esprit

Se former autrement

2022 - 2023

GRADUATION PROJECT

NATIONAL ENGINEERING DEGREE

SPECIALTY : INFORMATION TECHNOLOGY

HACKINI PLATFORM

By: HOUSSEM BEN MABROUK

Academic supervisor: ***AZIZA ZAOUGA***

Corporate Internship Supervisor: ***FEDI NAIMI***

DEDICATION

Praise to God, who has given me the strength, wisdom, and perseverance to carry out this project successfully.

To my dear parents, whose unwavering support, love, and encouragement have allowed me to move forward and achieve my goals. Their trust in me has been an invaluable source of motivation.

To my dear sister, your love, support, and encouragement have been a guiding light throughout my journey. This work is a testament to my gratitude toward you. Thank you for being there during the most decisive moments.

To my family and friends, for their presence and support throughout this adventure.

To **all of you**,

I dedicate this work.

HOUSSEM BEN MABROUK

ACKNOWLEDGEMENTS

rapport-style

I would like to sincerely thank **the ELEPZ'IA team** for their warm welcome, availability, and valuable feedback, which allowed me to enrich my knowledge. I also dedicate this work to my supervisors, **Mr. Fedi Naimi** , who guided, advised, and supported me throughout the process required to obtain my degree.

I would also like to thank **Mrs. AZIZA ZAOUGA**, my supervisor, for her valuable guidance, insightful advice, and continuous support throughout this project. Her expertise and guidance were of great help in successfully completing this work.

I am equally grateful to my professors at ESPRIT for the quality of their teaching and their support throughout my university studies.

Finally, a big thank you to all the people who, directly or indirectly, contributed to the completion of this project.

TABLE OF CONTENTS

LIST OF FIGURES

GENERAL INTRODUCTION

In a world where technological innovation is advancing at a breathtaking pace, technology competitions play a crucial role in skill development and fostering enthusiasm among enthusiasts and professionals in the field. They provide a stimulating environment where participants can test their knowledge, face challenges, and enhance their problem-solving abilities.

In this context, the company **Elepzia** is committed to developing **Hackini**, an innovative solution for organizing technology competitions. It aims to improve the participant experience by ensuring efficient challenge management, rigorous and professional evaluation of submissions that guarantees accurate and reliable results, and collaborative management through integrated tools such as an **Interactive Whiteboard** that allows teams to plan, brainstorm, and visualize ideas in real time—thus fostering creativity and coordination—and a **Kanban board** that provides a clear visualization of the workflow, limits work in progress, and optimizes task organization, ensuring effective management of projects and competitions.

This report is structured into ... chapters. The first chapter will present the general framework of the project. The second chapter will cover the design and technological choices, explaining the architecture of our application. The third chapter will describe ... Finally, a general conclusion and some future perspectives will be presented.

Chapitre

1 PRELIMINARY STUDY

1.1 Introduction

In this first chapter, we present the general framework and requirements of the project. We will start by introducing the organization. Then, we will examine projects similar to ours and consider solutions to the identified problems. Finally, we will describe the working methodology adopted for the project.

1.2 Host Organization

This section is dedicated to presenting the host organization, including a description of the company and its areas of activity.

1.2.1 General Presentation

Founded in 2020, **Elepzia** has established itself as a major player in the technology industry in Tunisia, with its headquarters located in Tunis. The company stands out for developing robust digital systems and innovative solutions to contemporary technological challenges. With approximately **80 employees**, recognized by programs such as **NVIDIA Inception**, **Microsoft for Startups Founders Hub**, and **GitHub for Startups**, Elepzia operates at the intersection of health and education, developing innovative solutions based on : Web and Mobile Development, Cybersecurity Services, and Advances in Artificial Intelligence. Under the leadership of its CEO, **Ms. Amina Trabelsi**, the organization has built a strong reputation thanks to the quality of its customer-focused products.

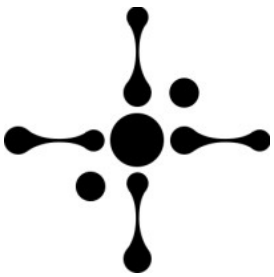


FIGURE 1.1 – Elepzia Company Logo

1.2.2 Services

Elepzia offers two flagship products :

- Episafe : a portable AI-powered platform that predicts epileptic seizures to improve patient safety and care.

- Hackini : an AI-based learning and innovation ecosystem designed to transform education through challenge-driven experiences.

Additionally, the company undertakes client projects in digital transformation and artificial intelligence.

1.3 Problem Statement

With the rise of technology competitions, it becomes crucial to provide a reliable, secure, and fair platform where participants can compete while ensuring the integrity of the results. However, several challenges arise :

- How to ensure objective evaluation of submissions ?
- How to design a scalable and secure platform capable of handling a large number of users and competitions simultaneously ?

1.4 Objective

The goal of this project is to create a unified and attractive environment to modernize and optimize the organization of technology competitions, making them more accessible, secure, and engaging for a wide audience.

The project aims to enhance user motivation and continuous improvement by providing interactive dashboards and gamified reward systems. Furthermore, it seeks to ensure fairness and transparency through jury evaluation.

1.5 Critical Study of Existing Platforms

This section presents an analysis of existing similar applications to identify their key features and limitations, in order to better understand current challenges and improvement opportunities.

1.5.1 Kaggle

Kaggle [1] is a **data science** competition platform owned by Google. It allows users to work on real datasets, develop AI models, and submit solutions for automatic evaluation (see Figure 1.2).

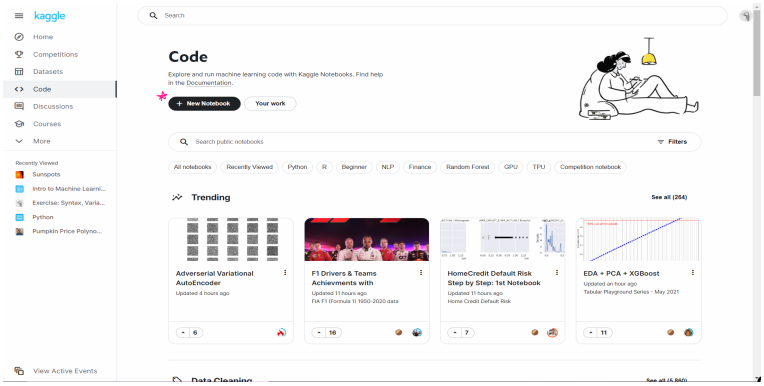


FIGURE 1.2 – Kaggle Platform Interface

Kaggle's limitations are :

- Platform exclusively dedicated to data science.
- Lack of interactive rewards to stimulate long-term engagement.

1.5.2 Codeforces & LeetCode

Codeforces [2] (see Figure 1.3) and LeetCode [3] (see Figure 1.4) are platforms dedicated to **algorithmic programming competitions**. They allow developers to practice coding challenges and participate in online tournaments.

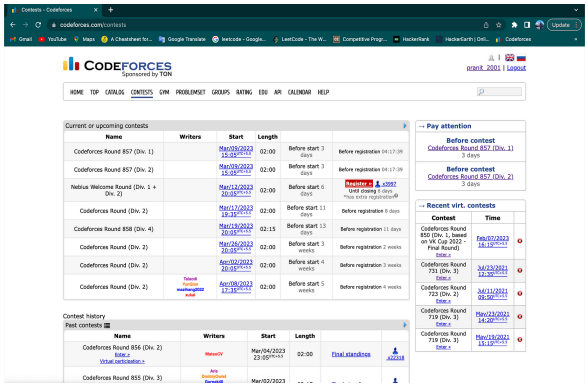


FIGURE 1.3 – Codeforces Platform

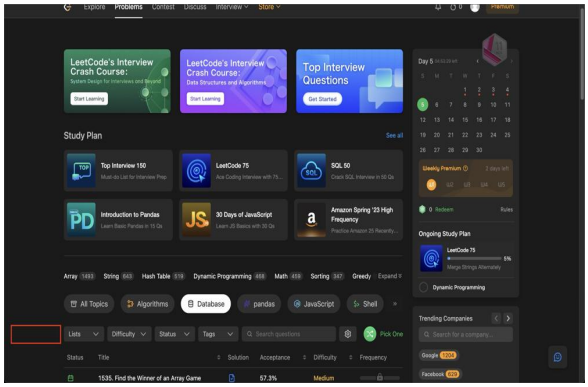


FIGURE 1.4 – LeetCode Platform

Their limitations include :

- Fragmented user experience and limited interaction between participants.
- Exclusively algorithm-oriented.

1.5.3 CTFtime & Hack The Box

CTFtime [4] (see Figure 1.5) and Hack The Box [5] (see Figure 1.6) are platforms specialized in **cybersecurity competitions**. They offer challenges where participants exploit vulnerabilities to "capture flags" (CTF - Capture The Flag).

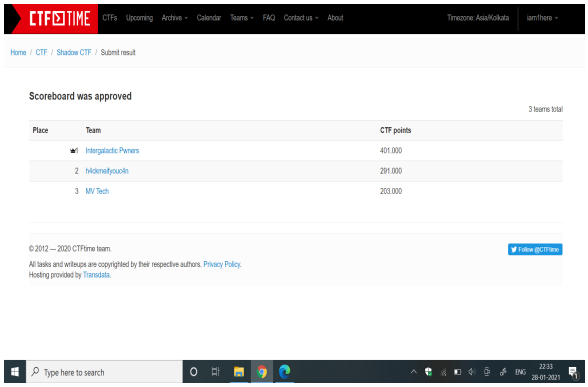


FIGURE 1.5 – CTFtime Platform

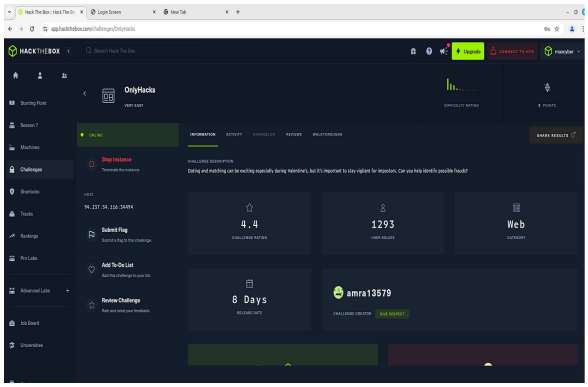


FIGURE 1.6 – Hack The Box Platform

Their limitations include :

- Highly specialized in cybersecurity, excluding other domains.
- Often very advanced levels, making access difficult for beginners.

1.6 Proposed Solution

To address the limitations of existing platforms, the company proposes **Hackini**, a web application dedicated to technology competitions. This platform will bring together **various types of challenges** in programming, AI, and cybersecurity, providing users with a diverse experience.

With **professional evaluation of submissions** and **gamified reward systems**, Hackini ensures continuous motivation, balanced matchups, and complete transparency. To facilitate collaborative management and team coordination, it integrates an **Interactive Whiteboard** for real-time planning, brainstorming, and visualization, as well as a **Kanban board** to provide clear workflow visualization, limit work in progress, and optimize task organization. Additionally, an **interactive dashboard** will allow users to track their group's progress.

1.7 Work Methodology

Adopting and rigorously implementing an appropriate work methodology is essential to ensure smooth project execution, effective coordination, and overall success. In this section, we present a comparative study between traditional and agile methodologies, followed by the justification for choosing the **Scrum-Kanban** approach.

1.7.1 Agile Approaches

- Scrum : an agile framework that promotes iterative and incremental development. Work is organized into short cycles called sprints, allowing regular reassessment and priority adjustment. It encourages close collaboration, frequent communication, and transparency through artifacts like the Product Backlog and Sprint Backlog. Regular meetings (daily stand-ups, sprint planning, sprint reviews) help the team stay aligned and responsive to changes [8] (see Figure 1.7).

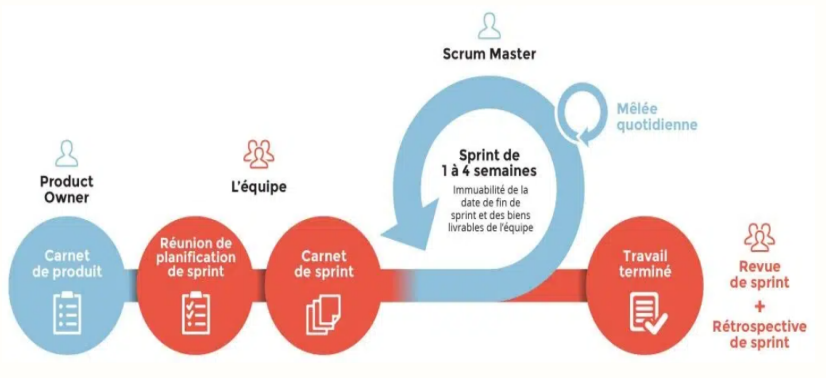


FIGURE 1.7 – Scrum Process [9]

- Kanban : an agile methodology focused on visualizing workflow and limiting work in progress (WIP), using a Kanban board to optimize efficiency and flow [10] (see Figure 1.8). Kanban is very flexible and can be adapted to different types of projects, but it does not define specific roles or time-boxed iterations.

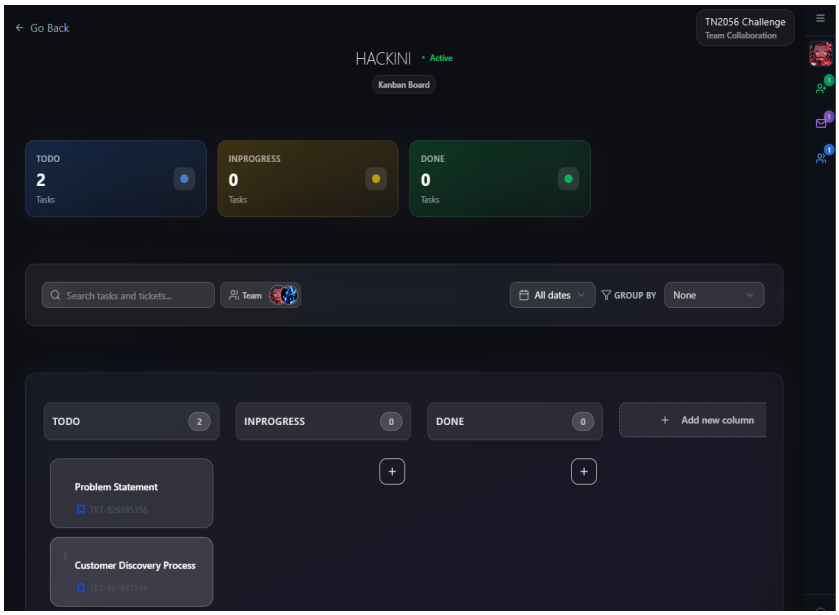


FIGURE 1.8 – Kanban Structure [11]

- Scrum-Kanban (Scrumban) : a hybrid approach combining Scrum's structure (sprints, roles, rituals) with Kanban's visual flow (board, WIP limits), providing flexibility and efficient management of evolving projects [12].

1.7.2 Comparison Table of Methodologies

To better distinguish project management approaches, Table 1.1 summarizes the characteristics of Scrum, Kanban, and their hybrid combination.

TABLE 1.1 – Comparison of Software Development Methodologies

Approach	Flexibility	Adaptability	Planning	Customer Feedback	Use Cases
Scrum	High	High	Iterative, by sprints	Frequent	Complex and evolving projects
Kanban	Very High	High	Continuous, flow-based	Continuous	Maintenance, support, variable requests
Scrum-Kanban	**Very High**	**Very High**	**Hybrid : iterative and continuous**	**Continuous and frequent**	**Dynamic and rapidly evolving environments**

1.8 Unified Modeling Language

The choice of Unified Modeling Language (UML) [13] as a modeling tool is due to its ability to clearly represent various aspects of our software system. UML is a graphical modeling language using pictograms, designed as a standardized visualization method in software development and object-oriented design.

UML is particularly suitable for this project for several reasons :

- Graphically visualize system components and their interactions.
- Variety of diagram types to capture different aspects of system structure and behavior.
- Clearly communicate ideas and relationships between system elements, facilitating team collaboration.

1.9 Conclusion

In conclusion, this first chapter laid the essential foundations of our project. After presenting the host company **Elepzia**, we identified the problem and chose the Scrumban methodology for its ability to address the project's challenges. The following chapter will delve deeper into the design of our solution.

Chapitre

2 CONCEPTUAL STUDY AND TECHNOLO CHOICES

Sommaire

2.1 Introduction

In this chapter, we will define the functional and non-functional requirements. We will identify the stakeholders involved and the essential features using various modeling diagrams. Next, we will select the tools required for application development. Finally, we will detail the application architecture and plan the sprints to structure and organize the project's progress.

2.2 Requirements Specification

The specification of functional and non-functional requirements is a crucial step. Its role is to define the project's demands and constraints, providing a clear framework to guide implementation and ensure conformity with expectations.

2.2.1 Functional Requirements

Functional requirements describe the main features that the Hackini platform must provide to achieve the following objectives :

- **Authentication :** Users must register and log in securely using a password, one-time password (OTP), and other robust authentication methods.
- **Competition Management :** Users can view available competitions based on dynamic criteria, participate in challenges, and submit solutions.
- **Submission Evaluation :** Juries must evaluate user submissions according to predefined criteria (accuracy, performance, etc.), generating scores and rankings in real time.
- **Interactive Dashboard :** Teams must have access to a personalized dashboard to track their rankings and scores.
- **Platform Administration :** The administrator must manage users, competitions, tickets, and invitations from a dashboard.
- **Secure and Scalable Cloud Deployment :** The platform must be deployed on a scalable and secure cloud infrastructure, with backup, encryption, firewall mechanisms, and load handling to ensure availability.

2.2.2 Non-Functional Requirements

Non-functional requirements are the technical criteria the software must meet to ensure quality. Our application must provide the following service qualities :

- **Performance :** Hackini must handle a large number of simultaneous participants without affecting responsiveness, optimizing performance for fast response times during registration, solution submission, and result consultation.

- **Security :** Sensitive user data, including personal information, financial transactions, and competition results, must be protected with advanced security mechanisms to prevent unauthorized access.

- **Usability :** Interfaces must be intuitive and easy to navigate, providing a satisfactory user experience.

- **Compatibility :** The platform must be compatible with a wide range of devices, particularly Android and iOS mobile devices, ensuring a smooth and consistent experience.

- **Scalability :** Hackini must be designed to scale with the growing number of users and competitions, with a cloud infrastructure allowing scaling without compromising performance or security.

2.3 Actors Specification

Table 2.1 summarizes the actors of our system and their respective roles :

2.4 Use Case Diagram

The use case diagram is a fundamental UML tool for visualizing interactions between actors and the main system functionalities. It facilitates understanding of the system's functional requirements by all stakeholders.

TABLE 2.1 – Actors Specification

Role	Actions
Participant	- Create an account and authenticate using secure methods - Participate in available hackathons - Manage tickets and articles - Submit solutions to challenges - Track rankings and scores - Send, accept, and reject invitations - Collaborate in real time via a shared interface (whiteboard) - Organize and track tasks using a Kanban board
Advisor	- Create an account and authenticate securely - Manage tickets and articles - Support and advise participants - Accept and reject invitations - Use the whiteboard interface to guide teams - Supervise and manage a shared Kanban board with tasks assigned to each member
Organizer	- Create an account and authenticate securely - Manage tickets and articles - Manage hackathons - Send invitations - View rankings and scores
Administrator	- Manage users and hackathons - Manage invitations, tickets, and articles - View rankings and scores
Jury	- Create an account and authenticate securely - View hackathons, scores, and rankings - Evaluate submitted solutions and add comments - Accept and reject invitations - Manage tickets and articles

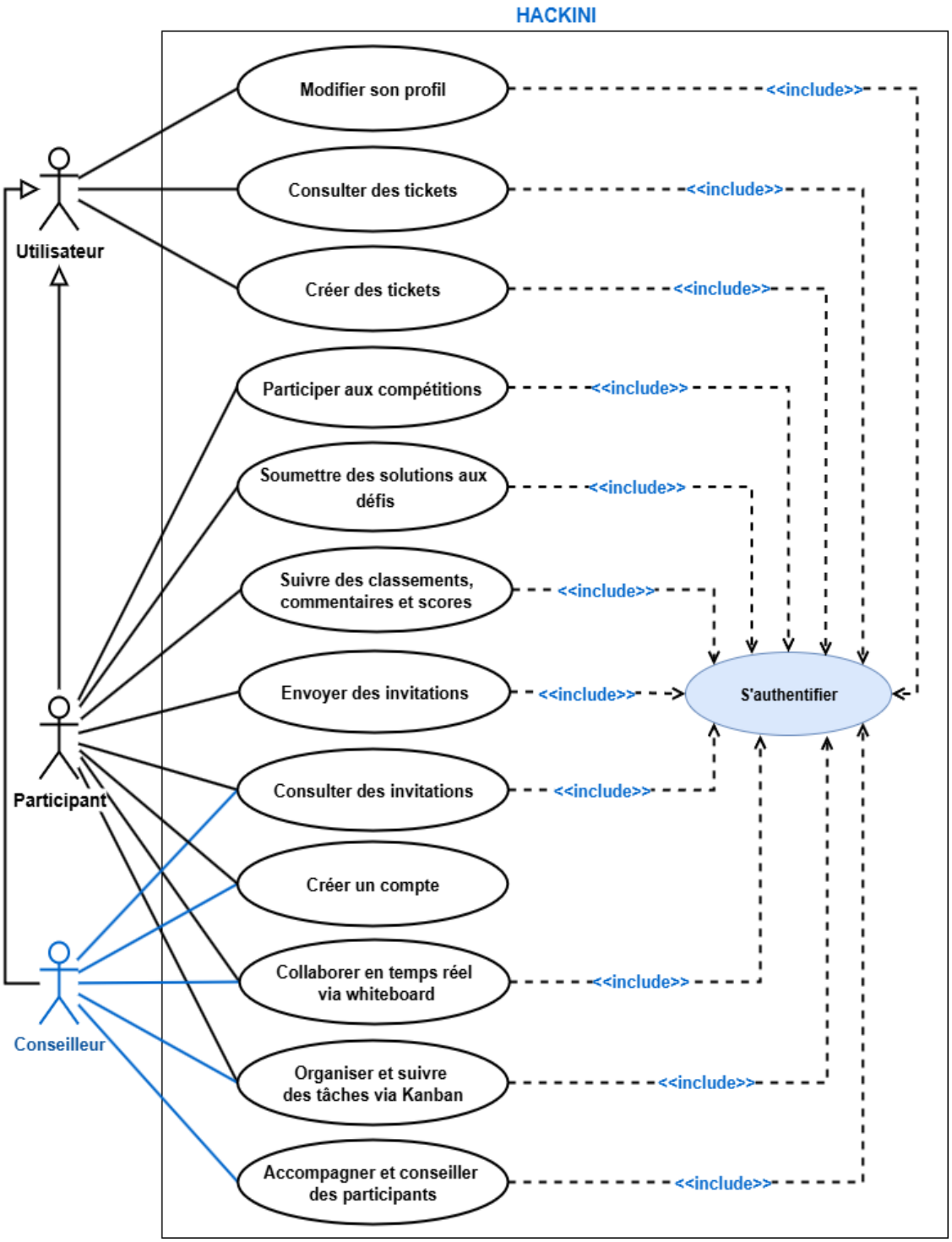


FIGURE 2.1 – Use Case Diagram – Front Office

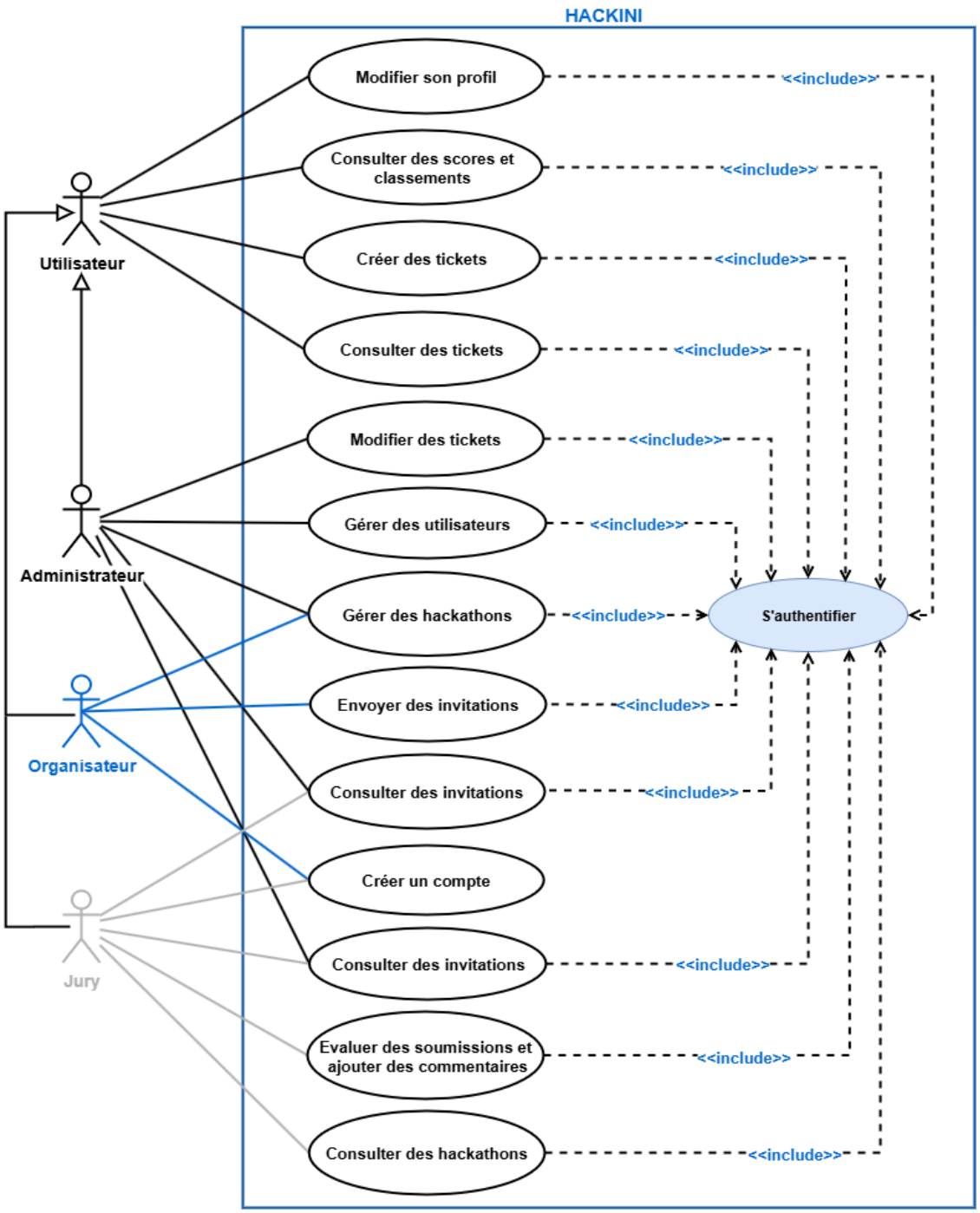


FIGURE 2.2 – Use Case Diagram – Back Office

2.5 Class Diagram

The class diagram is a core software design tool that describes the system's internal structure by showing classes, their attributes, and the relationships between them.

Figure 2.3 shows the class diagram of our system :

images/diag_classes.png

FIGURE 2.3 – Class Diagram

2.6 Working Environment

This section defines the hardware and software environment suitable for project implementation.

2.6.1 Hardware Environment

For this project, we used a computer with the following specifications : - Brand : ASUS TUF Gaming A15

- Processor : AMD Ryzen 5 4600H with Radeon Graphics 3.00GHz

- RAM : 16GB

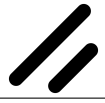
- Hard Drive : 500GB

- Operating System : Windows 11

2.6.2 Software Environment

This section details the complete software environment used throughout the project, highlighting the tools, technologies, and frameworks that ensured effective integration of components and smooth implementation of technical objectives.

TABLE 2.2 – Development Tools and Technologies

Logo	Description
	**Visual Studio Code :** Cross-platform code editor supporting multiple languages such as Java, JavaScript, Go, C++, etc.
	**GitHub :** Code hosting platform using Git. Facilitates collaboration with features like bug tracking and pull requests.
	**Jest :** JavaScript testing framework used to test components, functions, or modules.
	**Next.js :** React framework enabling server-side, static, or hybrid rendering.
	**NestJS :** Node.js framework for building scalable backend applications in TypeScript.
	**Prisma :** Open-source ORM for Node.js and TypeScript, compatible with PostgreSQL, MySQL, MongoDB, etc.
	**PostgreSQL :** Open-source relational database system, robust and high-performing.
	**Swagger :** Open-source suite to design, build, document, and test RESTful APIs.
	**Figma :** Online collaborative design tool for UI/UX and interactive mockups.
	**draw.io :** Online diagramming tool for UML, architecture, network, or database diagrams.
	**TailwindCSS :** Utility-first CSS framework for building modern responsive interfaces quickly.
	**shadcn/ui :** Reusable and customizable component library based on Radix UI and TailwindCSS.

2.7 Application Architecture

To ensure professional development, it is necessary to choose a suitable architecture for the system. For this reason, we selected the MVC architecture.

MVC : Model-View-Controller separates the application into three parts :

- **Model :** Contains the application logic and state.
- **View :** Represents the user interface.
- **Controller :** Manages synchronization between the view and the model.

2.8 Sprint Planning

Sprint planning is a key meeting in a Scrum project. Its objective is to create a clear and achievable action plan, focusing on the highest-priority items in the product backlog. We divided the work into four independent sprints :

- **Sprint 1 :** User Access Setup – Allow users to securely access the application and log out easily.
- **Sprint 2 :** Profile and User Settings Management – Complete, view, and edit the user profile, configure security, notifications, and privacy settings.
- **Sprint 3 :** Hackathon Management – Implement main hackathon functionalities including creation, participation, management, and tracking of competitions.

2.9 Timeline

The project spans approximately six months, from February 24 to August 18. The various development and implementation stages of Hackini are illustrated by the Gantt chart below, showing all planned tasks.

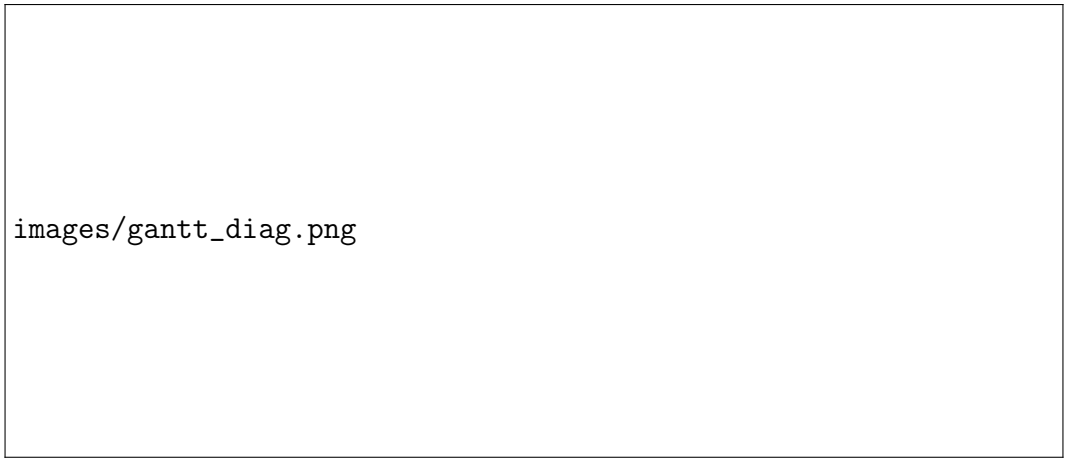


FIGURE 2.4 – Gantt Chart of Project Stages

2.10 Conclusion

This chapter laid the essential foundations for the design and development of our application. We identified system requirements and defined the actors involved. Modeling diagrams clarified system components, and the description of the working environment and architecture provided a framework for technical development. The next chapter will detail the first sprint of the project.

Chapitre

3 Sprint 1 – Implementation of the User Access System

Sommaire

3.1 Introduction

This first sprint is dedicated to the design and implementation of authentication, registration, and logout functionalities, which are essential elements for web application security. Authentication and registration ensure controlled and secure access to the platform's features, while logout guarantees smooth and reliable session termination, contributing to data protection and an optimal user experience.

3.2 Sprint 1 Backlog

As part of our first sprint, focused on implementing a user access module, we developed Table 3.1, which details the sprint 1 backlog :

TABLE 3.1 – Sprint 1 Backlog

ID	User Story	Description	Acceptance Criteria	Priority
1	Implementation of the user access system	As a user, I want to be able to create an account, authenticate to access the application, and log out when finished.	- User can create an account by providing the required information. - User enters username and password to log in. - The system verifies the input and authenticates the user. - User can log out at any time.	High
2	Password reset	As a user, I want to reset my password in case I forget it.	- A secure form allows entering a new password. - The system confirms the change.	High

3.3 Registration

This section presents the user registration process, illustrated through the use case diagram, sequence diagram, and corresponding interfaces.

3.3.1 Use Case Diagram for the "Register" Feature

Figure 3.1 shows the use case for the "Register" scenario.

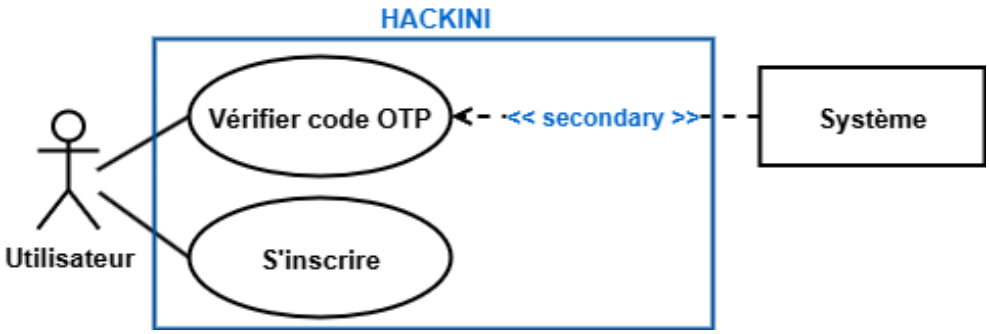


FIGURE 3.1 – Use Case Diagram for the "Register" Scenario

3.3.2 Textual Description of the "Register" Use Case

The textual description of a use case diagram outlines the steps, preconditions, and alternative scenarios of a process.

Table 3.2 shows the textual description for the "Register" use case :

TABLE 3.2 – Textual Description of the "Register" Use Case

Use Case	Register
Actors	User
Preconditions	User does not yet have an account.
Postconditions	User account is successfully created.
Main Scenario	1. User accesses the registration page. 2. User enters email address and name. 3. System sends an OTP code by email. 4. User enters the received OTP code in the app. 5. System verifies the OTP code. 6. User enters a password. 7. System verifies the password validity. 8. System creates the user account. 9. User is redirected to the login page.
Alternative Scenario	If the OTP code is invalid or expired, the system displays an error message and prompts the user to re-enter the code or request a new one.

3.3.3 Sequence Diagram for the "Register" Scenario

The sequence diagram shows the chronological interactions between the user and the system, highlighting the flow of messages and actions during registration.

The corresponding sequence diagram is illustrated in Figure 3.2 :

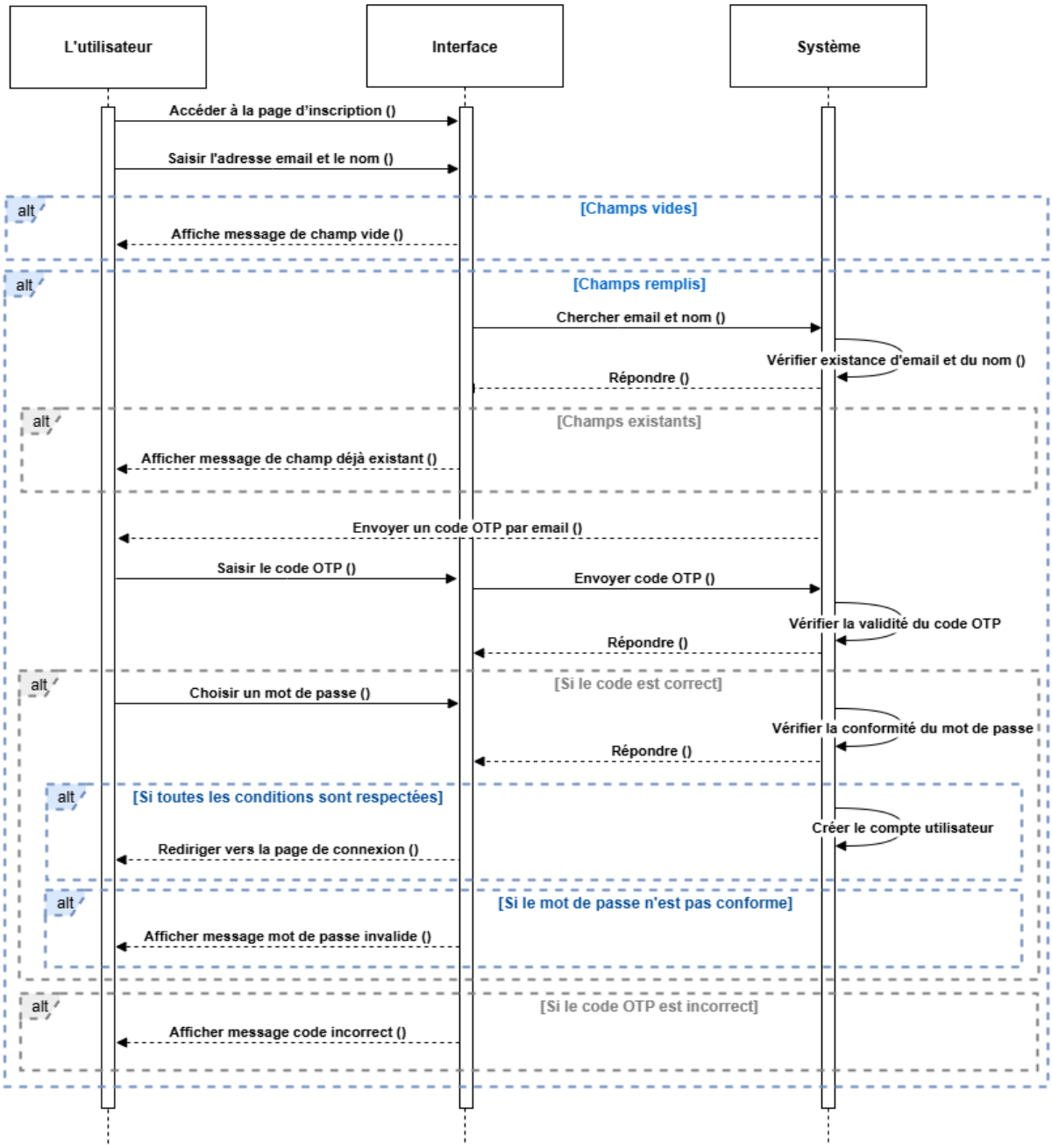


FIGURE 3.2 – Sequence Diagram for the "Register" Scenario

3.3.4 Registration Interface

To access all application features, the user must first register via the dedicated interface (Figures 3.3 and 3.4) before logging in.

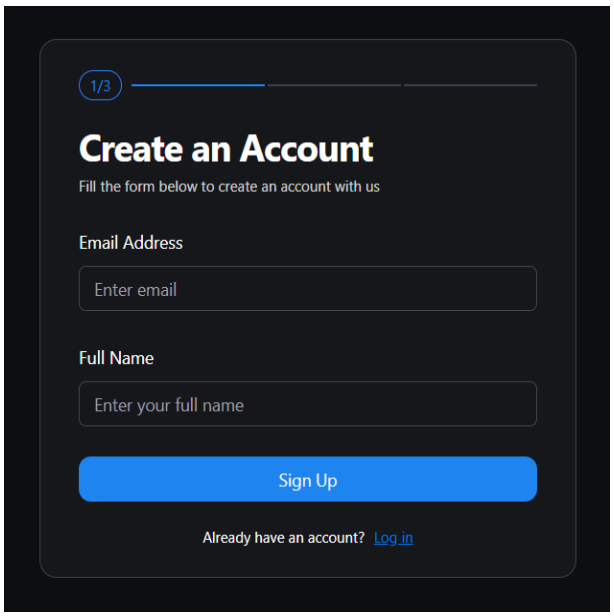


FIGURE 3.3 – Registration Interface (Arena)

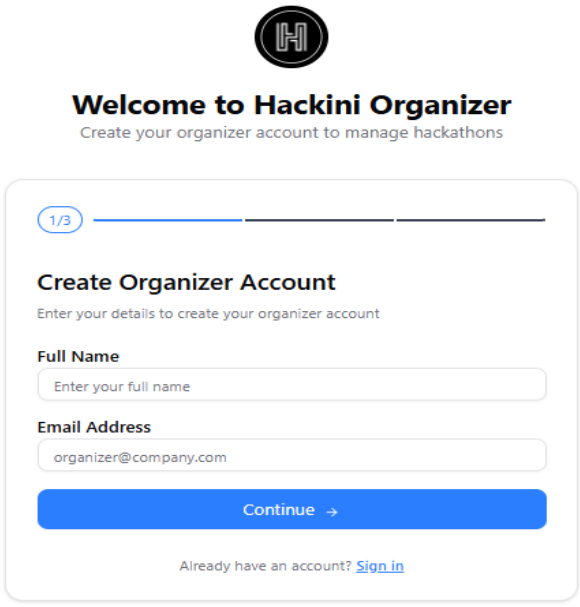


FIGURE 3.4 – Registration Interface (Venture)

Error messages notify the user if fields are empty or if an existing username is used (Figures 3.5 and 3.6).

Chapitre

4 Sprint 2 – User Profile and Settings Management

Sommaire

4.1 Introduction

This second sprint focuses on the design and development of the user profile and settings management interfaces, allowing users to view and update their account information for a secure and personalized experience.

4.2 Sprint 2 Backlog

For this sprint, which is centered on implementing the user profile and settings management module, we have prepared Table 4.1 detailing the sprint 2 backlog.

TABLE 4.1 – Sprint 2 Backlog

ID	User Story	Description	Acceptance Criteria	Priority
2	Profile Completion	As a user, I want to be able to complete my profile with optional information.	- User fills in the fields. - The system verifies the entered information. - Additional fields are successfully saved.	Medium
3	Access User Profile	As a user, I want to view my data.	- User can access their profile. - Data is dynamically retrieved from the database.	High
4	Profile Editing	As a user, I want to edit my profile.	- User can modify their data and save changes. - A confirmation message appears after update.	High
5	User Settings Management	As a user, I want to configure my account settings.	- A “Settings” page allows modifying certain options. - Preferences are saved and applied immediately. - User is notified in case of an error.	Medium

4.3 User Profile Management

This section presents the user profile management module, including access, editing, and completion of personal information, illustrated with use case diagrams, sequence diagrams, and associated interfaces.

4.3.1 Use Case Diagrams for Profile Management Features

Figure 4.1 shows the use case for the “Complete Profile” scenario.

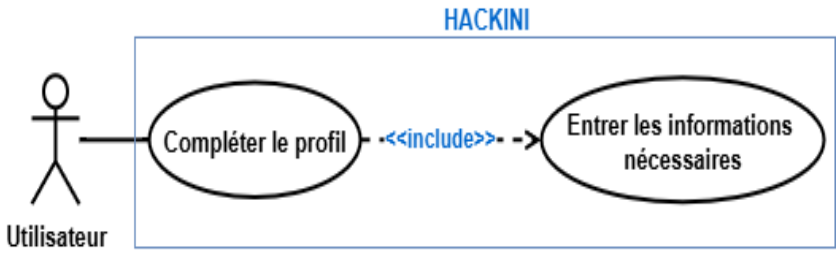


FIGURE 4.1 – Use case diagram for the “Complete Profile” scenario

Figure 4.2 details the use case for the “View User Profile Information” scenario.

FIGURE 4.2 – Use case diagram for the “View User Profile Information” scenario

Figure 4.3 shows the use case for the “Edit User Profile Information” scenario.

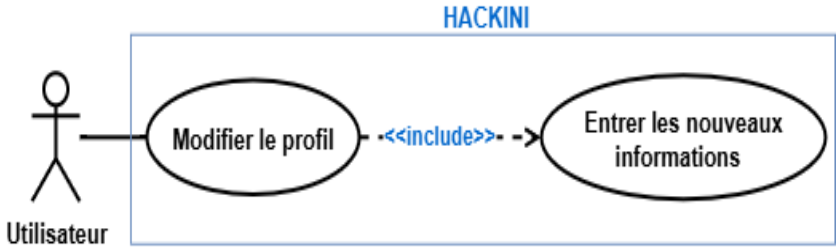


FIGURE 4.3 – Use case diagram for the “Edit User Profile Information” scenario

4.3.2 Textual Description of Profile Management Use Cases

Table 4.2 shows the textual description of the “Complete Profile” use case.

Table 4.3 describes the use case for viewing user profile information.

Table 4.4 describes the use case for editing profile information.

TABLE 4.2 – Textual description of the "Complete Profile" use case

Use Case	Complete Profile
Actors	User
Preconditions	The user profile is partially completed.
Postconditions	Optional fields are successfully saved.
Main Scenario	1. System displays profile completion suggestion. 2. User accesses the relevant section. 3. User fills in additional fields (photo, biography, etc.). 4. User validates the changes. 5. System saves the optional data.
Alternative Scenario	If the user does not wish to complete the profile, they can ignore the suggestion.

TABLE 4.3 – Textual description of the "View User Profile Information" use case

Use Case	View User Profile Information
Actors	User
Preconditions	User is logged into the platform.
Postconditions	User's personal information is displayed correctly.
Main Scenario	1. User accesses the "My profile" section. 2. System retrieves user information from the database. 3. Information is displayed in the profile interface.
Alternative Scenario	If data retrieval fails, an error message is displayed.

4.3.3 Sequence Diagrams for Profile Management Features

To understand profile management, the following sequence diagrams are presented.

Figure 4.4 illustrates the profile completion process.

TABLE 4.4 – Textual description of the "Edit User Profile" use case

Use Case	Edit User Profile Information
Actors	User
Preconditions	User is logged into the platform.
Postconditions	Changes are successfully saved.
Main Scenario	1. User accesses their profile and clicks "Edit". 2. User modifies desired fields (name, phone, etc.). 3. User clicks "Save Changes". 4. System updates the database. 5. Confirmation message is displayed.
Alternative Scenario	If validation or connection fails, an error message is shown.

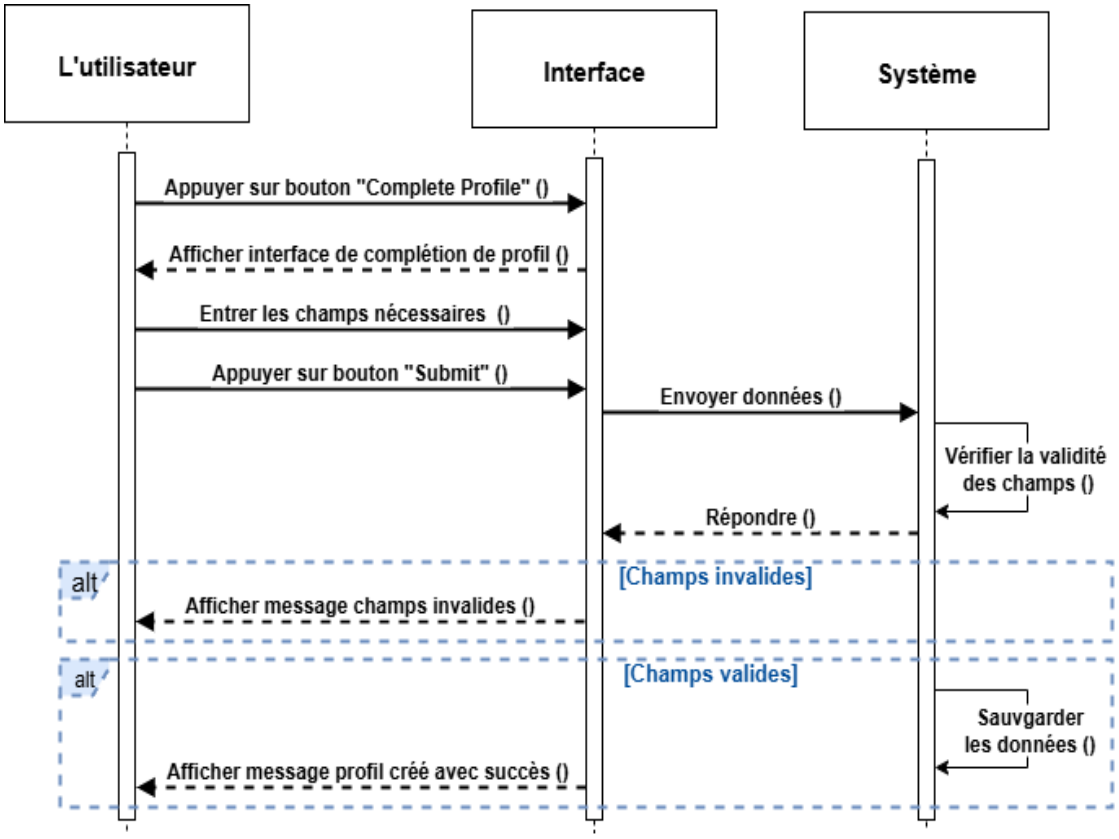


FIGURE 4.4 – Sequence diagram for the "Complete Profile" scenario

Figure 4.5 shows the sequence diagram for viewing profile information.

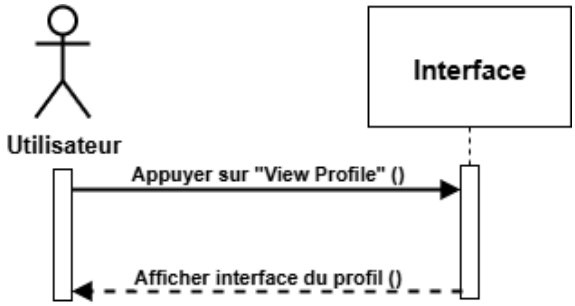


FIGURE 4.5 – Sequence diagram for the "View User Profile Information" scenario

Figure 4.6 shows the diagram describing the "Edit User Profile" scenario.

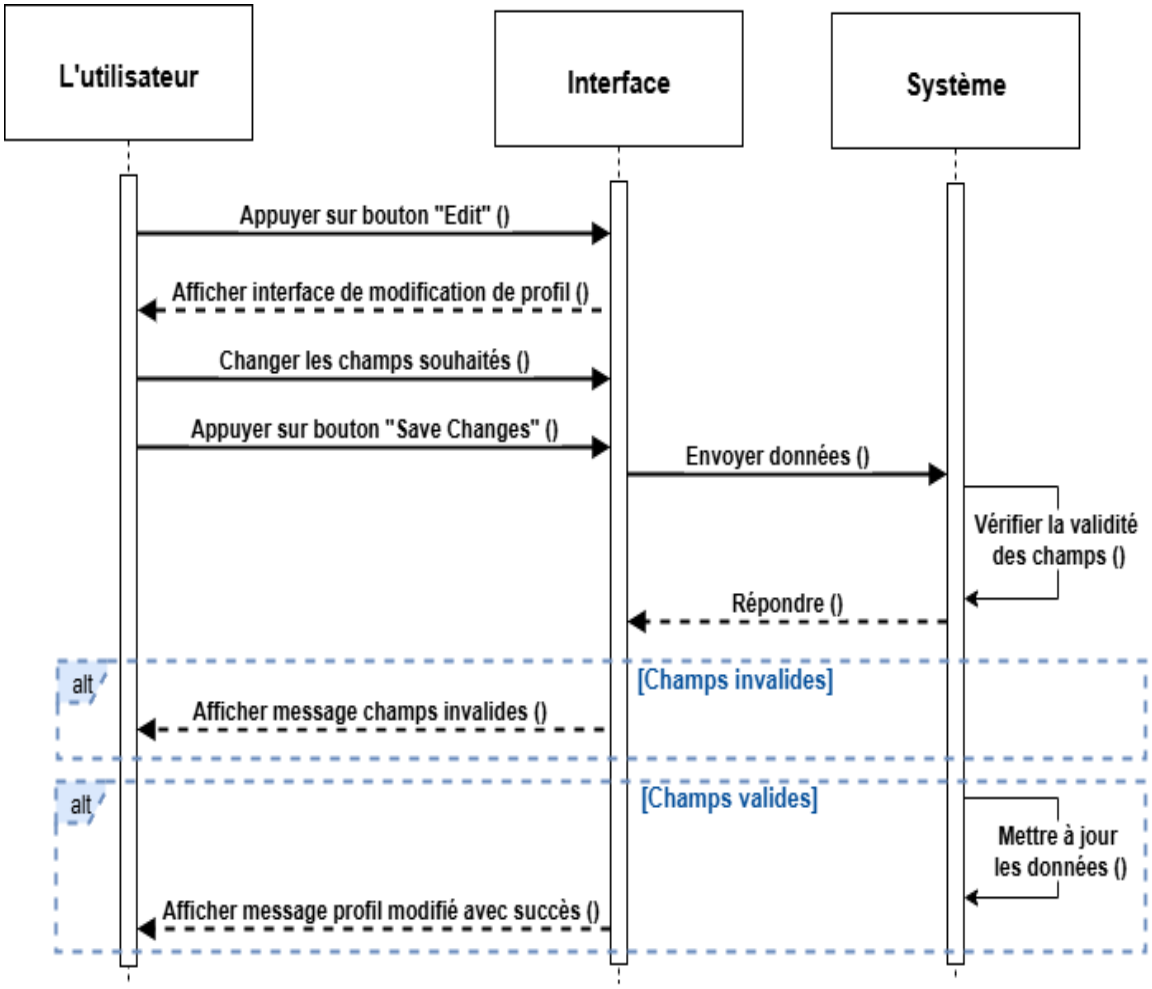


FIGURE 4.6 – Sequence diagram for the "Edit User Profile" scenario

4.3.4 User Profile Management Interface

To access all their information, the user must first complete their profile via the dedicated completion interface. Figure 4.7 illustrates this interface :

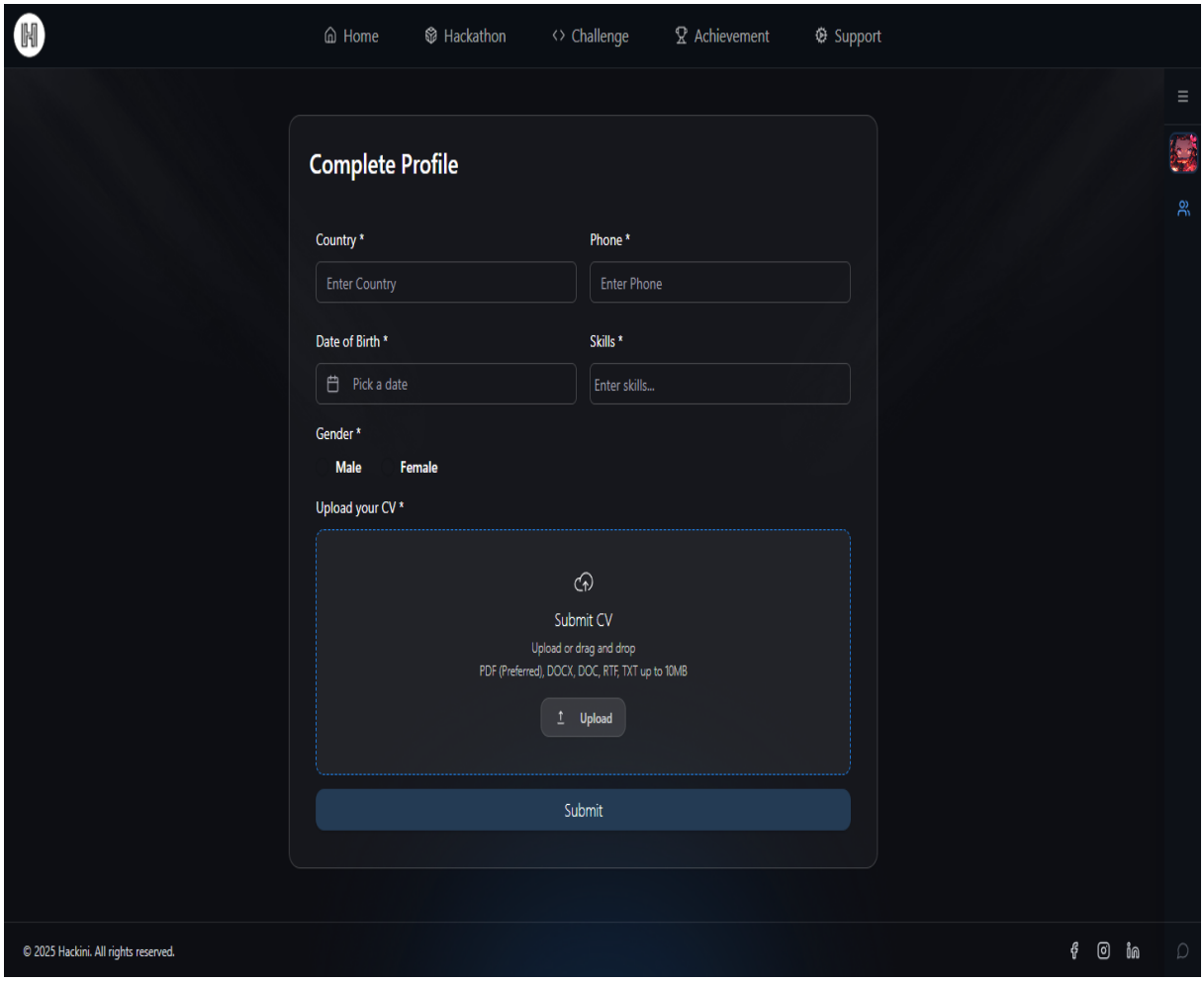


FIGURE 4.7 – Profile Completion Interface

If the user clicks “Submit” without entering correct information, error messages appear (see Figure 4.8). Otherwise, the user is redirected to the "Home" page.

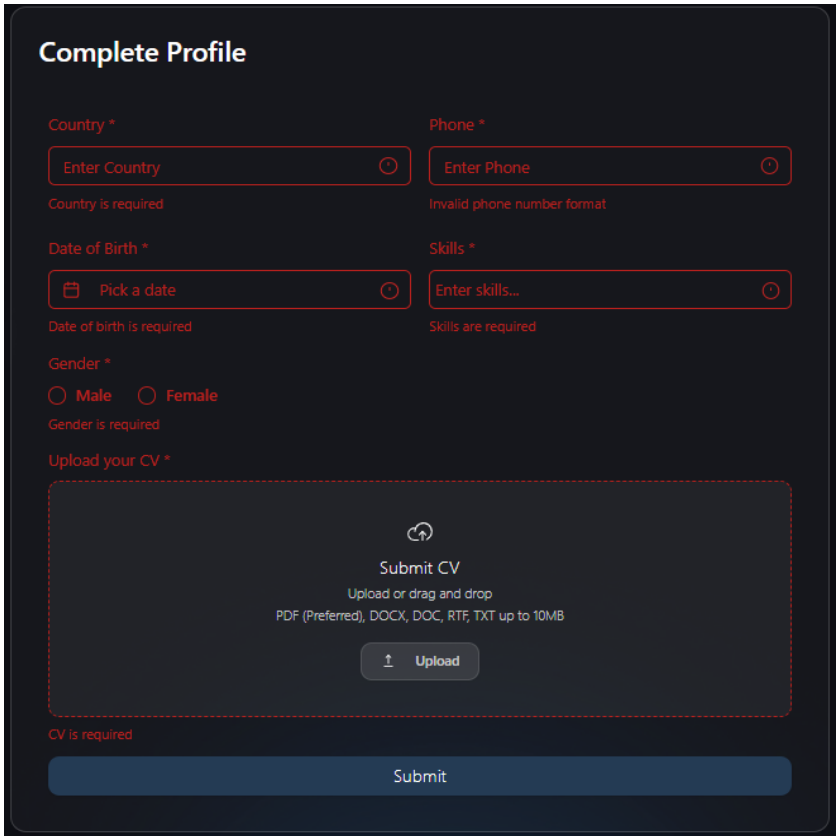


FIGURE 4.8 – Profile Completion Interface with Invalid Fields

They can then view their data through the profile interface shown in Figure 4.9.

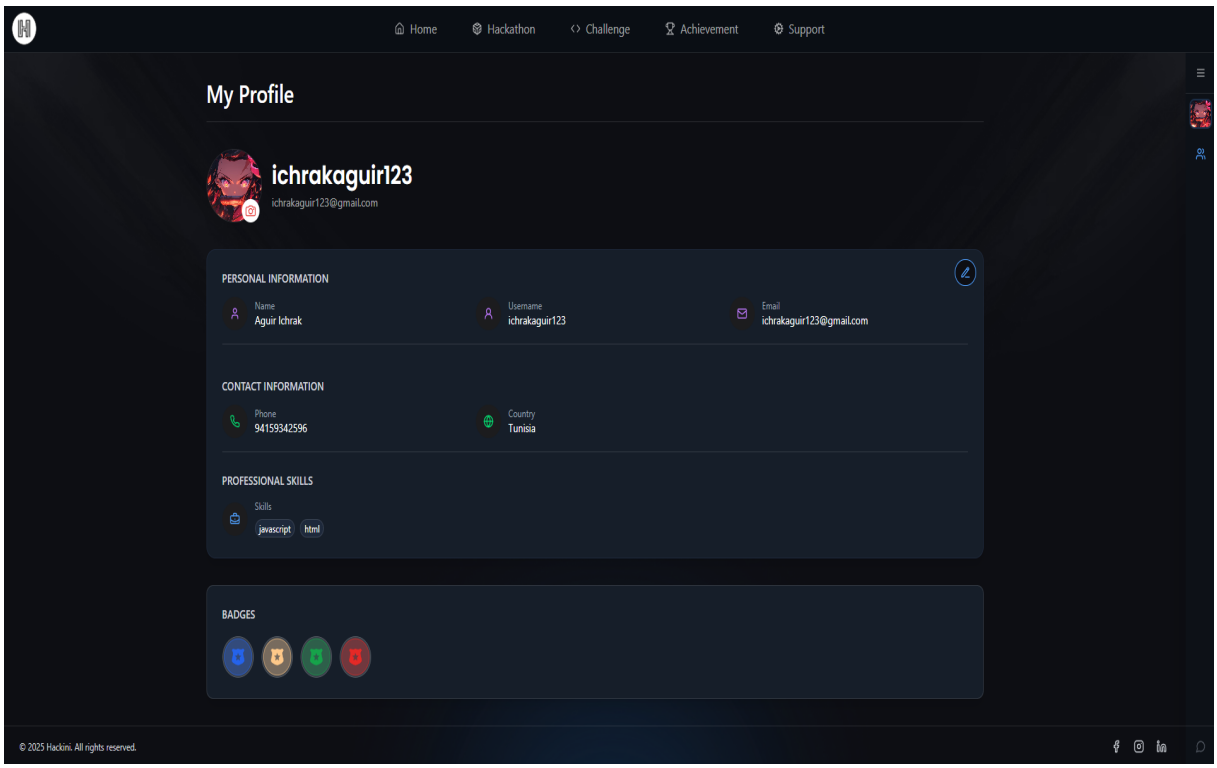


FIGURE 4.9 – User Profile Interface

If they wish to edit certain information, the user can do so via the profile edit interface (see Figure 4.10).

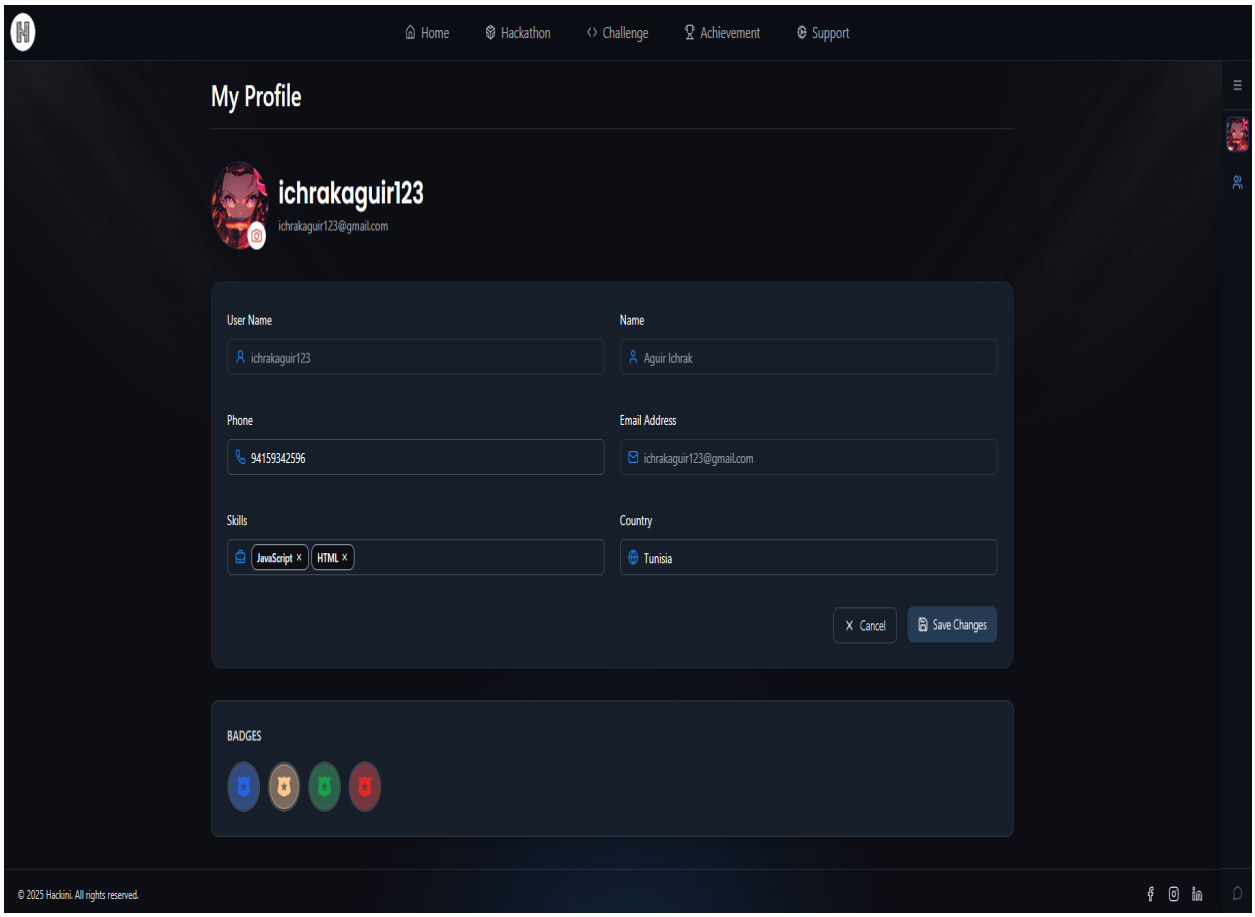


FIGURE 4.10 – Profile Edit Interface

4.4 User Settings Management

This section describes the process for managing user settings, based on sequence diagram analysis and interfaces dedicated to security, privacy, and notifications.

4.4.1 Textual Description of User Settings Management

Table 4.5 shows the textual description of the user settings management scenario :

TABLE 4.5 – Textual Description of the "Manage User Settings" Scenario

Use Case	Manage User Settings
Actors	User
Preconditions	User must be authenticated and have access to their personal space.
Postconditions	Selected settings are updated and successfully saved.
Main Scenario	1. User accesses the "Settings" section. 2. System displays the different settings categories (Security, Privacy, Notifications). 3. User selects a category. 4. System displays the corresponding options. 5. User modifies settings according to preferences (enable 2FA, disable email notifications, etc.). 6. System saves the new configurations. 7. A success message is displayed.
Alternative Scenario	If the user enters invalid information, the system displays an error message specifying the problem.

4.4.2 Sequence Diagram for the "Manage User Settings" Scenario

The following sequence diagram (see Figure 4.11) corresponds to the user settings management scenario.

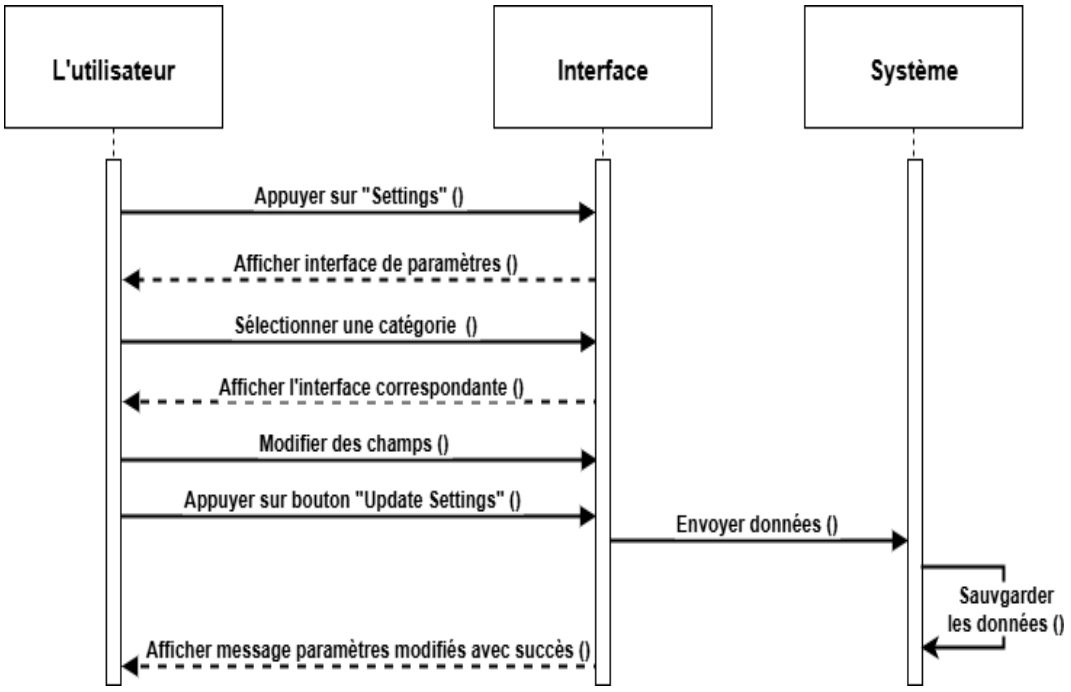


FIGURE 4.11 – Sequence Diagram for the "Manage User Settings" Scenario

4.4.3 User Settings Management Interfaces

To access settings, the user can navigate through different configuration interfaces. The following sections present interfaces for security (Figure 4.12), privacy (Figure 4.13), and notifications (Figure 4.14).

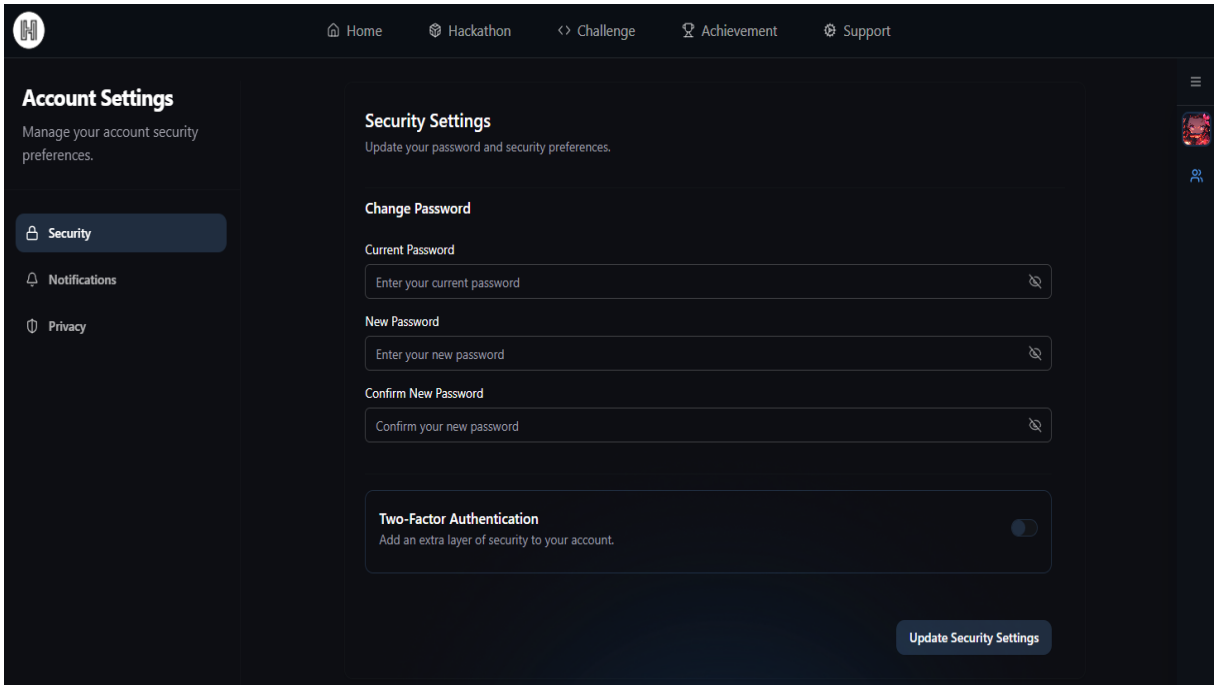


FIGURE 4.12 – Security Settings Interface

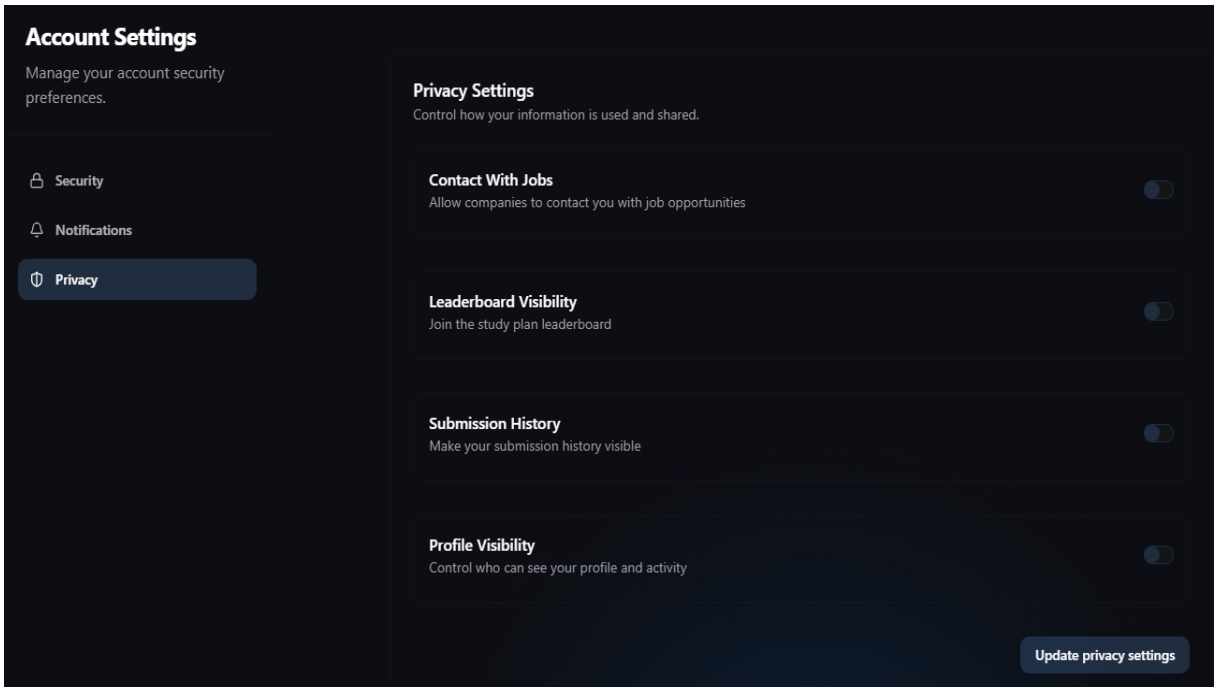


FIGURE 4.13 – Account Privacy Interface

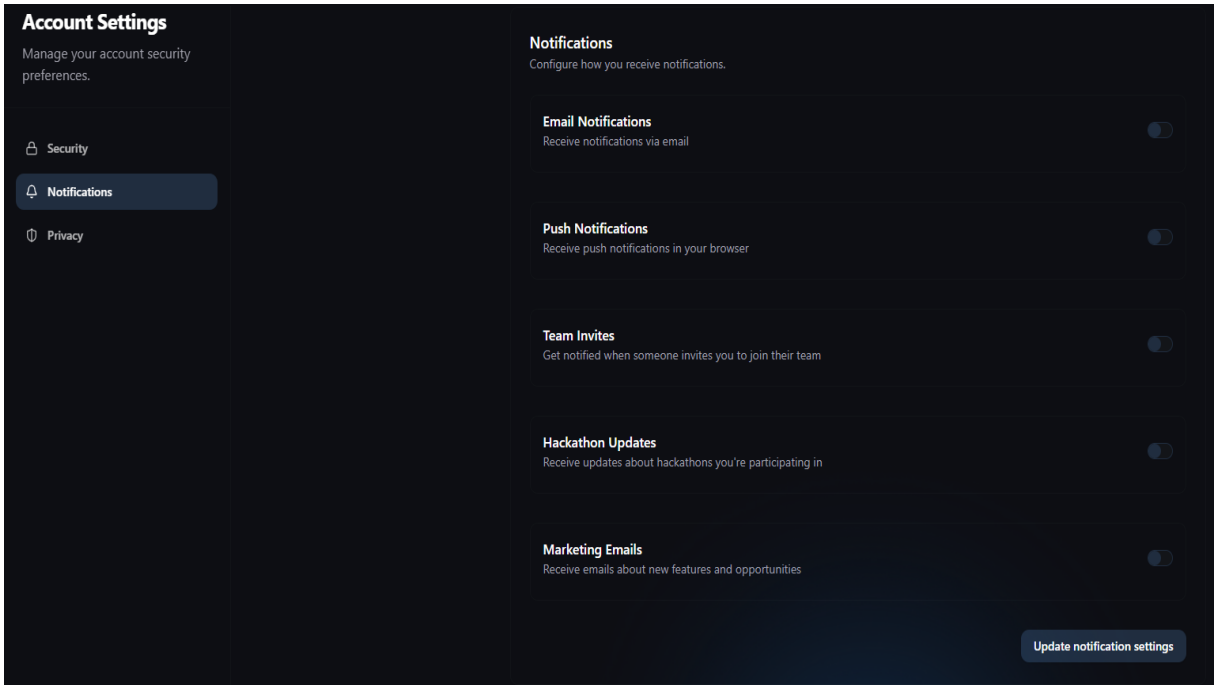


FIGURE 4.14 – Notification Settings Interface

4.5 Conclusion

In this second sprint, we implemented functionalities related to user profile management, including profile completion, viewing, and editing of personal information, as well as security, privacy, and notification settings. The next sprint will focus on ticket and invitation management.

Chapitre

5 Sprint 3 – Ticket and Invitation Managemen

Sommaire

5.1 Introduction

This third sprint focuses on implementing ticket and invitation management functionalities. It covers creating and viewing these items, as well as editing tickets when necessary.

5.2 Sprint 3 Backlog

Table 5.1 details the sprint 3 backlog.

TABLE 5.1 – Sprint 3 Backlog

ID	User Story	Description	Acceptance Criteria	Priority
1	Ticket Management	As a user, I want to manage my support tickets.	- User can create tickets for their requests. - User can view ticket details and exchange messages with the app manager. - User can view a list of their tickets. - Administrator can modify or close tickets.	Medium
1	Invitation Management	As a user, I want to manage my invitations as needed.	- User can send and receive friend invitations. - User can receive invitations to join groups. - Administrator can view all invitations. - Organizer can send invitations to mentors and jury members. - Jury members can view received invitations.	High

5.3 Ticket Management

This section presents the ticket management process through use case diagrams and the associated interfaces.

5.3.1 Use Case Diagram for Ticket Management

The use case diagram for ticket management is shown in Figure 5.1.

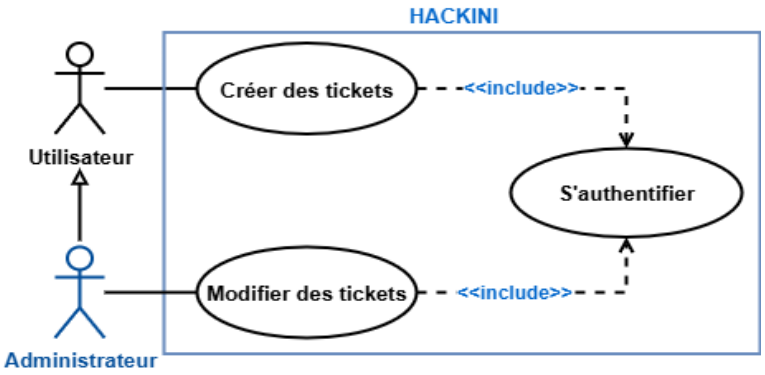


FIGURE 5.1 – Use Case Diagram for Ticket Management

5.3.2 Textual Description of Ticket Management

Table 5.2 shows the textual description of the “Create Ticket” use case :

TABLE 5.2 – Textual Description of the “Create Ticket” Use Case

Use Case	Create Ticket
Actors	User
Preconditions	User accesses the support section.
Postconditions	The ticket is saved in the database and a notification is generated.
Main Scenario	1. User accesses the "Support" or "Support Tickets" section. 2. Clicks the "Create Ticket" button. 3. Fills in the form fields. 4. Clicks the "Create" button. 5. The system saves the ticket and displays a creation message.
Alternative Scenario	If the OTP code is invalid or expired, the system shows an error message and prompts the user to re-enter the code or request a new one.

Table 5.3 details the “Edit Ticket” use case :

TABLE 5.3 – Textual Description of the "Edit Ticket" Use Case

Use Case	Edit Ticket
Actors	Administrator
Preconditions	Administrator accesses the "Support Ticket" section.
Postconditions	Changes are saved in the database.
Main Scenario	1. Administrator accesses the "Support Ticket" section. 2. Clicks the "Edit Ticket" button. 3. Modifies the desired fields. 4. Clicks the "Save" button. 5. The system saves the changes and displays a confirmation message.

5.3.3 Ticket Management Interfaces

The following interfaces allow users to view tickets. One interface is for administrators and mentors (Figure 5.2), while another is for participants (Figure 5.3), offering creation and detail viewing functionalities.

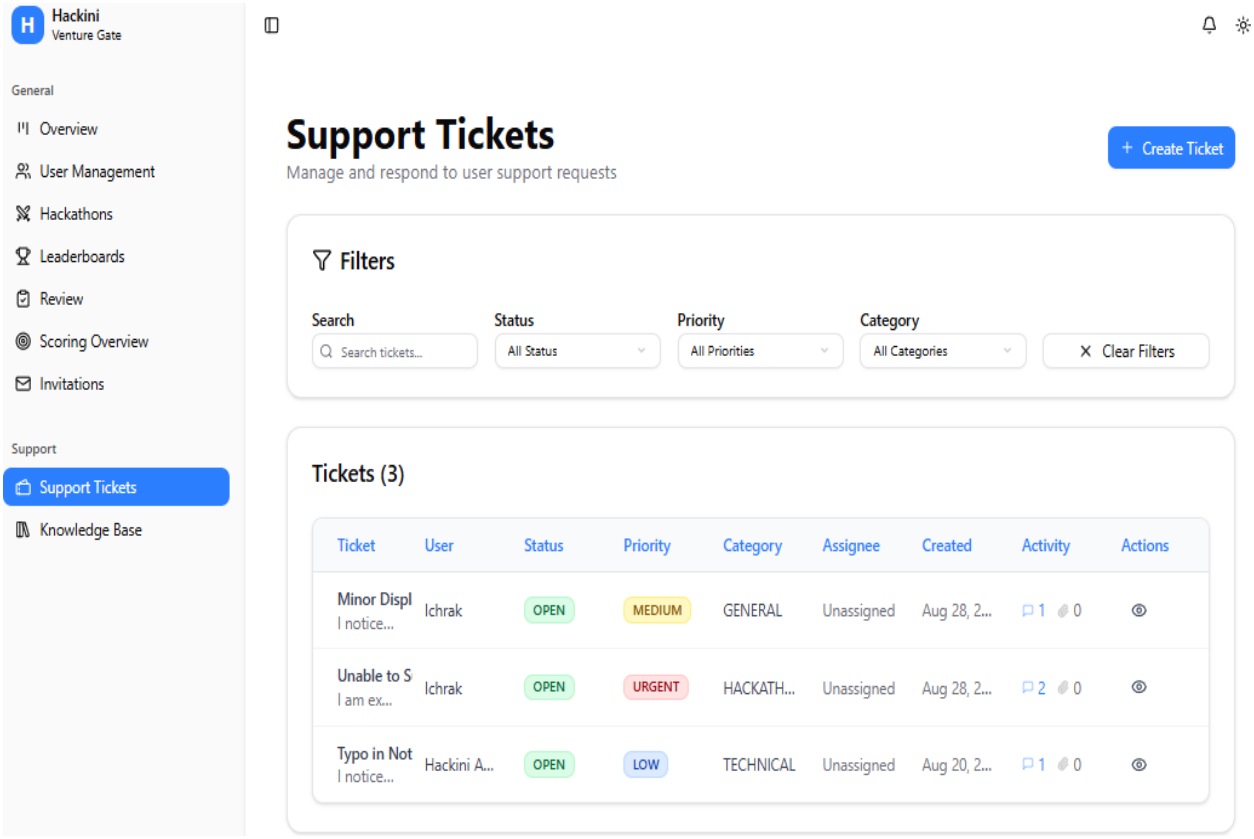


FIGURE 5.2 – Ticket Management Interface (Venture)

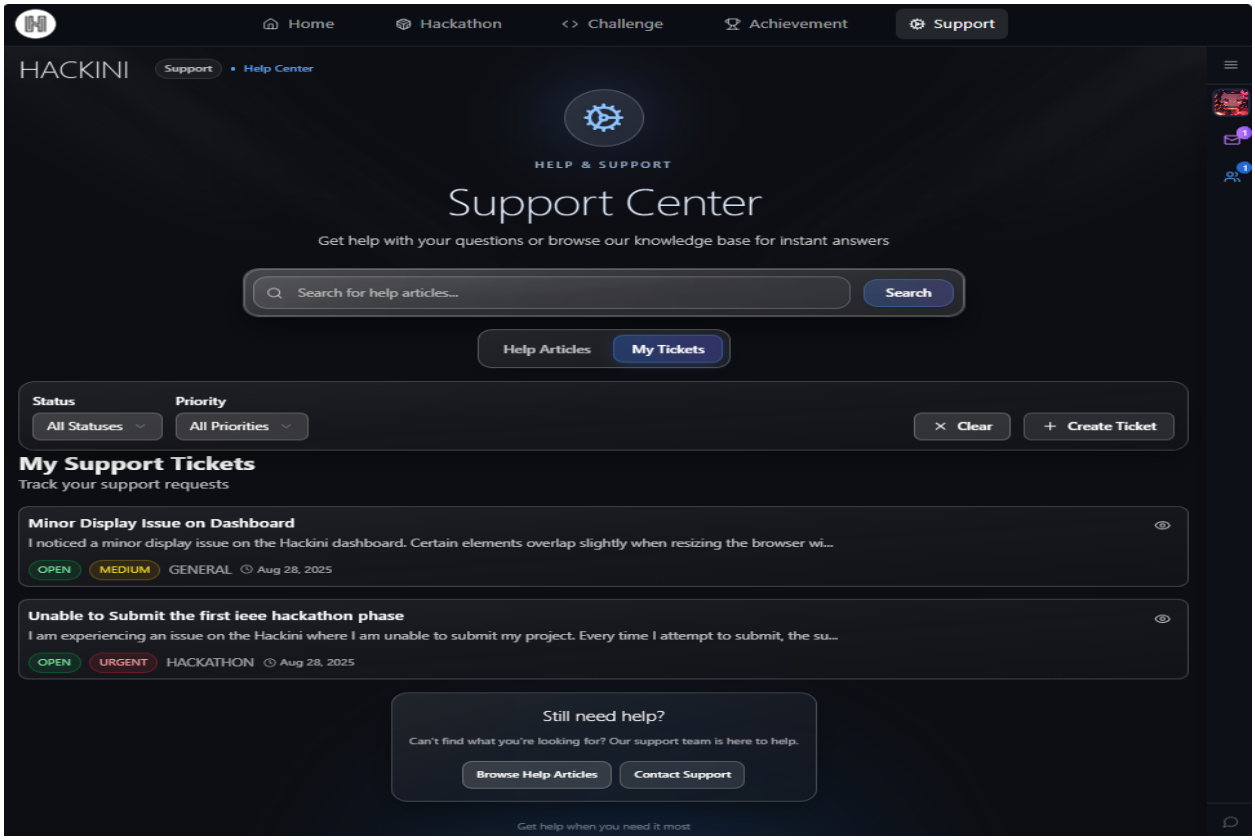


FIGURE 5.3 – Ticket Management Interface (Arena)

Clicking the eye icon lets users access ticket details and discuss the request with the responsible person (Figures 5.4 and 5.5 depending on the role).

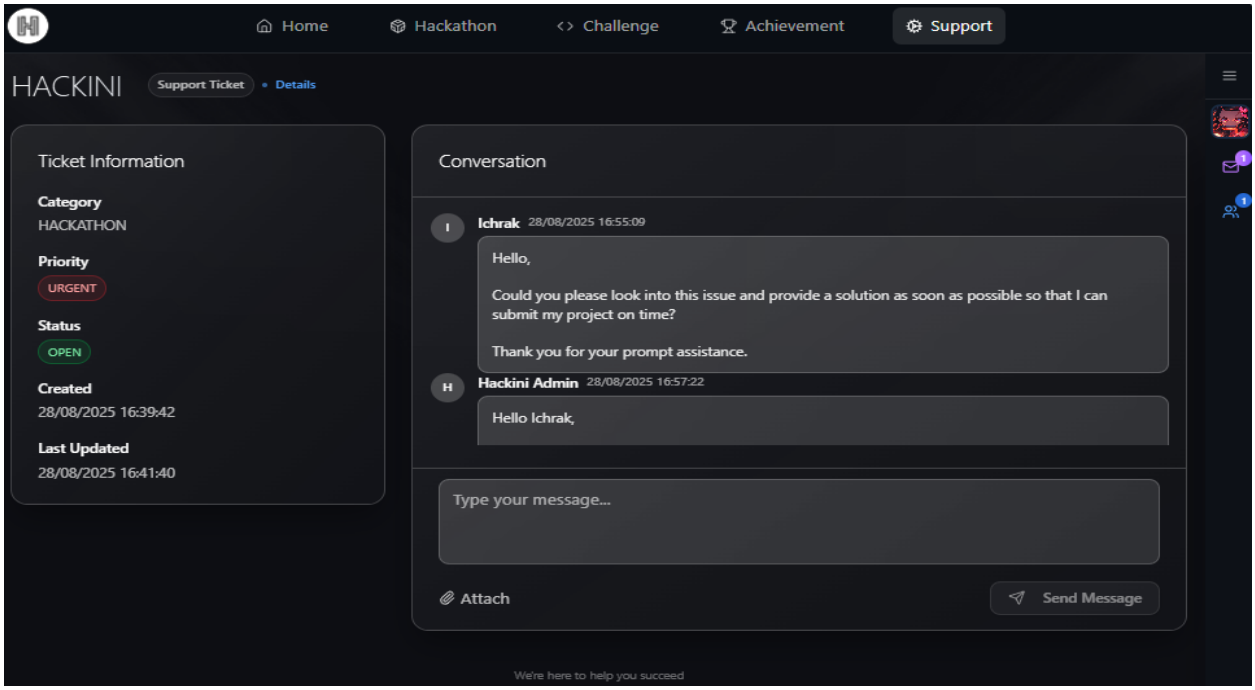


FIGURE 5.4 – Ticket Details Interface (Arena)

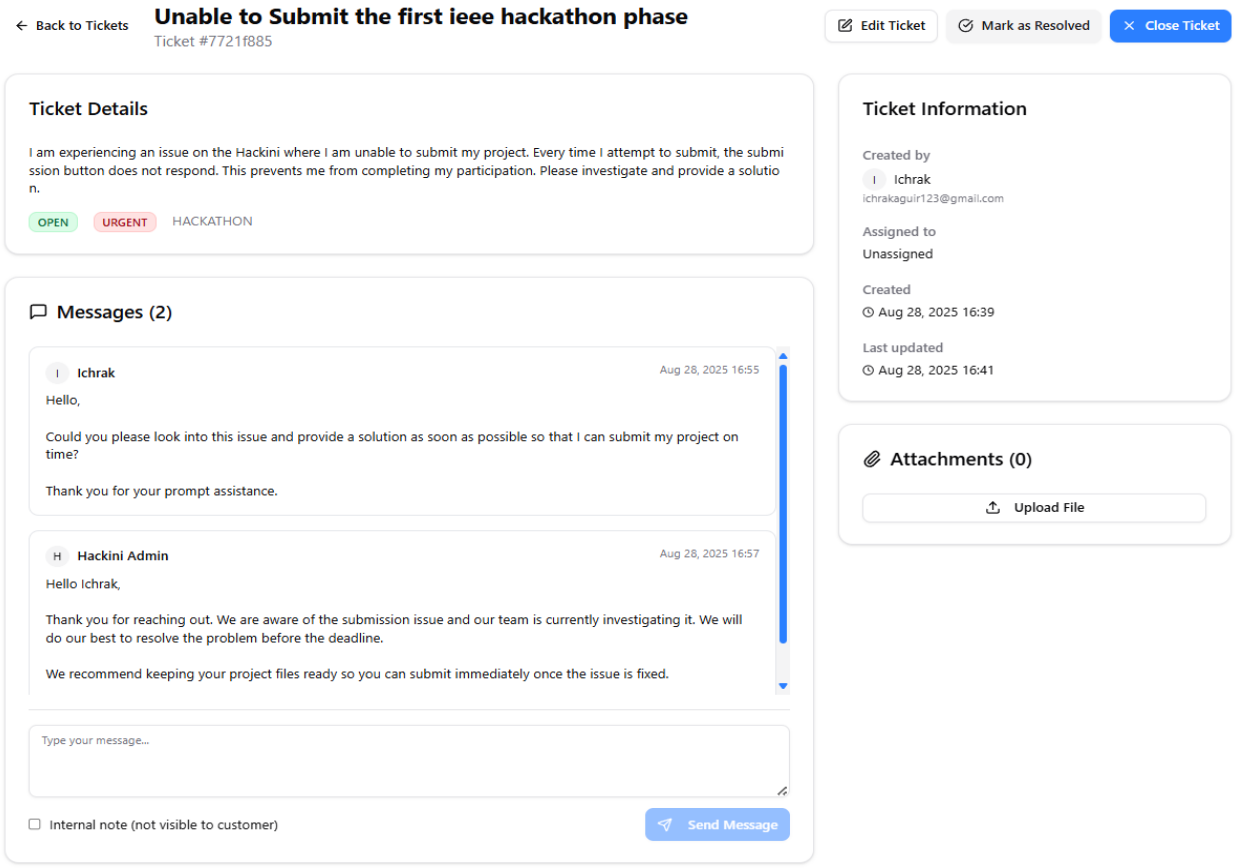


FIGURE 5.5 – Ticket Details Interface (Venture)

Users can submit tickets with a title, description, category, and priority via the interfaces shown in Figures 5.6 and 5.7.

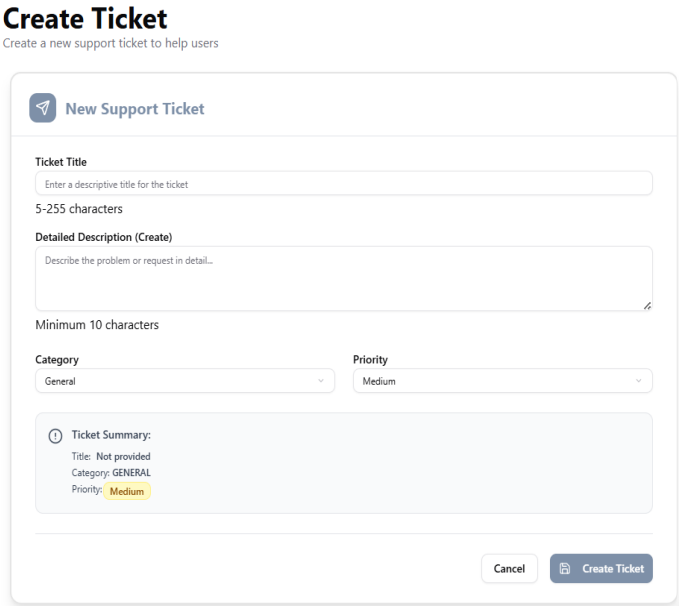


FIGURE 5.6 – Ticket Creation Interface (Venture)

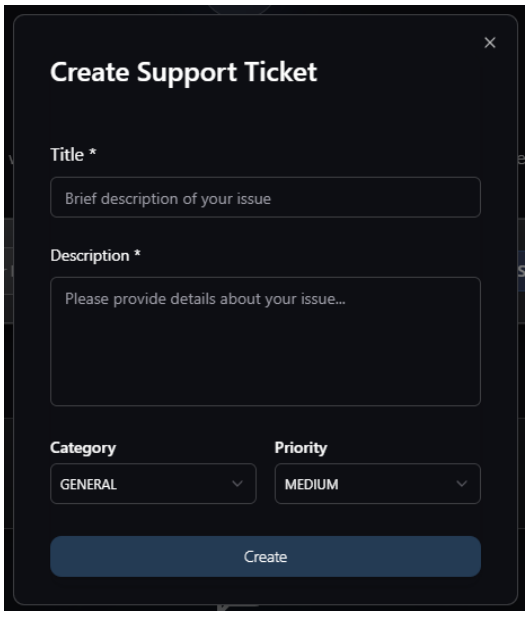


FIGURE 5.7 – Ticket Creation Interface (Arena)

Administrators can also edit tickets using the interface shown in Figure 5.8.

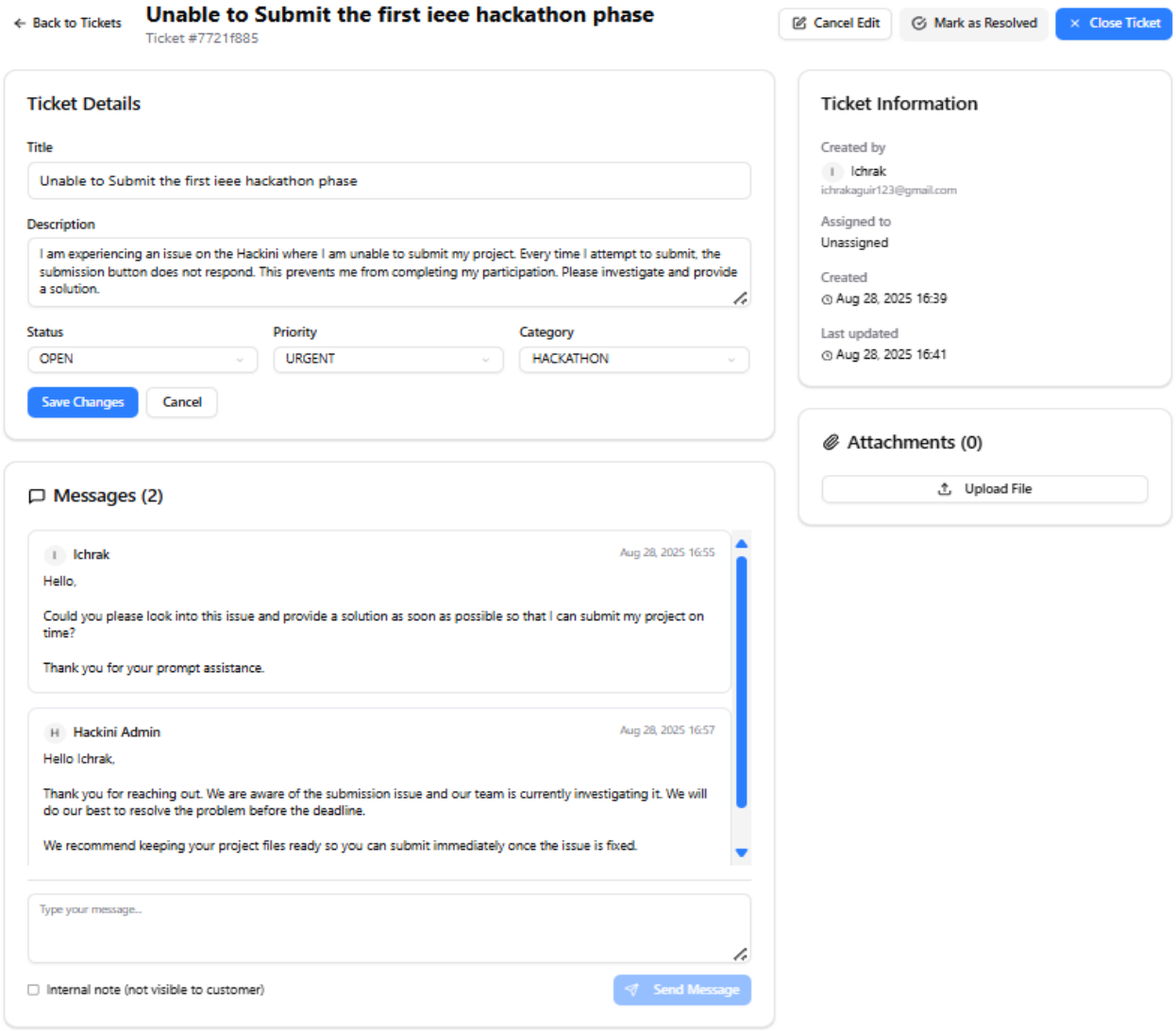


FIGURE 5.8 – Ticket Editing Interface

5.4 Invitation Management

This section presents the invitation management process, including use case diagrams and the associated interfaces.

5.4.1 Use Case Diagram for Invitation Management

The use case diagram for managing invitations is shown in Figure 5.9.

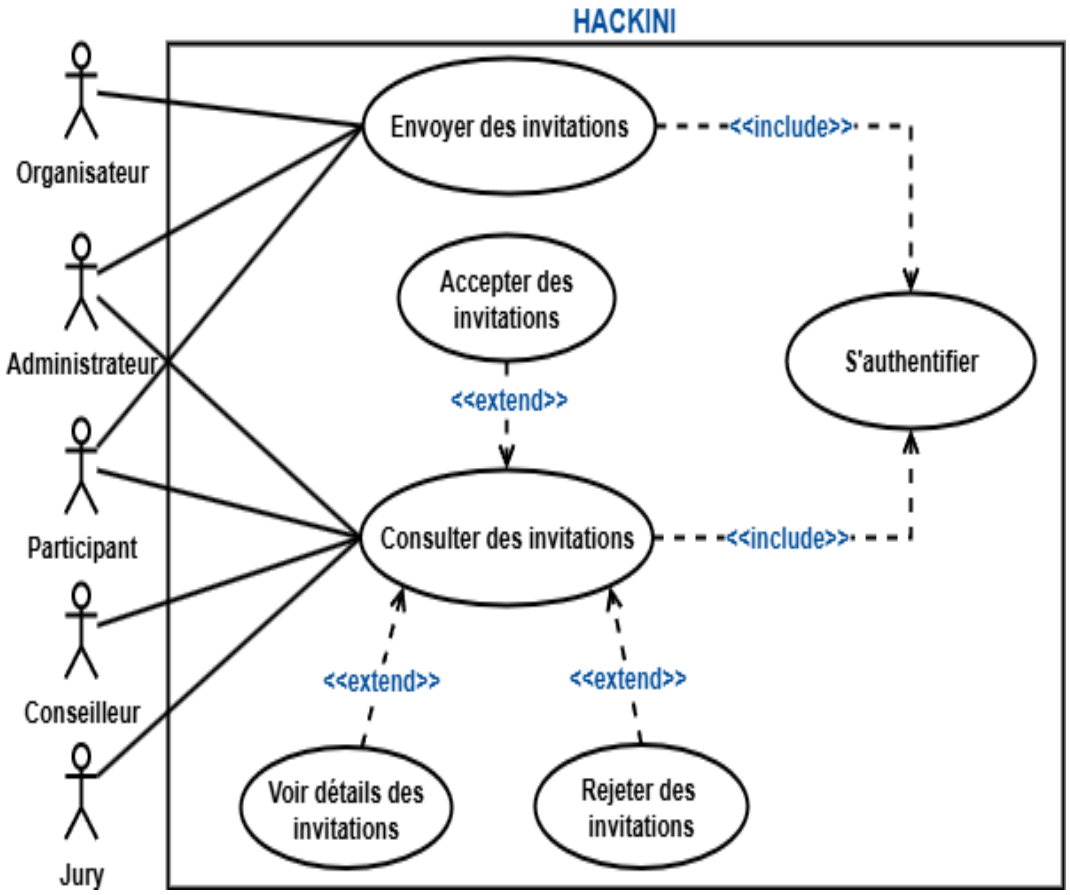


FIGURE 5.9 – Use Case Diagram for the "Manage Invitations" Scenario

5.4.2 Textual Description of Invitation Management Use Cases

Table 5.4 describes the front-office invitation management, and Table 5.5 details the back-office scenario.

5.4.3 Invitation Management Interfaces

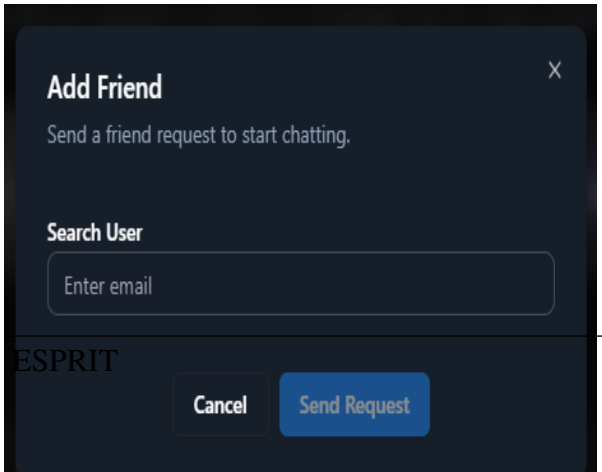
Participants can send invitations to other users to establish friendships (Figure 5.10). Groups can invite participants to join them (Figure 5.11). In Venture, administrators and organizers can send invitations to jury members and mentors for each hackathon phase (Figure 5.12).

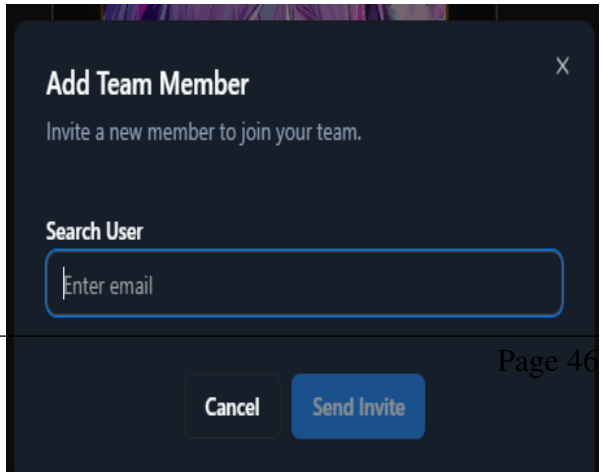
TABLE 5.4 – Textual Description of the "Manage Invitations" Use Case (Arena)

Use Case	Manage Invitations
Actors	Participant, Mentor
Preconditions	User must have an account and be authenticated.
Postconditions	Invitations are correctly sent, received, or viewed.
Main Scenario	1. User accesses the menu and views the list of received invitations and friends. 2. User can send a new invitation by clicking the button at the bottom. 3. System saves and delivers the invitation. 4. User can accept or reject a received invitation. 5. System updates the friends list or deletes the invitation based on the action. 6. User can view a friend's profile or remove them from the list. 7. System displays profile details or deletes the selected friend.
Alternative Scenario	If sending the invitation fails, an error message is displayed.

TABLE 5.5 – Textual Description of the "Manage Invitations" Use Case (Venture)

Use Case	Manage Invitations
Actors	Administrator, Organizer, Jury
Preconditions	User must have an account and be authenticated.
Postconditions	Invitations are correctly sent, received, or viewed.
Main Scenario	1. Administrator or organizer can send an invitation by clicking the "Invite" button in the jury interface. 2. System saves the invitation and notifies the sender by email. 3. User accesses the "Invitations" section. 4. User selects an invitation. 5. System displays the details of the selected invitation. 6. User accepts (✔) or declines (✖) the invitation. 7. System updates the list based on the action taken.
Alternative Scenario	If no invitation is available, an informational message is displayed.





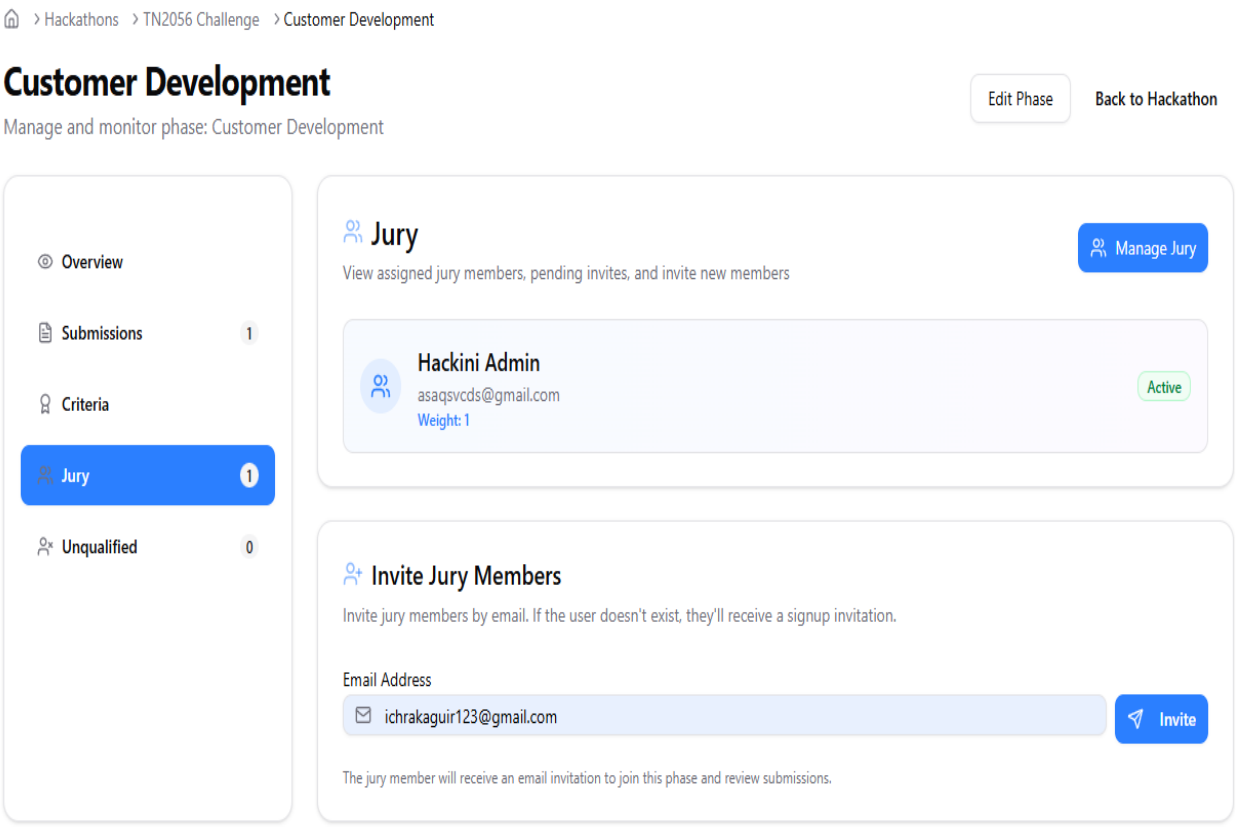


FIGURE 5.12 – Invitation Sending Interface (Venture)

For each invitation, the sender is notified by email : Figure 5.15 shows the invitation sent to a jury or mentor, and Figure 5.16 shows one sent by a group.

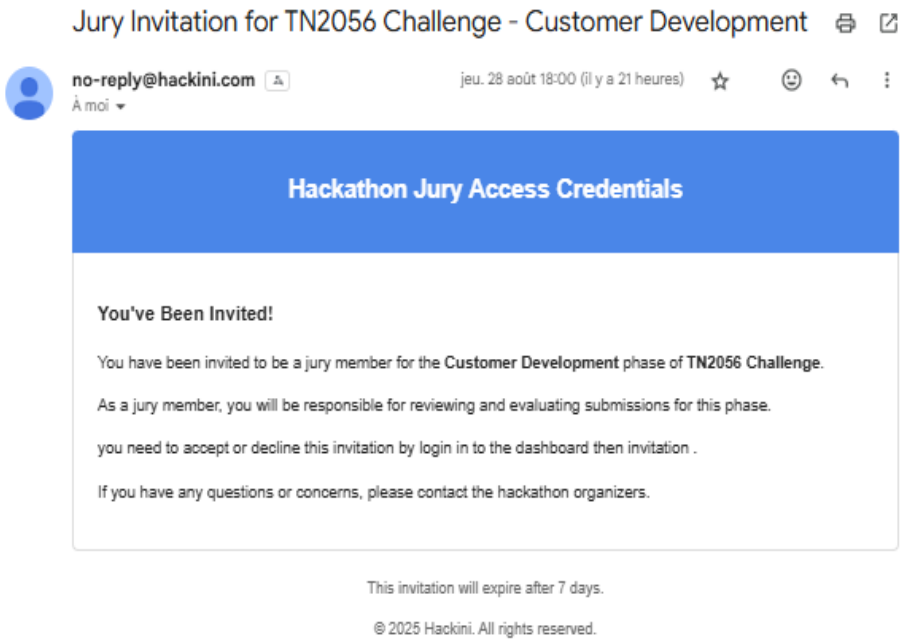


FIGURE 5.13 – Invitation Email Sent to Jury or Mentors

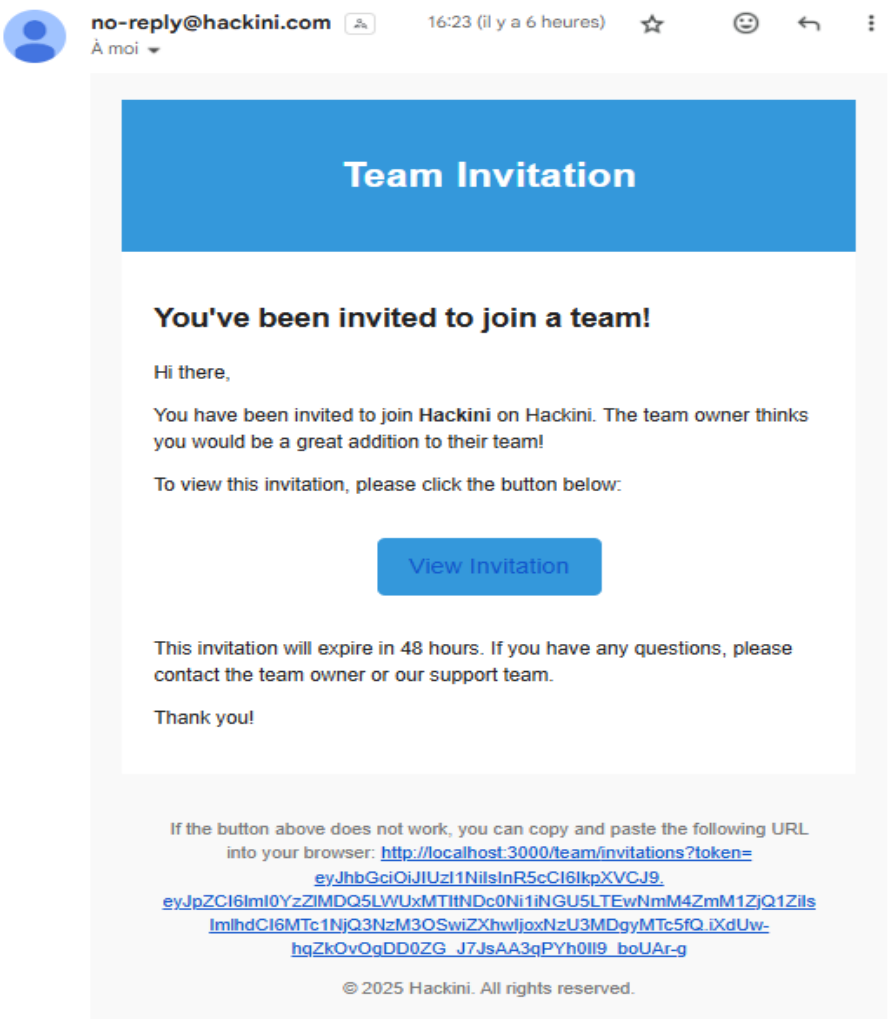


FIGURE 5.14 – Group Invitation Email

To view invitations, Figure 5.15 is for participants and mentors, while Figure 5.16 is for administrators and jury members.

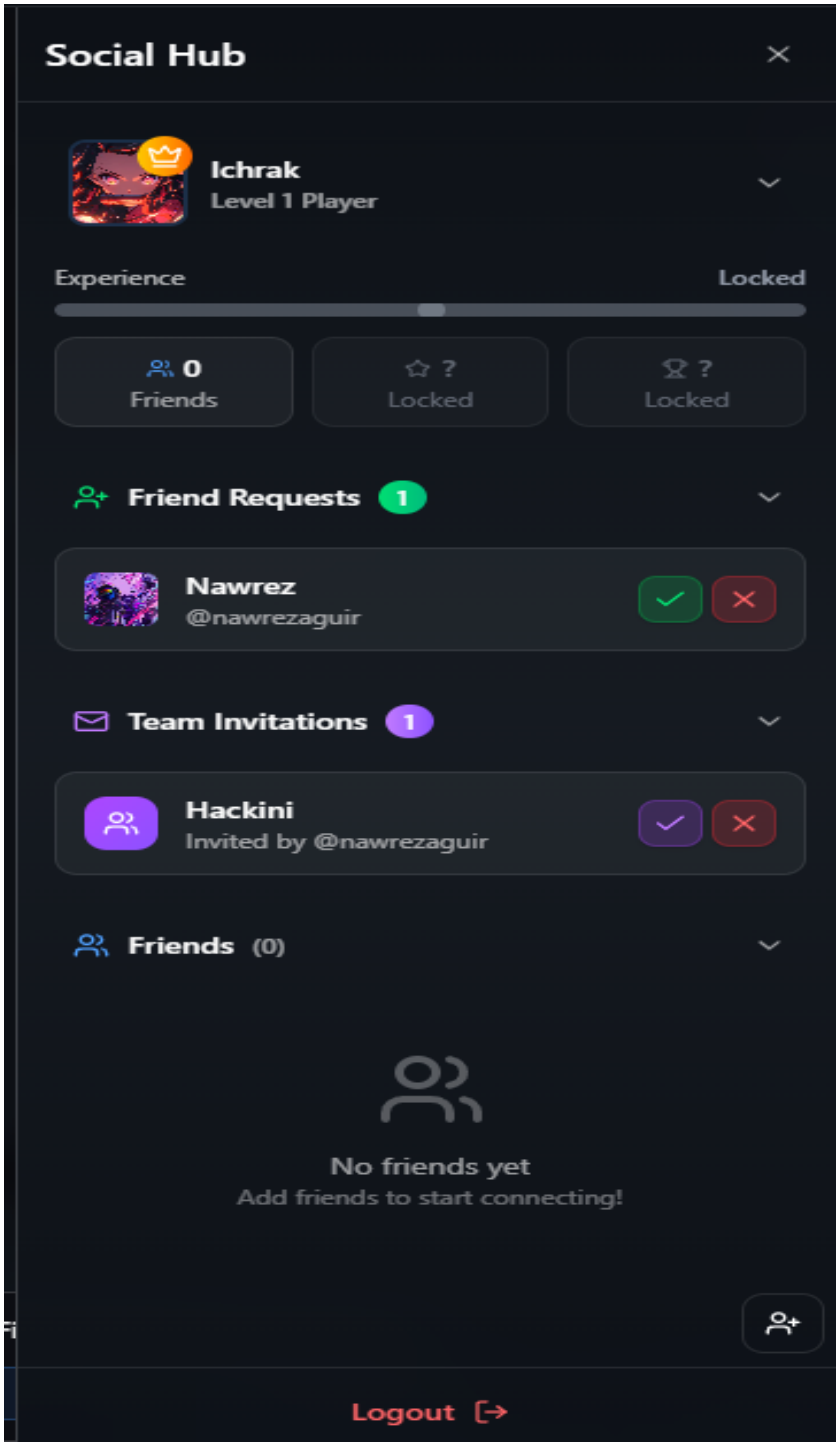


FIGURE 5.15 – Invitation Viewing Interface (Arena)

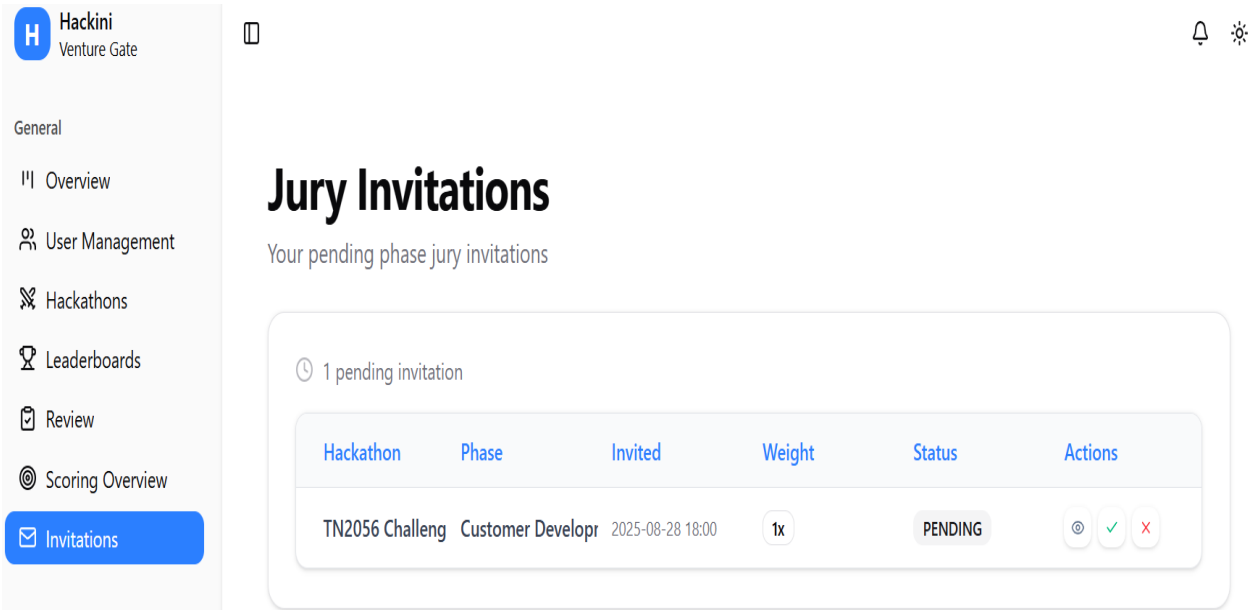


FIGURE 5.16 – Invitation Viewing Interface (Venture)

Venture users can view invitation details by clicking the eye icon, displaying the interface in Figure 5.17.

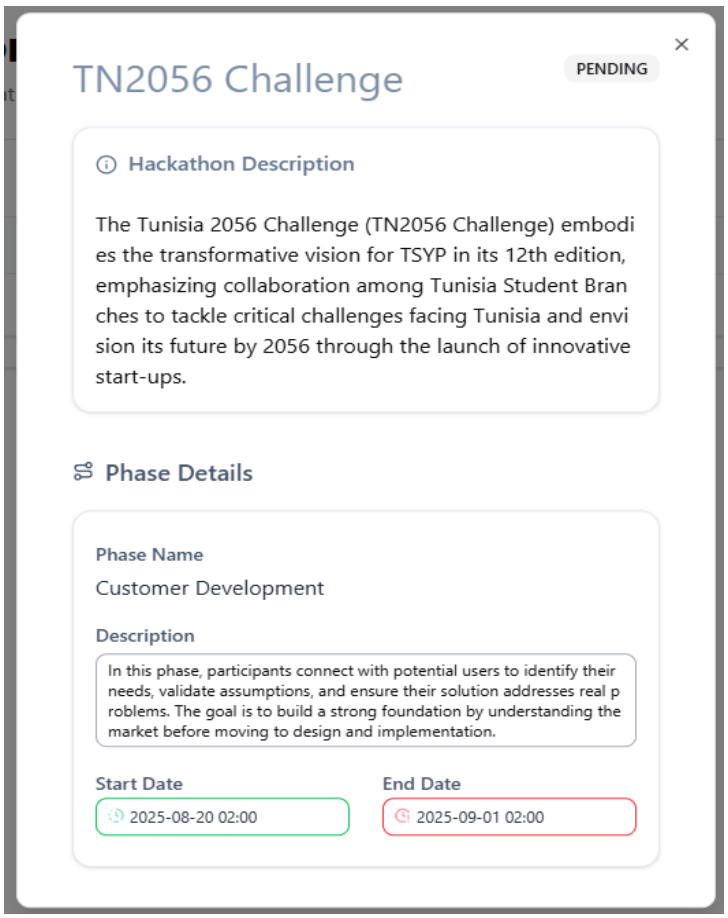


FIGURE 5.17 – Invitation Details Interface (Venture)

Arena users can view their friends list (Figure 5.18), view profiles (Figure 5.20), or remove friends using options shown in Figure 5.19.

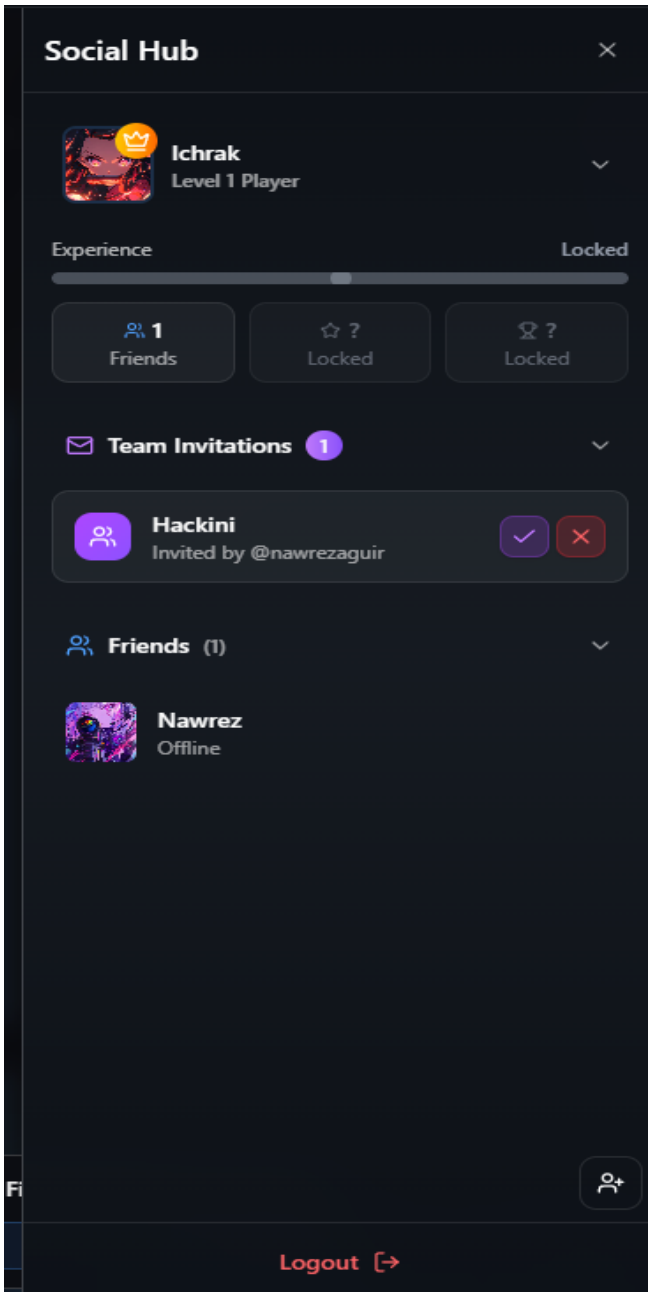


FIGURE 5.18 – Friends List Interface (Arena)

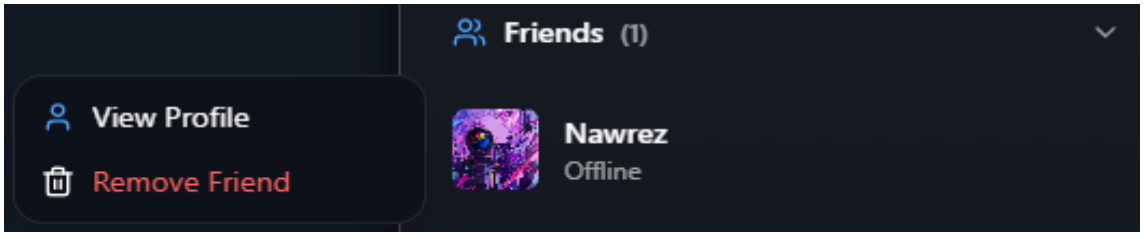


FIGURE 5.19 – Friend Actions Interface (Arena)

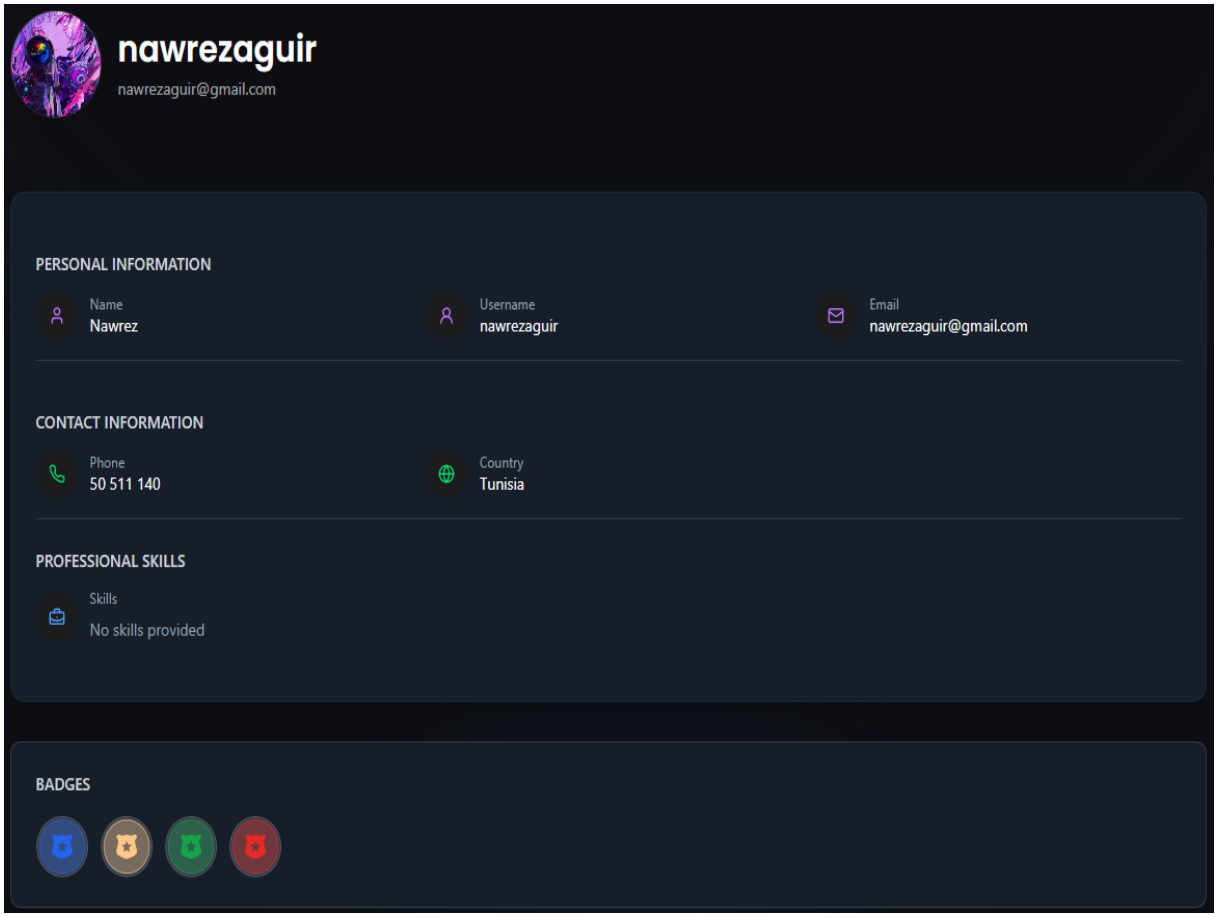


FIGURE 5.20 – Friend Profile Viewing Interface (Arena)

5.5 Conclusion

This third sprint implemented ticket and invitation management functionalities, allowing users, according to their roles, to create, view, and edit tickets, as well as send and receive invitations. These improvements enhance interaction and communication on the platform. The next sprint (Sprint 4) will focus on hackathon participation and solution submission.

Chapitre

6 Sprint 4 - Hackathon Participation Steps

Sommaire

6.1 Introduction

The fourth sprint focuses on the steps of participating in hackathons. It includes the management of **whiteboard** and **Kanban board**, displaying **hackathon details**, their **phases**, **submissions**, as well as the **leaderboard** to consult rankings and scores of participating teams. These features allow participants to effectively track project progress, collaborate in a structured way, and visualize team performance on the platform.

6.2 Sprint 4 Backlog

Table 6.1 below presents the backlog for sprint 4.

TABLE 6.1 – Sprint 4 Backlog

ID	User Story	Description	Acceptance Criteria	Priority
3	Display hackathon details, phases, and submissions	As a participant, I want to view the hackathon and phase details to track submissions.	- User can see general hackathon information. - User can view phases and their details. - User can visualize overall and phase-specific rankings via a leaderboard. - Participants can collaborate in real time via a whiteboard. - Participants can organize and track work using a Kanban board.	High
2	Whiteboard management	As a user, I want an interface to share my ideas with other members.	- User can create, edit, and delete notes on the whiteboard. - Team members can collaborate simultaneously. - Changes are saved in real time.	Medium
3	Kanban board management	As a user, I want to manage a board to track task progress.	- User can create, edit, and move tickets across columns. - User can add comments or assign members to tickets.	Medium

6.3 Participation Steps

This section presents the hackathon participation process through the corresponding interfaces.

6.3.1 Textual Description of Hackathon Participation Steps

6.3.1.1 Viewing Hackathon and Phase Details

Table 6.2 presents the textual description of participating in a hackathon.

TABLE 6.2 – Textual Description of Use Case "Participate in a Hackathon"

Use Case	Viewing Hackathon and Phase Details
Actors	Participant, Advisor
Preconditions	User must be authenticated and have access to the selected hackathon.
Postconditions	Hackathon and accessible phase information is accurately displayed.
Main Scenario	1. User selects a hackathon from the list of available hackathons. 2. The participant is redirected to the team member invitation interface. 3. The participant joins the hackathon. 4. The system displays general hackathon information and the list of its phases. 5. User can access the whiteboard or Kanban board interface to manage ideas and tasks. 6. User can view phase details only if its start date has been reached and the previous phase is validated. 7. Participant can submit their solution.
Alternative Scenario	If the phase is not yet accessible, the system informs the user and prevents access to the details.

6.3.1.2 Textual Description of Collaborative Whiteboard Use Case

Table 6.3 provides the textual description for the use case "Collaborative Whiteboard" :

6.3.1.3 Textual Description of Kanban Management

Table 6.4 provides the textual description of the use case "Kanban Management" :

6.3.1.4 Textual Description of Solution Submission

Table 6.5 describes the textual description of the use case "Solution Submission" :

TABLE 6.3 – Textual Description of Use Case "Collaborative Whiteboard"

Use Case	Collaborative Whiteboard
Actors	Participant, Advisor
Preconditions	User must be authenticated and access the whiteboard interface.
Postconditions	Changes on the whiteboard are saved and visible in real time to authorized members.
Main Scenario	1. User clicks the "Hackini Studio" button to access the whiteboard. 2. User can add text, drawings, or other graphic elements. 3. User can follow actions of other participants in real time. 4. System automatically saves all actions performed on the whiteboard.
Alternative Scenario	If the connection is interrupted or an error occurs while saving, the system displays an error message and allows the user to retry.

TABLE 6.4 – Textual Description of Use Case "Kanban Management"

Use Case	Kanban Management
Actors	Participant, Advisor
Preconditions	User must be authenticated and access the Kanban interface.
Postconditions	Tasks are correctly added, modified, moved, and visible to authorized members.
Main Scenario	1. User clicks the "Hackini Kanban" button to access the Kanban board. 2. User can create new tasks with title, description, and priority. 3. User can move tasks across columns (To Do, In Progress, Done) to track progress. 4. User can edit existing tasks. 5. Changes are visible in real time to other members.
Alternative Scenario	If an operation fails, the system displays an error message.

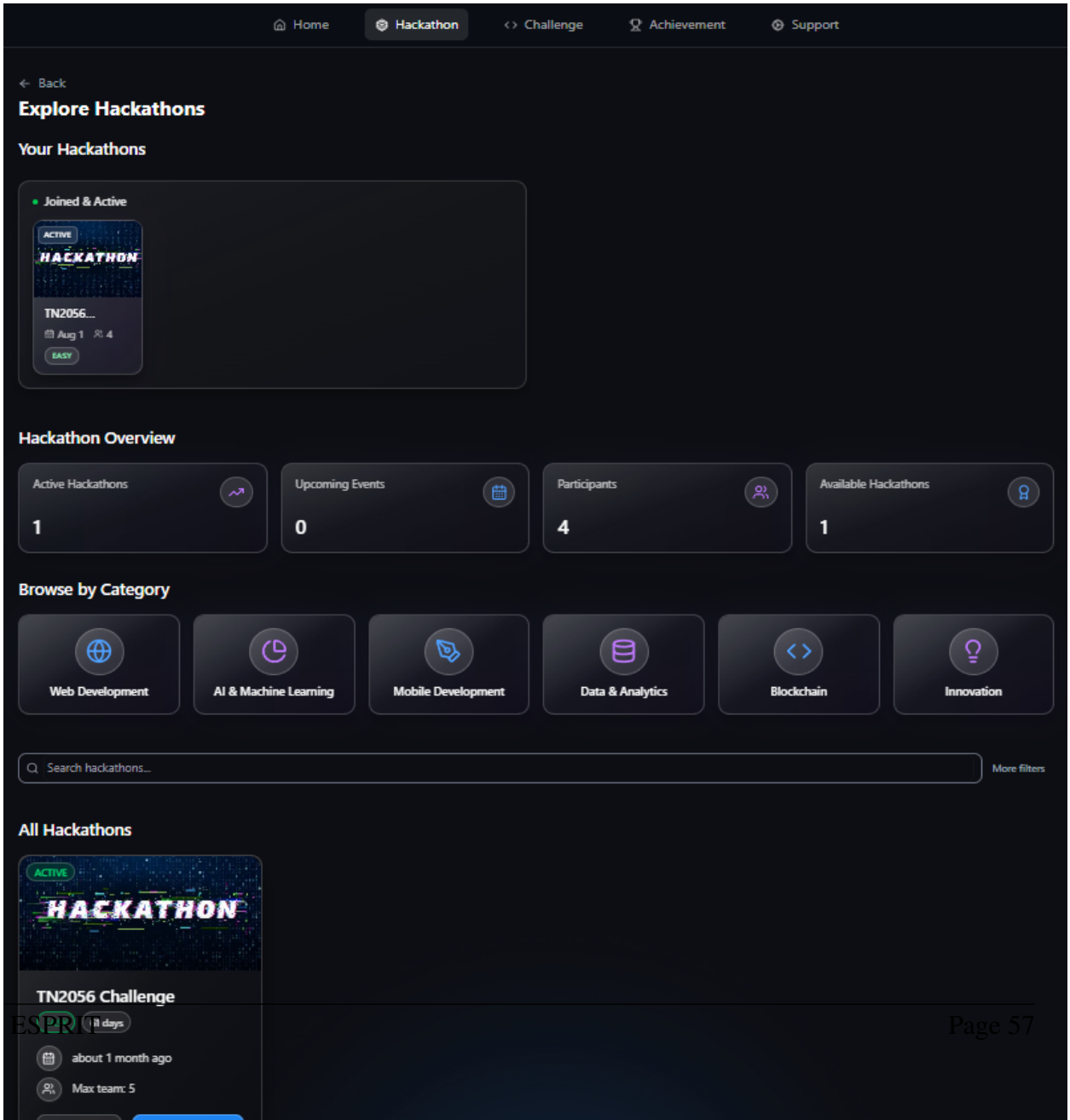
6.3.2 Participation Feature Interfaces

The following interfaces allow users to take part in a hackathon.

First, the participant accesses the interface shown in Figure 6.1 to view the list of available hackathons.

TABLE 6.5 – Textual Description of Use Case "Solution Submission"

Use Case	Solution Submission
Actors	Participant
Preconditions	The hackathon phase must be accessible and the user authenticated.
Postconditions	The solution is correctly submitted and recorded for the corresponding phase.
Main Scenario	1. User accesses the active hackathon phase. 2. Clicks the "Submit Project" button. 3. Fills in the submission form. 4. Clicks "Submit Project" to save the contribution. 5. System confirms the submission and makes it visible for evaluation.
Alternative Scenario	If the form is incomplete, the system displays an error message.



They can then get an overview of each hackathon via the sheet presented in Figure 6.2. By clicking the "Join Hackathon" or "Get Started" button, the user accesses their team invitation interface (Figure 6.3). By clicking "Join Hackathon," they can consult all hackathon details via the main interface shown in Figure 6.4.

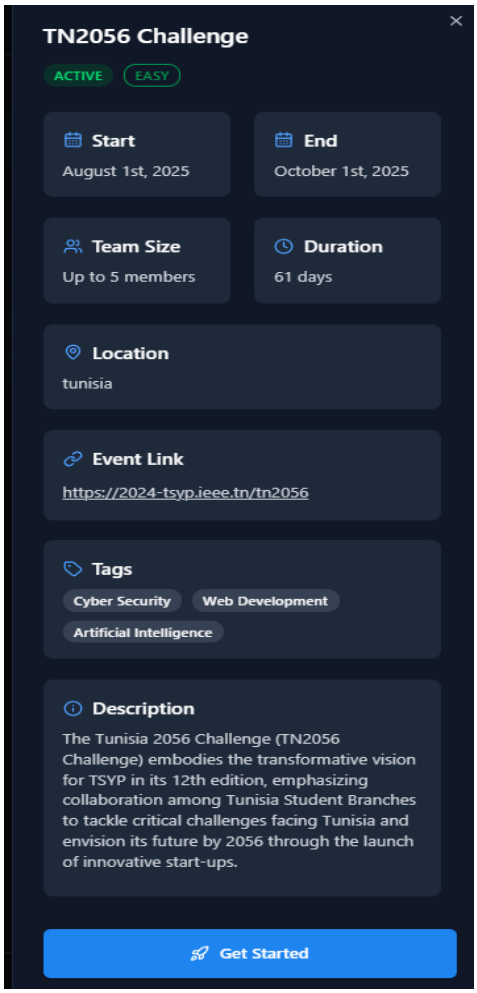


FIGURE 6.2 – Hackathon Details Interface

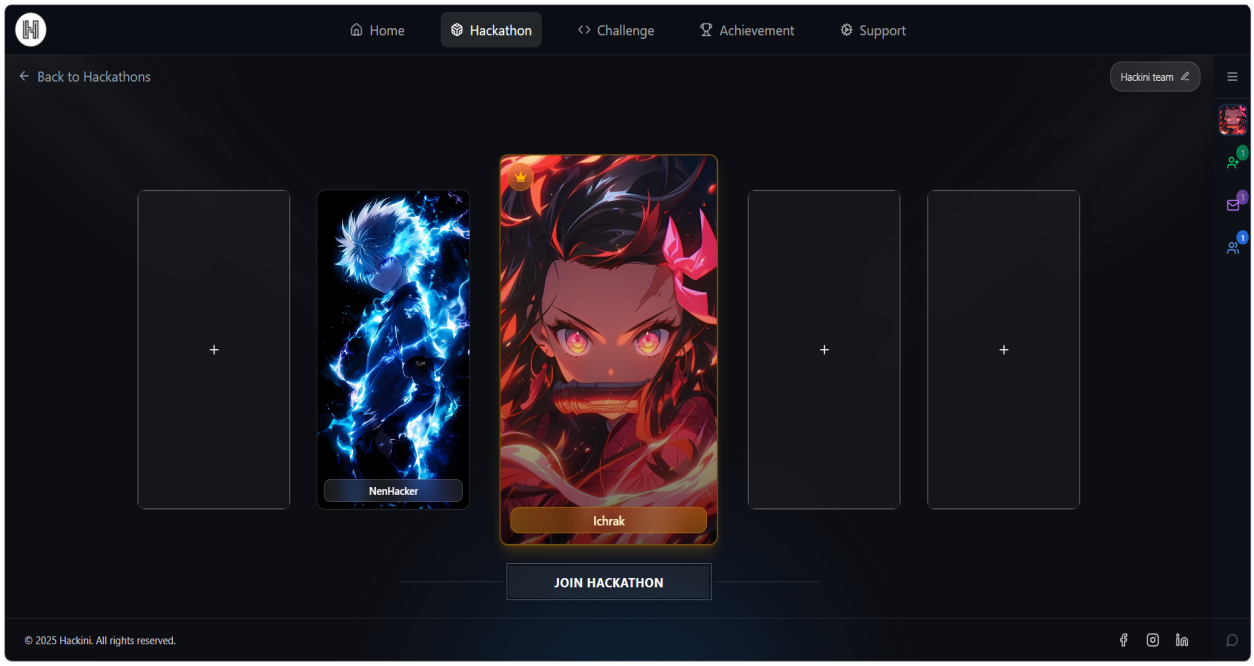


FIGURE 6.3 – Team Invitation Interface

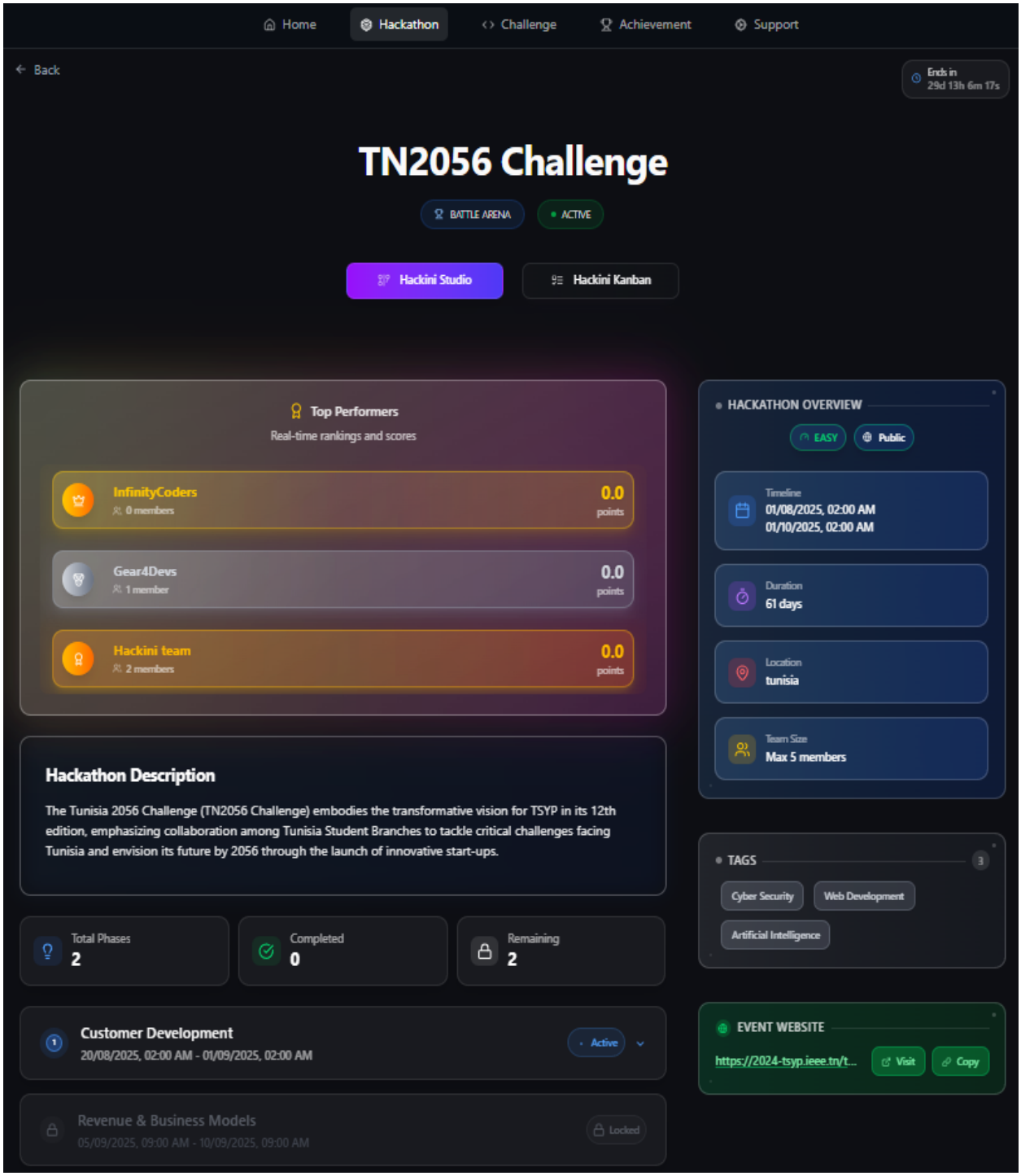


FIGURE 6.4 – Main Hackathon Interface

The hackathon provides participants with a collaborative whiteboard, accessible via the Hackini Studio interface (Figure 6.5). Users can write, draw, and view in real time the actions of their team members, promoting effective and interactive collaboration.

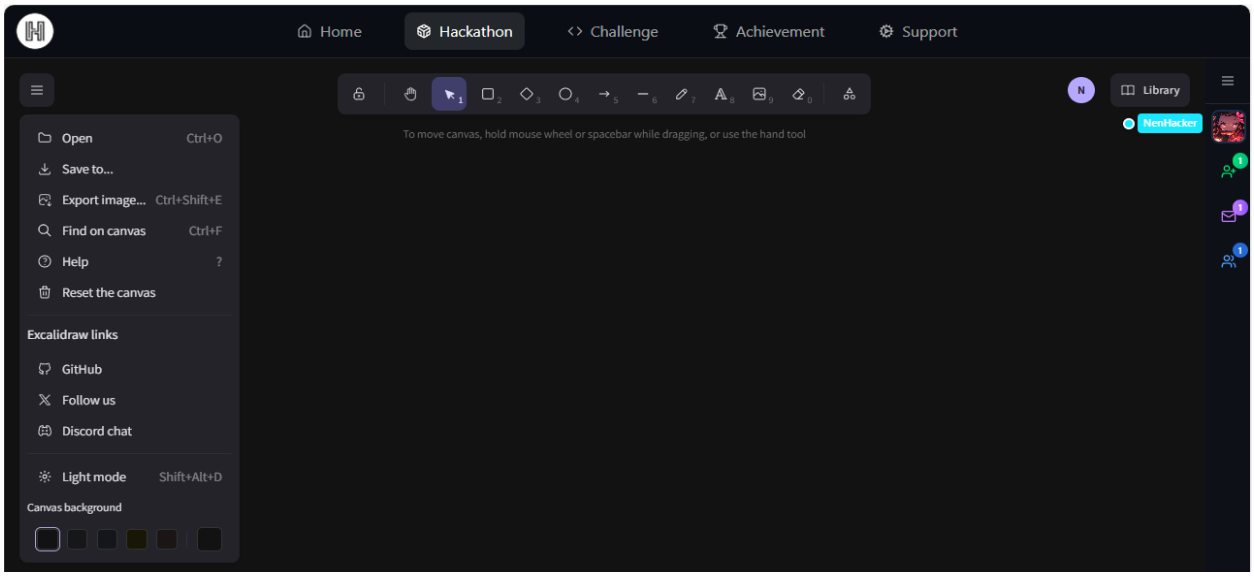


FIGURE 6.5 – Whiteboard Interface

In parallel, a Kanban board (Figure 6.6) is available for task management, including creation (Figure 6.7), modification, and consultation (Figure 6.8).

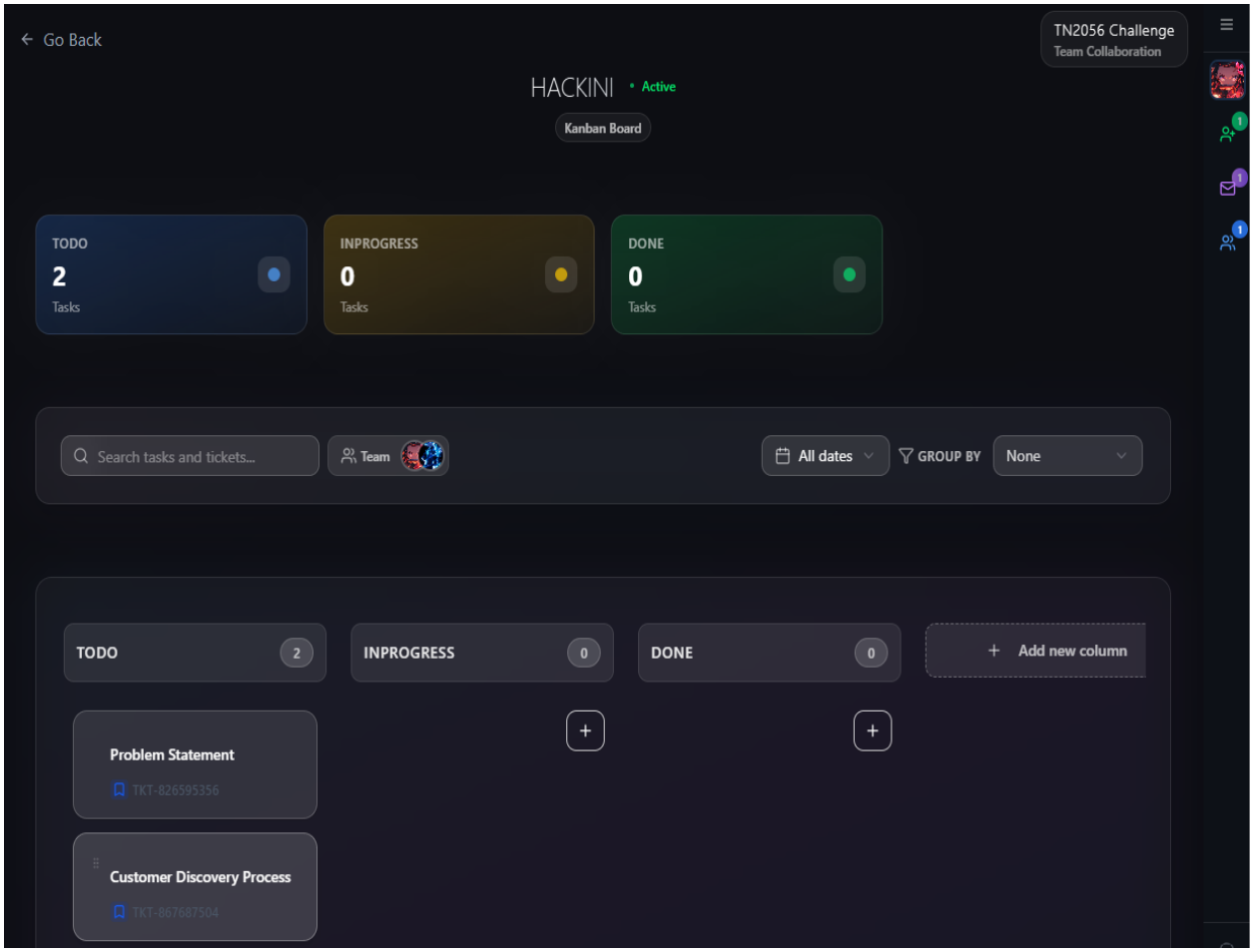


FIGURE 6.6 – Kanban Board Interface

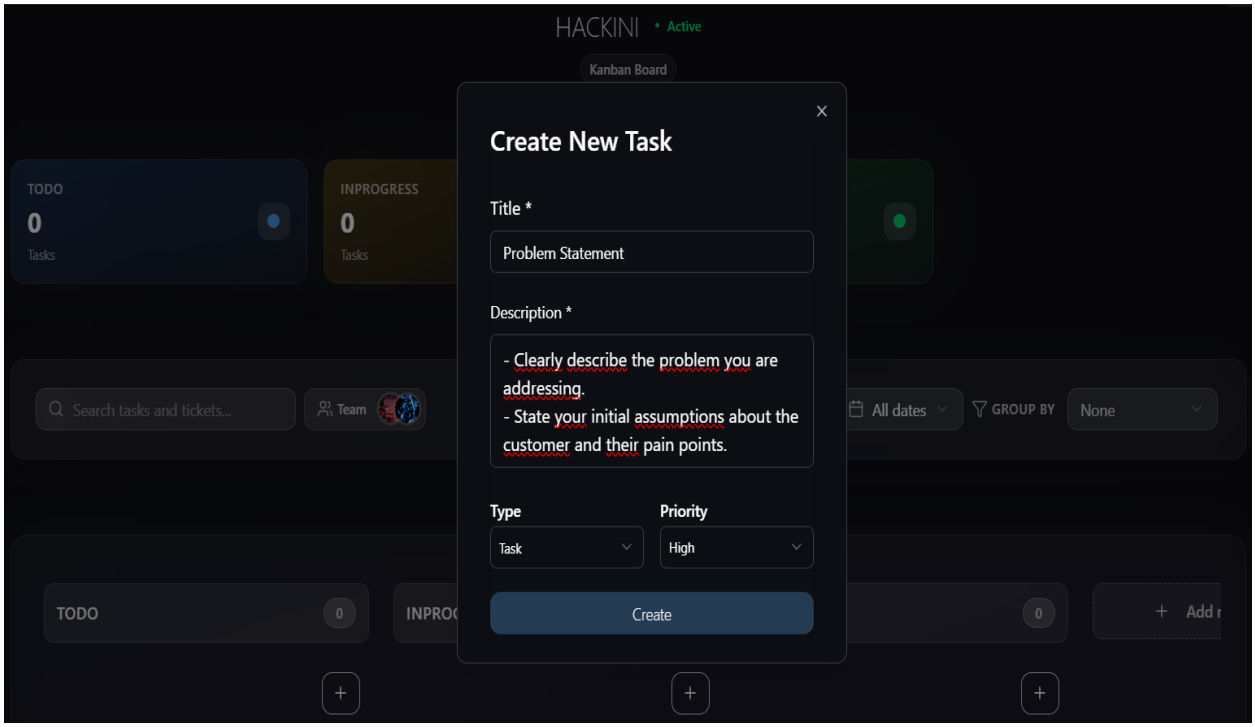


FIGURE 6.7 – Task Creation Interface

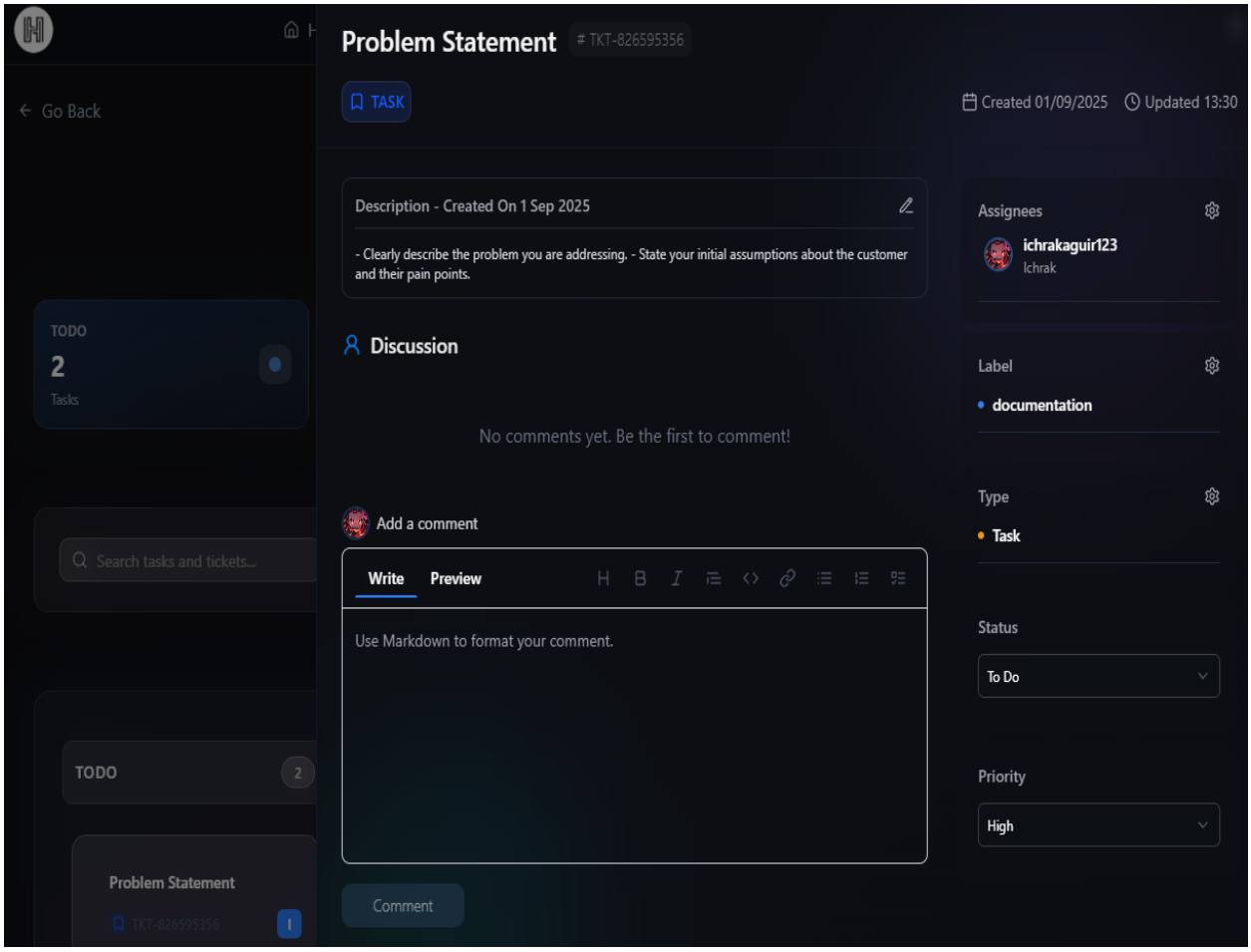


FIGURE 6.8 – Task Consultation and Edit Interface

Participants can then view the details of different hackathon phases as shown in Figure 6.9. Each phase becomes accessible only if its start date is reached and the previous phase is validated, ensuring structured project tracking.

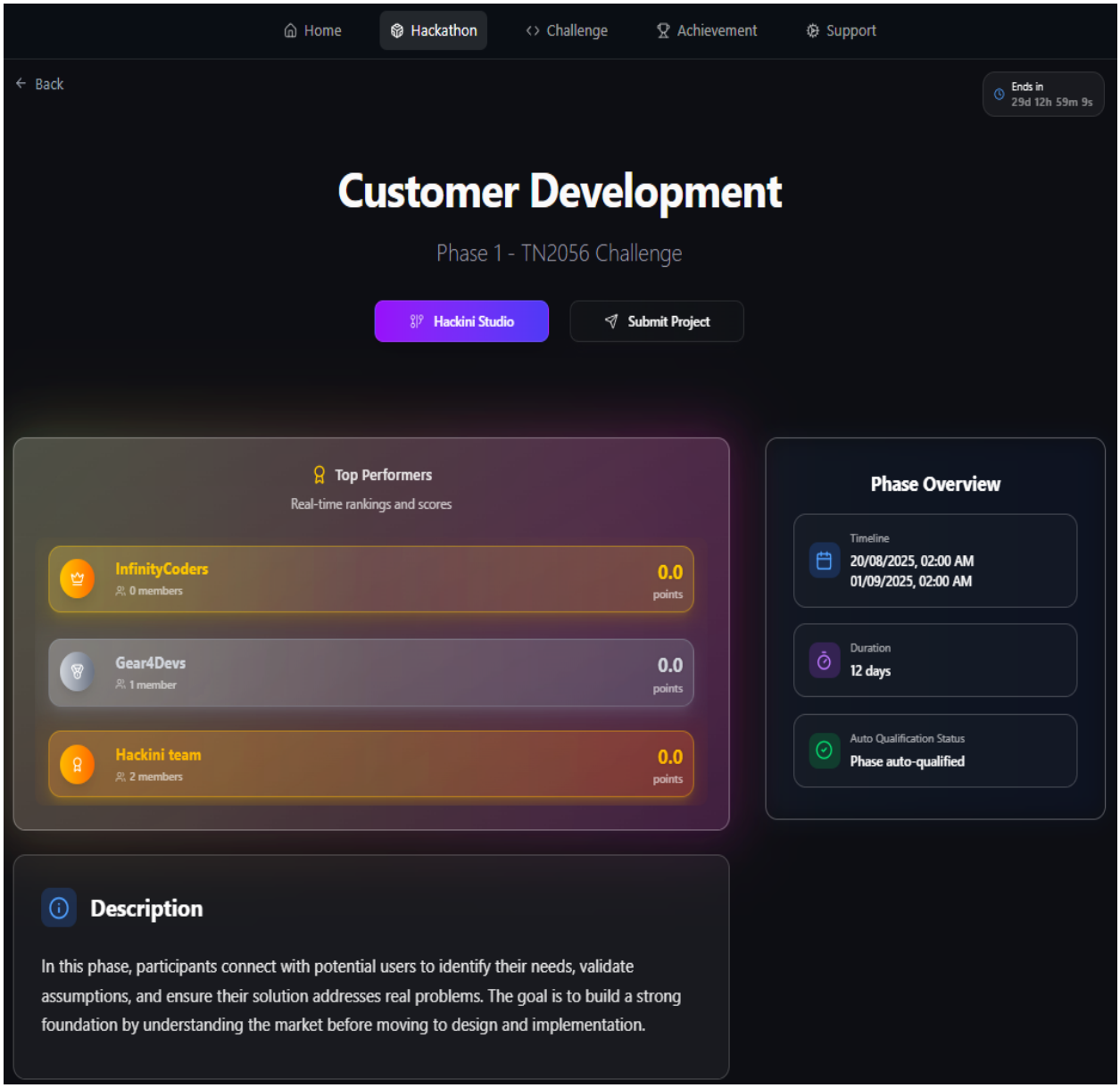


FIGURE 6.9 – Main Phase Interface

Once a phase is accessible, participants can submit their solutions via the dedicated interface (Figure 6.10). The system allows attaching required files and recording the submission, ensuring traceability and centralized management of contributions for each phase.

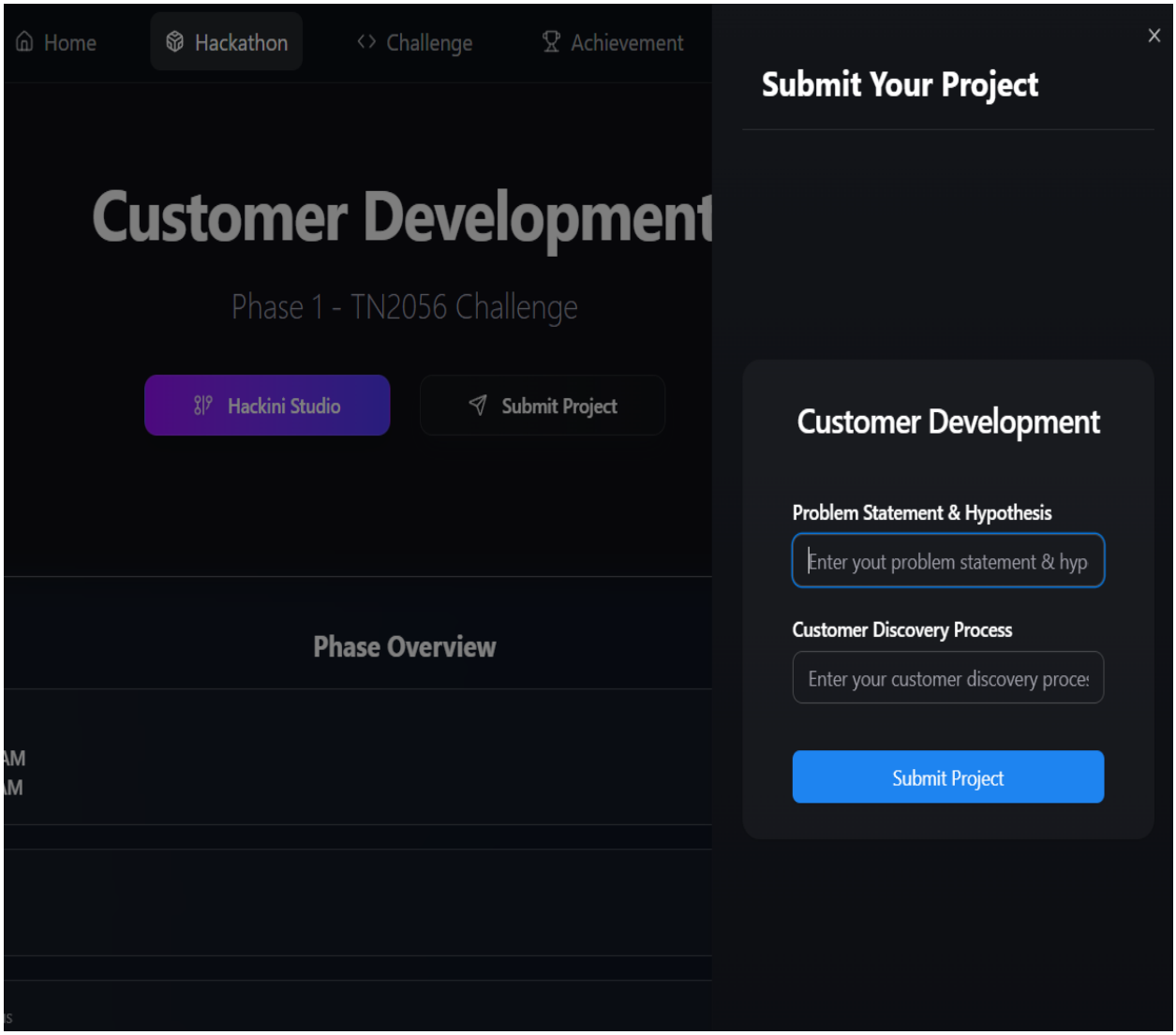


FIGURE 6.10 – Solution Submission Interface

6.4 Conclusion

This sprint strengthened hackathon participation features on the platform. With the integration of the **whiteboard** for real-time collaboration, the **Kanban board** for task management, detailed phase consultation, and **solution submission**, the application becomes more comprehensive and better suited to participant needs. These modules form the foundation for the next steps, focused on ...

Chapitre 7

Azure Cloud Architecture

7.1 Introduction

This chapter presents the Azure Cloud Architecture for the Hackini platform, which consists of three main applications : Hackini API, Hackini Arena, and Hackini Venture. The architecture is designed for scalability, security, and efficient DevOps automation, leveraging Azure's managed services and best practices.

7.2 Global Architecture Overview

7.2.1 Architecture Diagram

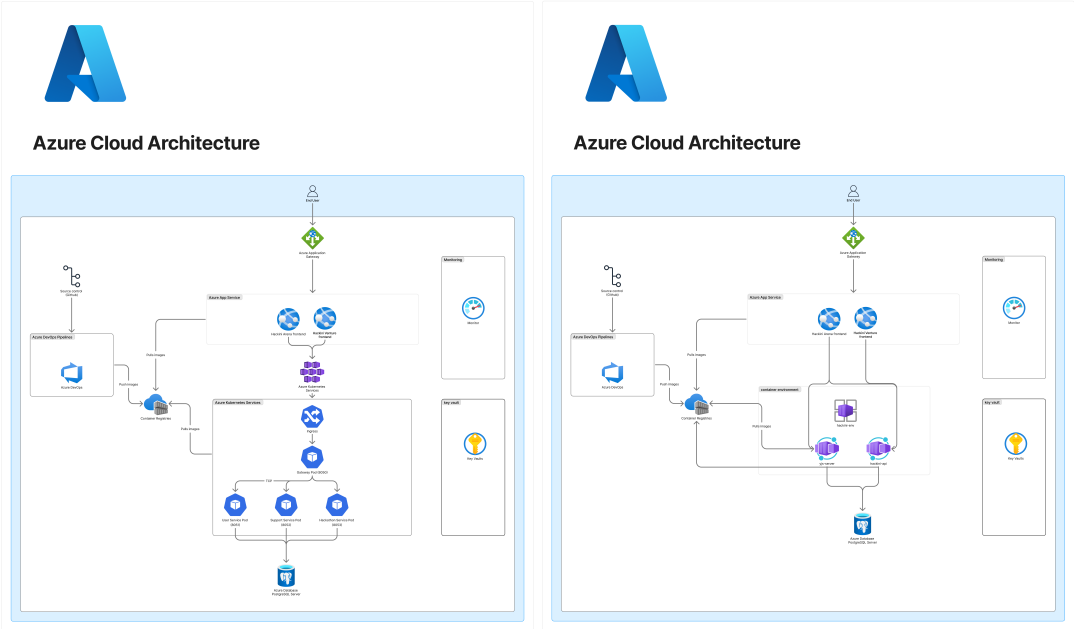


FIGURE 7.1 – Global Azure Cloud Architecture for Hackini Platform

The global architecture integrates all project components into a unified cloud environment, ensuring seamless communication, centralized security, and automated deployment.

7.3 Core Azure Services and Components

7.3.1 Resource Groups and Networking

— **Resource Groups :** Each environment (dev, staging, prod) is isolated in its own resource group for better management and security.

— **Virtual Network (VNet) :** All services are deployed within a secure VNet, with subnets for API, frontend apps, and databases.

7.3.2 Compute and Application Hosting

— **Azure Container Apps :** All backend services (API, microservices) are containerized and deployed as Azure Container Apps for scalability and isolation.

— **Azure App Service / Static Web Apps :** Frontend applications (Arena, Venture) are deployed as static web apps or containerized services.

7.3.3 Data Storage and Management

— **Azure SQL Database :** Centralized relational database for all persistent data.

— **Azure Blob Storage :** Stores user uploads, images, and other unstructured data.

7.3.4 Identity, Security, and Secrets Management

— **Azure Active Directory (AAD) :** Manages authentication and authorization for users and services.

— **Azure Key Vault :** Stores secrets, connection strings, and certificates securely.

— **Network Security Groups (NSG) :** Restrict access to resources within the VNet.

7.3.5 Monitoring, Logging, and Cost Management

— **Azure Monitor & Application Insights :** Collects telemetry, logs, and performance metrics for all services.

— **Azure Cost Management :** Tracks and optimizes resource usage and spending.

7.4 DevOps and CI/CD Pipeline

7.4.1 Pipeline Overview

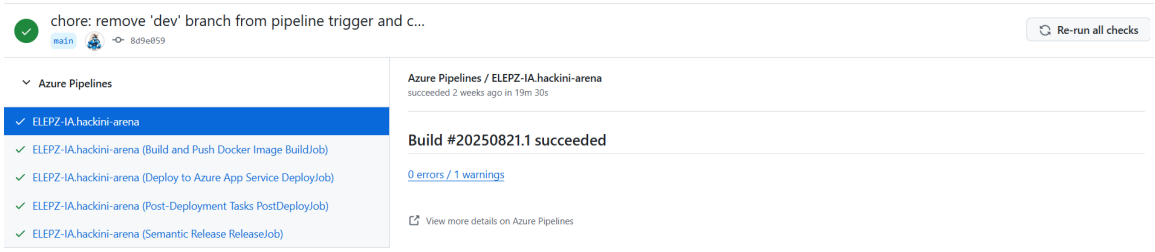


FIGURE 7.2 – CI/CD Pipeline with Azure DevOps

7.4.2 Pipeline Stages

— **Source Control :** All code is managed in Git repositories (monorepo for API, separate repos for Arena and Venture).

— **Build :** On each commit, Azure Pipelines build the code, run tests, and create Docker images.

— **Release :** Images are pushed to Azure Container Registry and deployed to Azure Container Apps.

— **Infrastructure as Code :** Bicep or ARM templates automate the provisioning of all Azure resources.

7.5 Application-Specific Architecture

7.5.1 Hackini API

— **Backend :** Built with NestJS, containerized, and deployed to Azure Container Apps.

— **Database :** Uses Azure SQL Database, with Prisma ORM for data access.

— **Secrets :** All sensitive data is managed via Azure Key Vault.

— **Endpoints :** Exposed securely within the VNet, accessible by frontend apps.

7.5.2 Hackini Arena

— **Frontend :** Next.js application, deployed as a static web app or container.

— **Integration :** Communicates with the API via secure endpoints.

— **Assets :** Static assets are stored in Azure Blob Storage.

— **Authentication :** Uses Azure AD for user login and role management.

7.5.3 Hackini Venture

— **Frontend :** Next.js application, similar deployment as Arena.

— **Features :** Handles hackathon management, team collaboration, and analytics.

— **CI/CD :** Automated via Azure Pipelines, with environment-specific configurations.

7.6 Security and Compliance

— **Role-Based Access Control (RBAC) :** Enforced at the resource group and service level.

— **Data Encryption :** All data at rest and in transit is encrypted.

— **Secret Rotation :** Automated via Azure Key Vault policies.

7.7 Scalability and High Availability

— **Auto-scaling :** Azure Container Apps scale automatically based on load.

— **Geo-redundancy :** Azure SQL Database and Blob Storage are configured for geo-replication.

— **Load Balancing :** Azure Front Door or Application Gateway distributes traffic efficiently.

7.8 Conclusion

The Hackini platform's Azure Cloud Architecture ensures robust, secure, and scalable operations for all its applications. By leveraging Azure's managed services, the platform achieves rapid

deployment, centralized management, and high availability, supporting the needs of both developers and end-users.

Chapitre 8

Mermaid Diagram Generation Agent

8.1 Introduction

8.1.1 Background

Diagram generation is essential in software engineering and documentation. Manual creation is time-consuming and error-prone. Automating this process using natural language and AI can improve productivity and accessibility.

8.1.2 Purpose

This project aims to build an intelligent agent that generates valid Mermaid diagrams from user prompts, supporting a wide range of diagram types and use cases.

8.1.3 Scope

The system includes a web interface, AI-powered backend, context retrieval, and SVG rendering. It supports multiple diagram types and is extensible for future enhancements.

8.2 Project Overview

8.2.1 System Description

The project is a Django web application that receives user prompts, detects the intended diagram type, retrieves relevant documentation, generates Mermaid code using AI, and renders the result as an SVG image.

8.2.2 Key Features

— Natural language prompt input

— Automatic diagram type classification

— Retrieval-augmented generation (RAG) for context-aware Mermaid code

— SVG rendering and download

— Support for over 15 diagram types

— Error handling and code correction

8.3 System Architecture

8.3.1 High-Level Architecture

— **Frontend :** Django templates for user interaction

— **Backend :** Python modules for processing, AI, and rendering

— **AI Layer :** Language model (LLM) for prompt understanding and code generation

— **Retrieval Layer :** FAISS vector search for context retrieval

— **Renderer :** Converts Mermaid code to SVG

— **Database :** SQLite for Django, Markdown for documentation

8.3.2 Component Diagram

FIGURE 8.1 – High-level component diagram of the system.

8.4 Technical Details

8.4.1 Django Web Application

— Handles HTTP requests and form submissions

— Renders templates for prompt input and SVG output

— Integrates with backend modules via Python imports

8.4.2 Agent Module

— Implements a state machine using `langgraph`

— Nodes : diagram type detection, Mermaid code generation, code correction

— Uses `ChatGroq` LLM for prompt analysis and code synthesis

— Handles error correction by re-prompting the LLM with error context

8.4.3 Retriever and Database

— Loads and splits Markdown documentation into chunks

— Uses FAISS for vector-based semantic search

— Retrieves relevant context for the LLM based on diagram type

— Stores diagram type metadata for filtering

8.4.4 Renderer

— Uses `mermaid_cli` to convert Mermaid code to SVG

— Supports both synchronous and asynchronous rendering

— Handles file I/O for temporary Mermaid files and SVG output

8.4.5 Configuration

— Environment variables for model selection, data paths, and chunk sizes

— Modular configuration via `config.py`

8.4.6 Summary Table of Main Modules

TABLE 8.1 – Summary of Main Modules and Technologies

Module	Responsibility	Key Technologies
Django Web App	User interface, HTTP request handling, template rendering	Django, HTML, CSS
Agent (`agent.py`)	Prompt analysis, diagram type detection, Mermaid code generation, error correction	Python, langgraph, ChatGroq (LLM)
Retriever (`retriever.py`)	Context retrieval, vector search, filtering by diagram type	FAISS, HuggingFace Embeddings, lang
Database (`database.py`)	Loading and splitting documentation, metadata management	Markdown, langchain, Python
Renderer (`renderer.py`)	Mermaid code to SVG conversion, file management	mermaid_cli, Python
Configuration (`config.py`)	Environment variables, model and path settings	dotenv, Python

8.5 Workflow

1. User submits a natural language prompt via the web interface.
2. The agent detects the most suitable diagram type using the LLM.
3. The retriever fetches relevant documentation chunks for the detected type.
4. The agent generates Mermaid code using the LLM and retrieved context.
5. The renderer converts the Mermaid code to SVG.
6. The SVG is displayed to the user and can be downloaded.
7. If rendering fails, the agent attempts to correct the code and re-render.

8.6 Supported Diagram Types

The system supports a wide range of diagram types, including :

— Flowchart

— Sequence diagram

— Class diagram

— State diagram

— Entity-Relationship (ER) diagram

— Gantt chart

— Mindmap

— Pie chart

— Radar diagram

— Quadrant chart

— XY chart

— Kanban board

— C4 diagrams

— Gitgraph diagram

— Sankey diagram

— Timeline diagram

8.7 Error Handling and Correction

— If Mermaid code fails to render, the error message and previous code are sent back to the LLM for correction.

— The system attempts to re-render the corrected code.

— All failed attempts are logged for further analysis.

8.8 Extensibility

— New diagram types can be added by updating the documentation and retriever.

— The system can be adapted to other diagramming libraries with minimal changes.

— Modular design allows for easy integration of new AI models or rendering engines.

8.9 Conclusion

This project demonstrates a robust, extensible approach to automated diagram generation using AI, retrieval-augmented generation, and web technologies. It streamlines technical documentation and can be expanded for broader use cases.

GENERAL CONCLUSION AND PERSPECTIVES

Our final year project marked a significant milestone in our academic journey, consolidating our theoretical knowledge through practical application in a demanding professional environment. It allowed us to confront real-world challenges while discovering and mastering tools, frameworks, and languages taught at ESPRIT, with the goal of developing Hackini over a six-month period.

Hackini provides users with the ability to register, manage competitions, invitations, tickets, articles, and groups, submit and view solutions, interact instantly via a whiteboard-style interface and a Kanban board, and track results in real time, all while ensuring a smooth and secure user experience.

For the future, several areas of improvement have been identified to offer an even more complete and secure user experience. These include the integration of a more advanced matchmaking system, an instant messaging interface, and an automatic evaluation system ; the implementation of a versatile recommendation system to suggest users to teams, teams to users, or candidates to companies for employment ; the addition of an educational module to enhance users' skills ; the development of a fraud detection system to ensure the integrity of competitions ; and the integration of a facial recognition solution to reliably verify participants' identities. These enhancements aim to make Hackini a smarter, more interactive, and secure platform, addressing the diverse needs of users and the challenges of the technology sector.

WEBOGRAPHY

[1] **Kaggle** – Your Machine Learning and Data Science Community : https://www.kaggle.com (Accessed April 30, 2025).

[2] **Codeforces** – Competitive Programming Platform : https://codeforces.com (Accessed April 30, 2025).

[3] **LeetCode** – Learn, Practice, and Enhance Your Coding Skills : https://leetcode.com (Accessed April 30, 2025).

[4] **CTFtime** – Capture The Flag Competitions Ranking : https://ctftime.org (Accessed April 30, 2025).

[5] **Hack The Box** – Cybersecurity Training and CTF Platform : https://www.hackthebox.com (Accessed April 30, 2025).

[6] **Wikipedia** – Waterfall Model : https://fr.wikipedia.org/wiki/Mod%C3%A8le_en_cascade (Accessed August 25, 2025).

[7] **Asana** – V-Model : https://asana.com/fr/resources/v-model (Accessed August 25, 2025).

[8] **Agiliste** – Example of Agile Project Organization : https://agiliste.fr/exemple-dorganisation-projet-agile (Accessed April 30, 2025).

[9] **Hello Pomelo** – 9 Key Points of the Scrum Agile Method : https://hello-pomelo.com/articles/9-points-cles-de-la-methode-agile-scrum (Accessed April 30, 2025).

[10] **Atlassian** – Kanban : https://www.atlassian.com/fr/agile/kanban (Accessed August 25, 2025).

[11] **Tcard** – Kanban Structure : https://tcard.leantransitionsolutions.com/tcard-kanban-boards#lt-kanban-board-features-and-components (Accessed August 25, 2025).

[12] **Atlassian** – Scrumban : https://www.atlassian.com/fr/agile/project-management/scrumban (Accessed August 25, 2025).

[13] **Wikipedia** – UML (Computing) : https://fr.wikipedia.org/wiki/UML_(informatique) (Accessed April 30, 2025).

[14] **Wikipedia** – Use Case Diagram : https://fr.wikipedia.org/wiki/Diagramme_de_cas_d%27utilisation (Accessed March 1, 2025).

[15] **IBM Docs** – Class Diagram : https://www.ibm.com/docs/fr/dmrt/9.5.0?topic=diagrams-class (Accessed March 1, 2025).

[16] **Visual Studio Code** : https://fr.wikipedia.org/wiki/Visual_Studio_Code (Accessed April 1, 2025).

[17] **GitHub** : https://fr.wikipedia.org/wiki/GitHub (Accessed April 1, 2025).

[18] **Jest** – Testing Frameworks : https://jestjs.io/fr/docs/testing-frameworks (Accessed April 1, 2025).

[19] **Next.js** : https://fr.wikipedia.org/wiki/Next.js (Accessed April 1, 2025).

[20] **NestJS Docs** – Official Documentation : https://docs.nest-js.fr (Accessed April 1, 2025).

[21] **NestJS Docs** – Prisma Recipes : https://docs.nestjs.com/recipes/prisma (Accessed April 1, 2025).

[22] **Oracle** – PostgreSQL Definition : https://www.oracle.com/fr/database/definition-postgresql (Accessed April 1, 2025).

[23] **Kodea** – Swagger Definition : https://www.kodea.fr/outils/swagger (Accessed August 26, 2025).

[24] **Figma** : https://www.makethegrade.fr/glossaire/figma (Accessed April 1, 2025).

[25] **Google Workspace Marketplace** – Draw.io : https://workspace.google.com/marketplace/app/drawio/671128082532?hl=fr (Accessed April 1, 2025).

[26] **TailwindCSS** – Official Documentation : https://tailwindcss.com/docs (Accessed August 15, 2025).

[27] **shadcn/ui Docs** – Components : https://ui.shadcn.com/docs/components (Accessed August 10, 2025).

[28] **Wikipedia** – Docker : https://fr.wikipedia.org/wiki/Docker_(logiciel) (Accessed August 10, 2025).

[29] **Wikipedia** – Kubernetes : https://fr.wikipedia.org/wiki/Kubernetes (Accessed August 10, 2025).

[30] **Microsoft** – Microsoft Azure : https://azure.microsoft.com/fr-fr/resources/cloud-computing-dictionary/what-is-azure (Accessed August 10, 2025).

[31] **Microsoft** – Azure Pipelines : https://learn.microsoft.com/fr-fr/azure/devops/pipelines (Accessed August 10, 2025).

HACKINI : UNE PLATEFORME DÉDIÉE AUX COMPÉTITIONS TECHNOLOGIQUES EN IA, PROGRAMMATION ET CYBERSÉCURITÉ

BEN MABROUK HOUSSEM

Résumé :

La plateforme Hackini est dédiée aux compétitions technologiques chez Elepz'ia. Elle permet aux utilisateurs de s'inscrire, valider leur identité, participer à des défis classés ou non classés, et être confrontés via un système de matchmaking intelligent. Les soumissions sont évaluées automatiquement, un système anti-fraude assure l'intégrité des résultats, et la plateforme offre un tableau de bord interactif avec un système de récompenses, le tout déployé sur une infrastructure cloud sécurisée et évolutive.

Abstract :

Hackini is developed at Elepz'ia, a platform dedicated to technological competitions. It allows users to register, verify their identity, and participate in ranked or unranked challenges, with an intelligent matchmaking system that adapts confrontations to the participants' level. Submissions are automatically evaluated, and an anti-fraud system ensures the integrity of results. The platform features an interactive dashboard and a reward system, all deployed on a secure and scalable cloud infrastructure.

ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siége Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : Z.I. Chotrana II - B.P. 160 - 2083 - Pôle Technologique - El Ghazala - Tél. : +216 70 685 685 - Fax. : +216 70 685 454