: Announcements System(from Sprint 3)**

**Feature 4.1 : Create Announcement (Employee)**

Table 6.6:Test Cases For Create Announcement

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC4.1.1 | Employee creates announcement with valid data | Employee enters full details | Saved successfully, status pending | Pass |
| TC4.1.2 | Employee tries to create without required fields | Title/content empty | Validation error | Pass |
| TC4.1.3 | Unauthorized user (citizen) tries to create announcement | Token for citizen put in header | Access denied | Pass |

**Feature 4.2: Approve and Reject Announcement (Institute Admin)**

Table 6.7:Test Cases For Approve and Reject Announcement

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC4.2.1 | Admin approve/reject a pending announcement. | Institute Admin review and approve/reject announcement | Status changes to Posted, visible to citizens/rejected | Pass |
| TC4.2.2 | Unauthorized user (citizen) tries to approve/reject | Token for citizen put in header | Access denied | Pass |

| | | | | |
|---|---|---|---|---|
| TC4.2.3 | Admin tries to approve/reject already approved/rejected announcementt | Institute Admin review and approve /reject announcement which posted/rejected in previous | Already approved/rejected | Pass |

**Feature 4.3 : Update and Delete pending Announcement (Employee )**

Table 6.8:Test Cases For Update and Delete pending Announcement

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC4.3.1 | Employee updates/delete a pending announcement. | Employee review and update/delete pending announcement | Announcement updated/deleted successfully ,remains pending | Pass |
| TC4.3.2 | Unauthorized user (citizen) tries to update/delete pending announcement. | Token for citizen put in header | Access denied | Pass |

**Feature 4.4 : Update and Delete posted Announcement (Institute )**

Table 6.9:Test Cases For Update and Delete posted Announcement

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC4.4.1 | Institute Admin updates/delete a posted announcement. | Admin review and update/delete posted announcement | Announcement updated successfully ,still visible to citizens/deleted | Pass |
| TC4.4.2 | Unauthorized user (citizen) tries to update/delete posted announcement. | Token for citizen put in header | Access denied | Pass |

**Feature 4.5: View , Like and  Comment Announcements (Citizen)**

Table 6.10:Test Cases For View,Like and Comment Announcements

| Test Case ID | Test Scenario | Input Example | Expected Result | Status |
|---|---|---|---|---|
| TC4.5.1 | Citizen views/like/comment posted announcements | Citizen make view,like and comment | All approved announcements visible/Like icon marked and count increases/Comment saved | Pass |
| TC4.5.2 | Citizen tries to view/like/comment pending/rejected announcements | Try to make view ,like, comment for unposted announcements | Access denied | Pass |

## 6.5 Conclusion

In conclusion, testing was conducted iteratively throughout the project in alignment with Agile Scrum methodology. We adopted manual black box testing supported by Insomnia for backend validation and manual testing for frontend[18]. Each sprint was tested upon completion, ensuring early detection and correction of issues. Core functionalities such as authentication, complaints and announcements were verified successfully, the tested system features demonstrated that the platform is functional, reliable, and aligned with the defined requirements.

# APPENDICES

## Appendix A: Brainstorming Mind Map

During the requirement gathering stage, a brainstorming mind map was created to visualize all the main components and features of the e-Gaza Platform. This map helped in organizing ideas and prioritizing functionalities before moving into detailed design

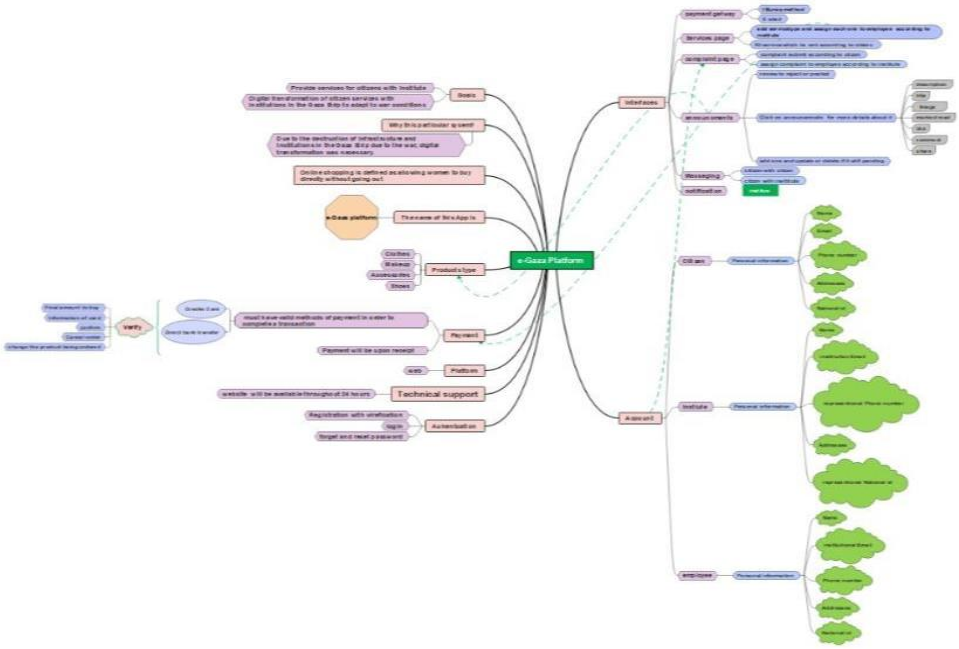


Figure A.1: Brainstorming Mind Map for e-Gaza Platform

## Appendix B: Interview Questions and Outcomes

To gather accurate requirements, interviews were conducted with different stakeholders (citizens, employees, and government institutes).

**Citizens Questions**

- What kind of government services do you use most frequently?
- What challenges do you face when trying to access government services today?
- Would you prefer to submit requests and complaints online instead of visiting offices? Why?
- How important is it for you to track the status of your requests or bills?

- Would you like to receive notifications (SMS/Email) about service updates, bills, or announcements?
- What features would make the platform easier and more useful for you (e.g., online payments, service history, digital documents)?

**Outcomes**

- Identify essential citizen needs like service requests, complaints, bills, notifications, and profile features.
- Highlight usability requirements (simple interface, mobile-friendly).
- Understand preferences for communication channels (notifications, email, SMS).

**Government Institutes Questions**

- What types of services do you provide to citizens?
- How do you currently manage citizen requests and complaints?
- What difficulties do you face in assigning tasks to employees?
- Would you find it helpful to track service request statistics (completed, pending, rejected)?
- How important is it for your institute to post announcements for citizens?
- Do you need a way to track your followers and their engagement?

**Outcomes**

- Define institute requirements: manage employees, assign tasks, post announcements, bills, dashboards.
- Show the need for data tracking and reports.
- Confirm features like followers tab and social media sharing.

**Government Employees Questions**

- What tasks do you usually handle in serving citizens?
- How do you receive assignments from your institute?
- Do you need notifications when a citizen pays a bill or submits a complaint?
- How should the approval/rejection process of announcements work from your perspective?

**Outcomes**

- Gather needs for assigned request management, bill assignment, notifications, and announcement submission.
- Confirm workflow between employees and institutes.

## Appendix C: Inspired Designs (Figma References)

To design the system interfaces,some benchmarking designs from different applications and websites were used as references, even if they weren't directly related to government platforms. The goal was to benefit from best practices in UXUI Design

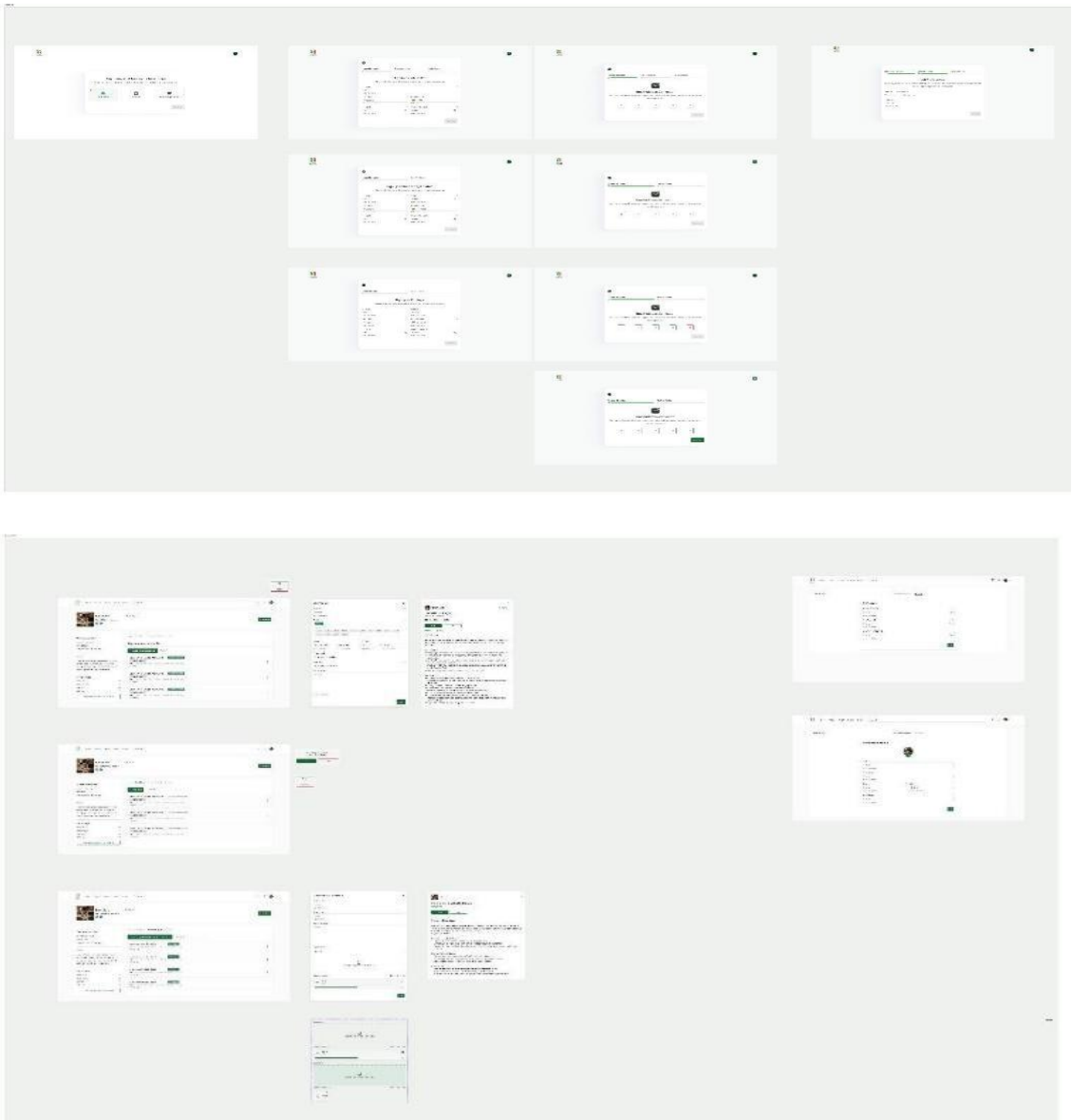

Figure C.1: Sample inspired UI Used as Reference for Figma Designs

## Appendix D:Development and Deployment Successfully Evidence

## D.1 Database Schema (Supabase)

The e-Gaza Platform database was hosted on Supabase (PostgreSQL) The schema defines the main entities such as Citizens, Government Institutes, Employees, Complaints, Announcements, and Friendships, along with their relationships. This schema ensures data consistency and supports efficient interactions between different actors in the system.

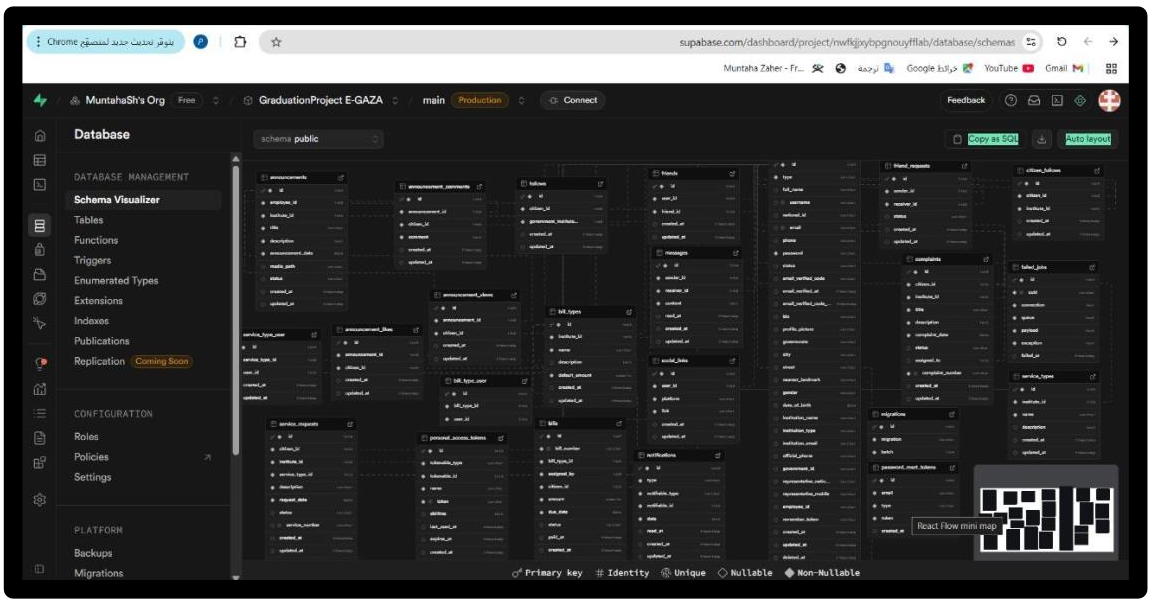


Figure D.1: Database Schema in Supabase

## D.2 Backend Deployment (Render)

The backend APIs of the e-Gaza Platform were containerized using Docker and deployed on Render. The screenshot below shows the deployed backend service along with its live status .

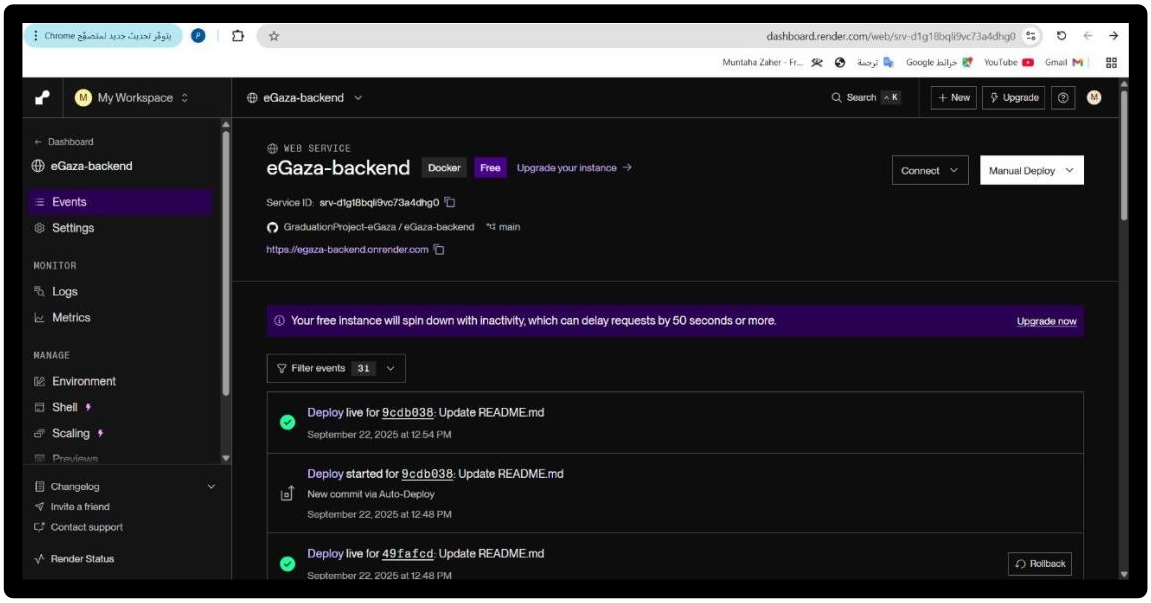


Figure D.2: Backend Successfully Deployed on Render

## D.3: Mailtrap Test

During development Mailtrap  used to capture outgoing emails in a safe environment. It was used to verify Laravel email configuration (such as verification emails and password reset emails)

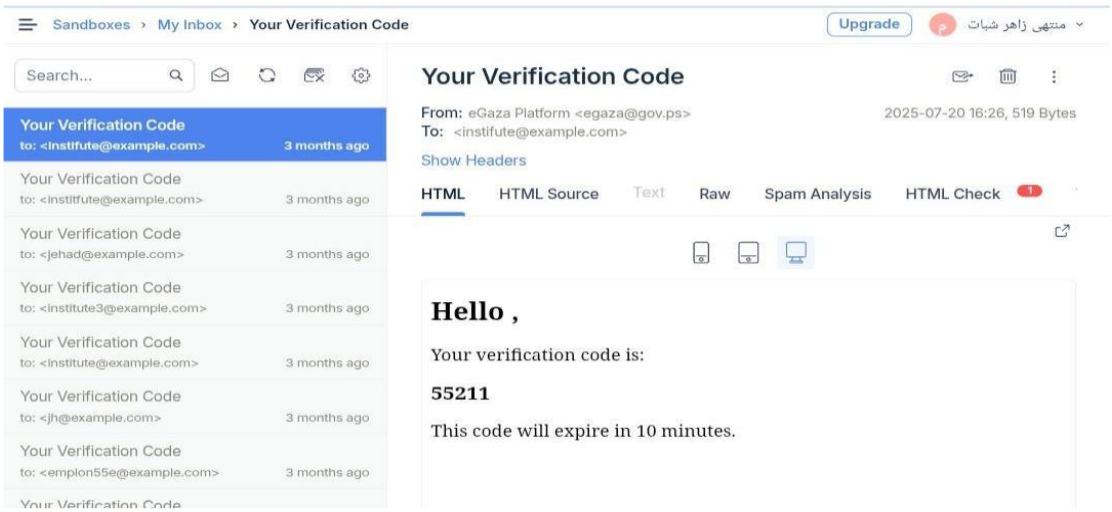


Figure D.3: Successful Verification Code Send Email

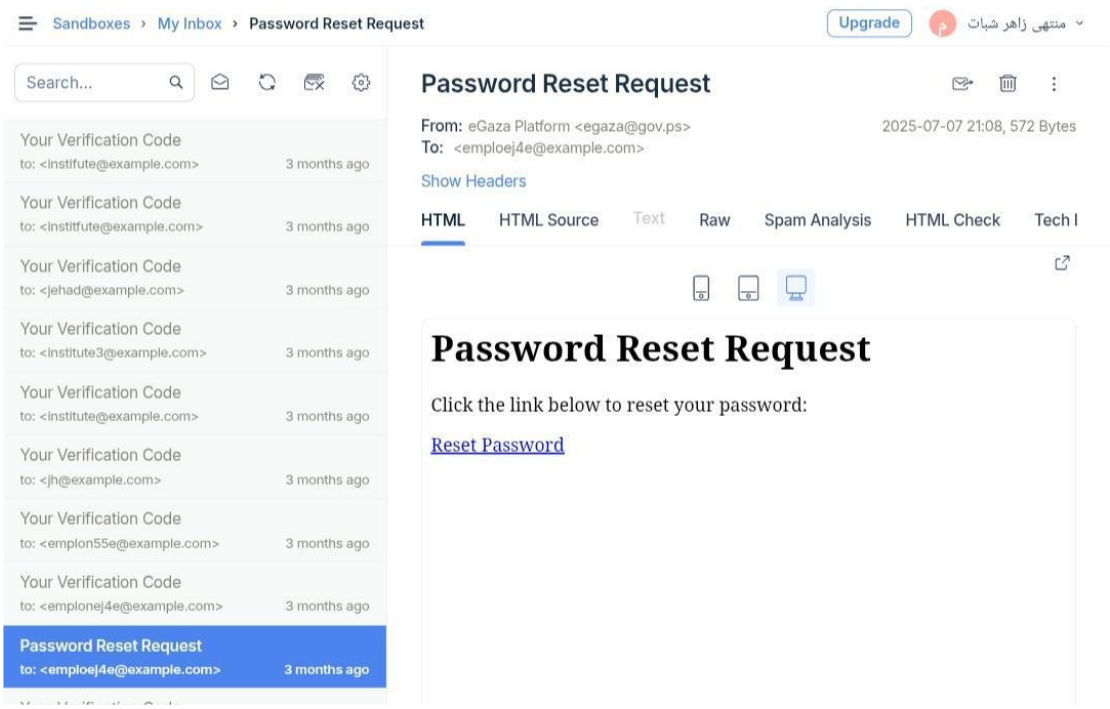


Figure D.3: Successful Reset Password Send Email

## Appendix E: Testing Reviews

During the testing phase, results were documented using tools such as Insomnia for API testing, sample test:

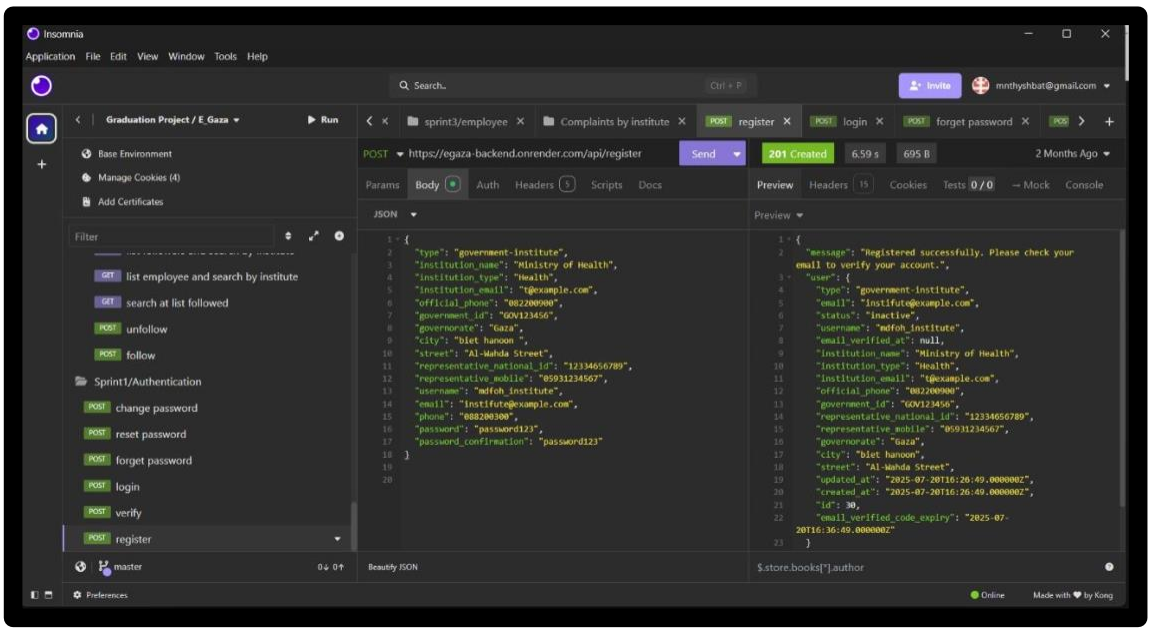


Figure E.1: Registration API Test in Insomnia

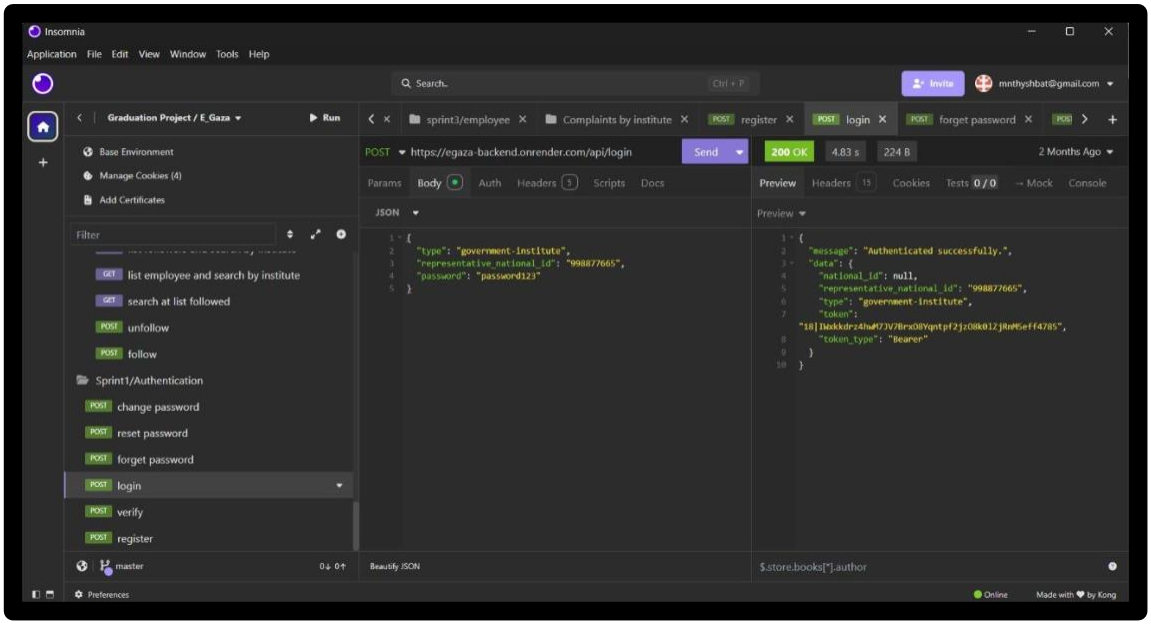


Figure E.2:Login API Test

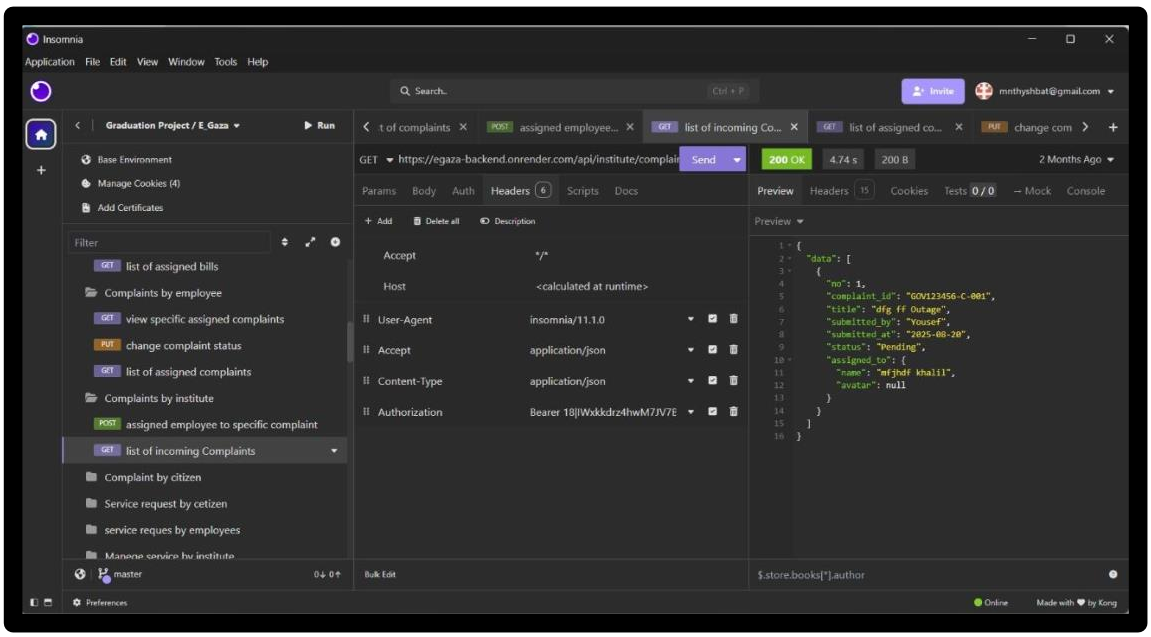


Figure E.3: List Of Incoming Complaints For Institutions  Test

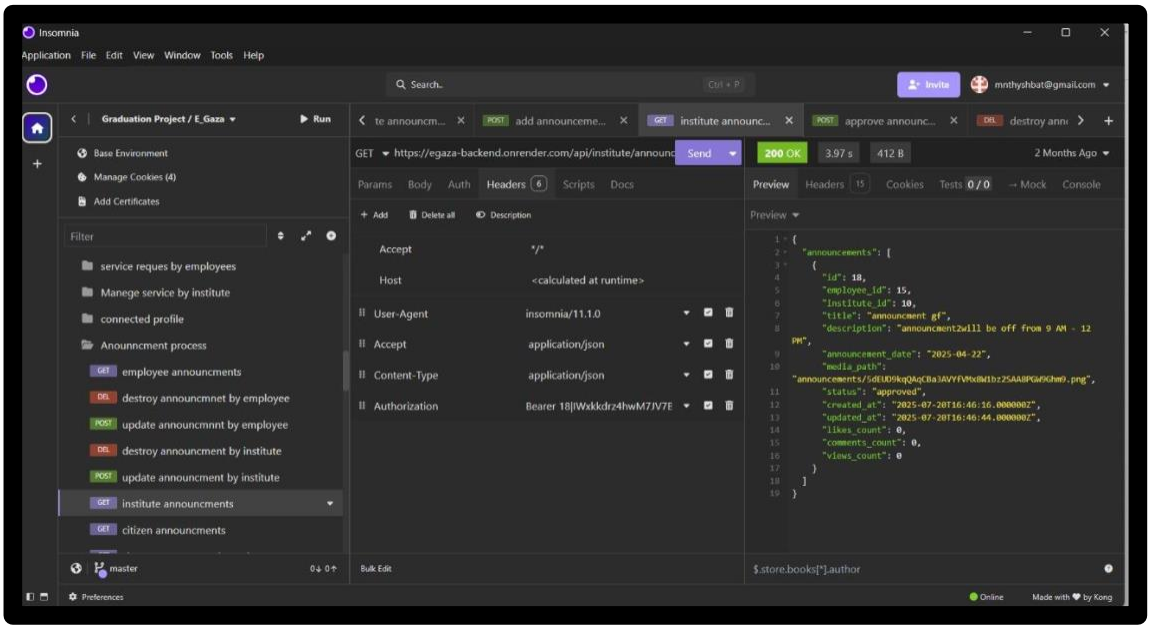


Figure E.4: Announcement List For Institutions API Test

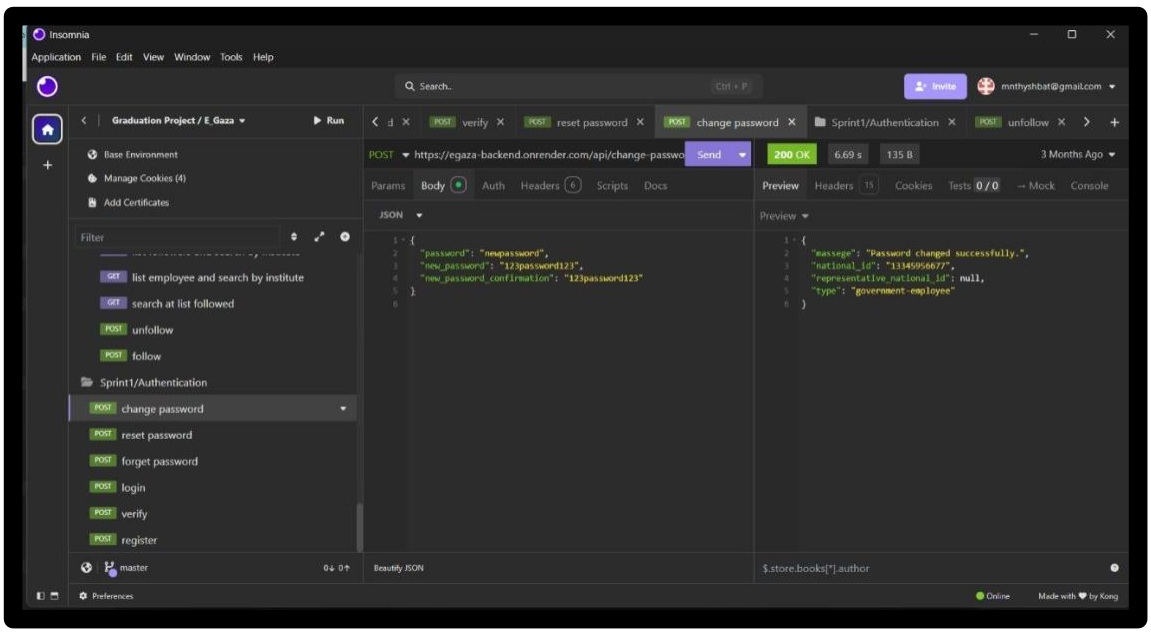


Figure E.5: Change Password API Test

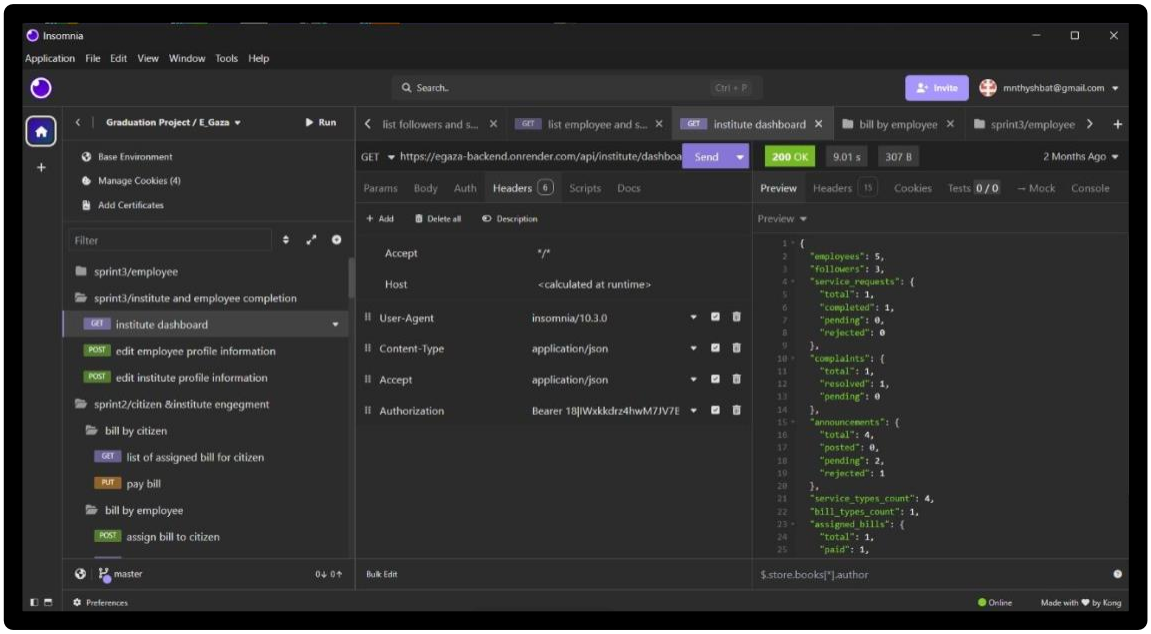


Figure E.6: Institutions Dashboard  API Test

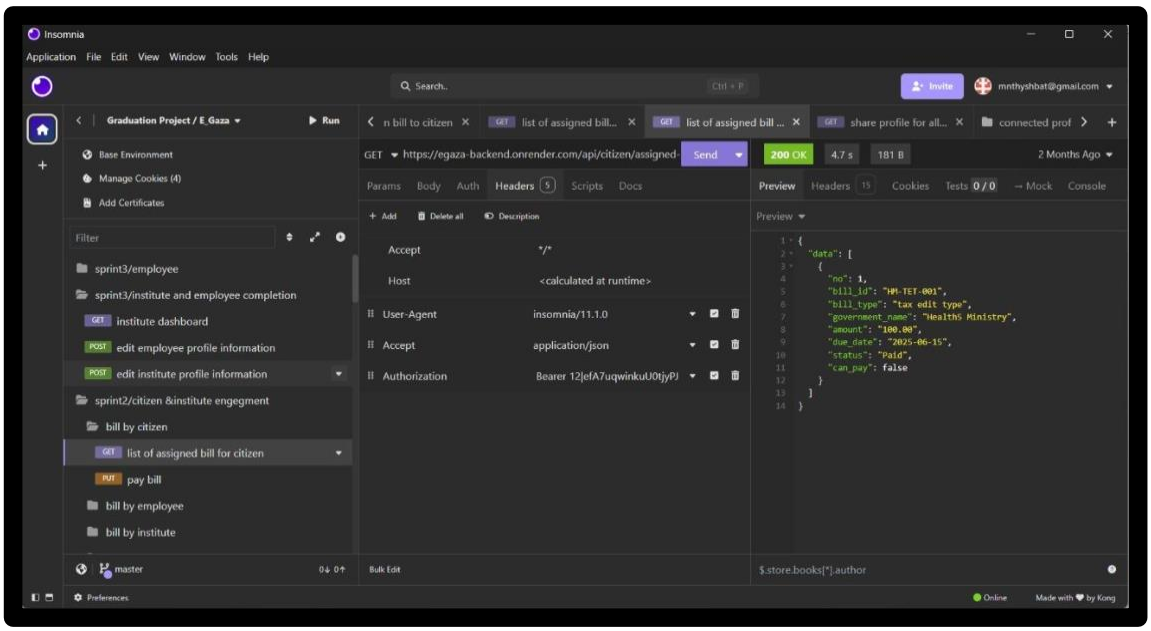


Figure E.7: List Of Bills For Citizen  API Test

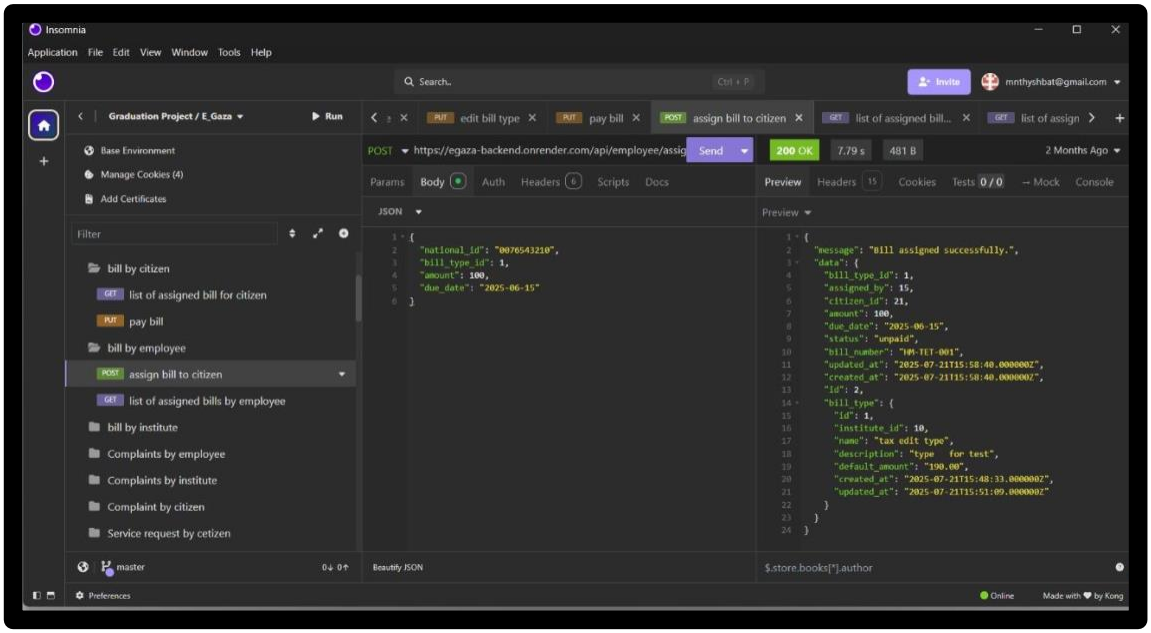


Figure E.8 :Assigned Bills To Citizen By Employee API Test

There are +70 APIs , all were tested and documented locally and after deployment to Render .

- API Testing File Locally: https://github.com/GraduationProject-eGaza/eGaza-docs/blob/main/eGaza-insomnia.json
- API Testing File on Render: https://github.com/GraduationProjecteGaza/eGazadocs/blob/main/Insomnia_Test_AfterDeploy.yaml

## Appendix F: Future Enhancements

Although the current version of the project works as a prototype, there are many possible enhancements that can be implemented in the future:

1. Admin dashboard for full system management.
2. Communication channels between institutions.
3. Mobile Application: Develop Android and iOS versions to increase accessibility.
4. Integration with Local Payment Gateways: Enable citizens to pay bills through platforms like Jawwal Pay or PalPay.
5. AI Complaint Classification: Use artificial intelligence to automatically categorize complaints and assign them to the right employee.
6. Multilingual Support: Add additional languages such as Arabic and Hebrew.

## REFERENCES

[1] Palestinian Central Bureau of Statistics (PCBS), Press Release on the Conditions of the Palestinian People, 2022. [Online]. Available: http://www.pcbs.gov.ps

[2] Gaza Electricity Distribution Company (GEDCo), Annual Report, 2022. [Online]. Available: https://www.gedco.ps

[3] World Bank, Digital Government Readiness Assessment, Washington DC: World Bank Group, 2021. [Online]. Available: https://www.worldbank.org

[4] A. Alshehri and S. Drew, "E-government principles: Implementation, avantages and challenges," International Journal of Electronic Business, vol. 9, no. 3, pp. 255–270, 2011.

[5] Smart Dubai Government. "Smart Dubai Initiative." [Online]. Available: https://smartdubai.ae

[6] Government of India. "UMANG App - Unified Mobile Application for New-age Governance." [Online]. Available: https://web.umang.gov.in

[7] Government of Kenya. "eCitizen Portal." [Online]. Available: https://www.ecitizen.go.ke

[8] M. Cohn, Succeeding with Agile: Software Development Using Scrum. Addison-Wesley, 2010.

[9] W. Royce, "Managing the development of large software systems," in Proceedings of IEEE WESCON, 1970.

[10] D. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010.

[11] A. Stellman and J. Greene, Learning Agile: Understanding Scrum, XP, Lean, and Kanban. O'Reilly Media, 2014.

[12] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide. Addison-Wesley, 2005.

[13] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Addison-Wesley, 2003.

[14] A. Dennis, B. Wixom, and R. Roth, Systems Analysis and Design. Wiley, 2018.

[15] M. Fowler, Continuous Integration: Improving Software Quality and Reducing Risk. Addison-Wesley, 2006.

[16] G. J. Myers, C. Sandler, and T. Badgett, The Art of Software Testing, 3rd ed. Wiley, 2011.

[17] M. Fowler, Continuous Integration: Improving Software Quality and Reducing Risk. Addison-Wesley, 2006.

[18] MDN Web Docs, "Web Browser Testing Guide", developer.mozilla.org. [Online]. Available: https://developer.mozilla.org