# Chapter 4: Design

## 4.1 Introduction

This chapter describe the design of the QuickFix Helper system. This phase is critical to the development of our project because it establishes the framework for the system's implementation and functioning. This chapter present a clear and straightforward blueprint of how the QuickFix Helper system will operate by using a series of diagrams and specifications adapted to the system and application development specialization. These visual representations, which include use case diagrams, test cases, class diagrams, and sequence diagrams, provide information about the system's design and process, assuring a thorough grasp of its operation.

Furthermore, we investigate the metadata linked with each entity in the system, such as user information and suggestions. We want to build strong data management procedures by rigorously documenting the data held for each component, including tables such as the User and Suggestions tables. This thorough attention to detail assures not just the efficient organization of information, but also the smooth operation and user experience.

Security issues are important to the design of any system (Chanchad et al, 2023), and the QuickFix Helper system is no exception. In this chapter, we describe the security features built into our design, which include database auditing, access control mechanisms, and reliable backup and recovery processes. These safeguards are critical for protecting sensitive information, ensuring data integrity, and limiting potential security risks. By prioritizing security throughout the design process (Muharam and Sulianta, 2023), we maintain our users' trust and confidence, creating a secure and reliable environment for their interactions with the QuickFix Helper system.

In conclusion, Chapter 4 highlights the thorough preparation and attention to detail that characterize the QuickFix Helper system's design phase. We hope to establish a strong and user-centric platform that satisfies our users' changing needs by thoroughly exploring system architecture, data management techniques, and security issues. By explaining the complexities of our design process, we establish the groundwork for the QuickFix Helper system's successful development and deployment, ensuring its effectiveness and dependability in offering exceptional help to our users.

## 4.2 Design Illustration

The schematics that show the system design are below. Various diagrams, including as flow charts, use case diagrams, Entity Relationship Diagram – ERD, Data Flow Diagram – DFD, Context Diagrams, Sequence Diagrams, are included in this section.

### 4.2.1 Context Diagram

Below is the context diagram for the system:

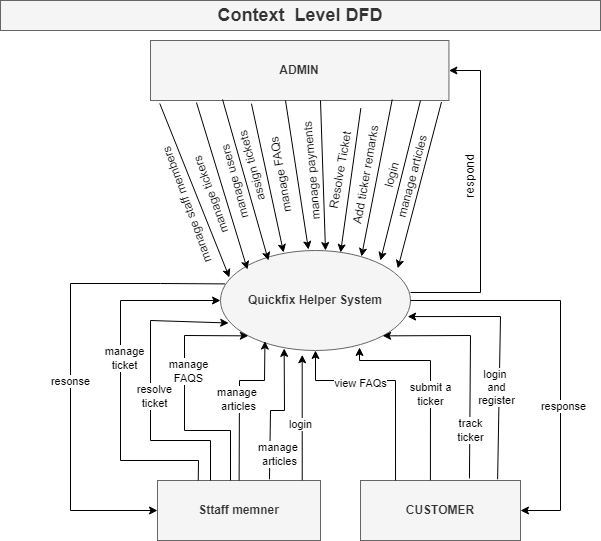


Figure Context Diagram

### 4.2.2 Entity Relationship Diagram

Below is the ER diagram for the system.

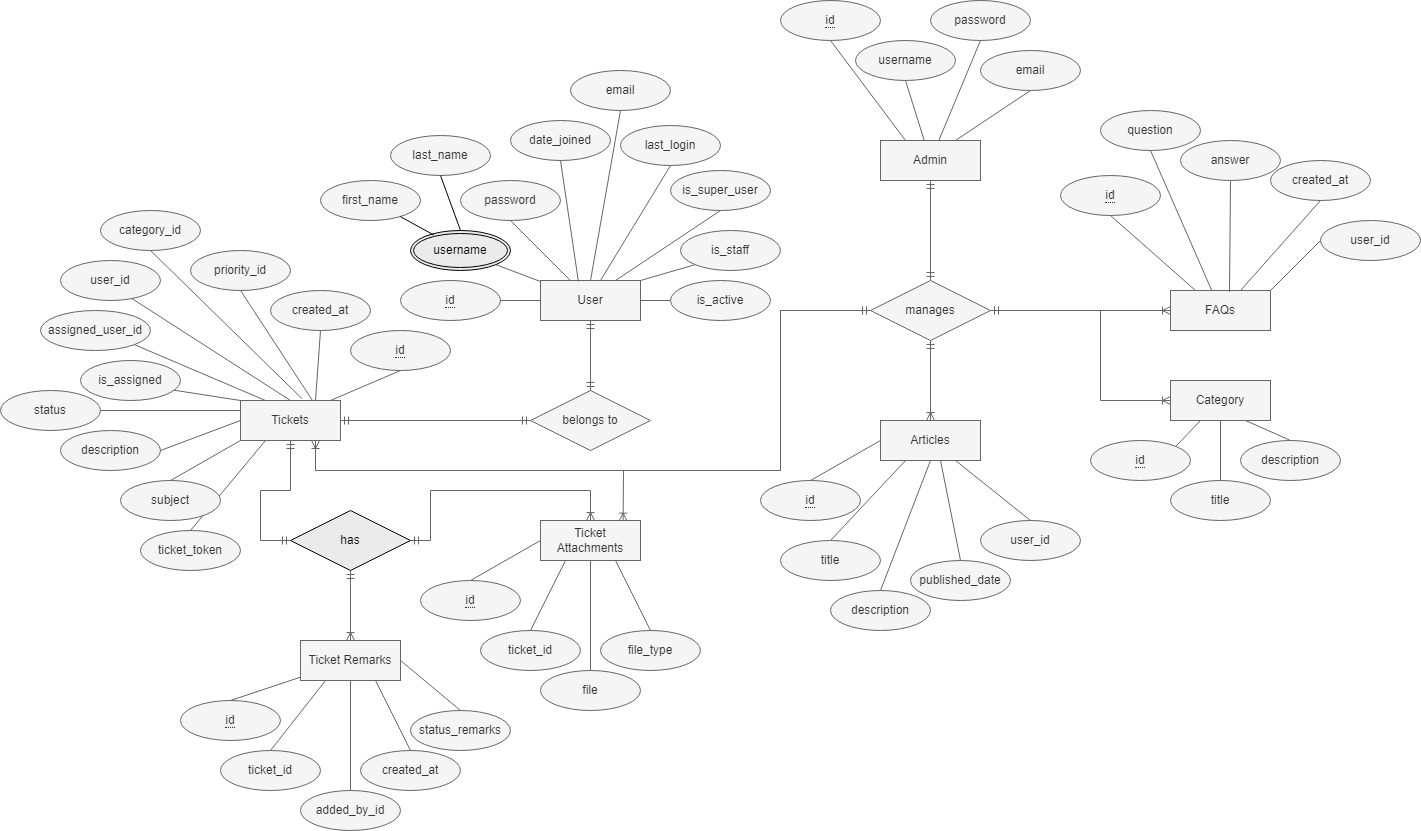


Figure : ER Diagram

### 4.2.3 Data Flow Diagram

Below is the dataflow diagram for the system:

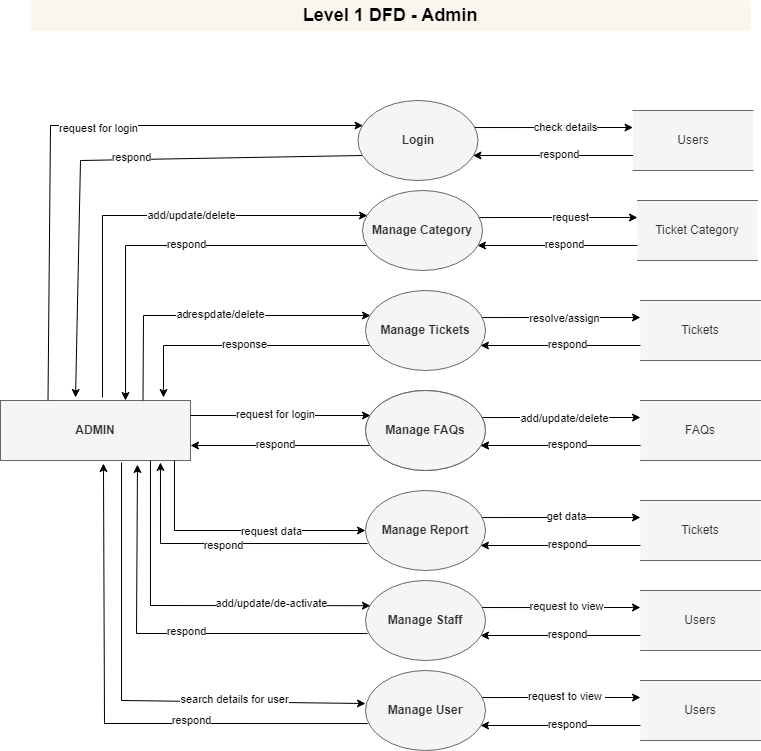


Figure : Dataflow Diagram for Admin

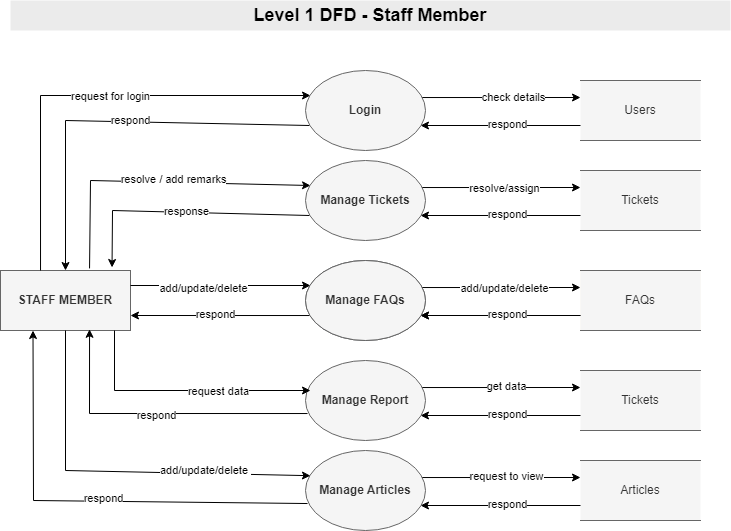


Figure : Dataflow Diagram for Staff Member

### 

Figure : Dataflow Diagram for User

### 4.2.4 Use Case Diagram

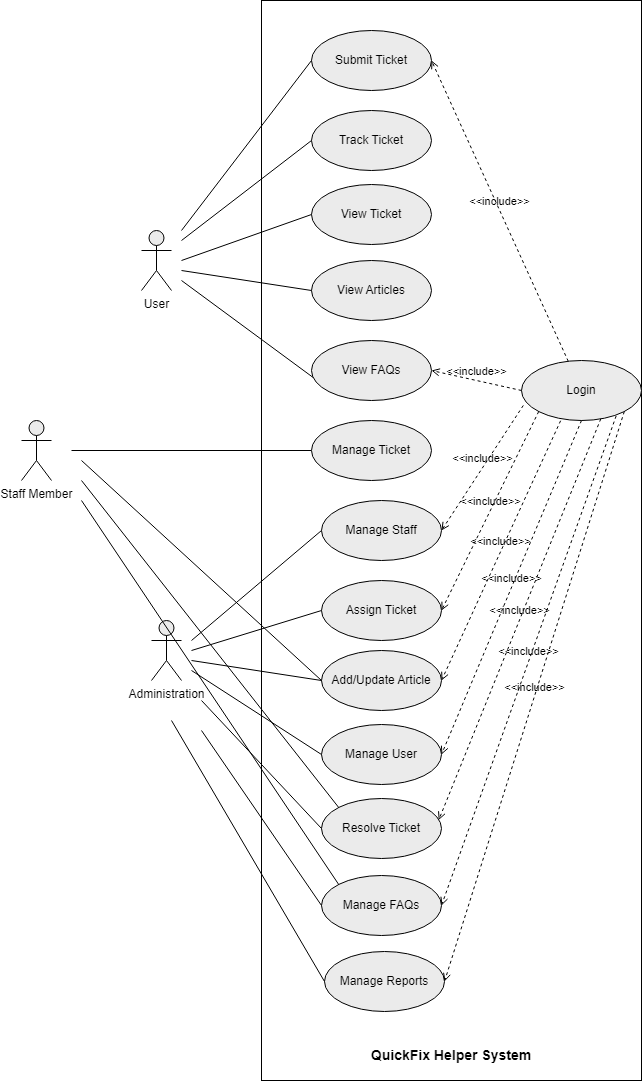


Figure ; Use Case Diagram

### 4.2.5 Sequence Diagram

Below are the sequence diagrams for the system:

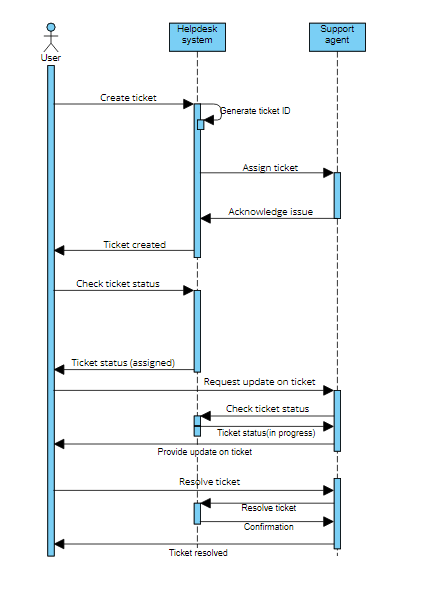


Figure : Sequence Diagram for the submitting ticket process

## 4.3 Security Measures

To guarantee the protection of user data within the QuickFix Helper system, many crucial security measures have been implemented: (Fields, 2024)

1. **Auditing databases:** In order to identify any suspicious activity or unauthorized access attempts, we monitor database activities. This enables us to react to security issues swiftly. (Satori Cyber, 2023)
2. **Access Control:** We have authority over what users within the system can access. By allocating roles and permissions in accordance with user responsibilities, we ensure that only authorized individuals have access to sensitive data. (Sarangi, 2024)
3. **Recovery and Backup of Databases:** Important data is regularly backed up to guard against loss from mishaps or cyberattacks. In this manner, we may promptly return the system to its initial condition in the event that something goes wrong. (Django Database Backup — Django-dbbackup 2.4.1 Documentation, n.d.)
4. **Encryption:** The use of encryption to safeguard data while it is being transmitted over the internet and while it is being stored in the system, we employ encryption techniques. This makes guarantee that the data cannot be understood or manipulated even if it is intercepted. (Goldberg, 2023)

The goal in putting these security measures in place is to protect user data and preserve users' faith in the QuickFix Helper system.

## Normalization

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name** | **First NF** | **Second NF** | **Third NF** |
| USERS | idPrimary PK  first\_name  last\_nameIndex  emailIndex  password  username  is\_supperuser  is\_staff  is\_active  last\_login  date\_joined | idPrimary PK  first\_name  last\_nameIndex  emailIndex  password  username  is\_supperuser  is\_staff  is\_active  last\_login  date\_joined | idPrimary PK  first\_name  last\_nameIndex  emailIndex  password  username  is\_supperuser  is\_staff  is\_active  last\_login  date\_joined |
| TICKETS | idPrimary PK  ticket\_token  subject  description  statusIndex  is\_assigned  assigned\_user\_id  user\_id  category\_id  priority  created\_at  fileIndexs  file\_typeIndex | idPrimary PK  Ticket\_token  subject  description  statusIndex  is\_assigned  assigned\_user\_id FK  user\_id FK  category\_id FK  priority  created\_at  fileIndexs  file\_typeIndex | idPrimary PK  Ticket\_token  subject  description  statusIndex  is\_assigned  assigned\_user\_id FK  user\_id FK  category\_id FK  priority  created\_at |
| TICKET  ATTACHMENTS | idPrimary PK  ticket\_idIndex  fileIndex  file\_typeIndex | idPrimary PK  ticket\_idIndex FK  fileIndex  file\_typeIndex | idPrimary PK  ticket\_idIndex FK  fileIndex  file\_typeIndex |
| TICKET  CATEGORIES | idPrimary PK  title  description | idPrimary PK  title  description | idPrimary PK  title  description |
| TICKET  REMARKS | id PK  ticket\_id  added\_by\_id  created\_at  status\_remarks | id PK  ticket\_id FK  added\_by\_id FK  created\_at  status\_remarks | id PK  ticket\_id FK  added\_by\_id FK  created\_at  status\_remarks |
| FAQs | id PK  question  answer  created\_at  user\_id | id PK  question  answer  created\_at  user\_id FK | id PK  question  answer  created\_at  user\_id FK |
| ARTICLES | idPrimary PK  title  description  published\_date  user\_id | idPrimary PK  title  description  published\_date  user\_id FK | idPrimary PK  title  description  published\_date  user\_id FK |

## 4.5 Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name** | **Attribute Name** | **Type** | **Constraints** |
| USERS | idPrimary | int(10) | Primary Key |
| First\_name | varchar(40) |  |
| Last\_nameIndex | varchar(40) |  |
| emailIndex | varchar(40) |  |
| password | varchar(40) |  |
| username | varchar(40) | Unique |
| Is\_supperuser | Boolean(2) |  |
| Is\_staff | Boolean(2) |  |
| Is\_active | Boolean(2) |  |
| Last\_login | datetime |  |
| Date\_joined | datetime |  |
| TICKETS | idPrimary | int (10) | Primary Key |
| Ticket\_token | varchar(40) | Unique |
| subject | varchar(255) |  |
| description | text |  |
| statusIndex | boolean(2) |  |
| is\_assigned | boolean (2) |  |
| assigned\_user\_id | tinyint(4) | Foreign key |
| user\_id | int(10) | Foreign key |
| category\_id | int(10) | Foreign key |
| priority | varchar(40) |  |
| created\_at | datetime |  |
| TICKET  ATTACHMENTS | idPrimary | bigint(20) | Primary Key |
| Ticket\_idIndex | int(20) | Foreign Key |
| fileIndex | text |  |
| file\_typeIndex | varchar(20) |  |
| TICKET  CATEGORIES | idPrimary | int(10) | Primary Key |
| title | varchar(20) |  |
| description | text |  |
| TICKET  REMARKS | id | int(10) | Primary Key |
| ticket\_id | int(20) | Foreign Key |
| added\_by\_id | Int(20) | Foreign Key |
| created\_at | datetime |  |
| status\_remarks | text |  |
| FAQs | id | int(10) | Primary Key |
| question | varchar(20) |  |
| answer | bigint(20) |  |
| created\_at | datetime |  |
| user\_id | Int(20) | Foreign Key |
| ARTICLES | idPrimary | int(10) | Primary Key |
| title | varchar(40) |  |
| description | text |  |
| published\_date | datetime |  |
| user\_id | int(20) | Foreign Key |

## 4.6 SQL Table Design

**Table 1: Users Table**

This table is to stores the detail of users, which also contain the information for the admin and staff as well.

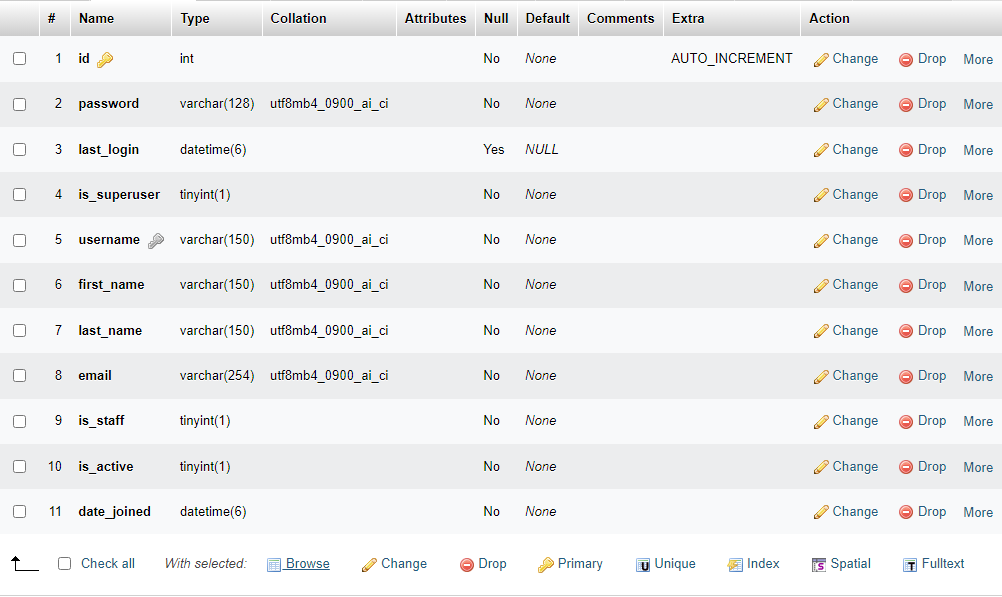


Figure 9: User Table

**Table 2: Tickets Table**

User can create a ticket and system administration can assign ticket to the user, this table contain the details tickets.

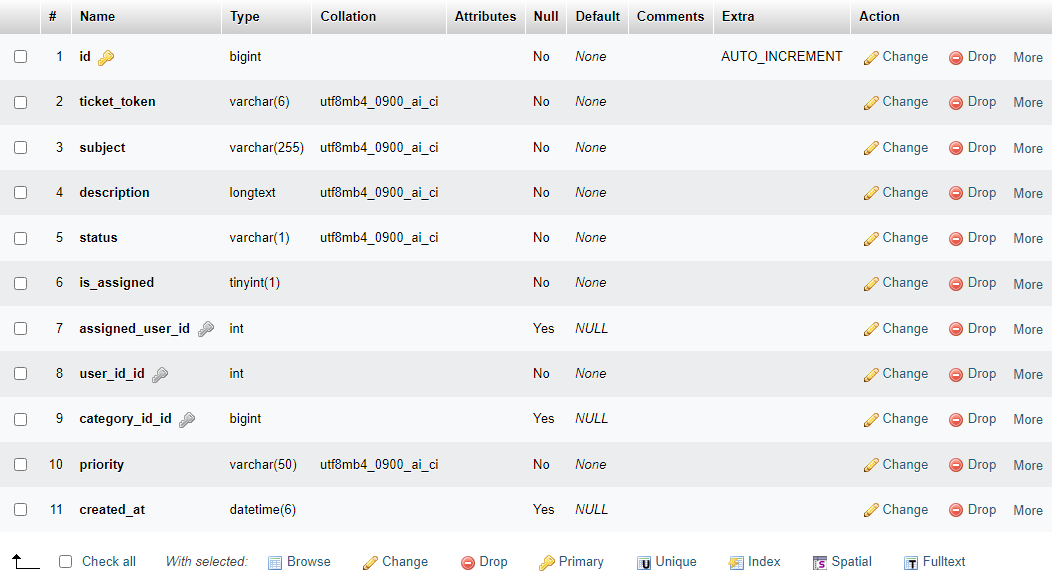


Figure 10: Tickets Table

**Table 3: Ticket Attachments**

To store the detail screenshot or other attachments regarding tickets, this table store the attachments.

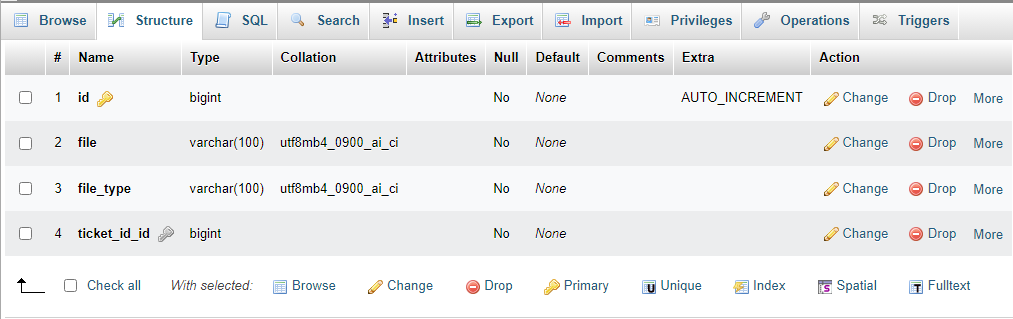


Figure 11: Ticket Attachments Table

**Table 4: Ticket Categories**

This stores ticket categories so which can be linked to the ticket later.

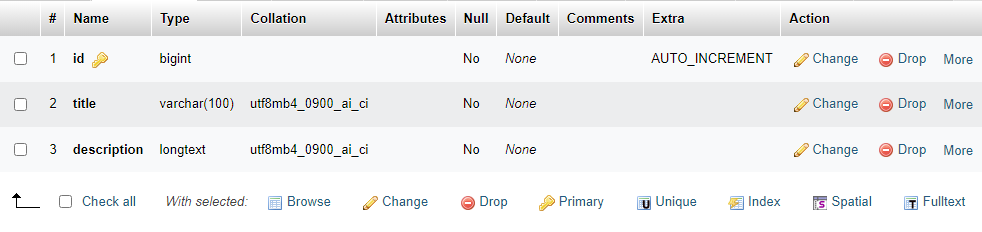


Figure 12: Ticket Categories Table

**Table 5: Ticket Remarks**

Administration and other staff member can create ticket remarks and these remarks store in this table.

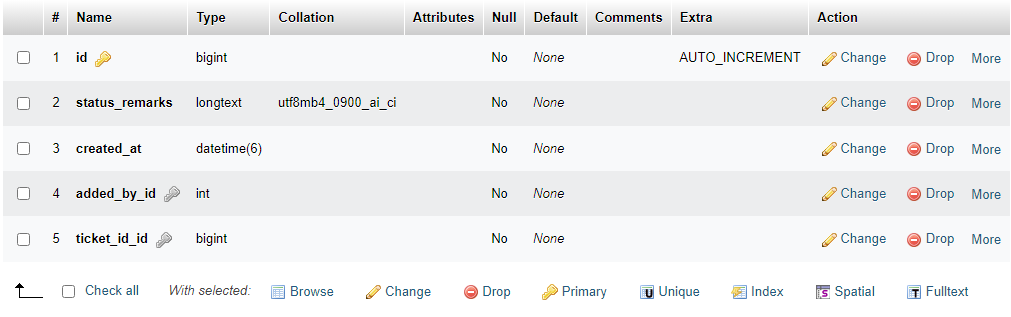


Figure 13: Ticket Remarks Table

**Table 6: FAQs Table**

Staff and administration can create frequently asked question which will later help the users. This table store the content which contains questions and answer.

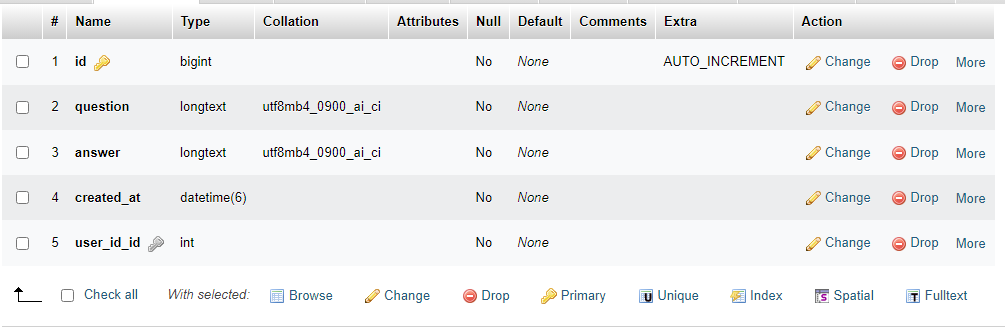


Figure 14: FAQs Table

**Table 7: Articles Table**

This table store the articles which can be created by the administration and staff.

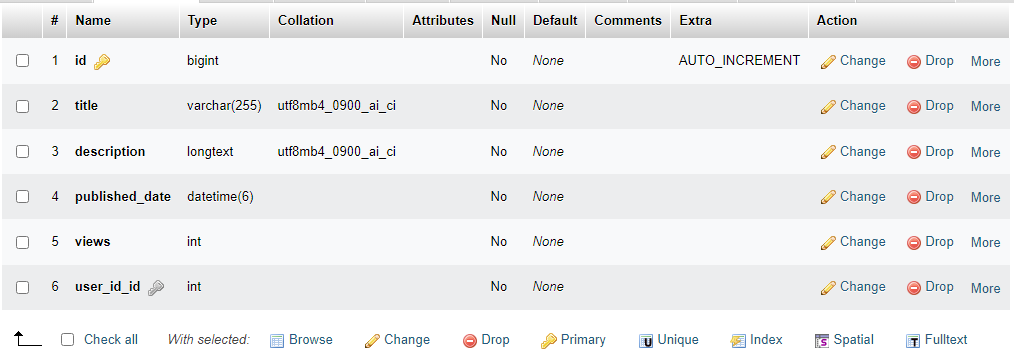


Figure 15: Articles Table

## 4.7 Screenshot of system

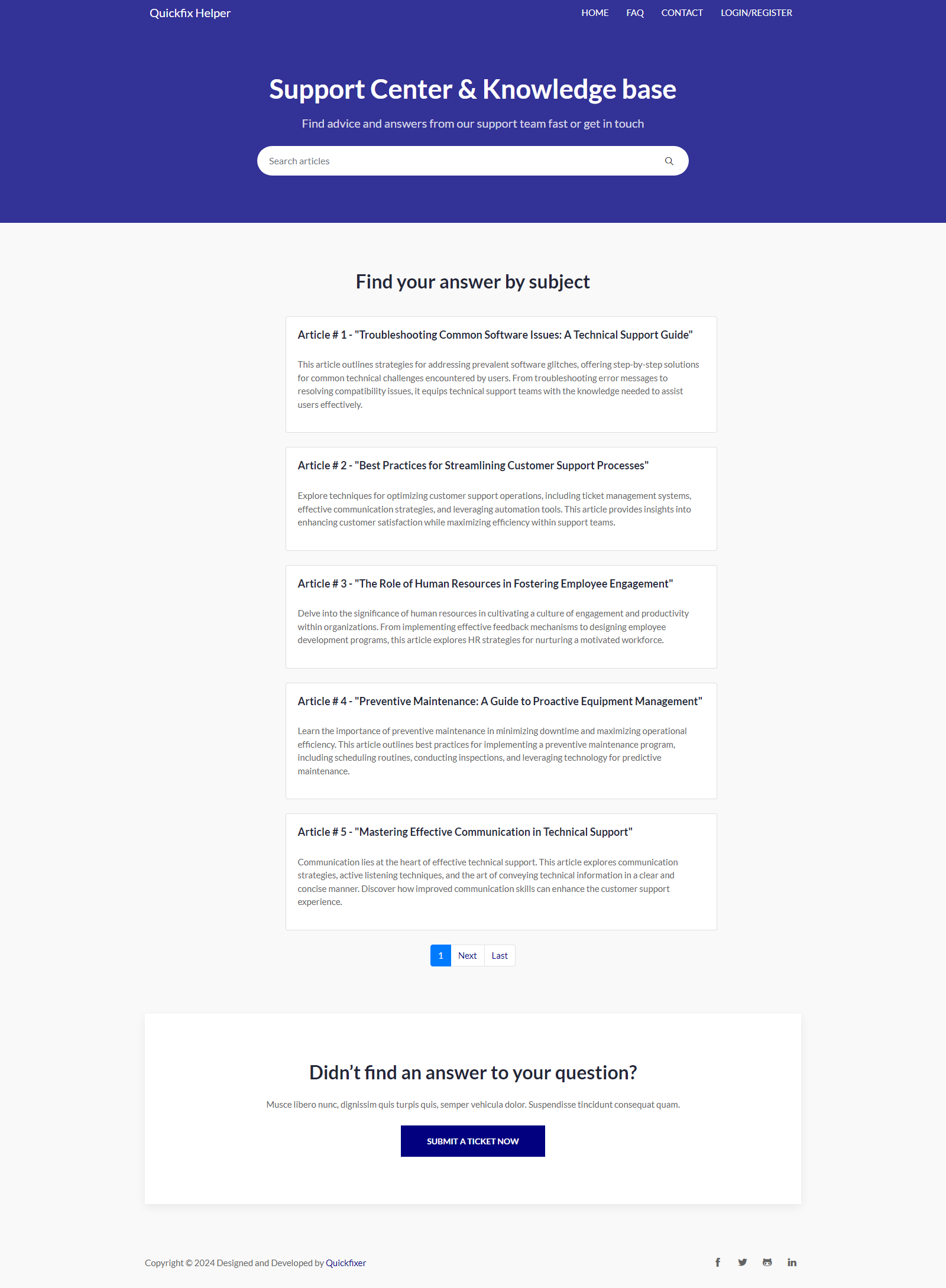


Figure 16: Home Page

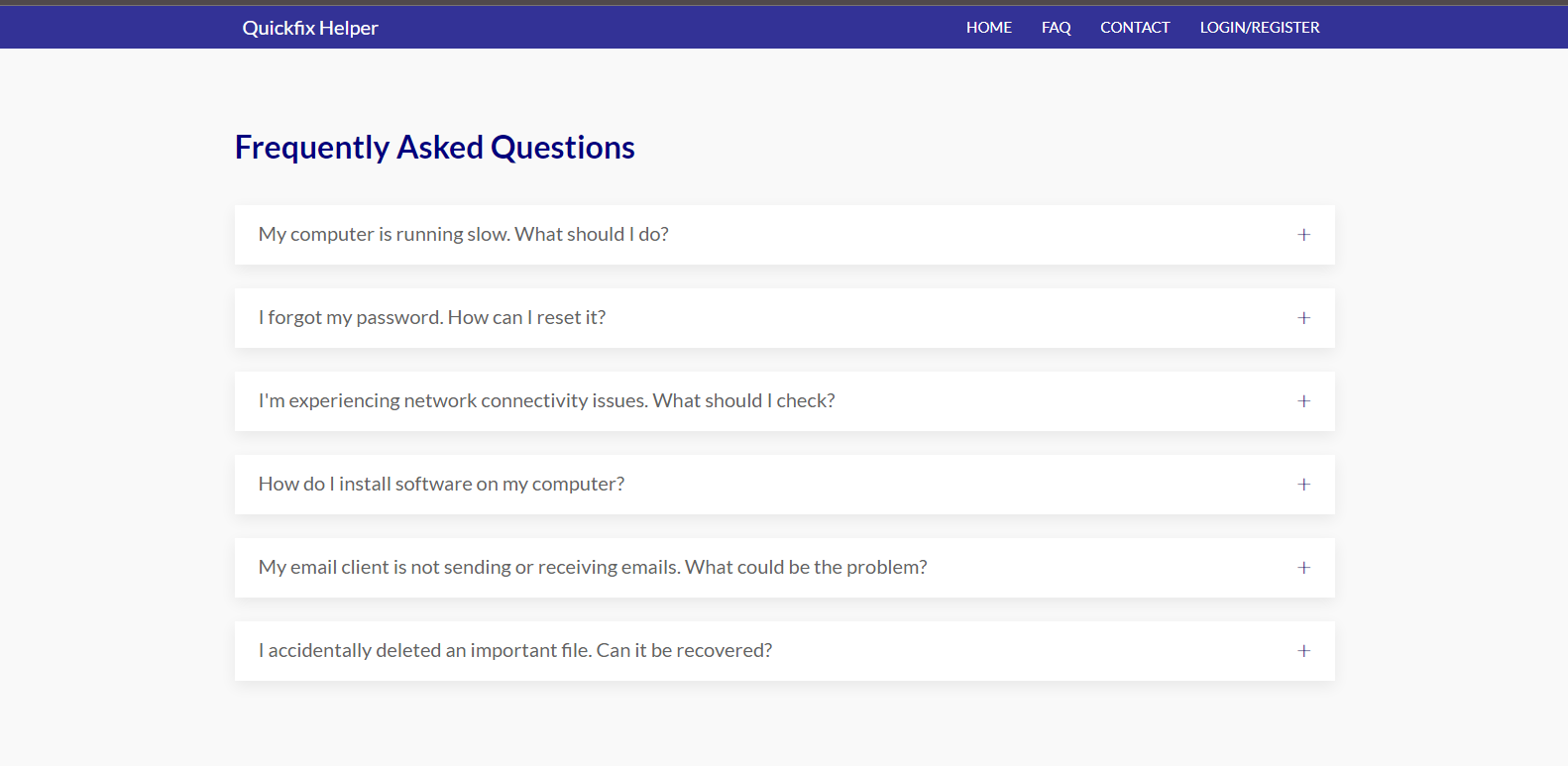


Figure 17: FAQs Page

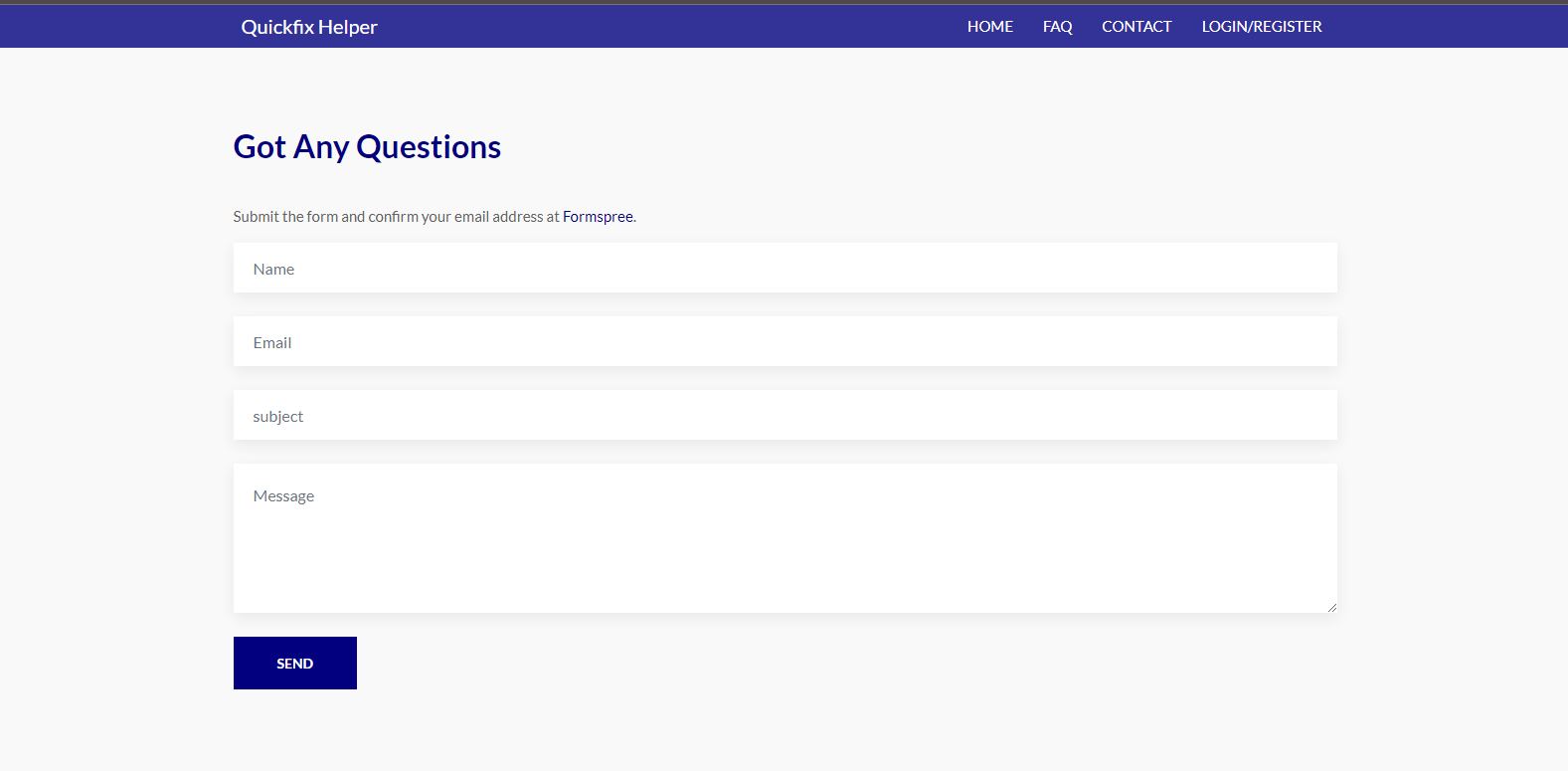


Figure 18: Contact Us Page

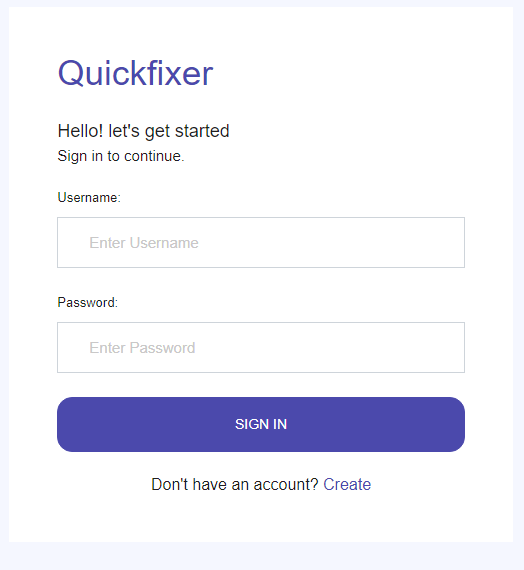


Figure 19: Login Page

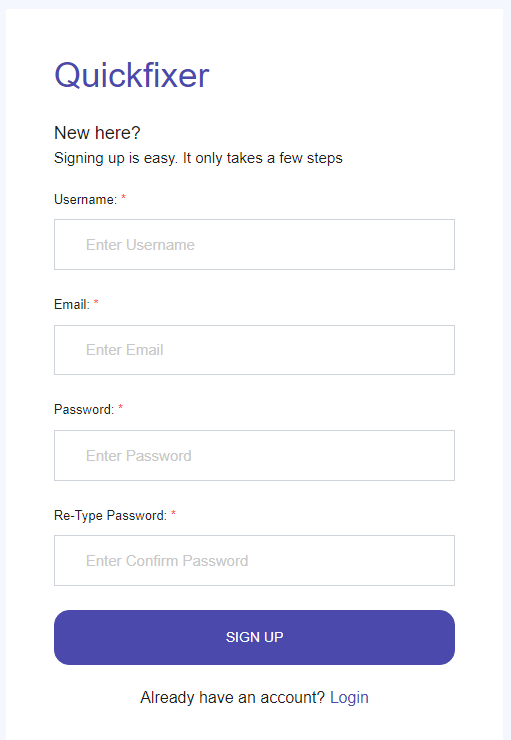


Figure 20: Registration Page

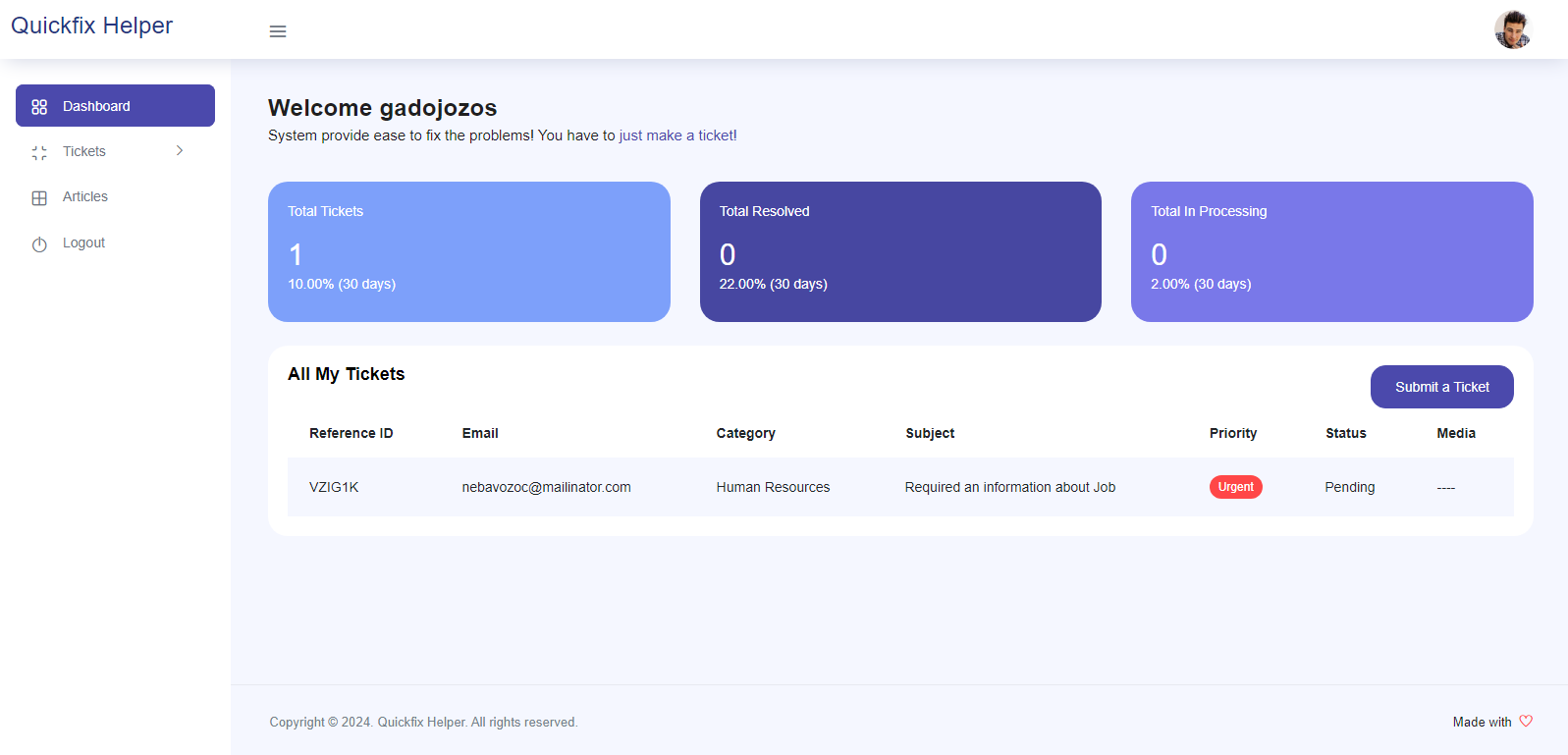


Figure 21: User Dashboard

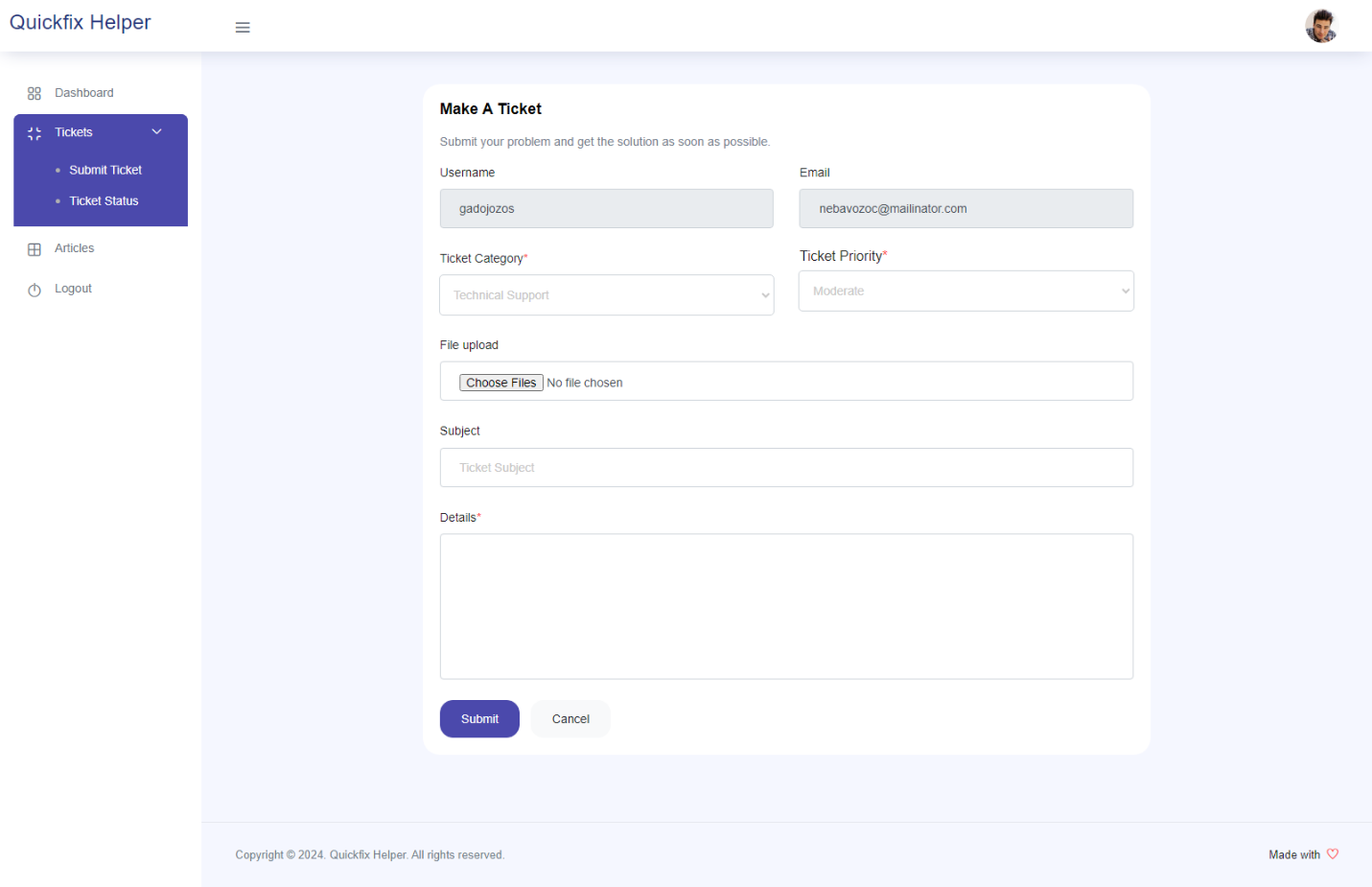


Figure 22: Create Ticket

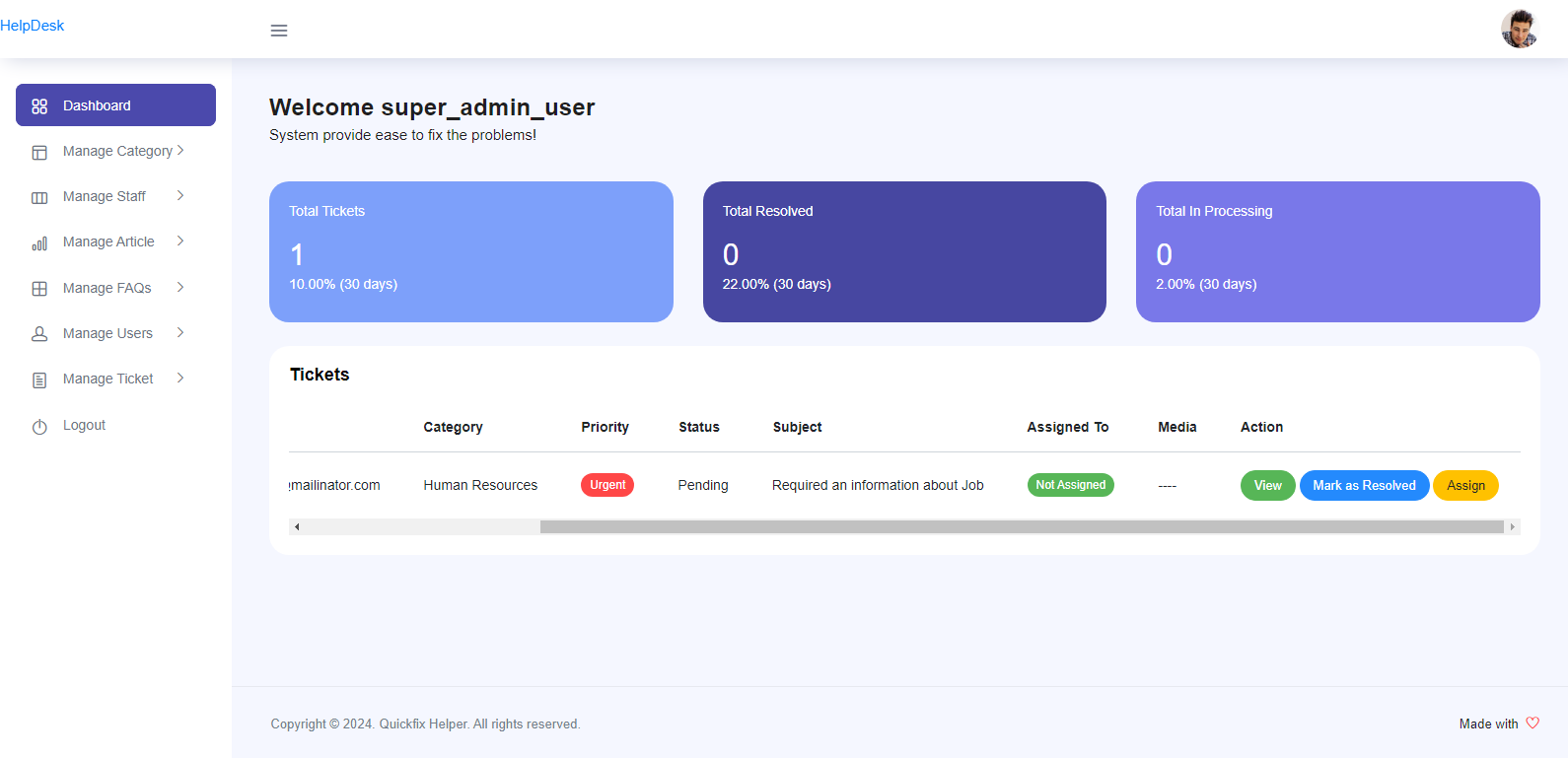


Figure 23: Admin Dashboard

**Reference:**

1. Sarangi, S. (2024, February 13). Role-Based Access Control in Django: The right features to the right users. *Medium*. <https://medium.com/@subhamx/role-based-access-control-in-django-the-right-features-to-the-right-users-9e93feb8a3b1>
2. *Django Database Backup — django-dbbackup 2.4.1 documentation*. (n.d.). <https://django-dbbackup.readthedocs.io/en/2.5.0/>
3. Goldberg, I. (2023, October 27). *Exploring approaches to field-level encryption in Python for Django applications*. <https://www.piiano.com/blog/field-level-encryption-in-python-for-django-applications>
4. Fields, A. (2024, February 27). *Website security measures*. WebFX. <https://www.webfx.com/web-development/learn/website-security-measures/>
5. Satori Cyber. (2023, November 6). *Database Auditing - Satori*. Satori. <https://satoricyber.com/cloud-data-governance/database-auditing/>