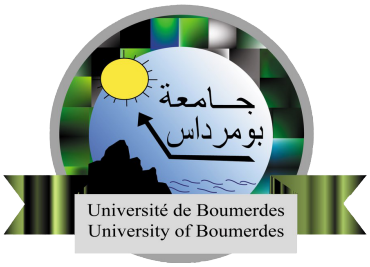
People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
M'Hamed Bouguerra University - Boumerdes



Faculty of Sciences
Department of Computer Science

Field: Mathematics and Computer Science
Branch: Computer Science
Speciality: Information Systems

Final-Year Dissertation submitted in partial fulfillment of the requirements for the Academic Bachelor's Degree

Crowdsourced Smart City Platform for Urban Problem Detection and Prioritization

Prepared by:
AICHAOUI Dalia
BELMOKHTAR Mounira
DJEMEL Tarek
DJEMMANE Bilal

Examinated by:
Dr. CHABANI Mohamed
Supervised by:
Mr. BOUDAOUD Sid Ahmed
Academic Year :
2025/2026

Dedication

We would first like to express our deepest gratitude to ALLAH Almighty for granting us the strength, courage, and patience necessary to accomplish this modest work.

We would like to extend our sincere thanks to our supervisor Mr.BOUDAOUD Sid Ahmed, for his valuable advice and unwavering support throughout this project.

We also wish to thank our examiner, Mr.CHABANI Mohamed , for accepting to examine and evaluate this work.

Our gratitude also goes to all our teachers at the Faculty of Exact Sciences of M'Hamed Bougara University of Boumerdès for the education and guidance they provided throughout our academic journey.

Finally, our most heartfelt thanks go to our parents for their help, advice, and above all, for their endless support and unwavering trust.

To everyone who contributed, directly or indirectly, to the completion of this dissertation, we express our highest appreciation.

Thank you all for everything.

Abstract

Urban areas are facing many problems related to infrastructure and public services, such as damaged roads, waste accumulation, and public lighting failures. These issues affect the daily life of citizens, and traditional reporting methods are often slow and inefficient.

This project presents UrbanFix, a web-based platform that allows citizens to report urban problems easily by providing basic information such as location, description, and images. At the same time, administrators can view and manage these reports in an organized way.

The system improves communication between citizens and authorities and helps make urban management more efficient. The results show that UrbanFix is a useful and practical solution for smart city applications.

Résumé

Les zones urbaines font face à de nombreux problèmes liés aux infrastructures et aux services publics, tels que les routes endommagées, l'accumulation des déchets et les pannes d'éclairage public. Ces problèmes affectent la vie quotidienne des citoyens, et les méthodes traditionnelles de signalement sont souvent lentes et inefficaces.

Ce projet présente UrbanFix, une plateforme web qui permet aux citoyens de signaler facilement les problèmes urbains en fournissant des informations de base telles que la localisation, la description et des images. En même temps, les administrateurs peuvent consulter et gérer ces signalements de manière organisée.

Le système améliore la communication entre les citoyens et les autorités et aide à rendre la gestion urbaine plus efficace. Les résultats montrent que UrbanFix est une solution utile et pratique dans le domaine des villes intelligentes.

Contents

Dedication **I**

Abstract **II**

Résumé **II**

List of Figures **IV**

List of Tables **V**

List of Abbreviations **V**

General Introduction **1**

1 Requirements Analysis **2**
1.1 Problem Statement 2
1.2 Objectives of the Project 2
1.3 Proposed Solution 2
1.4 System Actors 3
1.4.1 Citizen 3
1.4.2 City Worker 3
1.4.3 Administrator 3
1.5 Functional Requirements 3

2 System Design **5**
2.1 Use Case Diagram 5
2.2 Sequence Diagram 7
2.3 Class Diagram 14
2.4 MCD 15

3 Implementation **16**
3.1 Technologies Used 16
3.1.1 Front-end 16
3.1.2 Back-end 17
3.1.3 Database 17
3.1.4 Development Tools 17

3.1.5 Documentation Tool 18
3.2 System Development 18
3.3 Visual Identity and Branding 18
3.3.1 The Logo 19
3.3.2 Color Palette 19
3.4 User Interface 19
3.4.1 Citizen Interface 19
3.4.2 Admin Interface 22
3.4.3 City Worker Interface 23

General Conclusion **25**

List of Figures

2.1 Use Case Diagram 6
2.2 Authentification 7
2.3 Report a problem 9
2.4 Track reports 11
2.5 Processes a report 13
2.6 Class Diagram 15
2.7 MCD 15

3.1 Technologies used for Front-end 16
3.2 Technologies used for Back-end 17
3.3 Technologies used for Database 17
3.4 Technologies used in development tools 18
3.5 Technologies used for documentation tool 18
3.6 Official logo of the platform. 19
3.7 Color Palette 19
3.8 report a problem(Location) 20
3.9 report a problem (Upload Picture) 20
3.10 report a problem (Submit) 21
3.11 Explore Reports 21
3.12 Admin Dashboard 22
3.13 Admin Dashboard(Staff management) 22
3.14 change reports Status 23

3.15 Worker Dashboard 24

List of Tables

1.2 Functional Requirements of the System 4

2.1 Authentification (Admin, city worker, citizen). 8
2.2 Report a problem (citizen). 10
2.3 Track my reports (citizen). 12
2.4 Processes a report (city worker). 14

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CSS	Cascading Style Sheets
DBMS	Database Management System
HTML	HyperText Markup Language
JS	JavaScript
MCD	Modèle Conceptuel de Données (Conceptual Data Model)
PHP	Hypertext Preprocessor
REST	Representational State Transfer
SQL	Structured Query Language
UML	Unified Modeling Language
UI	User Interface
UX	User Experience

General Introduction

Nowadays, cities are experiencing rapid growth due to urbanization and technological development. As urban populations continue to increase, managing public services and urban infrastructure has become a major challenge for local authorities. Modern cities must ensure efficient management of roads, waste collection, public lighting, water distribution, and many other essential services that directly affect citizens' quality of life.

In recent years, the concept of Smart Cities has emerged as an important approach for improving urban management through digital technologies. Smart city solutions aim to facilitate communication, optimize public services, and encourage citizen participation in the management of urban environments. Among these solutions, web and mobile platforms play a significant role in collecting and processing information related to urban problems in real time.

Citizen participation is considered a key element in the success of smart city systems. Citizens are the first to observe issues such as damaged roads, waste accumulation, water leaks, or lighting failures. However, traditional reporting methods are often inefficient, slow, and poorly organized, which can delay interventions and reduce the effectiveness of urban management.

To address these limitations, this project proposes UrbanFix, a web-based platform dedicated to urban issue reporting and management. The system allows citizens to report problems by providing descriptions, images, and geographical locations, while enabling administrators and city workers to manage reports efficiently through a centralized platform. The proposed solution aims to improve communication between citizens and local authorities and contribute to more efficient urban management.

This report is organized into four chapters. The first chapter presents the requirements analysis and the identification of system needs. The second chapter describes the system design using UML models and database structures. The third chapter explains the implementation technologies and developed interfaces. Finally, the last chapter presents the testing phase and the obtained results.

Chapter 1

Requirements Analysis

This chapter presents the requirements analysis of the UrbanFix system, including the project objectives, the proposed solution, and the main system requirements and actors. It provides the foundation for the system design and implementation.

1.1 Problem Statement

Urban areas face many challenges related to infrastructure and public service management. Problems such as damaged roads, waste accumulation, water leaks, and public lighting failures are common in many cities. These issues negatively affect the quality of life of citizens and require quick and efficient solutions. However, traditional methods used to report these problems are often slow, unstructured, and inefficient. Citizens usually rely on phone calls or physical visits to report issues, which may lead to delays, loss of information, or lack of proper follow-up. In many cases, there is no clear communication between citizens and local authorities. Therefore, there is a need for a digital solution that simplifies the reporting process, improves communication, and allows better management of urban issues.

1.2 Objectives of the Project

The main objective of this project is to develop a web-based platform that improves the reporting and management of urban issues. The specific objectives are:

- To simplify the process of reporting urban problems for citizens.
- To improve communication between citizens and local authorities.
- To ensure better organization and tracking of reported issues.
- To provide an efficient way to manage and monitor urban problems.

1.3 Proposed Solution

To achieve these objectives, a web-based system called UrbanFix is proposed. The platform is designed with a user-friendly interface that allows citizens to report urban issues

by providing key information such as location, description, category, and images. The system also provides an administrative interface that allows administrators to view reported issues, manage them efficiently, and update their status when necessary.

This approach ensures better organization of information and improves the efficiency of urban issue management while offering a practical solution for smart city environments.

1.4 System Actors

The UrbanFix platform involves three primary actors and one secondary actor. Each of them interacts with the system according to a specific role within the urban issue reporting and management process.

1.4.1 Citizen

The citizen is the principal actor of the UrbanFix platform. This actor uses the system to report urban issues encountered in the city, such as damaged roads, water leaks, or public lighting failures. Furthermore, the citizen can track the progress of submitted reports and consult other public reports available on the platform.

1.4.2 City Worker

The city worker is an operational actor of the UrbanFix platform. This actor is responsible for handling assigned reports, reviewing their details, updating their processing status, and adding remarks related to the intervention process

1.4.3 Administrator

The administrator is a supervisory actor of the UrbanFix platform. This actor is responsible for managing users, supervising reported issues, and ensuring the overall proper functioning of the system.

1.5 Functional Requirements

This section presents the main functional requirements of the UrbanFix platform according to the different actors interacting with the system.

Actor	Functional Requirements
Citizen	• Create an account. • Securely log into the platform. • Submit a report concerning an urban issue. • Fill in a reporting form. • Upload a photo related to the reported issue. • Specify the location of the issue on the map. • Provide a textual description of the problem. • Track the status of submitted reports. • View the history of submitted reports. • Consult other public reports available on the platform. • Filter public reports by area and severity.
City Worker	• Securely authenticate to access role-specific functionalities. • Consult the list of reports assigned by the administrator. • View report details including title, description, location, photo, category, severity, submission date, and status. • Open a specific report to access complete information. • Monitor the progress of assigned reports. • Update the status of reports (pending, in progress, resolved, etc.). • Add remarks or observations related to interventions. • Consult the history of updates associated with a report. • Manage assigned reports until resolution. • Ensure structured follow-up of interventions for better coordination and traceability.
Administrator	• Securely authenticate to access administrative functionalities. • Manage user accounts within the platform. • Add new city workers to the system. • Update user information when necessary. • Deactivate, ban, or remove users according to platform rules. • Consult all reports submitted through the platform. • View detailed information for each report. • Supervise the overall progress of reported issues. • Assign reports to city workers according to responsibilities. • Update the status of reports when necessary. • Monitor platform activity and coordination between actors. • Access dashboards and statistical data related to reports and users. • Ensure the overall proper functioning of the platform.

Table 1.2: Functional Requirements of the System

Chapter 2

System Design

This chapter presents the design of the UrbanFix system by translating the identified requirements into a structured technical model. It describes the system architecture, database design, and the main UML diagrams used to illustrate the system's behavior, structure, and interactions, providing a solid foundation for the implementation phase.

2.1 Use Case Diagram

This use case diagram represents the interaction between the main actors of the UrbanFix system and its functional modules. It illustrates the different services offered by the platform, including reporting urban issues, managing reports, and system administration.

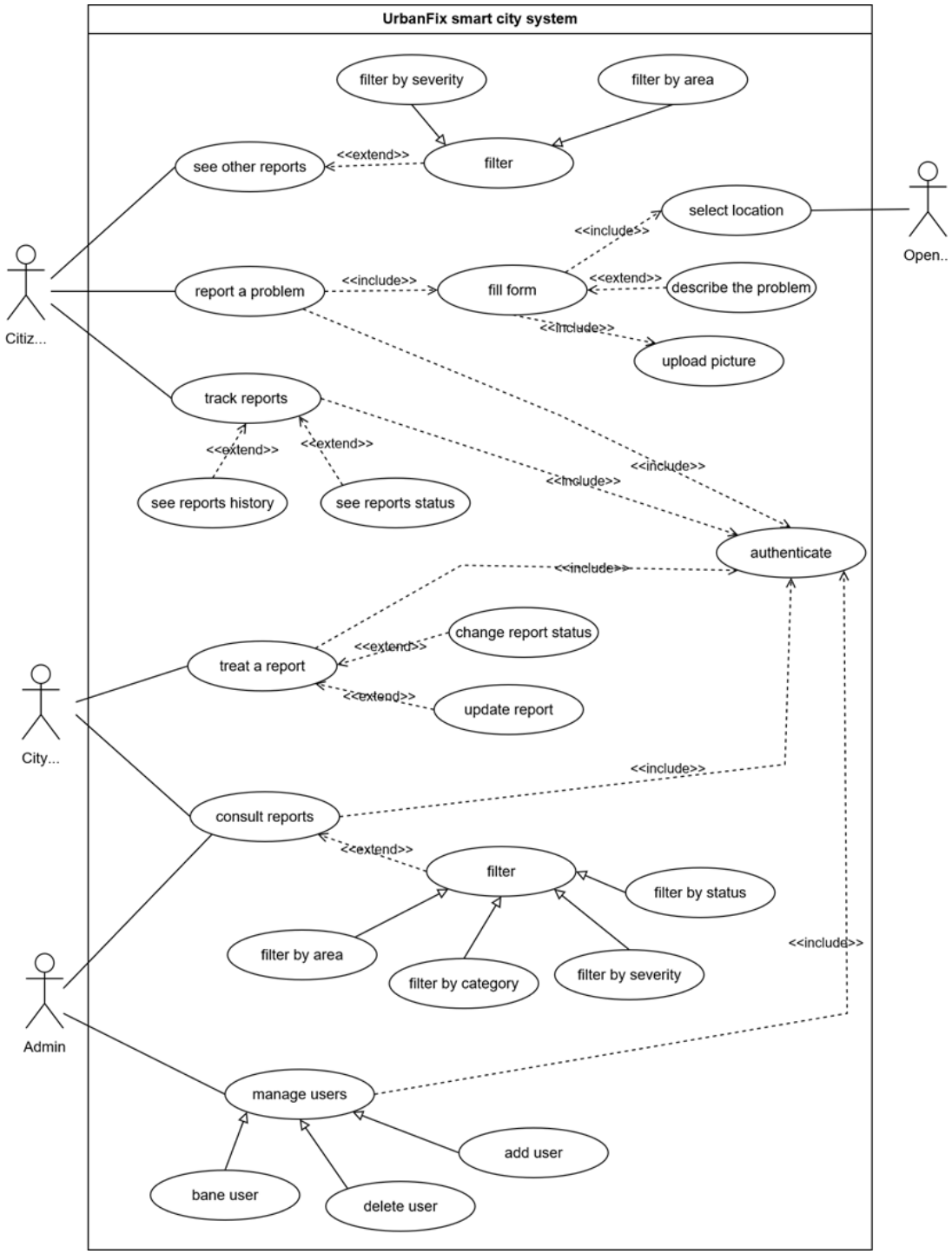


Figure 2.1: Use Case Diagram

2.2 Sequence Diagram

Authentification (Admin , city worker , citizen).

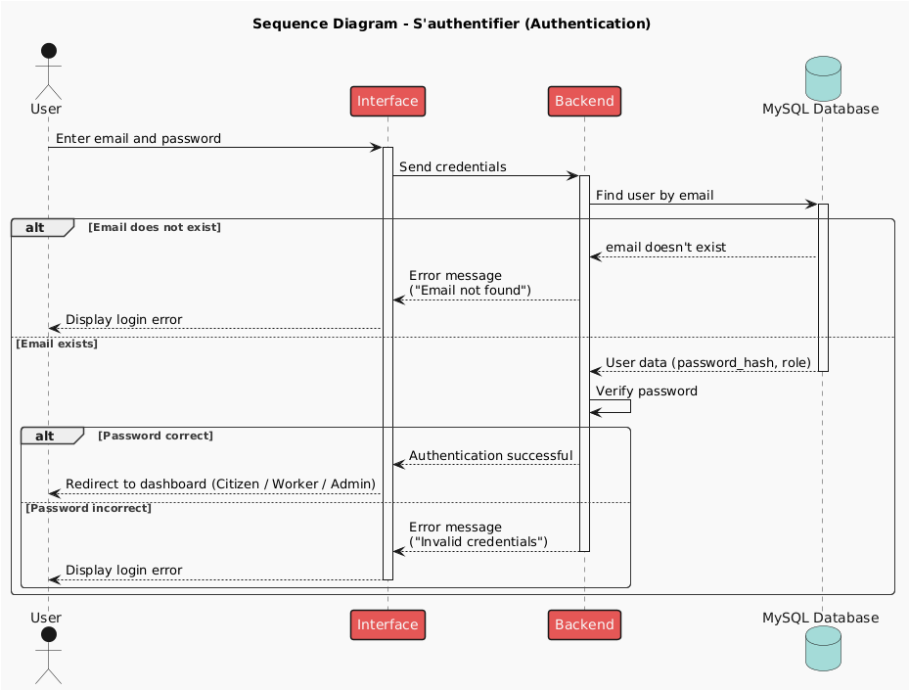


Figure 2.2: Authentification

Authentification (Admin, city worker, citizen).

Titre	Authentification
Actors	Citizen , Admin , City Worker
Objectif	To secure the access to the system , the user must authenticate themselve to access the platform.
Pre-condition	/
Post-condition	access to the sytem

Nominal scenario	1. User enters credentials: User inputs email and password in the interface. 2. Interface sends credentials to backend: React sends these credentials to the API. 3. Backend looks up user: API queries the database for the user by email. 4. User exists: Database returns user data (password hash, role). 5. Password verification: API checks if the entered password matches the stored hash. 6. Successful authentication: Password is correct, API confirms success. 7. Interface redirects user: React redirects the user to the dashboard according to their role (Citizen, Worker, Admin).
Alternative scenario	• A1: The scenario starts at step 4 of the nominal scenario (API queries database). • 1.The database returns “email not found”. • 2.The backend sends an error message to the interface.

Table 2.1: Authentification (Admin, city worker, citizen).

Report a problem (citizen).

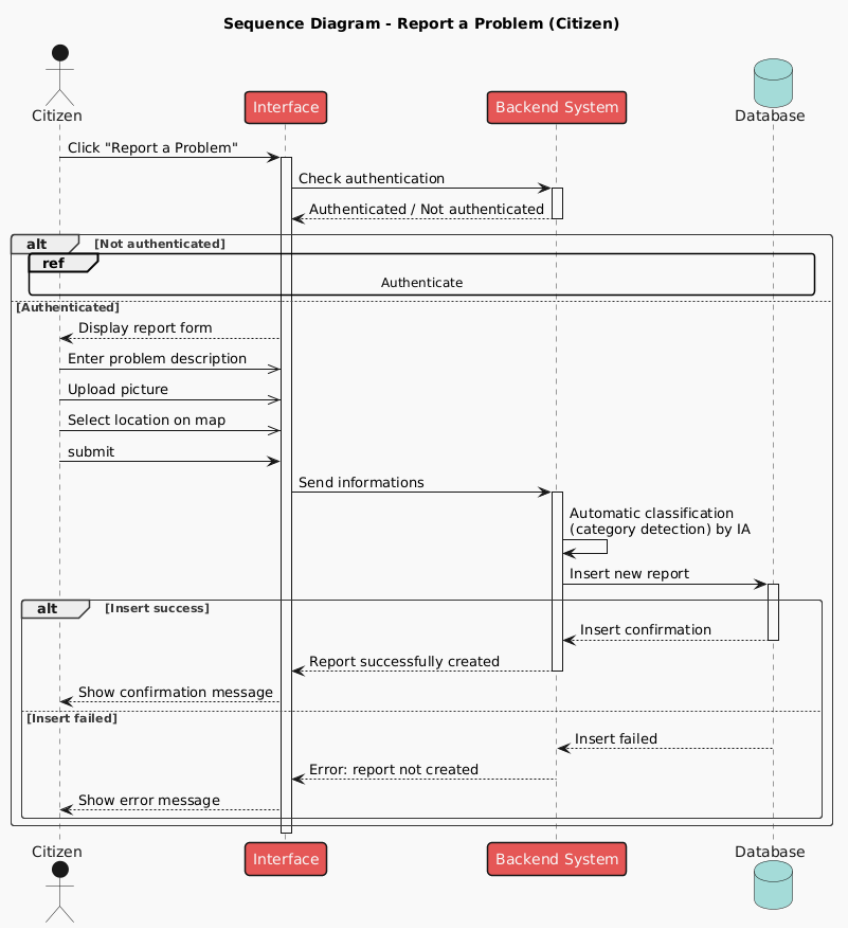


Figure 2.3: Report a problem

Report a problem (citizen).

Titre	Report a problem.
Actors	Citizen.
Objectif	to let the citizem to report a problm urban in the platform.
Pre-condition	authentification
Post-condition	ensure successful delivery of the report.

Nominal scenari	1. The citizen accesses the platform. 2. The citizen clicks "Report a Problem". 3. The system checks whether the citizen is authenticated. 4. The system displays the report form. 5. The citizen enters the problem description, uploads a picture, selects the location, then submits the report. 6. The interface sends the report information to the backend system. 7. The backend automatically classifies the report (category and severity level) using AI. 8. The backend inserts the new report into the database. 9. The system confirms that the report was successfully created.
Alternative scenario	• A1. At step 3, if the citizen is not authenticated,the system redirects the citizen to the "Authenticate" use case. • A2. After successful authentication, the scenario resumes at step 4. • B1. At step 8, the database fails to insert the report. • B2. The backend system returns an error message to the interface. • B3. The interface displays an error message to the citizen.

Table 2.2: Report a problem (citizen).

Track reports (citizen)

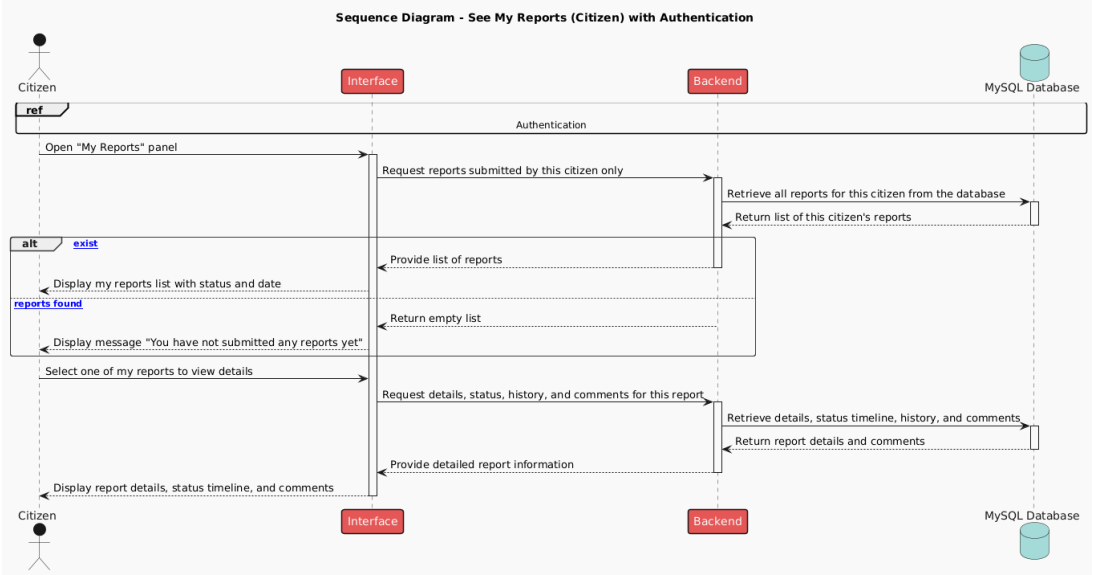


Figure 2.4: Track reports

Track my reports (citizen)

Titre	Report a problem.
Actors	city worker.
Objectif	trait the assigned reports and update the state of the reports.
Pre-condition	authentification as city worker.
Post-condition	ensure fixing the problm in the report assinged.

Nominal scenari	1. Citizen opens the “My Reports” panel. 2. Interface sends a request to the Backend for reports submitted by this citizen only. 3. Backend requests the citizen’s reports from the database. 4. Database returns the list of reports to the Backend. 5. Backend sends the reports list to the Interface. 6. Interface displays the list of reports with status and date to the Citizen. 7. Citizen selects one of their reports to view details. 8. Interface requests details, status timeline, history, and comments for the selected report from the Backend. 9. Backend retrieves the report details from the database. 10. Database returns the report details and comments to the Backend. 11. Backend sends the detailed report information to the Interface. 12. Interface displays the report details, status timeline, and comments to the Citizen.
Alternative scenario	• A1. Citizen has not submitted any reports • 1. At step 5, if the citizen has no reports, the Backend sends an empty list to the Interface. • 2. Interface displays the message:“You have not submitted any reports yet.”

Table 2.3: Track my reports (citizen).

Processes a report (city worker).

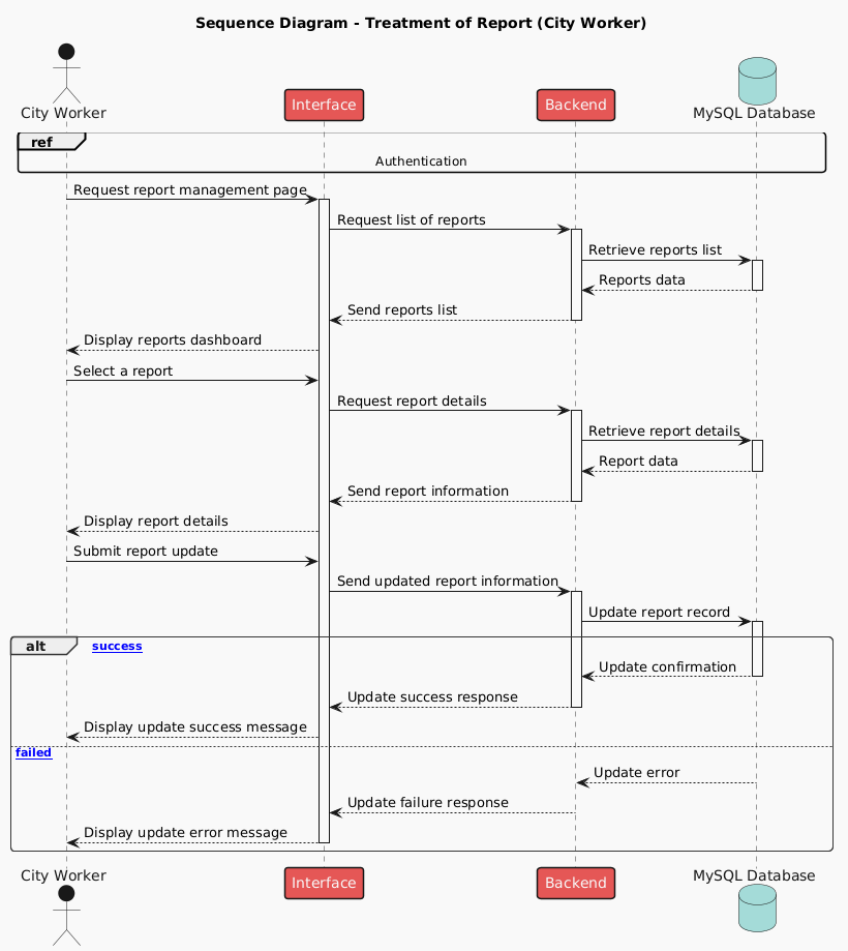


Figure 2.5: Processes a report

Processes a report (city worker).

Titre	Processes a report.
Actors	city worker.
Objectif	trait the assigned reports and update the state of the reports.
Pre-condition	authentification as city worker.
Post-condition	ensure fixing the problm in the report assinged.

Nominal scenario	1. The City Worker accesses the report management page. 2. The system requests the list of reports from the backend. 3. The backend retrieves the reports list from the database. 4. The database returns the reports data to the backend. 5. The system displays the reports dashboard to the City Worker. 6. The City Worker selects a report. 7. The system requests the report details from the backend. 8. The backend retrieves the report details from the database. 9. The database returns the report information to the backend. 10. The system displays the report details to the City Worker. 11. The City Worker submits an update to the report. 12. The system sends the updated report information to the backend. 13. The backend updates the report record in the database. 14. The database confirms the update. 15. The system displays a success message confirming that the report has been updated.
Alternative scenario	• A: The scenario starts at step 6. If there's no new assinged report. • 1. the system displays an empty report message. • 2. the system return to dashboard page.

3

Table 2.4: Processes a report (city worker).

2.3 Class Diagram

The class diagram of UrbanFix presents the main entities and their relationships, including User, Report, Category, Photo, Comment, and Assignment. It shows how reports are created, categorized, and managed, as well as how users interact with the system. This diagram provides a clear structural view and serves as a basis for the database design. Class Diagram

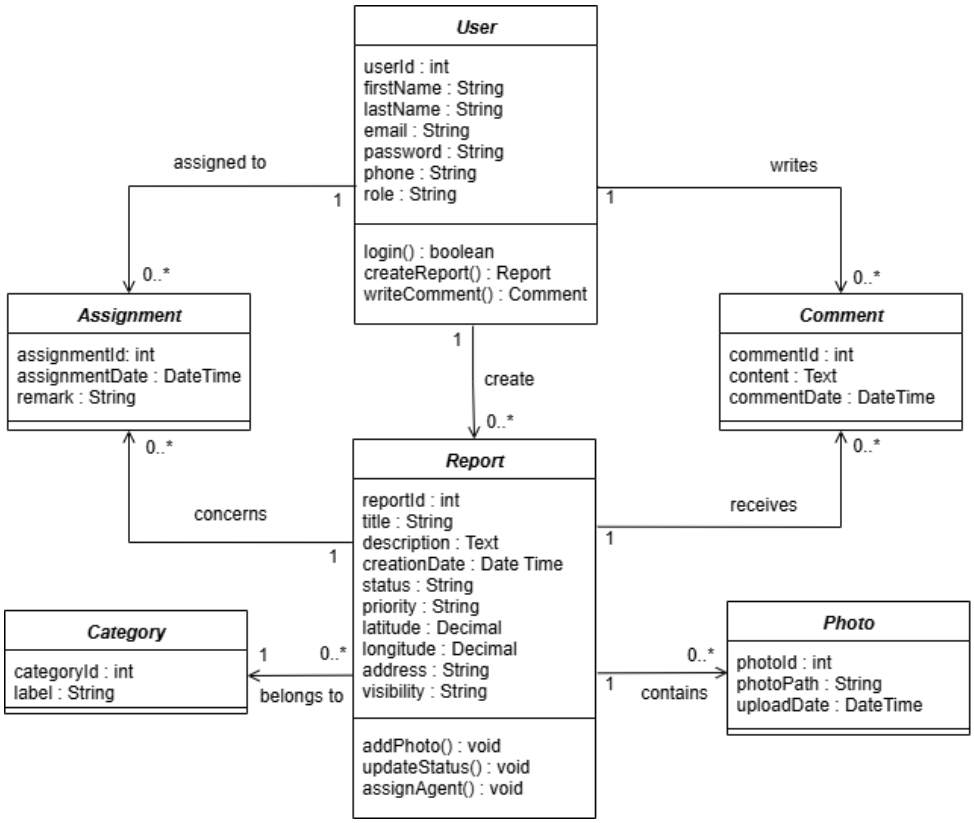


Figure 2.6: Class Diagram

2.4 MCD

The MCD (Conceptual Data Model) of UrbanFix describes the main entities and their relationships, providing a foundation for database design.

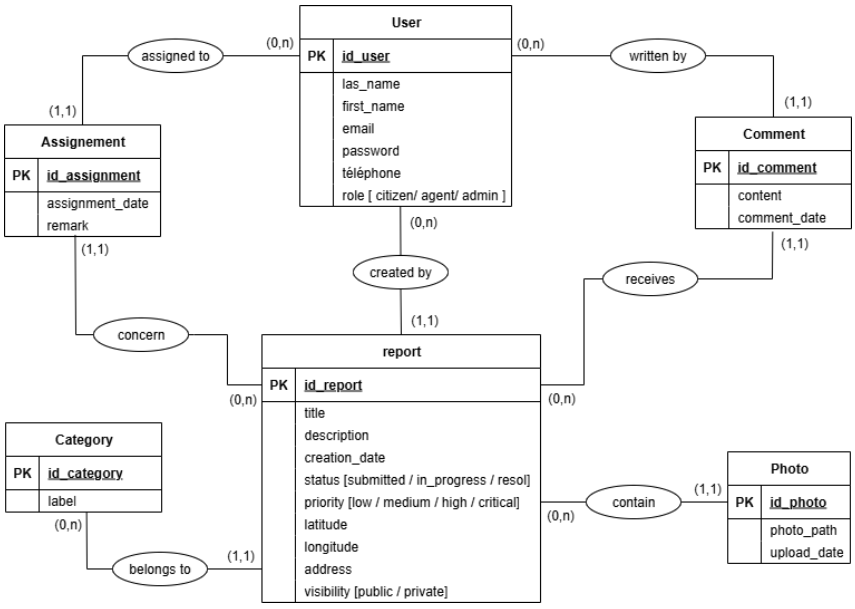


Figure 2.7: MCD

Chapter 3

Implementation

This chapter presents the implementation of the UrbanFix platform, including the technologies, system architecture, and user interfaces developed to build a functional and efficient urban issue reporting system.

3.1 Technologies Used

3.1.1 Front-end

This project was developed using a combination of modern technologies to ensure performance, scalability, and a smooth user experience.

Front-end Technologies: The front-end of the application was developed using React.js, a JavaScript library that enables the creation of dynamic and interactive user interfaces through reusable components [3]. To improve development speed and performance, Vite was used as a modern build tool. It provides fast compilation and an efficient development environment. The project also uses TypeScript, which introduces static typing to JavaScript. This improves code quality, reduces errors, and makes the application easier to maintain. For the design of the user interface, Tailwind CSS was used. It allows the creation of responsive and consistent layouts using utility-based classes. Additionally, several supporting libraries were integrated:

- React Router for managing navigation between pages
- React Hook Form for handling user input and forms
- Recharts for data visualization and statistics
- Leaflet for interactive maps based on OpenStreetMap



(a) React (b) Typescript (c) tailwindcss

Figure 3.1: Technologies used for Front-end

3.1.2 Back-end

The back-end is based on a hybrid architecture combining FastAPI and Laravel. FastAPI, a modern Python framework, was used to integrate artificial intelligence functionalities through RESTful APIs. It handles high-performance endpoints and enables automatic classification of incoming reports by category and severity level using AI models, improving automation and reducing manual processing.

Laravel, a PHP framework, was used to manage the core back-end structure of the application. It handles authentication, routing, controllers, application components, database management, and communication between the front-end and the database through RESTful APIs [9, 7].



(a) Laravel (b) FastAPI

Figure 3.2: Technologies used for Back-end

3.1.3 Database

MySQL was used as the relational database management system. It stores all application data, including user information, reports, categories, and status updates[6]. Its reliability and structured nature make it suitable for managing large amounts of data.



Figure 3.3: Technologies used for Database

3.1.4 Development Tools

Several tools were used to support the development process: Visual Studio Code [4] as the main development environment Git [10] for version control and source code management Postman [8] for testing API endpoints DrawSQL [1]for designing and visualizing the database structure Node.js[5] for managing front-end dependencies and running development tools Browser Developer Tools for debugging and testing the application



(a) VS Code



(b) Node.js



(c) Git

Figure 3.4: Technologies used in development tools

3.1.5 Documentation Tool

LaTeX was used to write and format the project report. It is widely used for academic and technical documentation due to its ability to produce well-structured and professional documents[2].



(a) Overleaf



(b) LaTeX

Figure 3.5: Technologies used for documentation tool

3.2 System Development

The UrbanFix platform was developed using a client-server architecture to ensure efficient communication between the front-end and the back-end. The front-end was built with React.js, TypeScript, and Tailwind CSS to provide a responsive and user-friendly interface. The back-end was implemented using FastAPI and Laravel to manage APIs, business logic, and server-side operations. The system includes several functionalities such as user authentication, issue reporting, report tracking, worker assignment, and dashboard management. OpenStreetMap and Leaflet were integrated for geolocation services, while MySQL was used for database management. Development and testing were supported using tools such as Git, Postman, and Visual Studio Code. The result is a functional and efficient platform that simplifies urban issue reporting and management.

3.3 Visual Identity and Branding

To establish a unique and recognizable identity for our platform, we have designed specific visual guidelines. This visual identity aims to provide a user-friendly experience while reflecting the technological and participatory (crowdsourcing) aspects of the project.

3.3.1 The Logo



Figure 3.6: Official logo of the platform.

Significance of the logo elements:

- **The Location Pin:** Represents geolocation, mapping, and the core reporting functionality.
- **The City Skyline:** Signifies the urban environment and smart city infrastructure.
- **The Circular Shape:** Symbolizes community connectivity and collaborative crowdsourcing.

3.3.2 Color Palette

The colors used in the application were carefully selected to convey specific meanings related to urban management and technology:

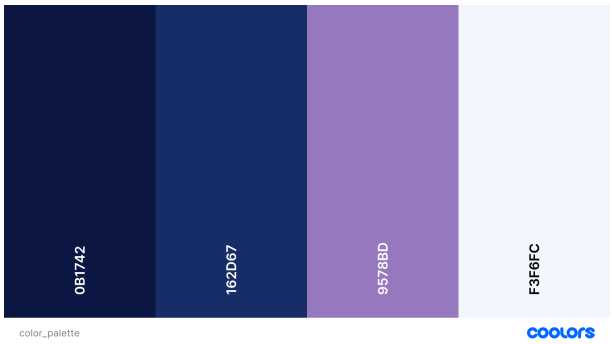


Figure 3.7: Color Palette

3.4 User Interface

3.4.1 Citizen Interface

The citizen interface is designed to provide a simple and user-friendly environment for reporting urban issues. Citizens can submit reports by selectiong location directly on the map , uploading images and providing descriptions . The interface also allows users to track the status of submitted reports and explore other public reports available on

the platform. The responsive design improves accessibility and ensures a smooth user experience across different devices.

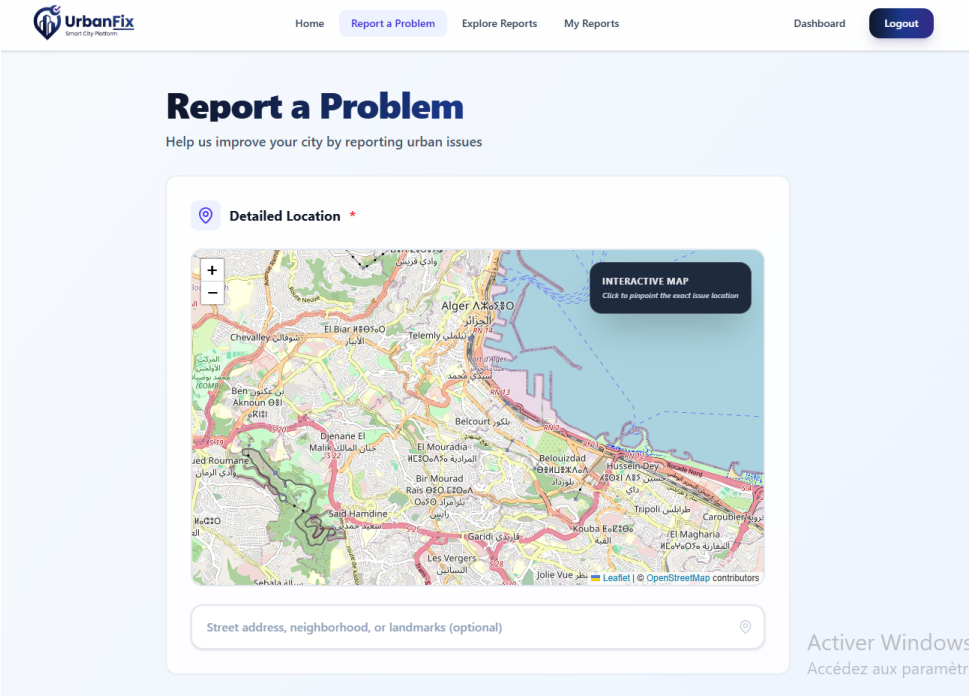


Figure 3.8: report a problem(Location)

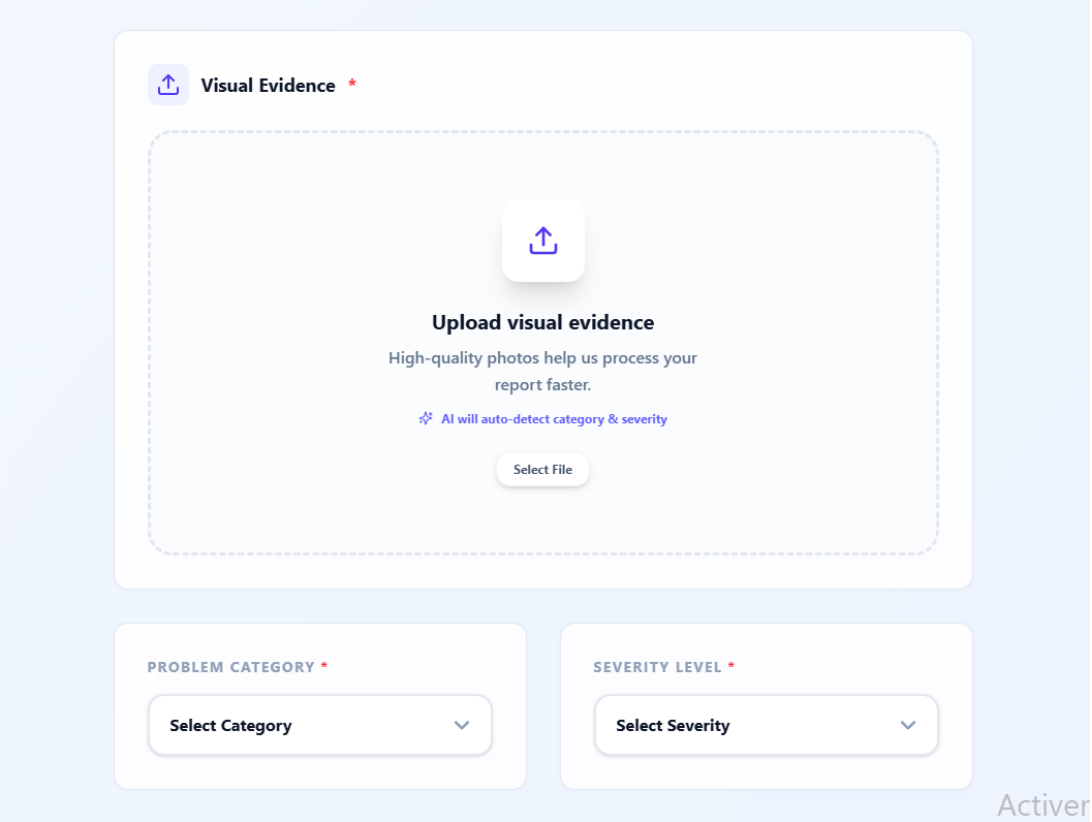


Figure 3.9: report a problem (Upload Picture)

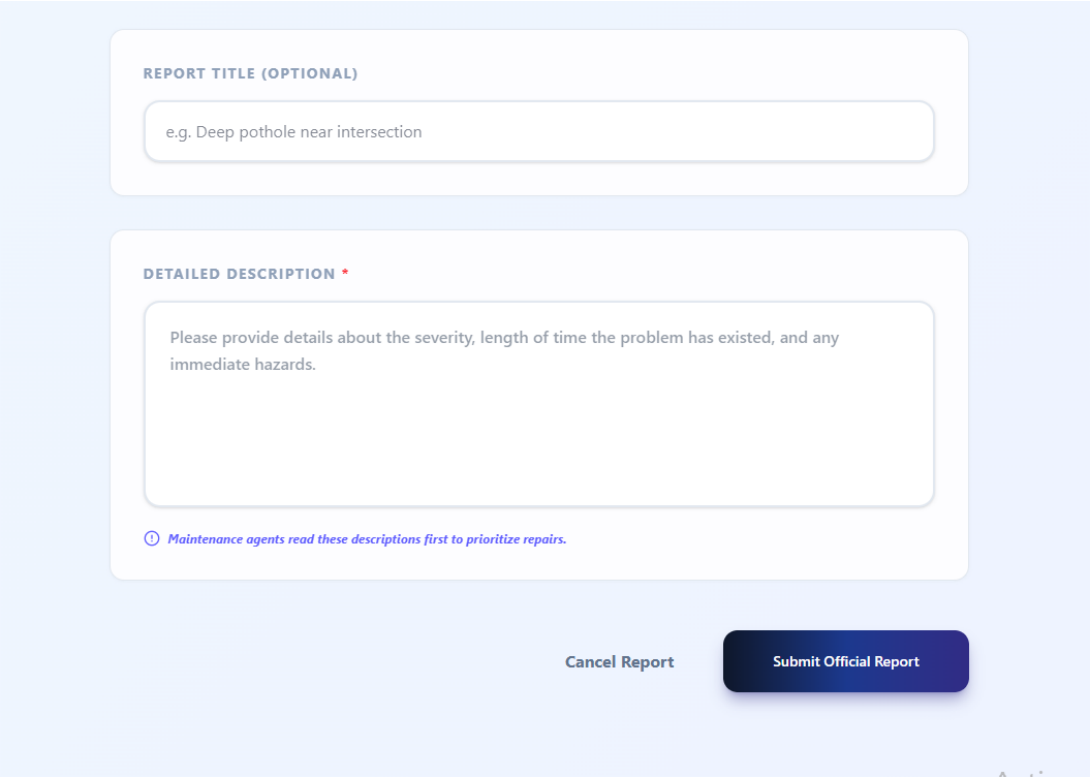


Figure 3.10: report a problem (Submit)

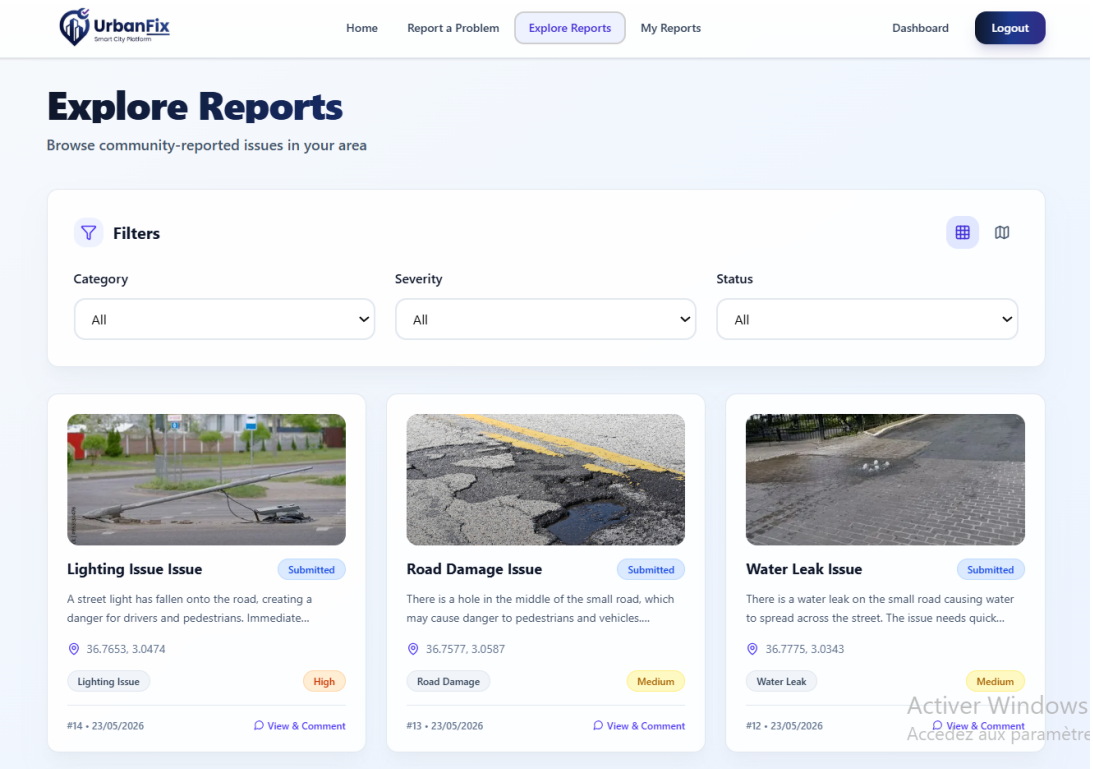


Figure 3.11: Explore Reports

3.4.2 Admin Interface

The admin interface is designed to supervise and manage all activities within the platform. Administrators can monitor submitted reports, manage users and city workers, assign tasks, and update report statuses. The dashboard also provides statistical information and visual indicators that help improve coordination and decision-making during urban issue management.

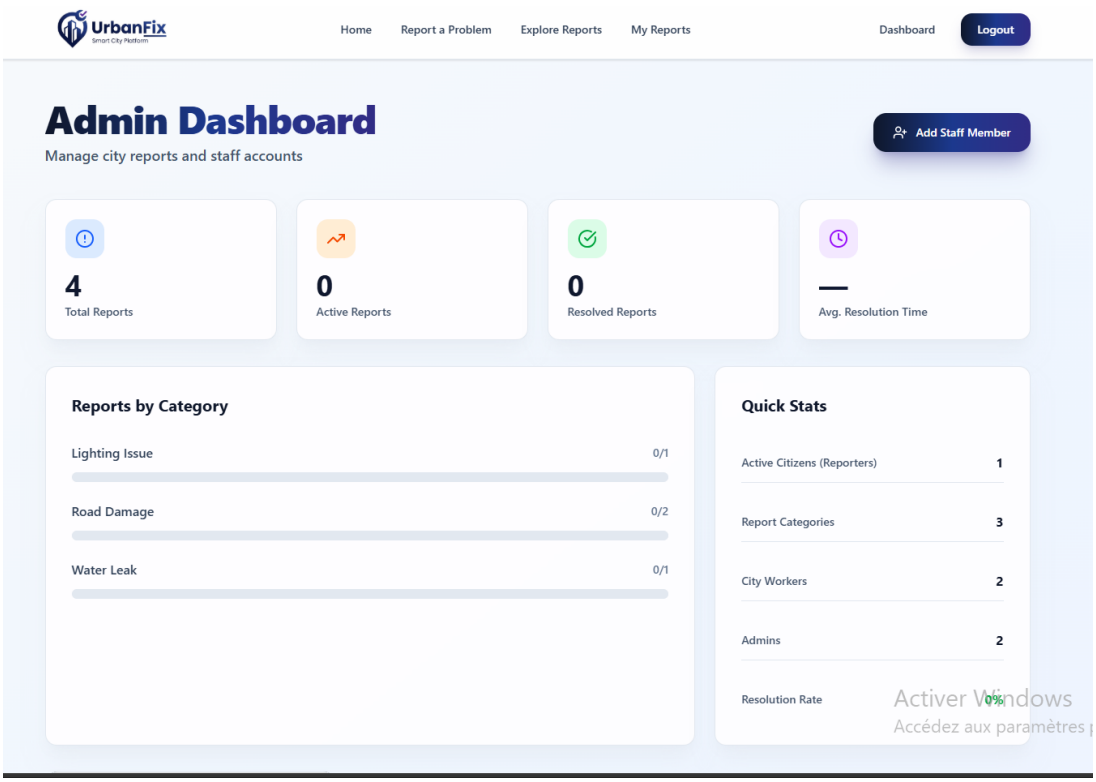


Figure 3.12: Admin Dashboard

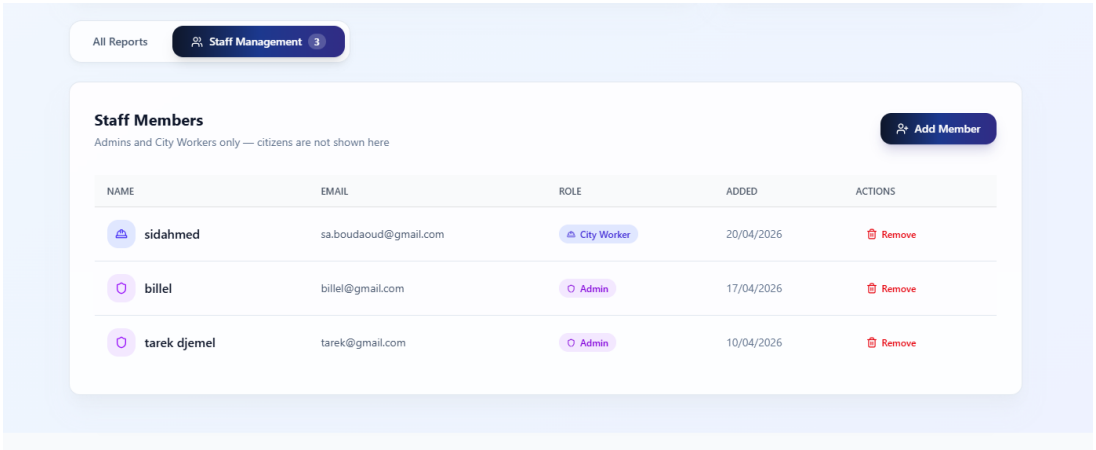


Figure 3.13: Admin Dashboard(Staff management)

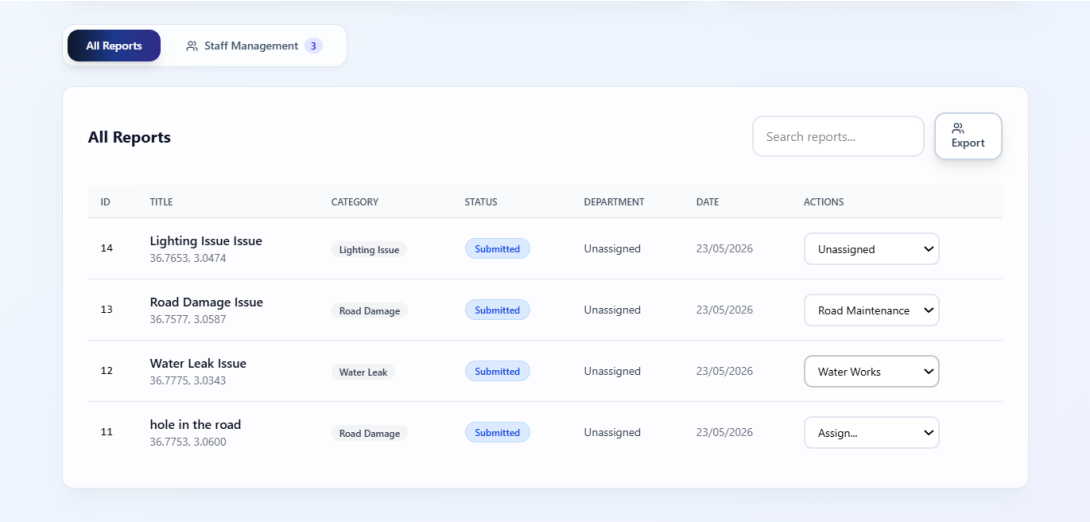


Figure 3.14: change reports Status

3.4.3 City Worker Interface

The city worker interface is designed to help field workers manage assigned urban issues efficiently. Workers can access report details, monitor task progress, and update report statuses during the intervention process. The interface also allows workers to add remarks and follow the history of updates related to each assigned report.

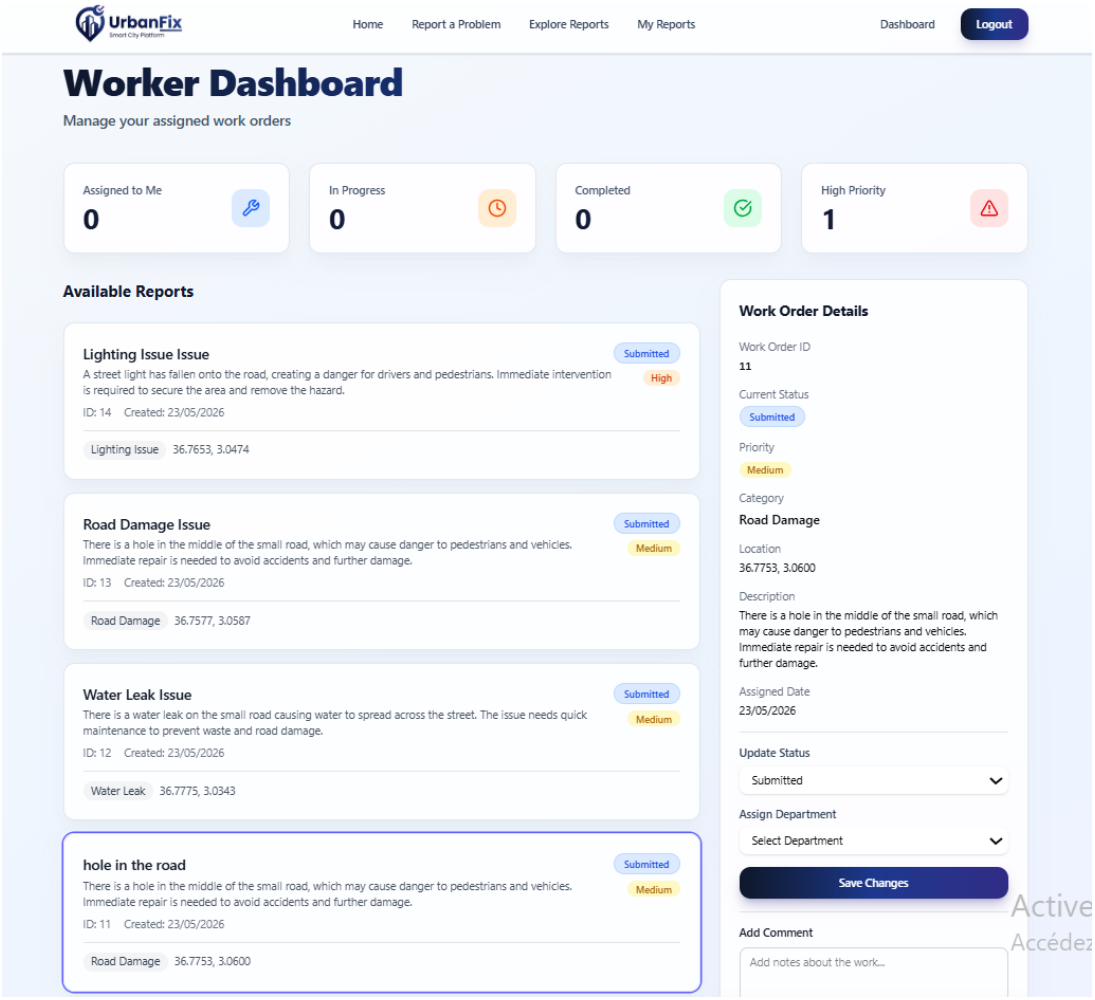


Figure 3.15: Worker Dashboard

General Conclusion

The rapid growth of urban areas and the increasing complexity of public service management have created new challenges for modern cities. In this context, smart city technologies play an important role in improving communication between citizens and local authorities and in providing more efficient urban management solutions.

The main objective of this project was to design and develop UrbanFix, a web-based platform dedicated to urban issue reporting and management. The system allows citizens to report problems such as damaged roads, waste accumulation, water leaks, and lighting failures by providing descriptions, images, and locations. At the same time, administrators and city workers can manage and monitor these reports through a centralized platform.

Throughout this project, several phases were completed, including requirements analysis, system design using UML diagrams and database modeling, and the implementation of the application using modern technologies such as React.js, FastAPI, Laravel, MySQL, and Tailwind CSS. The developed platform demonstrates how digital technologies can contribute to improving urban management and encouraging citizen participation in smart city environments.

This project also allowed us to strengthen our technical and practical skills in web development, database management, system design, and teamwork. Although the obtained results are satisfactory, the platform can still be improved in the future through the integration of a mobile application, real-time notifications, and more advanced artificial intelligence features.

In conclusion, UrbanFix represents a practical and useful solution for urban issue management and contributes to the development of smarter and more connected cities.

Bibliography

[1] DrawSQL. *DrawSQL - Database Schema Visualization and Collaboration Tool*. https://drawsql.app/. Accessed: 2026-05-17. 2026.

[2] LaTeX Project Team. *LaTeX Project*. https://www.latex-project.org/. Accessed: 2026-05-17. 2026.

[3] Meta Platforms, Inc. *React: A JavaScript Library for Building User Interfaces*. https://react.dev/. Accessed: 2026-05-17. 2026.

[4] Microsoft Corporation. *Visual Studio Code*. https://code.visualstudio.com/. Accessed: 2026-05-17. 2026.

[5] OpenJS Foundation. *Node.js JavaScript Runtime*. https://nodejs.org/. Accessed: 2026-05-17. 2026.

[6] Oracle Corporation. *MySQL Database Management System*. https://www.mysql.com/. Accessed: 2026-05-17. 2026.

[7] Taylor Otwell. *Laravel - The PHP Framework for Web Artisans*. https://laravel.com/. Accessed: 2026-05-17. 2026.

[8] Postman, Inc. *Postman API Platform*. https://www.postman.com/. Accessed: 2026-05-17. 2026.

[9] Sebastián Ramírez. *FastAPI Framework*. https://fastapi.tiangolo.com/. Accessed: 2026-05-17. 2026.

[10] Linus Torvalds and Junio Hamano. *Git Distributed Version Control System*. https://git-scm.com/. Accessed: 2026-05-17. 2026.