A REPORT
ON

CORMSQUARE SUPPORT HUB: CENTRALIZED KNOWLEDGE & SOLUTIONS

Submitted by,

Ms. MEDHA JEENOOR - 20211CSE0209

Under the guidance of,

Dr. SANDEEP ALBERT MATHIAS

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internship/Project report **"CORMSQUARE SUPPORT HUB: CENTRALIZED KNOWLEDGE & SOLUTIONS"** being submitted by "MEDHA JEENOOR" bearing roll number "20211CSE0209" in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. SANDEEP ALBERT MATHIAS
Assistant Professor Selection Grade-
PSCS
Presidency University

Dr. ASIF MOHAMMED H.B
Associate Professor & HoD
PSCS
Presidency University

Dr. MYDHILI NAIR
Associate Dean
PSCS
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-Vice Chancellor - Engineering
Dean –PSCS / PSIS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I hereby declare that the work, which is being presented in the report entitled **"CORMSQUARE SUPPORT HUB: CENTRALIZED KNOWLEDGE & SOLUTIONS"** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of my own investigations carried under the guidance of **Dr. Sandeep Albert Mathias, Assistant Professor Selection Grade-PSCS, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

Medha

MEDHA JEENOOR

20211CSE0209

INTERNSHIP COMPLETION CERTIFICATE

CormSquare

09-May-2025

Ms. Medha Jeenoor,
No. 010, Kalpataru Apartments,
Kodigehalli Main Road, Sahakarnagar
Bengaluru - 560092

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Ms. Medha Jeenoor** has successfully completed her internship at our organization from **27th January 2025 to 25th April 2025** and her project **"CormSquare Support Hub: Centralized Knowledge & Solutions"** has been completed judiciously.

During her internship, she has exhibited great understanding of the various technologies that were employed during her project in the domain of Software Development including but not limited to:

FrontEnd: HTML, CSS, JavaScript, JQuery
Backend: Microsoft .NET
Database: SQL Server

Her association with us was fruitful and we wish her all the best in her future endeavors.

With warm regards,
For Cormorant Engineering India Private Limited.,

RA Narendra

Narendra RA
Director

Cormorant Engineering India Private Limited
CEIPL

Cormorant Engineering India Private Limited
Head Office: #683/A, 2nd Floor, 13th Main Road, 15th Cross, J P Nagar 2nd Phase, Bangalore - 560078
Ph: +91 8048902364 | Email: info@cormsquare.com | Website: https://cormsquare.com

ABSTRACT

This project is about building a user-friendly **Knowledge Management System (KMS)** that organizes and simplifies how a company handles its knowledge, answers questions, and solves problems. **External users**, like customers or partners, can check out published solutions, report issues, or ask for information. Inside the system, **KM Creators** whip up solutions using templates designed by **Admins**. **KM Champions** take a look, deciding if the solutions are good to go, need a bit of polishing, or should be sent back to the drawing board. Admins are the ones calling the shots—they handle user accounts, create templates, and set up details for clients or products. When someone from outside wants to join, they sign up, and an Admin checks it out to make sure only the right people get access. This straightforward process for creating, checking, and sharing content keeps things running smoothly, makes it easy to follow, and helps everyone in the organization share and reuse knowledge like pros.

ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presiency School of Computer Science and Engineering & Presiency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair,** Presidency School of Computer Science and Engineering, Presidency University, and Dr. "Dr. Asif Mohammed H.B", Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Sandeep Albert Mathias, Assistant Professor Selection Grade-PSCS** and Reviewer **Dr. Vishwanath Y, Professor-SCSE**, Presdiency School of Computer Science and Engineering, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the internship work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 Internship/University Project Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K,** department Project Coordinators **Mr. Jerrin Joe Francis** and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

MEDHA JEENOOR

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 General Introduction

An organization's most valuable asset is knowledge. In a large-scale company, it is crucial to manage the vast amounts of data. A Knowledge Management System (KMS) is designed to collect, store, organize, and retrieve knowledge to aid in decision-making and improve productivity.

During my internship at CormSquare, I was assigned to the Development Domain with a focus on creating a robust Knowledge Management System (KMS) using Microsoft .NET (C#), SQL Server, HTML, CSS, JavaScript, and jQuery.

The purpose of this system is to allow employees to access the different kinds of knowledge base, ensuring a streamlined workflow and improved collaboration within the company.

1.2 Introduction to the Domain of the Problem Statement

The development domain comprises of designing, implementing, and optimizing software solutions to meet the client's specifications and needs. There are 3 parts to this: Front-end development for user interface (UI), Back-end development for processing logic, and Database management for storing and retrieving information in the server.

My work revolved around implementing a centralized KMS using the following technologies:

- Front-End: HTML, CSS, JavaScript, Bootstrap, jQuery
- Back-End: Microsoft .NET (C#)
- Database: SQL Server
- Version Control: GitHub

This project aligns with modern enterprise-level knowledge management systems where information retrieval is faster, redundancy is almost nil, and ensures a structured documentation approach.

Chapter 2

LITERATURE SURVEY

A Knowledge Management System (KMS) is a structured digital framework that allows for the creation, organization, retrieval, and dissemination of knowledge across an organization. The following literature review presents critical developments in the domain, covering KMS design principles, role-based access control, searchability, quality metrics, and visual analytics.

2.1. Knowledge Management Foundations and Frameworks

Dalkir outlines that KMS frameworks are composed of three primary layers: knowledge capture, storage, and dissemination. A well-implemented KMS promotes organizational learning, innovation, and collaboration by turning tacit knowledge into explicit knowledge shared across teams (Dalkir, 2013).

Nonaka and Takeuchi's SECI model (Socialization, Externalization, Combination, Internalization) has been widely adopted to conceptualize knowledge transformation and flow within systems (Nonaka, 2009). However, traditional systems often fall short in dynamically adapting to evolving organizational structures and workflows.

2.2. Role-Based Access and Workflow Enforcement

Effective access control is essential for secure and meaningful knowledge flow. Role-Based Access Control (RBAC) models ensure that knowledge assets are editable or viewable only by authorized users (Sandhu, 1998). Furthermore, workflow-enforced editing (e.g., draft → review → publish) is rarely enforced in legacy systems, despite being crucial for auditability and accountability.

Modern frameworks like ASP.NET Core Identity provide granular control over user access and roles, yet integrating these with real-world editorial workflows in KMS applications remains underutilized.

2.3. Rich Text Content Capture and Document Flexibility

Superior WYSIWYG editors, such as TinyMCE and Tiptap, make it simple to add rich formatting, embed media, and even change code, making them ideal for producing in-depth knowledge documents (TinyMCE 7 Documentation, n.d.). The drawback? They usually preserve their material in HTML, which might make it difficult to read, search for, or display uniformly across devices, such as PDFs or mobile apps. My project focused for ways to address these issues while still preserving rich content's adaptability.

2.4. Search Optimization via Full-Text Indexing

Knowledge is only useful if you can find it quickly. Full-text search tools, like SQL Server's FTS or ElasticSearch, make document retrieval faster (Microsoft Docs, n.d.). But HTML-heavy content can confuse searches unless plain text is extracted first, as (Vogels, 2018) suggests. I focused on improving search accuracy to enhance the user experience in my KMS.

2.5. Scoring and Quality Metrics in KM Systems

How do you measure the value of knowledge? (Xu, 2019) recommends tracking metrics like how fast content is created, how many revisions it goes through, or how long approvals take. Adding these metrics to a KMS can help rank content, encourage contributions, and provide feedback. Since these features are rare in existing systems, I explored integrating them into my project.

2.6. Trending Algorithms and Usage-Based Boosting

Few studies discuss how to maintain the relevance of knowledge. In order to promote content with a high impact, recent models integrate trending algorithms that measure usage frequency (e.g., in the past 7 and 30 days) (Ricci, 2010). According to behavioral learning theory, content that is accessed more frequently indicates more relevance and

ought to be promoted as such.

2.7. Attachments, References, and Metadata Granularity

In KMS platforms, references and attachments frequently lack context. Documents become richer and easier to understand when metadata tags are included, such as when content is labeled as internal or external or its purpose is explained (Gostoji{\'c}, 2014). In order to balance security and flexibility, Sharma et al. also recommend stricter controls, which I took into account when designing my system.

2.8. Visual Dashboards and Decision Support Tools

Chart.js and similar libraries are increasingly used for real-time data visualizations, offering dashboard views for administrators and contributors alike. Visualizing metrics like document distribution by category or top contributors helps stakeholders make informed decisions (Heer). Still, most KM systems lack interactive, data-driven insights.

Chapter 3

RESEARCH GAPS OF EXISTING METHODS

Even with all the progress in Knowledge Management Systems (KMS), there are still some big gaps that my project aims to fix. Here's a rundown of the issues I found and how my system tackles them:

3.1. Weak Implementation of Editorial Workflows

Most KMS platforms let users log in with different roles, but they don't guide content through a clear process—like getting approvals, limiting who can edit what, or tracking changes to documents. My system sets up a smoother, more organized workflow to keep everything on track.

3.2. Absence of Efficiency-Based Scoring Models

Hardly any systems use scores to measure things like how quickly content is created or revised. Without these metrics, contributors miss out on feedback, and managers can't spot where things are slowing down or how good the content is.

3.3. No Real-Time Trending Promotion

Some platforms show basic stats, but very few use them to highlight hot topics or popular solutions in real time. That's a missed chance to push the most useful knowledge to users who need it. My system will spotlight trending content to make it easier to find what's relevant.

3.4. Underutilized Attachment and Reference Metadata

Many systems treat attachments and links like afterthoughts, skipping details like captions, security tags, or how they should open (like in a new tab). My approach gives these elements proper attention, making them more useful and secure.

3.5. Search Limitations Due to Rich HTML Content

Searching in a lot of KMS platforms is either limited to titles and tags or gets messy because of unprocessed HTML content. Proper HTML parsing and full-text indexing are not consistently implemented.

3.6. Lack of Visual Analytics

Few platforms embed visual dashboards that track metrics like usage trends, publishing efficiency, or category-based distributions. This restricts the ability of admins and contributors to improve content strategy.

3.7. Inefficient User Experience for Solution Creation

Template-based creation using rich editors is often a disconnected process. Your approach of using category templates with editable formatting and independent file/reference handling significantly streamlines knowledge contribution.

Chapter 4

PROPOSED MOTHODOLOGY

The proposed solution for the CormSquare Support Hub is developed using the ASP.NET Core MVC framework, structured into a modular and scalable architecture comprising four interconnected projects: CormSquareSupportHub, SupportHub.DataAccess, SupportHub.Models, and SupportHub.Utility. The methodology emphasizes clean separation of concerns, maintainability, and extensibility to accommodate future enhancements such as interactive dashboards and role-based visibility controls.

4.1. Modular Project Architecture

- **CormSquareSupportHub**: Serves as the main web application project containing Razor views, controllers, JavaScript files, and front-end logic. It is organized using Areas (Admin, Identity, and Public) to separate responsibilities across user roles.

- **SupportHub.DataAccess**: Implements the data access layer using the Repository and Unit of Work patterns for centralized database operations. It interacts with a SQL Server backend via ApplicationDbContext, and each entity has a corresponding repository interface and implementation.

- **SupportHub.Models**: Contains all domain models and shared base entities. It includes audit-tracking logic via the AuditableEntity class and segregates ViewModels for forms and data transfer.

- **SupportHub.Utility**: Provides reusable utility functions including SMTP-based email services and constants like role names in a centralized class (SD.cs).

4.2. Authentication and Role Management

Authentication is implemented using ASP.NET Core Identity, extended with a custom ExternalUser class. The system supports five roles: Admin, KM Champion, KM Creator, Internal User, and External User.

Role-based access is enforced across different areas:

- **Admins** manage users, categories, and global settings.
- **KM Creators** handle content creation.
- **KM Champions** review and publish solutions.
- **External Users** raise issues.
- **Internal Users** gain access to content marked as internal.

4.3. Content Creation with TinyMCE

TinyMCE is integrated for rich text editing in both category templates and solution creation workflows. Separate JavaScript files manage TinyMCE behavior across category and solution contexts. Templates defined by Admins are reused during solution creation, allowing contributors to customize formatting and content without altering the original template.

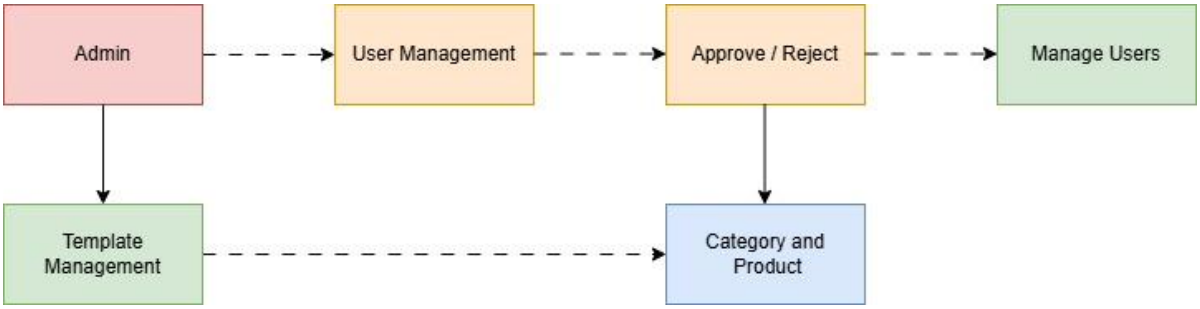


Figure 1: Admin Role & Management

4.4. Attachment and Reference Management

The application allows file attachments and reference links to be added both at the template (category) level and during solution creation. Each file or reference can be marked as

internal, restricting visibility only to internal users. Attachments are saved both in the database and in a dedicated upload folder configured in appsettings.json.

Two separate JS files handle:

- Attachment/reference logic for categories.
- Attachment/reference logic for user-generated solutions.

This modular design ensures code reusability and prevents duplication.

4.5. Content Visibility & Internal Access Controls

A future-facing feature, now partially implemented, introduces the ability to flag solution content as internal-only, beyond just attachments and references. Visibility is dynamically enforced based on the logged-in user's role. This helps protect sensitive internal knowledge while still supporting public-facing documentation.

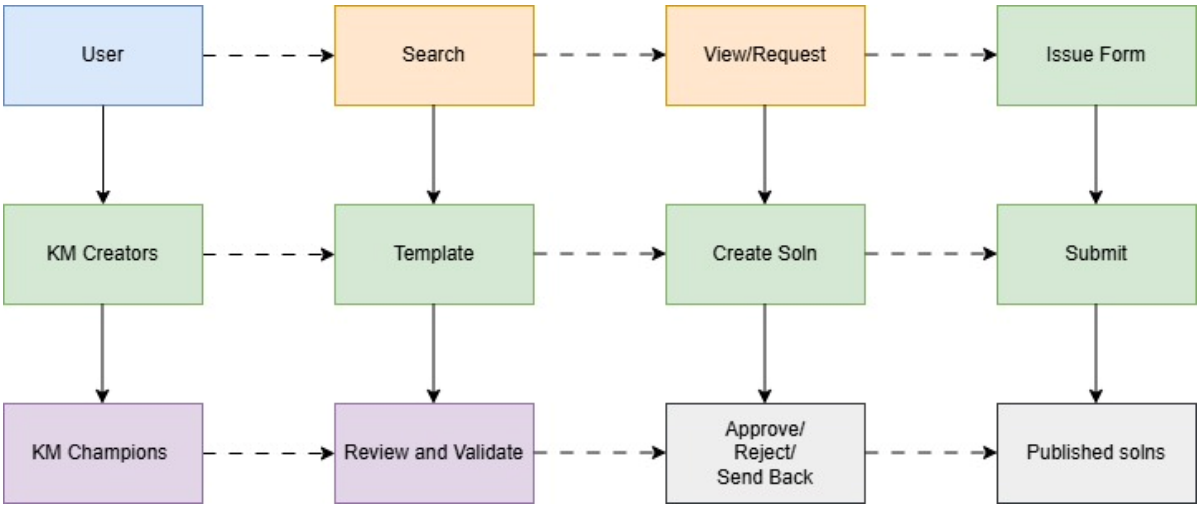


Figure 2: User Flow and Interaction

4.6. Email Notifications

Email services are configured using SMTP (Gmail). The system sends automatic notifications for events like user registration, password resets, and content approval status updates. Configuration is stored securely in appsettings.json.

4.7. Areas and Routing

The project utilizes ASP.NET Areas for logical separation:

- **Admin** handles categories, users, issues, products, and solutions.
- **Public** includes the main landing page and FAQs.
- **Identity** manages login, registration, and password recovery.

Routing is configured to redirect unauthenticated users to the login page by default, ensuring security and compliance with business access requirements.

4.8. Audit Logging and Soft Deletes

All entities inherit from a shared AuditableEntity base, allowing tracking of CreatedBy, UpdatedBy, and soft deletion metadata. This design supports compliance and traceability across the system without permanently deleting data.

4.9. Future Enhancements

- **Interactive Dashboards**: Development is planned to provide analytical visualizations like category-wise solution distribution, usage trends, and contributor statistics.

- **Search and Filtering Improvements**: Planned integration of SQL Server Full-Text Search on plain text versions of solution content.

- **Concurrency Control**: Future support for collaborative editing with lock mechanisms to avoid overwrites when multiple contributors are working on the same solution.

Chapter 5

OBJECTIVES

The goal of this project is to create a role-based Knowledge Management System (KMS) that makes it easy to create, review, view, and control access to organizational knowledge with keeping in mind that it had to be organized, secure, and user-friendly.

5.1. Centralized Knowledge Base

To create a single hub for all published solutions, FAQs, and documents, making it simple for everyone in the organization to find, use, and reuse information easily.

5.2. Role-Based Workflow and Access Control

Clear roles are defined to keep things structured and to ensure a smooth workflow:

- External Users can check out published content and submit issues or questions.
- KM Creators write up new solutions.
- KM Champions review draft or asking for tweaks.
- Admins handle users, clients, products, and templates.

This setup ensures role-based access and authentication.

5.3. Template-Driven Content Authoring

To keep content consistent, Admins create templates (like FAQs, How-To Guides, or Case Studies) that KM Creators use with the TinyMCE editor.

5.4. Secure Issue and Request Handling

I wanted users to be able to raise issues or ask for info easily, which then turns into tasks for KM Creators. This creates a smooth loop where feedback leads to new content.

5.5. Review and Approval Mechanism

I built a review process where KM Champions check drafts and decide to approve, reject, or send them back for changes. This ensures that only high quality content gets published.

5.6. User Authentication and Admin Approval

To keep things secure, I added login, registration, and password recovery features. External users need Admin approval before they can join, ensuring only the right people get access.

5.7. Attachments and References with Visibility Control

KM Creators can add files or links to solutions and mark them as internal-only, so that only internal users can see them. This keeps sensitive stuff safe while still being useful.

5.8. Internal Content Flagging

To allow parts or entire content of a solution to be marked as internal-only, restricting its visibility to authorized users and safeguarding sensitive or confidential information.

5.9. Future Integration of Interactive Dashboards

To plan for future enhancements, including interactive dashboards and visual analytics (such as category-wise pie charts and usage insights), improving content discoverability and user engagement.

5.10. Modular and Scalable Design

To architect the system using .NET Core MVC and SQL Server, ensuring scalability, maintainability, and ease of future integration with additional tools and features.

Chapter 6

SYSTEM DESIGN & IMPLEMENTATION

6.1. System Design

I built the **CormSquare Support Hub** to be flexible, easy to maintain, and ready to grow. It uses a layered setup with the **Model-View-Controller (MVC)** pattern, **Entity Framework Core** to handle data, and the **Identity Framework** for secure logins and permissions. My goal was to keep everything neat and organized so it's simple to update or add new features later.

6.1.1. Architectural Overview

The The system is split into four main parts, each with a clear role:

1. **CormSquareSupportHub (Presentation Layer):**
    - It is organized into Areas (Admin, Identity, Public) to separate for role-based access.
    - Controllers and Views handle requests and show the right web pages.
    - Includes JavaScript (like TinyMCE for editing), CSS, and other front-end files.

2. **SupportHub.DataAccess (Data Access Layer):**
    - Uses Repository and Unit of Work patterns to keep database tasks tidy.
    - Connects to SQL Server using Entity Framework Core.
    - Has repositories for things like Issues, Products, and Solutions.

3. **SupportHub.Models (Domain Layer):**
    - Defines the core data structures, all tied to an AuditableEntity class for tracking changes.
    - Includes ViewModels to move data between layers smoothly.

4. **SupportHub.Utility (Infrastructure Layer):**
    - Offers shared tools like email services and constants (e.g., role names in SD.cs).
    - Handles things like sending notification emails.

6.1.2. Component Interaction

- Controllers take user requests, talk to the Data Access Layer through repositories, and send back the right web pages.
- Views use Razor to create the user interface, pulling in data from models.
- Models hold the app's data and rules.
- JavaScript adds interactivity, like rich text editing with TinyMCE or managing attachments.

6.1.3. Security and Authentication

- ASP.NET Core Identity handles user accounts and logins.
- Role-based access ensures Admins, KM Champions, KM Creators, Internal Users, and External Users only see what they're allowed to.
- I added security basics like HTTPS, HSTS, and secure cookies to keep things safe.

6.2. System Implementation

Backend Implementation

- Database Connectivity:
    - Set up in appsettings.json with a secure SQL Server connection.
    - Keeps settings flexible for different environments.
- Entity Framework Core:
    - Models use AuditableEntity to track who created or changed data and when.
    - Migrations keep the database schema up to date.

- Repository and Unit of Work Patterns:
    - Organize data tasks to avoid repeats and keep things consistent.
    - Interfaces like ISolutionRepository make data handling clean and abstract.

Frontend and UX

- TinyMCE Integration:
    - Lets users create rich content for templates and solutions.
    - Separate JavaScript files handle templates vs. solutions for smoother use.
- Attachments and References:
    - Files get saved in a specific folder, with details stored in the database.
    - JavaScript manages the interface for adding and viewing attachments or links.
- Routing and Navigation:
    - Routes are set up to send unauthenticated users to the login page.
    - Each Area has its own routes to guide users based on their role.

Email and Notifications

- Uses SMTP (Gmail) for emails, configured in appsettings.json.
- Sends automatic emails for things like user approvals or password resets.

Deployment and Logging

- Enforces HTTPS and HSTS for secure connections.
- Includes console and debug logs to track what's happening.
- Ready to deploy on IIS or Kestrel, with settings for different environments.

6.3. System Architecture Diagram

Below is a high-level architecture diagram illustrating the layered structure of the CormSquare Support Hub:

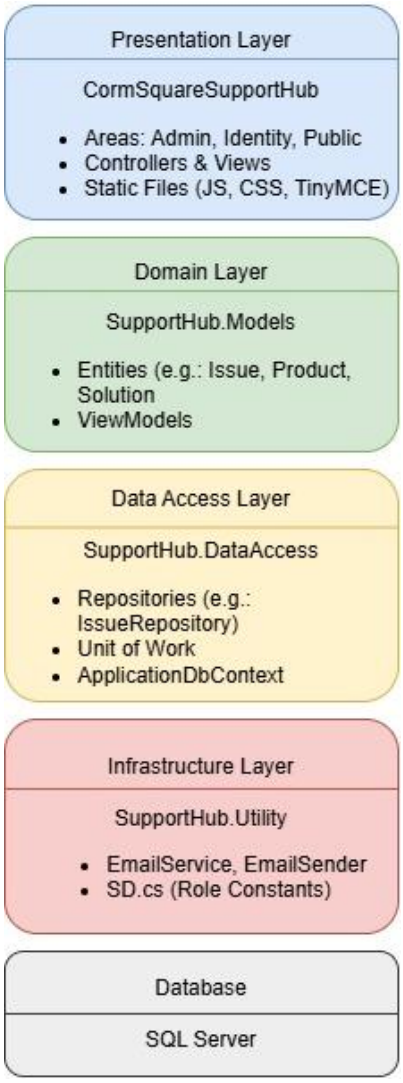


Figure 3: System Architecture Diagram

Legend:

- Presentation Layer: Handles user interactions and UI rendering.
- Domain Layer: Contains business entities and ViewModels.
- Data Access Layer: Manages data persistence and retrieval.
- Infrastructure Layer: Provides utility services and configurations.
- Database: Stores application data.

6.4. Summary of Technologies Used

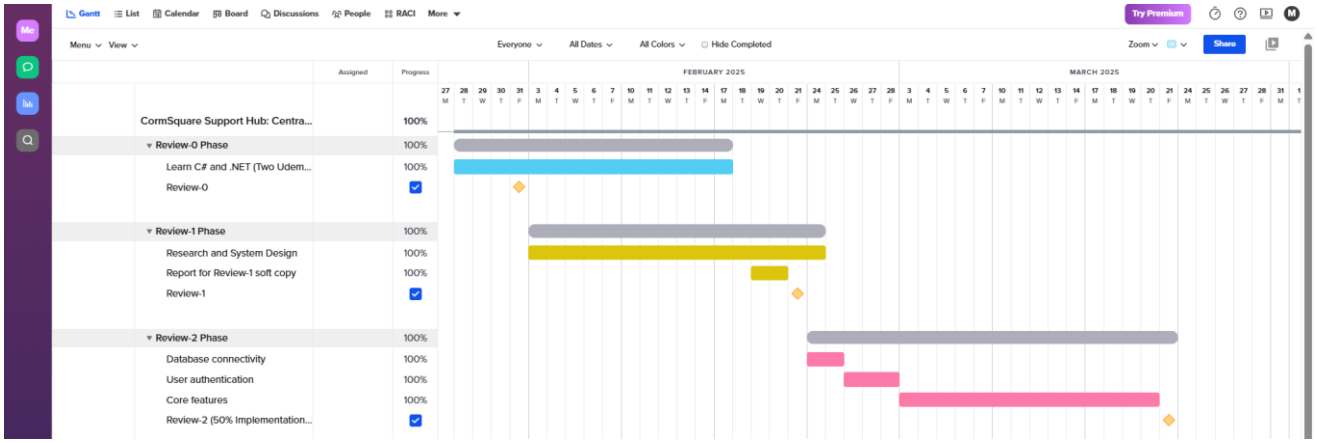
Technology	Purpose
ASP.NET Core MVC	Web application framework
Entity Framework Core	ORM for data access
SQL Server	Backend relational database
Identity Framework	Authentication and authorization
TinyMCE	Rich-text content editing
JavaScript/jQuery	Frontend interactivity and logic
SMTP (Gmail)	Email notifications
Razor Pages	UI rendering with server-side logic
Repository Pattern	Abstracted data operations
Unit of Work Pattern	Transaction management

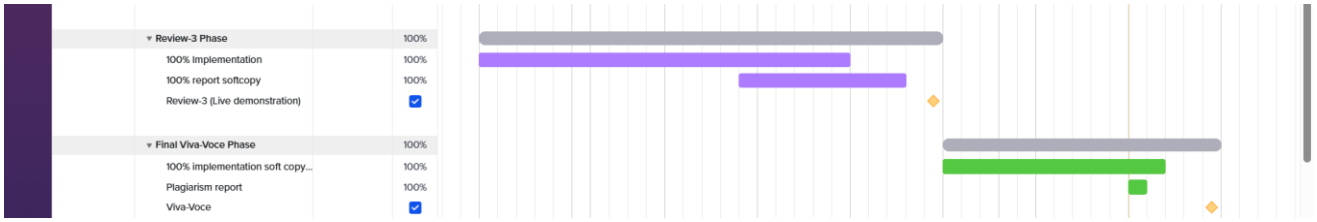
Table 1: Technologies Used and Their Functional Roles in the Knowledge Management System

Chapter 7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)





Chapter 8

OUTCOMES

Here's what I achieved with the CormSquare Support Hub—a practical, user-friendly Knowledge Management System that delivers on its goals. These outcomes show how the system came together to make knowledge sharing easier, more secure, and organized.

8.1. Centralized Knowledge Repository

- Built a platform where solutions, FAQs, and documents are neatly organized using categories and templates.
- Templates let users add rich text (thanks to TinyMCE), attach files, and include links, making it a one-stop hub for knowledge.

8.2. Role-Based Access Control

- Set up clear roles—Admin, KM Creator, KM Champion, Internal User, and External User—each with specific permissions.
- Locked down internal documents, attachments, and links so only internal users can see them, keeping sensitive info safe.

8.3. Template-Based Solution Creation

- Made it easy to create solutions using reusable templates (like categories) for consistency.
- Users can tweak formatting, add files or links, and save their work without messing with the original template.

8.4. Review & Approval Workflow

- KM Champions can check submitted solutions, add feedback, and mark them as Under Review, Needs Changes, Rejected, or Approved.
- This process ensures only top-notch content makes it to the knowledge base.

8.5. Email Notification Integration

- Added automatic emails using MailKit (via Gmail SMTP) for things like user approvals and password resets.
- Kept it secure and reliable so users stay in the loop without a hitch.

8.6. Secure File Upload Management

- Added automatic emails using MailKit (via Gmail SMTP) for things like user approvals and password resets.
- Kept it secure and reliable so users stay in the loop without a hitch.

8.7. Rich Text and Reference Support

- Added automatic emails using MailKit (via Gmail SMTP) for things like user approvals and password resets.
- Kept it secure and reliable so users stay in the loop without a hitch.

8.8. Searchable Knowledge Base

- Added automatic emails using MailKit (via Gmail SMTP) for things like user approvals and password resets.
- Kept it secure and reliable so users stay in the loop without a hitch.

8.9. Scalable, Maintainable Architecture

- Added automatic emails using MailKit (via Gmail SMTP) for things like user approvals and password resets.
- Kept it secure and reliable so users stay in the loop without a hitch.

8.10. User-Friendly Interface

- Added automatic emails using MailKit (via Gmail SMTP) for things like user approvals and password resets.
- Kept it secure and reliable so users stay in the loop without a hitch.

Chapter 9

RESULTS AND DISCUSSIONS

9.1. Role-Based Access and User Management

The application successfully enforces strict access control based on user roles:

- Admins manage users and templates.
- KM Creators draft and submit solutions using pre-defined templates.
- KM Champions review, approve, or reject submitted solutions.
- Internal and External Users have appropriately restricted access to content, especially internal-only documents.

This granular role-based system improves system security and workflow integrity across different user types.

9.2. Template-Driven Solution Creation

A category-based template system was implemented to ensure structured and consistent documentation. This led to:

- Faster creation of solutions by reusing standardized layouts.
- Reduced formatting inconsistencies between solutions.
- Effective inheritance of reference and attachment settings, enhancing modularity and reuse.

The use of the TinyMCE editor enabled rich formatting while preserving a separation between stored HTML content and plain text used for search.

9.3. Enhanced Reference and Attachment Management

The platform supports:

- Uploading categorized attachments with captions.
- Tagging references to open in new or same tabs.
- Flagging content (attachments or references) as internal to restrict visibility.

This feature ensures that sensitive information is appropriately protected and that content is displayed meaningfully to the intended audience.

9.4. Email Communication Integration

Email functionalities were implemented using the MailKit library:

- Account activation, password reset, and notification emails are handled via Gmail's SMTP server.
- Email templates were configured to provide consistent formatting.
- Asynchronous sending ensures system responsiveness during user operations.

The system can easily adapt to other SMTP providers if needed, supporting long-term flexibility.

9.5. Robust Document Search Capabilities

The application supports full-text search using SQL Server's Full-Text Search engine on the plain-text version of the content:

- HTML tags are stripped during content storage using HtmlAgilityPack.
- Users can find relevant documents using keyword-based search without requiring exact matches.

This improves knowledge retrieval and ensures quicker access to solutions.

9.6. Data Persistence and File Handling

Attachments are stored on disk with metadata (file name, path, caption) saved in the database. The system supports:

- Organized folder structures for categories and solutions.
- Independent storage for solution-specific files, even if derived from templates.
- Internal-only content is consistently filtered based on user roles during retrieval.

9.7. Interactive Visualization using Chart.js

An interactive donut chart was integrated on the home page using Chart.js to visually represent the distribution of published solutions across different category templates.

- The chart dynamically fetches data from the backend, reflecting real-time counts of published solutions per category.
- This enhances the user experience by offering immediate visual insights into which knowledge areas are most populated.
- KM Champions and Admins can use this at a glance to identify underrepresented categories and focus efforts accordingly.

Benefits:

- Encourages data-driven decision-making.
- Makes the system feel responsive and modern.
- Aids stakeholders in identifying documentation gaps.

9.8. Discussion and Insights

- **System Usability:** User testing indicated that the interface is intuitive, especially the template-based solution creation and category-wise filters.
- **Security & Access Control:** Role-based access and internal/external flags ensure that sensitive data is not exposed inadvertently.
- **Extensibility:** The project structure (separated into logical layers and services) makes future enhancements — such as dashboards, analytics, and collaboration tools — straightforward to implement.
- **Challenges Encountered:**
    - Real-time editing and version control are not yet supported.
    - SMTP credentials are currently hardcoded and should be moved to secure storage (e.g., environment variables or secrets manager).
    - Performance optimizations will be required as the data volume increases, especially for attachments and search.

Chapter 10

CONCLUSION

This project brought to life a practical, role-based Knowledge Management System using .NET Core MVC. By setting up category-based templates, clear workflows for different user roles, and solid handling of attachments and links, I created a system that keeps documentation consistent and makes managing solutions a breeze.

I nailed key features like full-text search, rich text editing with TinyMCE, email notifications, and tight access controls, all tested and working smoothly. There's still room to grow—things like scoring, trending content, and real-time collaborative editing are on the to-do list—but what's here is a strong, scalable starting point.

Plus, I added a neat Chart.js donut chart on the homepage that shows how solutions are spread across categories in real time. It's not just eye-catching; it helps admins and users spot gaps and make smarter decisions about where to focus. All in all, this platform is sturdy, ready for future upgrades, and perfectly geared to help organizations share knowledge effectively.

REFERENCES

(n.d.). Retrieved from Microsoft Docs. "Full-Text Search - SQL Server.": https://learn.microsoft.com/en-us/sql/full-text-search/

Dalkir, K. (2013). *Knowledge management in theory and practice.* routledge.

Gostoji{\'c}, S. a. (2014). Semantic driven document and workflow management. In *Proceedings of the international conference on applied internet and information technologies (ICAIIT 2014). Zrenjanin, Serbia* (pp. 229--234).

Heer, J. a. (n.d.). A tour through the visualization zoo. *Communications of the ACM, 53*, 59--67.

Nonaka, I. (2009). The knowledge-creating company. In *The economic impact of knowledge* (pp. 175--187). Routledge.

Ricci, F. a. (2010). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1--35). Springer.

Sandhu, R. S. (1998). Role-based access control. In *Advances in computers* (Vol. 46, pp. 237--286). Elsevier.

TinyMCE 7 Documentation. (n.d.). Retrieved from TinyMCE Documentation. "WYSIWYG HTML editor.": https://www.tiny.cloud/docs/tinymce/latest/

Vogels, T. a.-E. (2018). Web2text: Deep structured boilerplate removal. (pp. 167--179). Springer.

Xu, L. a. (2019). Data-driven approach for quality evaluation on knowledge sharing platform. In *2019 International Conference on Machine Learning and Cybernetics (ICMLC)* (pp. 1--6). IEEE.

APPENDIX-A

PSUEDOCODE

1. Upsert GET Action (SolutionController.cs)

This action renders the "Create" or "Edit" page and populates ViewData["AttachmentLinks"] with attachment URLs.

```
FUNCTION Upsert(solutionId: Optional Integer, issueId: Optional Integer)
    // Authenticate user
    user = GetCurrentUser()
    IF user IS NULL THEN
        RETURN Unauthorized()
    END IF

    // Initialize view model
    model = CreateNewSolutionViewModel()
    model.Categories = FetchAllCategories()
    model.Products = FetchAllProducts()

    IF solutionId EXISTS THEN
        // Edit mode: Fetch existing solution
        solution = FetchSolutionById(solutionId, user.Id)
        IF solution IS NULL THEN
            RETURN NotFound()
        END IF

        // Populate model with solution data
        model.Id = solution.Id
        model.Title = solution.Title
        model.ProductId = solution.ProductId
        model.SubCategoryId = solution.SubCategoryId
        model.CategoryId = solution.CategoryId
        model.HtmlContent = solution.HtmlContent
        model.IssueDescription = solution.IssueDescription
        model.Attachments = solution.Attachments
        model.References = solution.References
        model.SubCategories = FetchSubCategoriesByProduct(solution.ProductId)

        // Populate ViewData with attachment links for rendering
        attachmentLinks = EMPTY_LIST
        FOR EACH attachment IN model.Attachments
            IF NOT attachment.IsDeleted THEN
                link = CREATE_OBJECT
                link.Id = attachment.Id
                link.FileName = attachment.FileName
                link.Url = GenerateUrl("DownloadAttachmentForReview", {area: "Admin", id: attachment.Id})
                link.IsInternal = attachment.IsInternal
                ADD link TO attachmentLinks
            END IF
        END FOR
        SetViewData("AttachmentLinks", attachmentLinks)

        // Populate ViewData with reference links
        referenceLinks = EMPTY_LIST
        FOR EACH reference IN model.References
            IF NOT reference.IsDeleted THEN
                link = CREATE_OBJECT
                link.Id = reference.Id
                link.Url = reference.Url
                link.Description = reference.Description
                link.IsInternal = reference.IsInternal
                link.OpenOption = reference.OpenOption
                ADD link TO referenceLinks
            END IF
        END FOR
        SetViewData("ReferenceLinks", referenceLinks)
    ELSE IF issueId EXISTS THEN
        // Create mode with issue: Pre-fill from issue
        issue = FetchIssueById(issueId)
        IF issue EXISTS THEN
            model.IssueDescription = issue.Description
            model.ProductId = issue.ProductId
            model.SubCategoryId = issue.SubCategoryId
            model.SubCategories = FetchSubCategoriesByProduct(issue.ProductId)
        END IF
    END IF

    SetViewData("HtmlContent", model.HtmlContent OR "")
    RETURN RenderView("Upsert", model)
END FUNCTION
```

2. Upsert POST Action (SolutionController.cs)

This action handles form submission for creating or editing a solution, including saving attachments.

```
FUNCTION UpsertPost(model: SolutionViewModel, files: List<File>, ReferenceData: String, AttachmentData: String, submitAction: String)
    // Authenticate user
    user = GetCurrentUser()
    IF user IS NULL THEN
        RETURN Json({success: false, message: "Unauthorized"})
    END IF

    // Handle cancellation
    IF submitAction = "Cancel" THEN
        redirectUrl = IF model.Id > 0 THEN "MySolutions" ELSE "IssueList"
        RETURN Json({success: true, redirectTo: redirectUrl})
    END IF

    // Validate model
    IF NOT ModelIsValid() THEN
        errors = CollectModelErrors()
        PopulateViewModel(model)
        RETURN Json({success: false, message: "Validation failed", errors: errors})
    END IF

    // Begin database transaction
    BeginTransaction()

    TRY
        solution = NULL
        savedAttachments = EMPTY_LIST
        savedReferences = EMPTY_LIST

        IF model.Id > 0 THEN
            // Edit mode: Update existing solution
            solution = FetchSolutionById(model.Id)
            IF solution IS NULL THEN
                RollbackTransaction()
                RETURN Json({success: false, message: "Solution not found"})
            END IF

            solution.Title = model.Title
            solution.CategoryId = model.CategoryId
            solution.ProductId = model.ProductId
            solution.SubCategoryId = model.SubCategoryId
            solution.HtmlContent = model.HtmlContent
            solution.PlainTextContent = ConvertHtmlToPlainText(model.HtmlContent)
            solution.IssueDescription = model.IssueDescription
            solution.Status = IF submitAction = "Save" THEN "Draft" ELSE "Submitted"
            solution.AuthorId = user.Id
            UpdateAudit(solution, user.Id)
            UpdateSolution(solution)

        ELSE
            // Create mode: Create new solution
            solution = CREATE_OBJECT
            solution.Title = model.Title
            solution.CategoryId = model.CategoryId
            solution.ProductId = model.ProductId
            solution.SubCategoryId = model.SubCategoryId
            solution.HtmlContent = model.HtmlContent
            solution.PlainTextContent = ConvertHtmlToPlainText(model.HtmlContent)
            solution.IssueDescription = model.IssueDescription
            solution.Status = IF submitAction = "Save" THEN "Draft" ELSE "Submitted"
            solution.AuthorId = user.Id
            UpdateAudit(solution, user.Id)
            AddSolution(solution)
        END IF

        SaveChanges()

        // Process references
        IF ReferenceData IS NOT EMPTY THEN
            savedReferences = ProcessReferences(solution, ReferenceData, user.Id)
            SaveChanges()
        END IF

        // Process attachments
        IF AttachmentData IS NOT EMPTY THEN
            savedAttachments = ProcessAttachments(solution, files, AttachmentData, user.Id)
            SaveChanges()
        END IF
```

```
        IF submitAction = "Save" OR submitAction = "Submit" THEN
            // Save attachments to file system
            solutionPath = ConstructPath("solutions", solution.Id)
            IF NOT DirectoryExists(solutionPath) THEN
                CreateDirectory(solutionPath)
            END IF

            FOR EACH attachment IN savedAttachments
                destPath = ConstructPath(solutionPath, attachment.fileName)
                IF FileExists(destPath) THEN
                    CONTINUE
                END IF

                sourcePath = FindSourcePath(attachment, files)
                IF sourcePath EXISTS AND FileExists(sourcePath) THEN
                    CopyFile(sourcePath, destPath)
                ELSE
                    uploadedFile = FindUploadedFile(files, attachment.originalFileName)
                    IF uploadedFile EXISTS THEN
                        SaveFile(uploadedFile, destPath)
                    END IF
                END IF
            END FOR

            SaveChanges()
            CommitTransaction()

            redirectUrl = IF submitAction = "Save" THEN "MySolutions" ELSE "Approvals"
            RETURN Json({success: true, redirectTo: redirectUrl, attachments: savedAttachments, references: savedReferences})
        ELSE
            RollbackTransaction()
            RETURN Json({success: false, message: "Invalid submit action"})
        END IF
    CATCH Exception ex
        RollbackTransaction()
        RETURN Json({success: false, message: "Error: " + ex.Message})
    END TRY
END FUNCTION
```

3. DownloadAttachmentForReview Action (SolutionController.cs)

This action serves attachments with Content-Disposition: inline to open in a new tab for viewable file types.

```
FUNCTION DownloadAttachmentForReview(id: Integer)
    // Authenticate user
    user = GetCurrentUser()
    IF user IS NULL THEN
        RETURN Unauthorized()
    END IF

    // Fetch attachment
    attachment = FetchAttachmentById(id)
    IF attachment IS NULL OR attachment.IsDeleted THEN
        RETURN NotFound()
    END IF

    // Construct file path
    fullPath = ConstructPath(attachmentSettings.UploadPath, attachment.FilePath)

    IF NOT FileExists(fullPath) THEN
        RETURN NotFound()
    END IF

    TRY
        // Determine MIME type and Content-Disposition
        mimeType = "application/octet-stream"
        contentDisposition = "attachment"
        extension = GetFileExtension(attachment.FileName)

        IF extension EXISTS THEN
            IF extension = ".pdf" THEN
                mimeType = "application/pdf"
                contentDisposition = "inline"
            ELSE IF extension = ".png" THEN
                mimeType = "image/png"
                contentDisposition = "inline"
            ELSE IF extension = ".jpg" OR extension = ".jpeg" THEN
                mimeType = "image/jpeg"
                contentDisposition = "inline"
            ELSE IF extension = ".txt" THEN
                mimeType = "text/plain"
                contentDisposition = "inline"
            ELSE IF extension = ".docx" THEN
                mimeType = "application/vnd.openxmlformats-officedocument.wordprocessingml.document"
                contentDisposition = "attachment"
            END IF
        END IF
```

```
        // Serve file
        fileStream = OpenFile(fullPath)
        SetResponseHeader("Content-Disposition", contentDisposition + "; filename=\"" + attachment.FileName + "\"")
        RETURN ServeFile(fileStream, mimeType)
    CATCH IOException ex
        RETURN StatusCode(500, "Error reading the attachment file")
    CATCH Exception ex
        RETURN StatusCode(500, "An unexpected error occurred")
    END TRY
END FUNCTION
```

4. DownloadAttachment Action (SolutionController.cs)

This action forces a download with Content-Disposition: attachment, which was the root cause of the issue when used in ViewData["AttachmentLinks"].

```
FUNCTION DownloadAttachment(attachmentId: Integer)
    // Authenticate user
    user = GetCurrentUser()
    IF user IS NULL THEN
        RETURN Unauthorized()
    END IF

    // Fetch attachment
    attachment = FetchAttachmentById(attachmentId)
    IF attachment IS NULL OR attachment.IsDeleted THEN
        RETURN NotFound()
    END IF

    // Construct file path
    fullPath = ConstructPath(attachmentSettings.UploadPath, attachment.FilePath)

    IF NOT FileExists(fullPath) THEN
        RETURN NotFound()
    END IF

    TRY
        // Determine MIME type
        mimeType = "application/octet-stream"
        extension = GetFileExtension(attachment.FileName)
        IF extension EXISTS THEN
            IF extension = ".pdf" THEN
                mimeType = "application/pdf"
            ELSE IF extension = ".png" THEN
                mimeType = "image/png"
            ELSE IF extension = ".jpg" OR extension = ".jpeg" THEN
                mimeType = "image/jpeg"
            ELSE IF extension = ".txt" THEN
                mimeType = "text/plain"
            ELSE IF extension = ".docx" THEN
                mimeType = "application/vnd.openxmlformats-officedocument.wordprocessingml.document"
            END IF
        END IF

        // Serve file with forced download
        fileStream = OpenFile(fullPath)
        SetResponseHeader("Content-Disposition", "attachment; filename=\"" + attachment.FileName + "\"")
        RETURN ServeFile(fileStream, mimeType, attachment.FileName)
    CATCH IOException ex
        RETURN StatusCode(500, "Error reading file")
    CATCH Exception ex
        RETURN StatusCode(500, "Unexpected error")
    END TRY
END FUNCTION
```

5. **Client-Side Attachment Rendering (Upsert.cshtml)**
   This pseudocode represents how Upsert.cshtml renders the attachment links.

```
BEGIN VIEW Upsert.cshtml
    // Fetch ViewData
    attachmentLinks = GetViewData("AttachmentLinks")
    referenceLinks = GetViewData("ReferenceLinks")

    // Render form
    RenderFormStart("Upsert", {area: "Admin"})
    RenderFormFields(model)

    // Render attachments section
    IF attachmentLinks EXISTS AND NOT EMPTY THEN
        FOR EACH link IN attachmentLinks
            IF NOT link.IsInternal OR UserHasInternalAccess THEN
                attachmentUrl = link.Url
                fileName = link.FileName
                RenderHtml("<a href=\"" + attachmentUrl + "\" class=\"attachment-link\" target=\"_blank\" rel=\"noopener noreferrer\">" + fileName + "</a>")
            END IF
        END FOR
    END IF

    // Render references section
    IF referenceLinks EXISTS AND NOT EMPTY THEN
        FOR EACH link IN referenceLinks
            IF NOT link.IsInternal OR UserHasInternalAccess THEN
                target = IF link.OpenOption = "NewTab" THEN "_blank" ELSE "_self"
                RenderHtml("<a href=\"" + link.Url + "\" target=\"" + target + "\" rel=\"noopener noreferrer\">" + link.Description + "</a>")
            END IF
        END FOR
    END IF

    RenderFormEnd()
END VIEW
```

6. **Client-Side Attachment Handling (solution-attachments-references.js)**
   This pseudocode represents the JavaScript logic for dynamically adding and rendering attachments in the UI.

```
BEGIN SCRIPT solution-attachments-references.js

// Global state
window.attachments = EMPTY_LIST
window.references = EMPTY_LIST

FUNCTION addAttachment(event)
    file = event.target.files[0]
    IF NOT file THEN RETURN

    guidFileName = GenerateGuid() + GetFileExtension(file.name)
    attachment = CREATE_OBJECT
    attachment.id = 0
    attachment.fileName = file.name
    attachment.guidFileName = guidFileName
    attachment.url = NULL
    attachment.isInternal = GetCheckboxValue("isInternal")
    attachment.caption = GetInputValue("caption")
    ADD attachment TO window.attachments

    RenderAttachments()
END FUNCTION

```

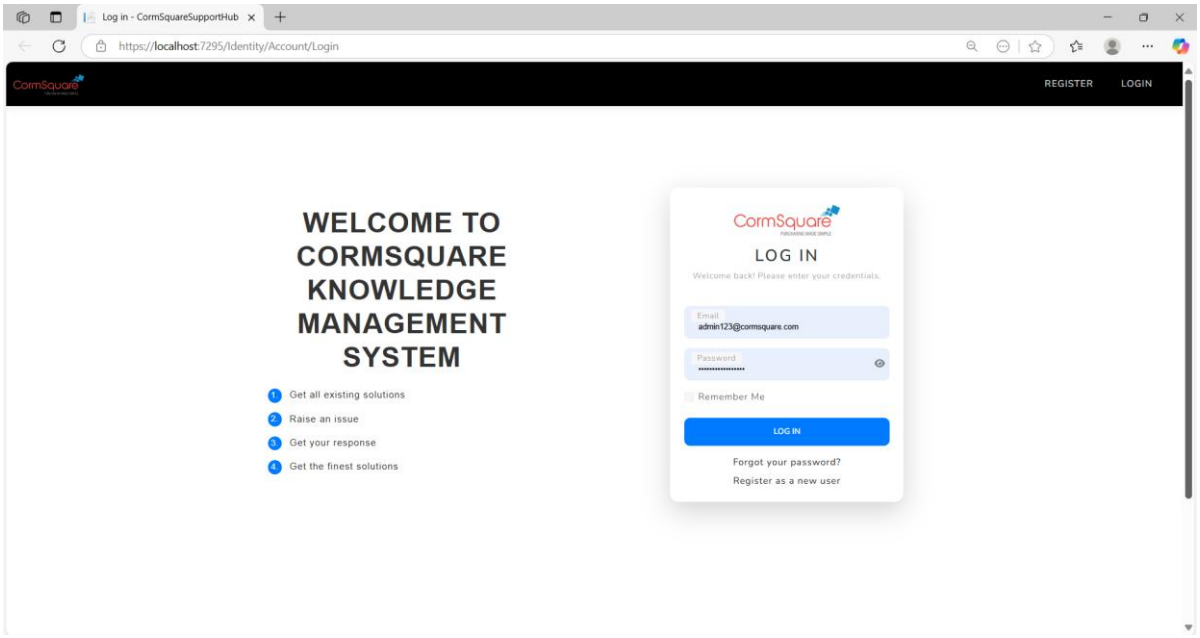
```
23
24  FUNCTION RenderAttachments()
25      container = GetElementById("attachment-list")
26      ClearElement(container)
27
28      FOR EACH attachment IN window.attachments
29          IF NOT attachment.isDeleted THEN
30              fileName = attachment.caption OR attachment.fileName
31              fileNameHtml = IF attachment.url EXISTS THEN
32                  "<a href=\"" + attachment.url + "\" class=\"attachment-link\" target=\"_blank\" rel=\"noopener noreferrer\"
33                  onclick=\"console.log('Opening: " + attachment.url + "')\">" + fileName + "</a>"
34              ELSE
35                  "<strong>" + fileName + "</strong>"
36              END IF
37
38              html = "<li data-id=\"" + attachment.id + "\">" + fileNameHtml + " (Internal: " + attachment.isInternal + ")</li>"
39              AppendHtml(container, html)
40          END IF
41      END FOR
42  END FUNCTION
```

```
43
44  FUNCTION OnFormSubmitSuccess(response)
45      IF response.success THEN
46          IF response.attachments EXISTS THEN
47              FOR EACH serverAttachment IN response.attachments
48                  localAttachment = FindAttachmentByFileName(window.attachments, serverAttachment.originalFileName)
49                  IF localAttachment EXISTS THEN
50                      localAttachment.id = serverAttachment.id
51                      localAttachment.url = GenerateUrl("DownloadAttachmentForReview", {area: "Admin", id: serverAttachment.id})
52                  END IF
53              END FOR
54              RenderAttachments()
55          END IF
56          RedirectTo(response.redirectTo)
57      ELSE
58          ShowError(response.message)
59      END IF
60  END FUNCTION
61
62  END SCRIPT
```

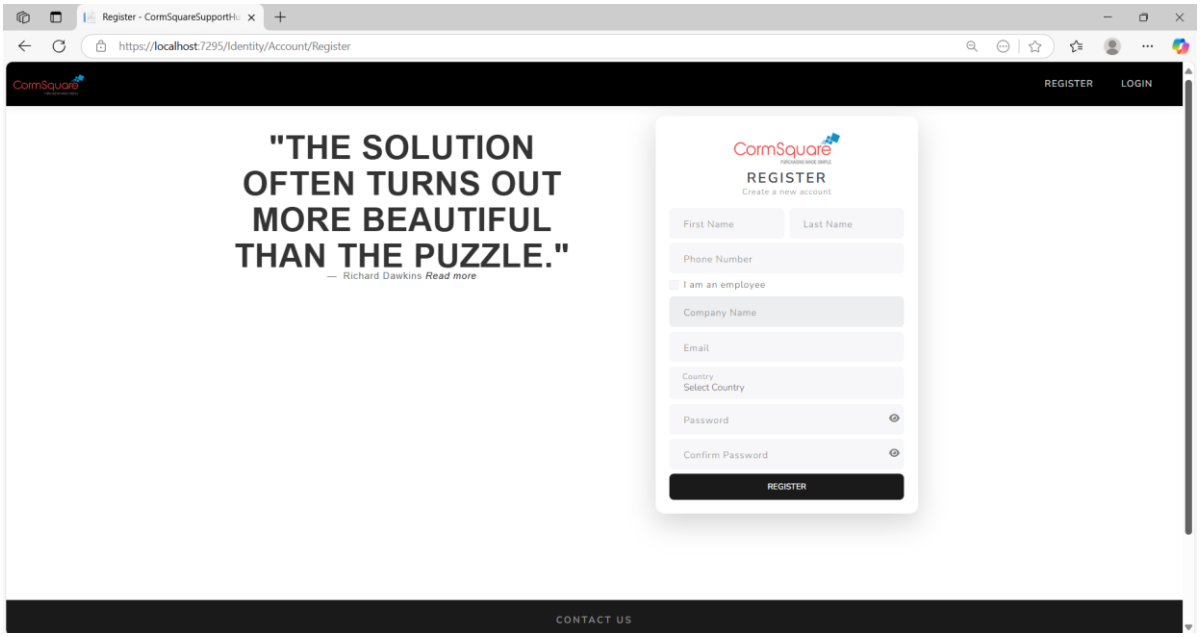
APPENDIX-B

SCREENSHOTS

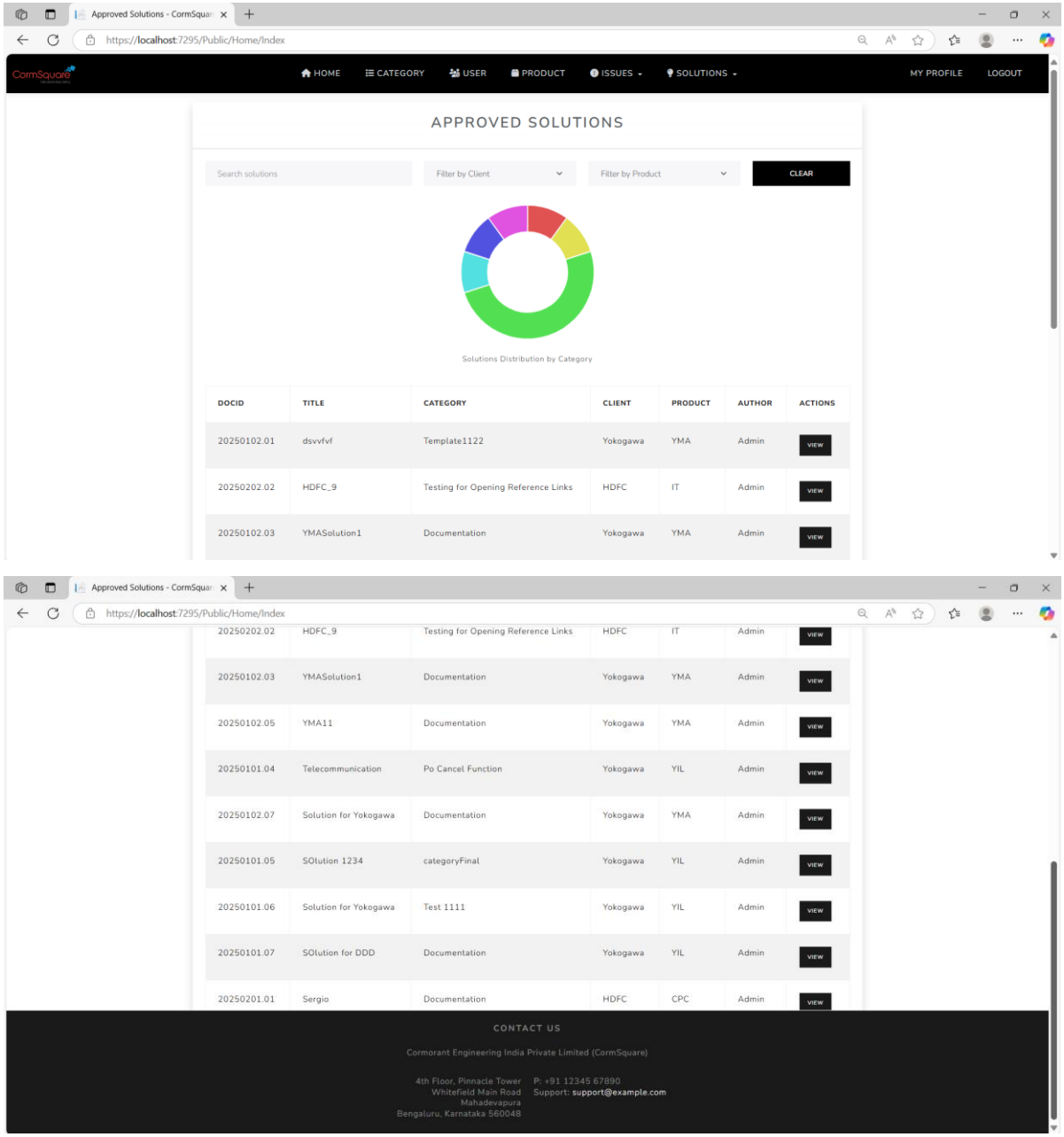
Login Page:



Register Page:

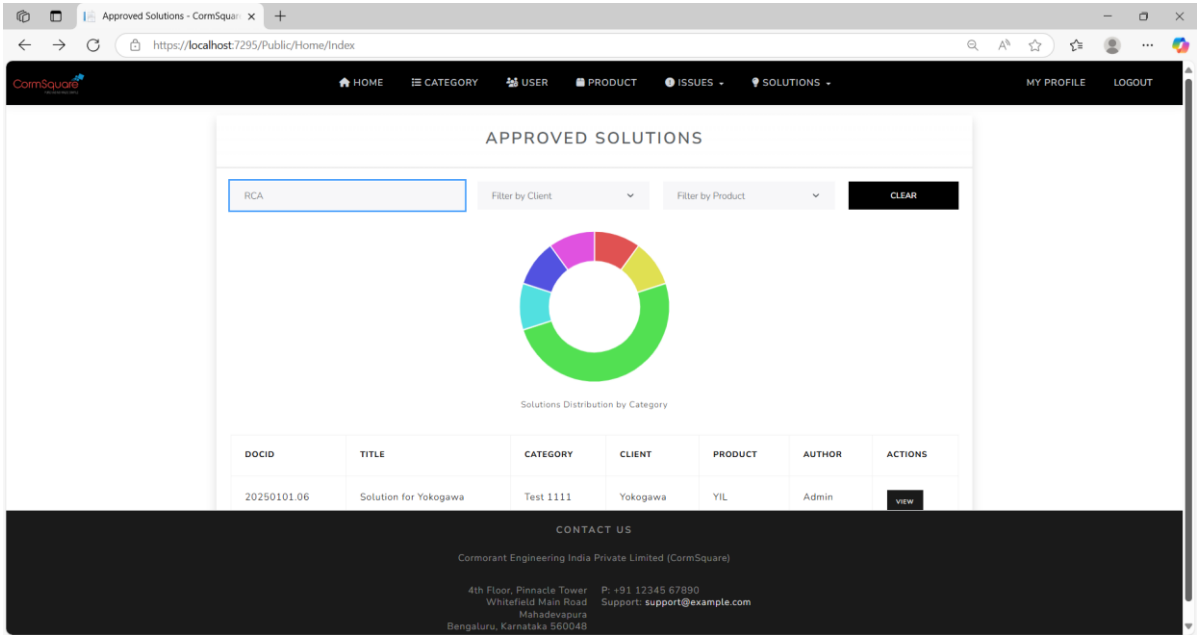


Home Page:

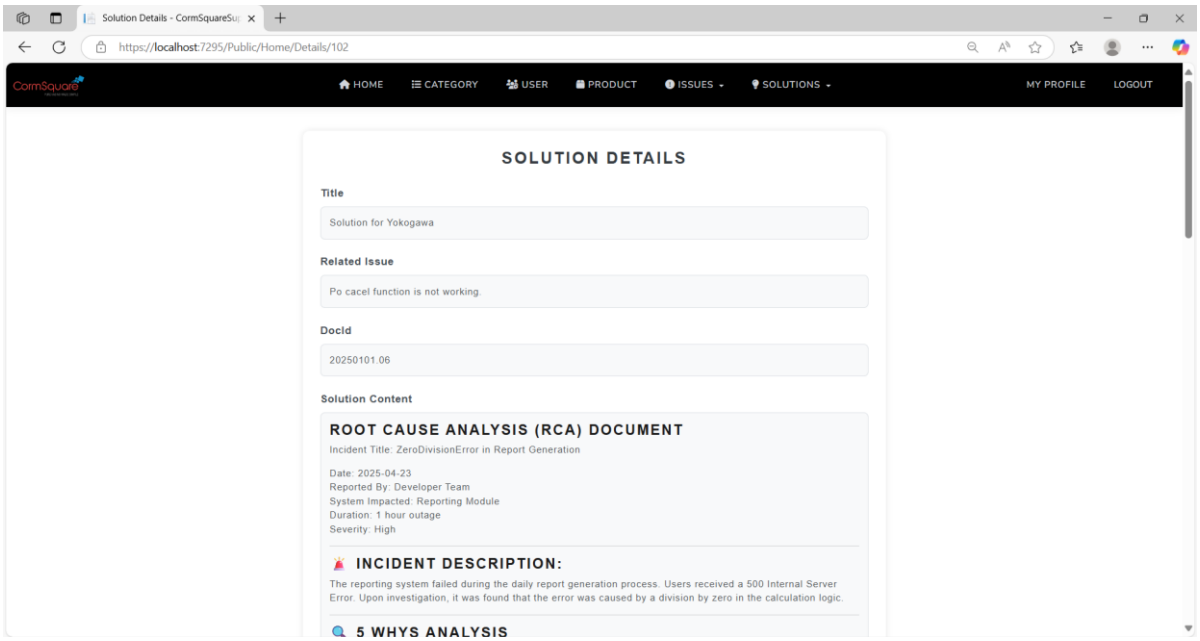

Approved Solutions - CormSquar
https://localhost:7295/Public/Home/Index
CormSquare
HOME
CATEGORY
USER
PRODUCT
ISSUES
SOLUTIONS
MY PROFILE
LOGOUT
APPROVED SOLUTIONS
Search solutions
Filter by Client
Filter by Product
CLEAR
Solutions Distribution by Category
DOCID
TITLE
CATEGORY
CLIENT
PRODUCT
AUTHOR
ACTIONS
20250102.01 dsvvfvf Template1122 Yokogawa YMA Admin VIEW
20250202.02 HDFC_9 Testing for Opening Reference Links HDFC IT Admin VIEW
20250102.03 YMASolution1 Documentation Yokogawa YMA Admin VIEW

Approved Solutions - CormSquar
https://localhost:7295/Public/Home/Index
20250202.02 HDFC_9 Testing for Opening Reference Links HDFC IT Admin VIEW
20250102.03 YMASolution1 Documentation Yokogawa YMA Admin VIEW
20250102.05 YMA11 Documentation Yokogawa YMA Admin VIEW
20250101.04 Telecommunication Po Cancel Function Yokogawa YIL Admin VIEW
20250102.07 Solution for Yokogawa Documentation Yokogawa YMA Admin VIEW
20250101.05 SOlution 1234 categoryFinal Yokogawa YIL Admin VIEW
20250101.06 Solution for Yokogawa Test 1111 Yokogawa YIL Admin VIEW
20250101.07 SOlution for DDD Documentation Yokogawa YIL Admin VIEW
20250201.01 Sergio Documentation HDFC CPC Admin VIEW
CONTACT US
Cormorant Engineering India Private Limited (CormSquare)
4th Floor, Pinnacle Tower
Whitefield Main Road
Mahadevapura
Bengaluru, Karnataka 560048
P: +91 12345 67890
Support: support@example.com

Searching in the content:



Viewing a Solution:

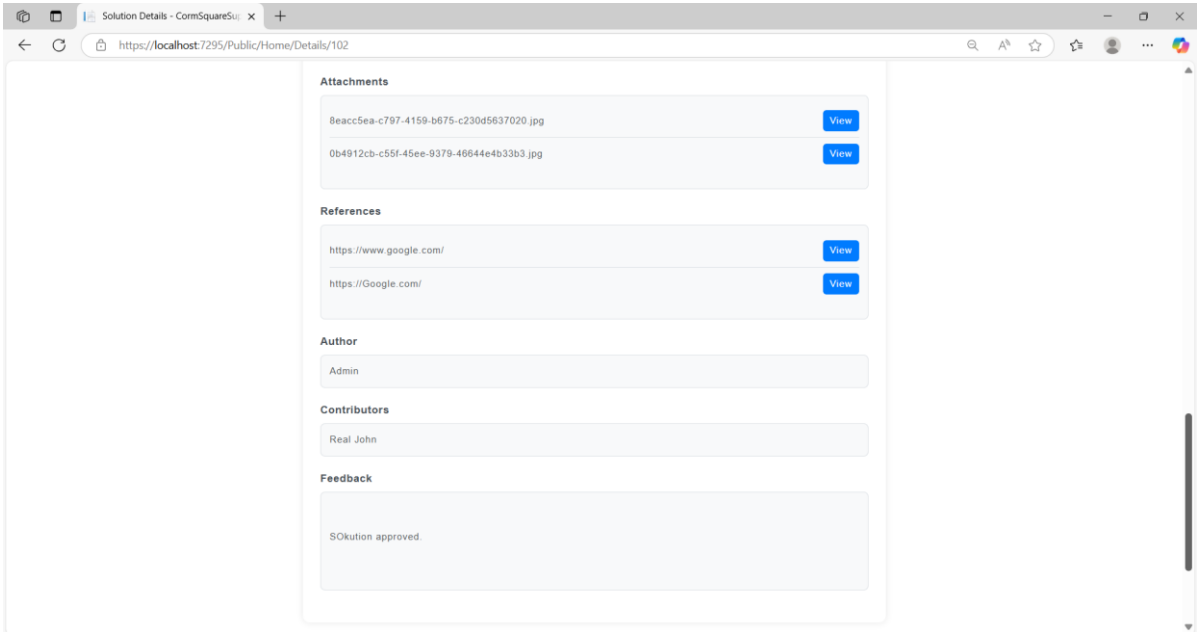


Copy button for code:

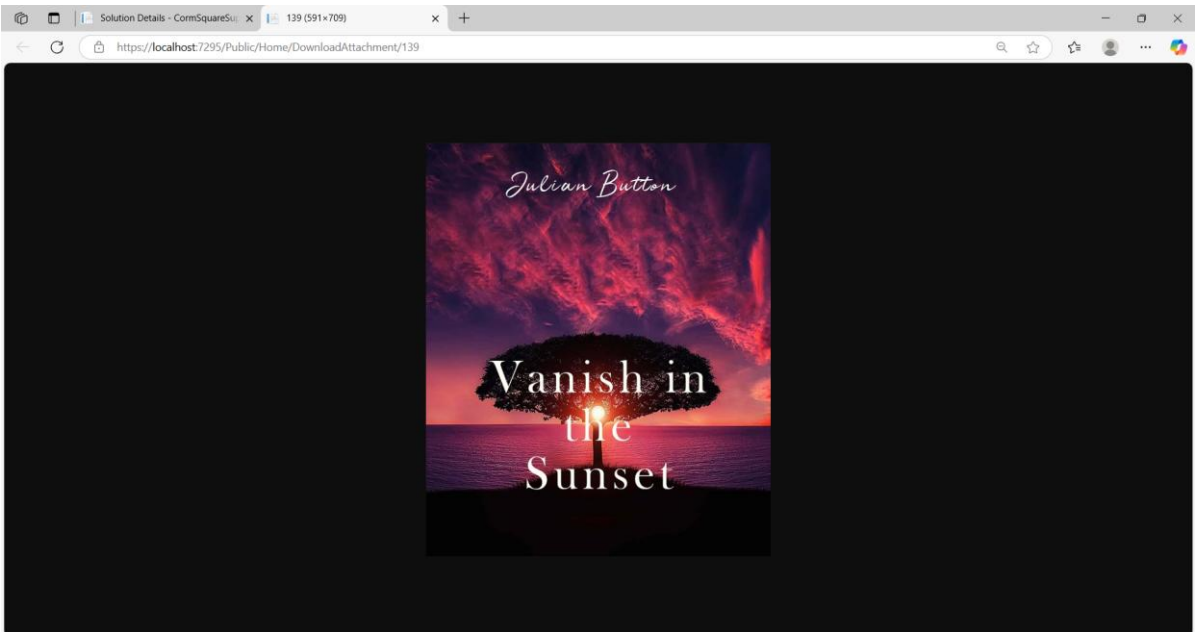
```
def generate_report(data):
    total_users = len(data.get("users", []))

    if total_users == 0:
        logger.error("Total users is zero — skipping division to avoid error.")
        return {"error": "No users found, report not generated."}

    engagement_rate = data["engaged_users"] / total_users
    return {"engagement_rate": engagement_rate}
```

Copy

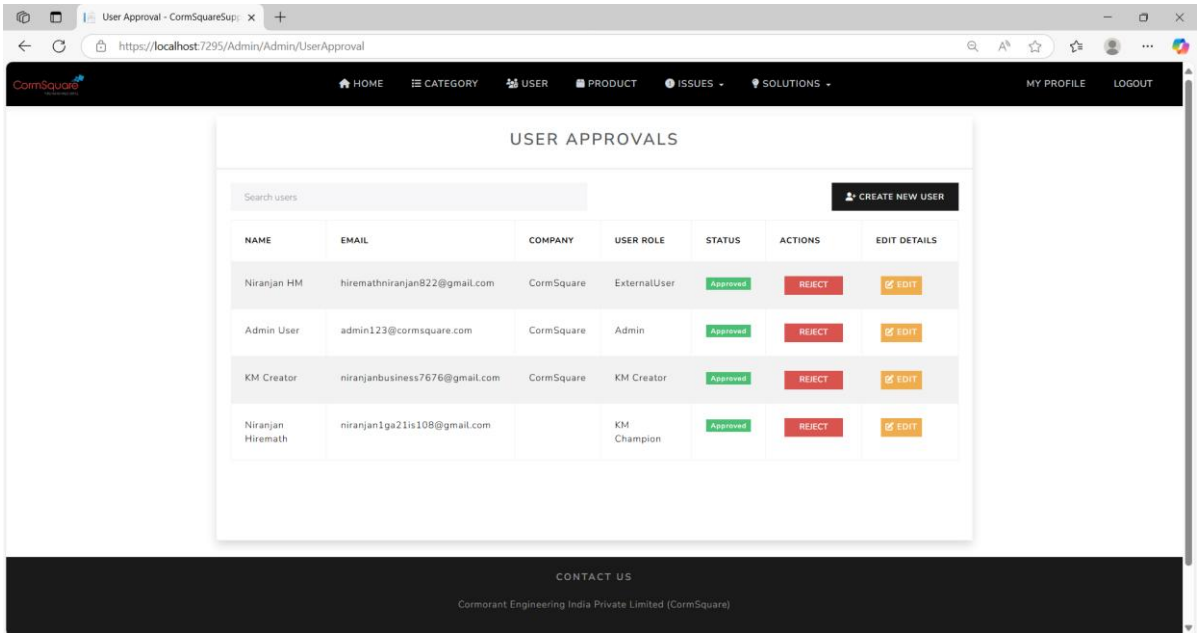
All other details (References + Attachments)



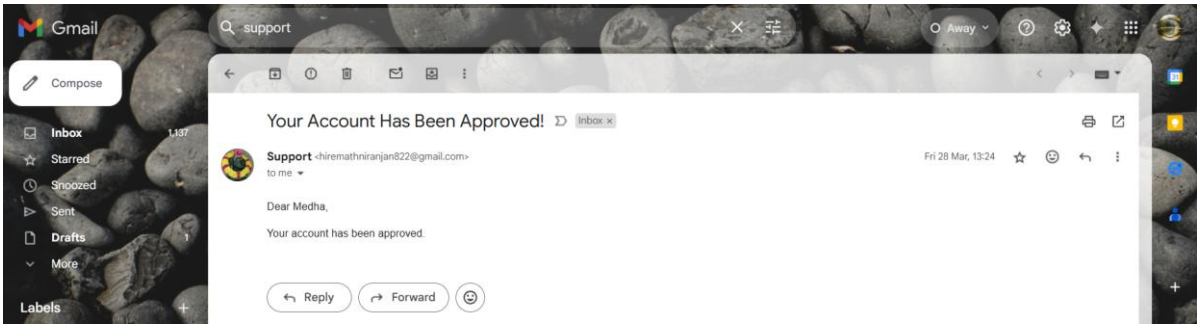
Previewing Sample Attachments:



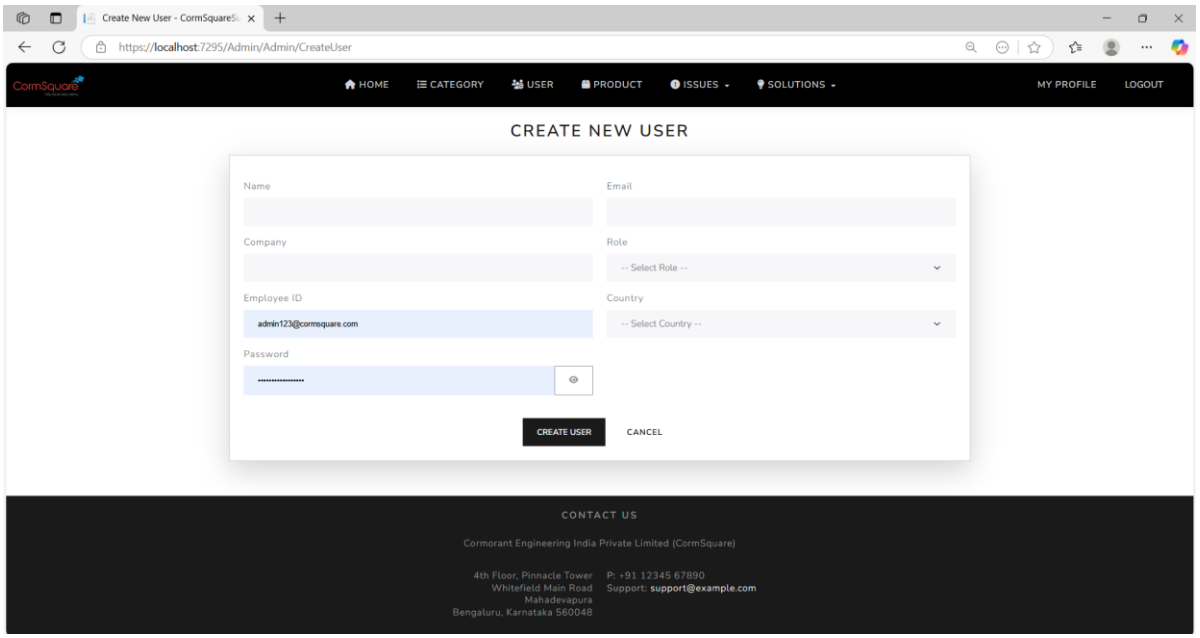
User Management:



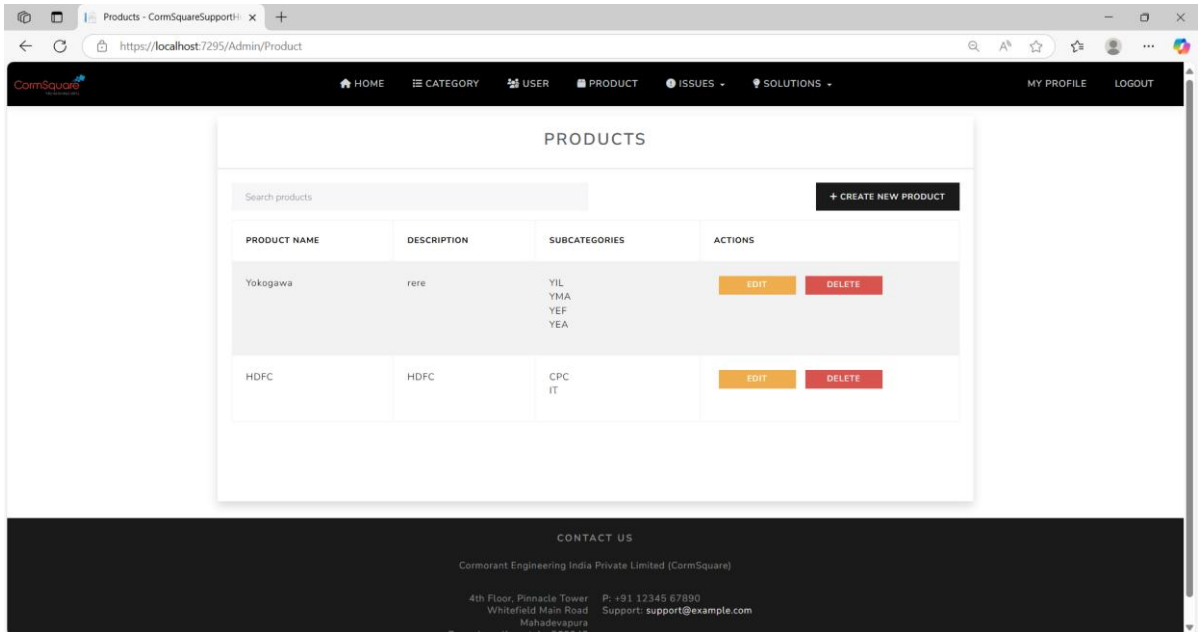
Approval Mail:

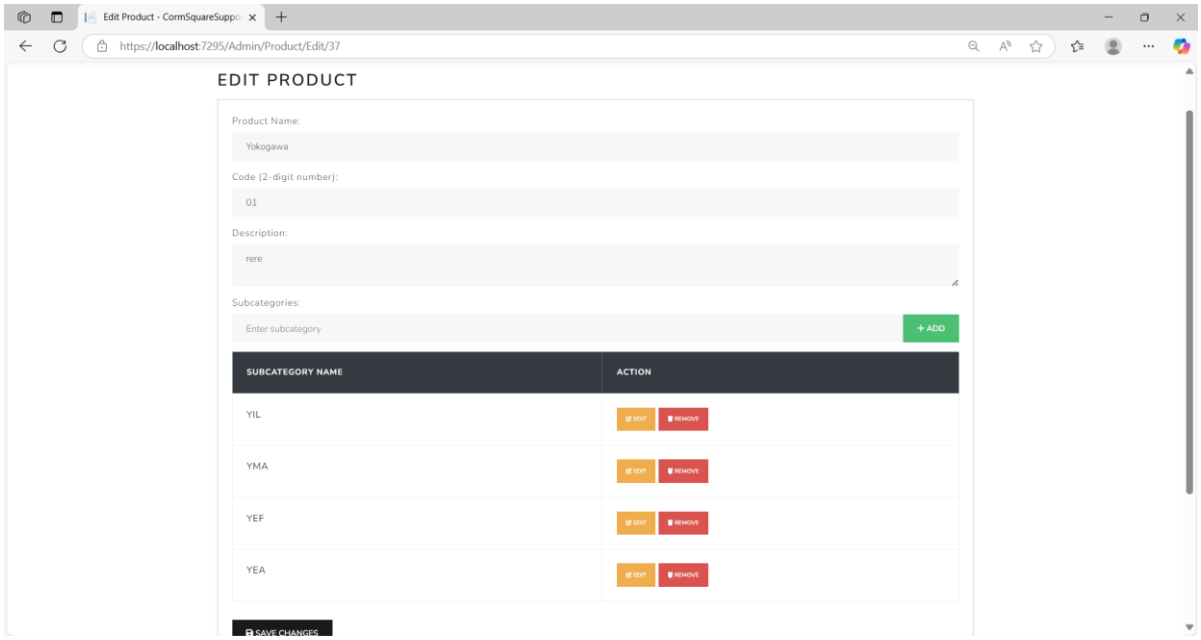


Create New User:

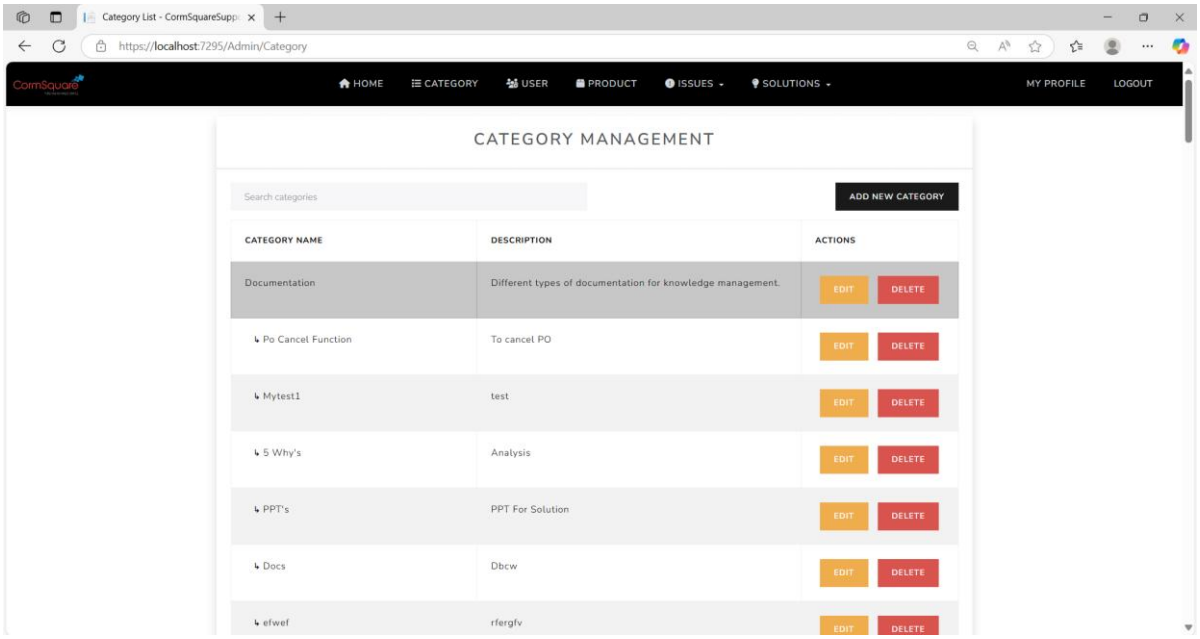


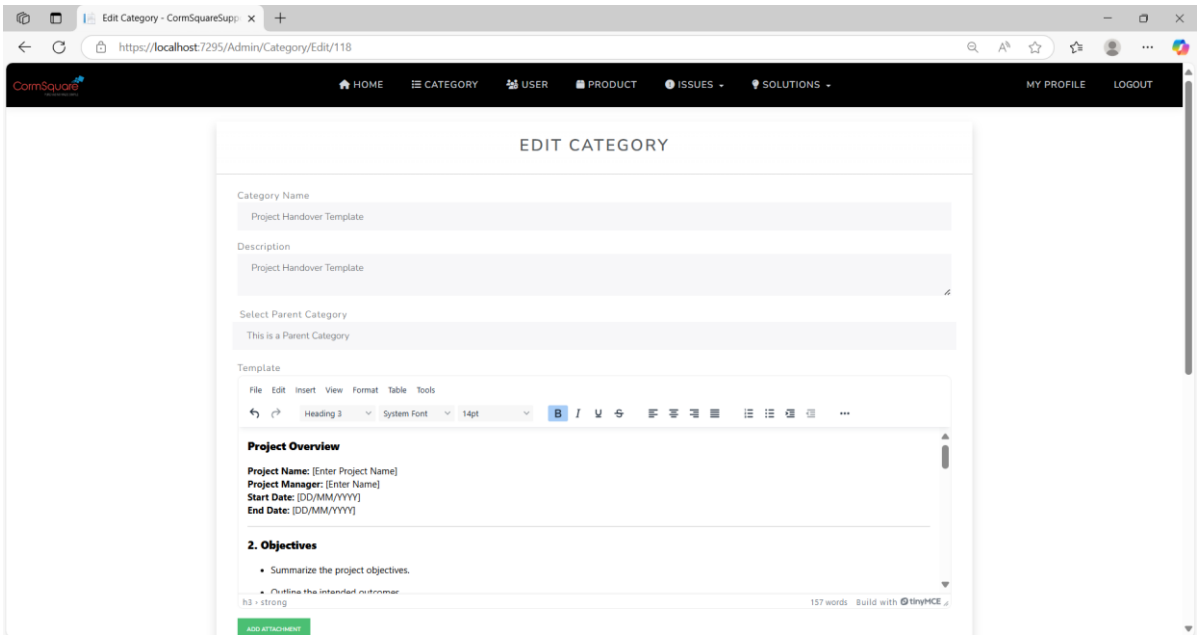
Product Management:

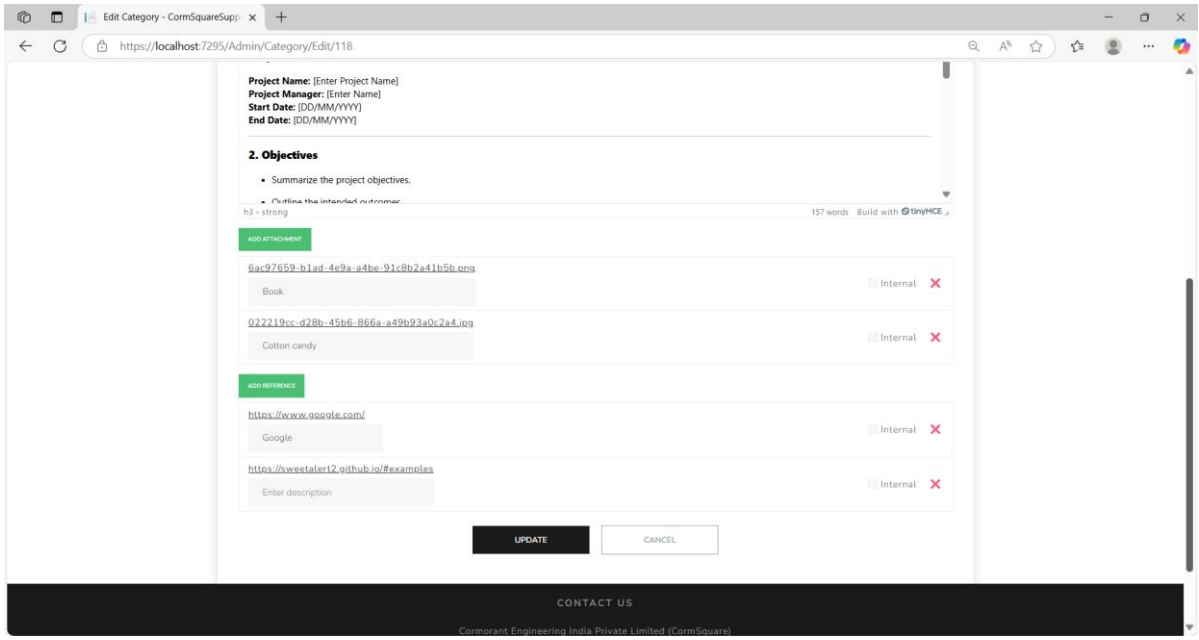




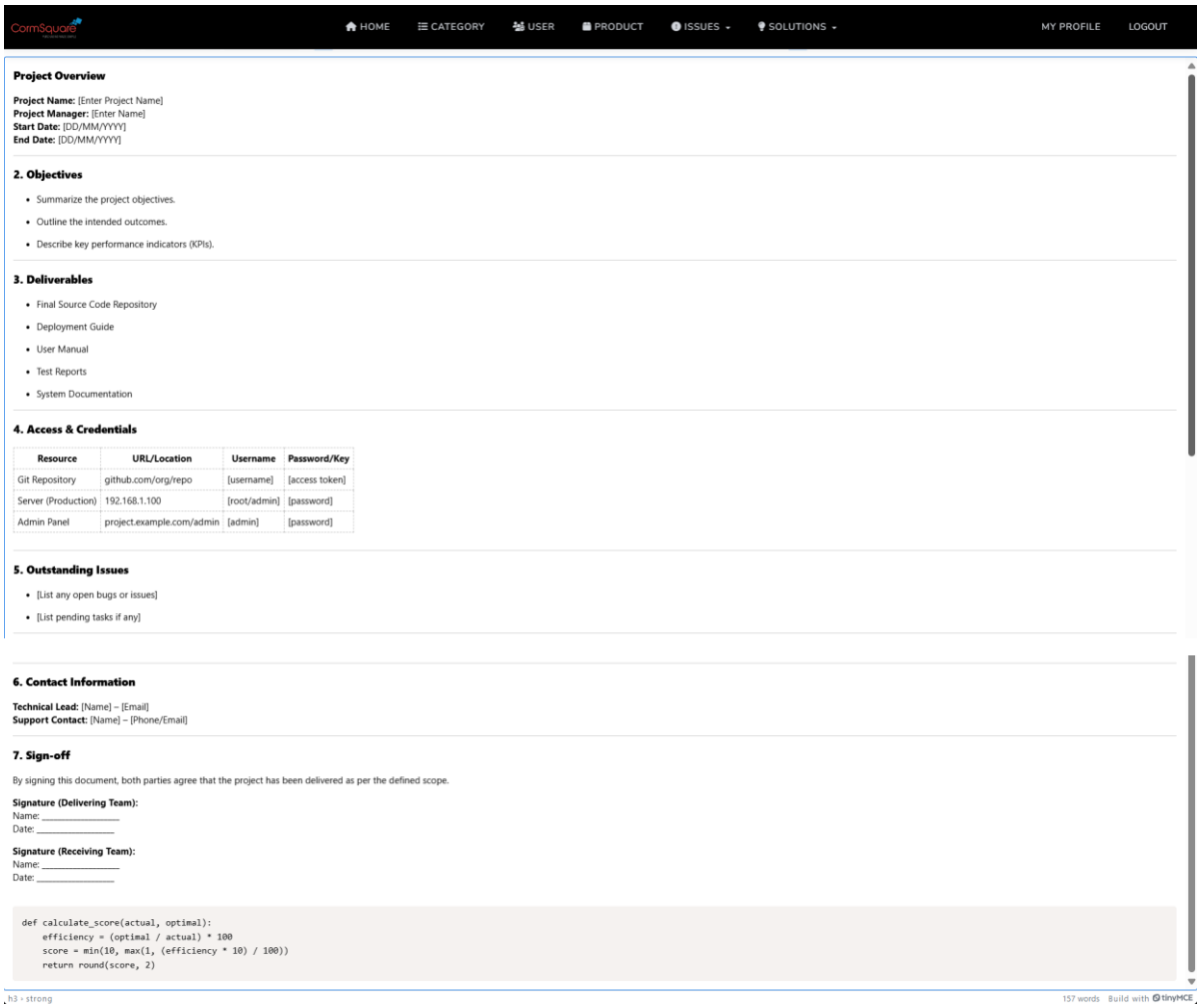
Category Management:



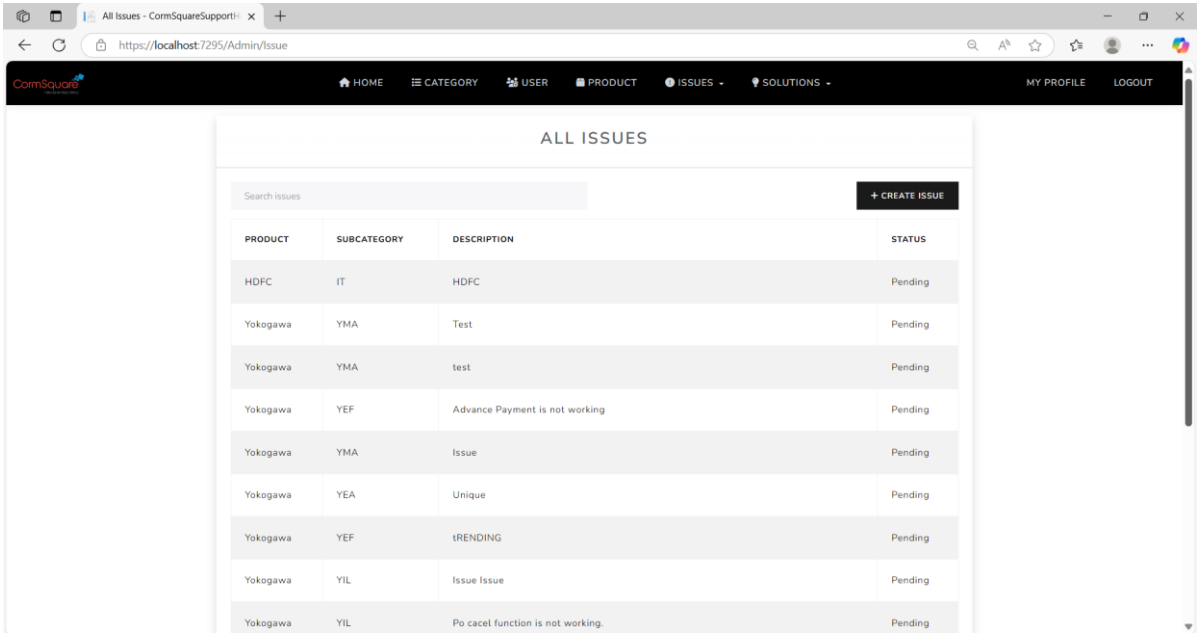


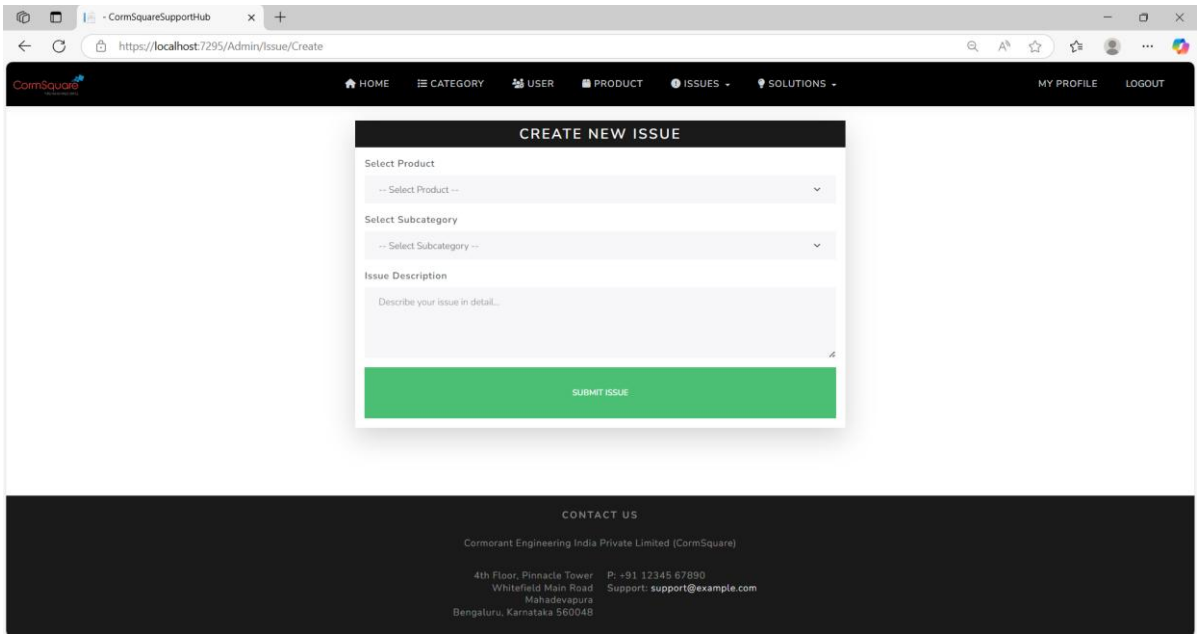

Edit Category - CormSquareSupp
https://localhost:7295/Admin/Category/Edit/118
Project Name: [Enter Project Name]
Project Manager: [Enter Name]
Start Date: [DD/MM/YYYY]
End Date: [DD/MM/YYYY]
2. Objectives
Summarize the project objectives.
h3 › strong
157 words Build with tinyMCE
ADD ATTACHMENT
6ac97659-b1ad-4e9a-a4be-91c8b2a41b5b.png
Book
Internal
022219cc-d28b-45b6-866a-a49b93a0c2a4.jpg
Cotton candy
Internal
ADD REFERENCE
https://www.google.com/
Google
Internal
https://sweetalert2.github.io/#examples
Enter description
Internal
UPDATE
CANCEL
CONTACT US
Cormorant Engineering India Private Limited (CormSquare)

Editor:

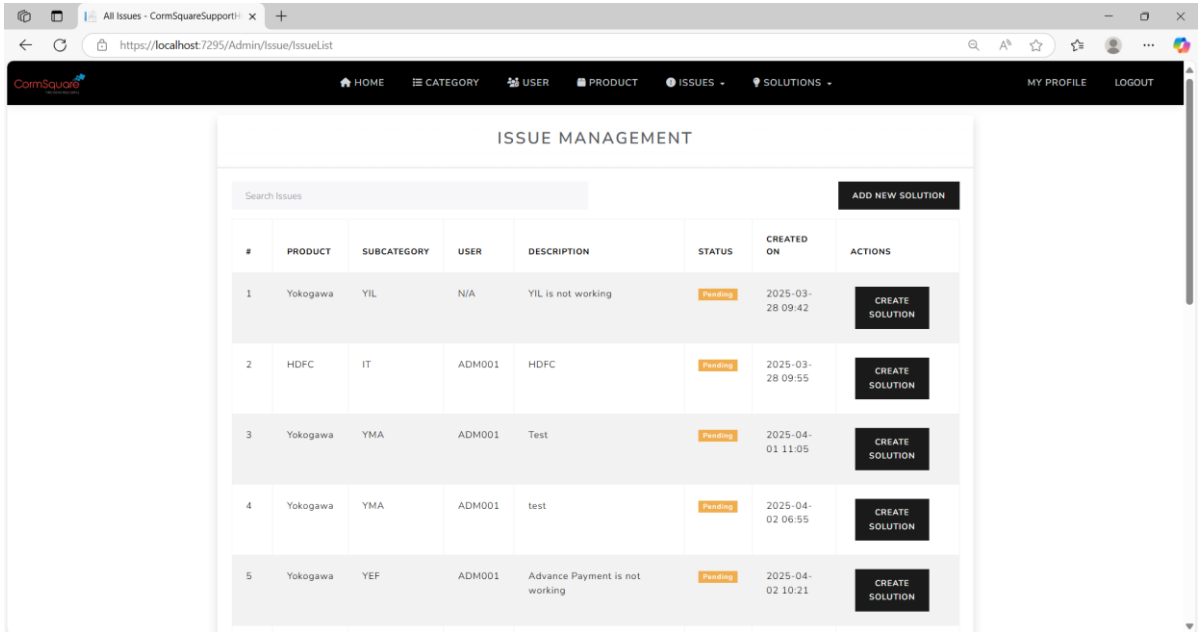

CormSquare
HOME
CATEGORY
USER
PRODUCT
ISSUES
SOLUTIONS
MY PROFILE
LOGOUT
Project Overview
Project Name: [Enter Project Name]
Project Manager: [Enter Name]
Start Date: [DD/MM/YYYY]
End Date: [DD/MM/YYYY]
2. Objectives
Summarize the project objectives.
Outline the intended outcomes.
Describe key performance indicators (KPIs).
3. Deliverables
Final Source Code Repository
Deployment Guide
User Manual
Test Reports
System Documentation
4. Access & Credentials
Resource
URL/Location
Username
Password/Key
Git Repository
github.com/org/repo
[username]
[access token]
Server (Production)
192.168.1.100
[root/admin]
[password]
Admin Panel
project.example.com/admin
[admin]
[password]
5. Outstanding Issues
[List any open bugs or issues]
[List pending tasks if any]
6. Contact Information
Technical Lead: [Name] – [Email]
Support Contact: [Name] – [Phone/Email]
7. Sign-off
By signing this document, both parties agree that the project has been delivered as per the defined scope.
Signature (Delivering Team):
Name: _____
Date: _____
Signature (Receiving Team):
Name: _____
Date: _____
def calculate_score(actual, optimal):
efficiency = (optimal / actual) * 100
score = min(10, max(1, (efficiency * 10) / 100))
return round(score, 2)
h3 › strong
157 words Build with tinyMCE

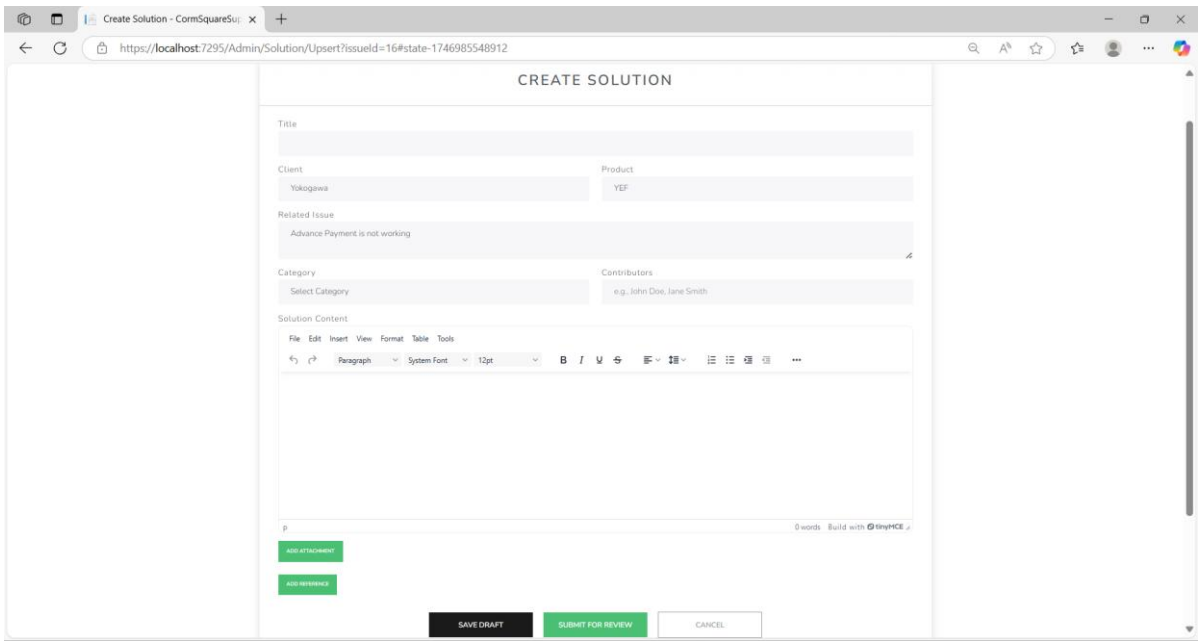
Issue Management:

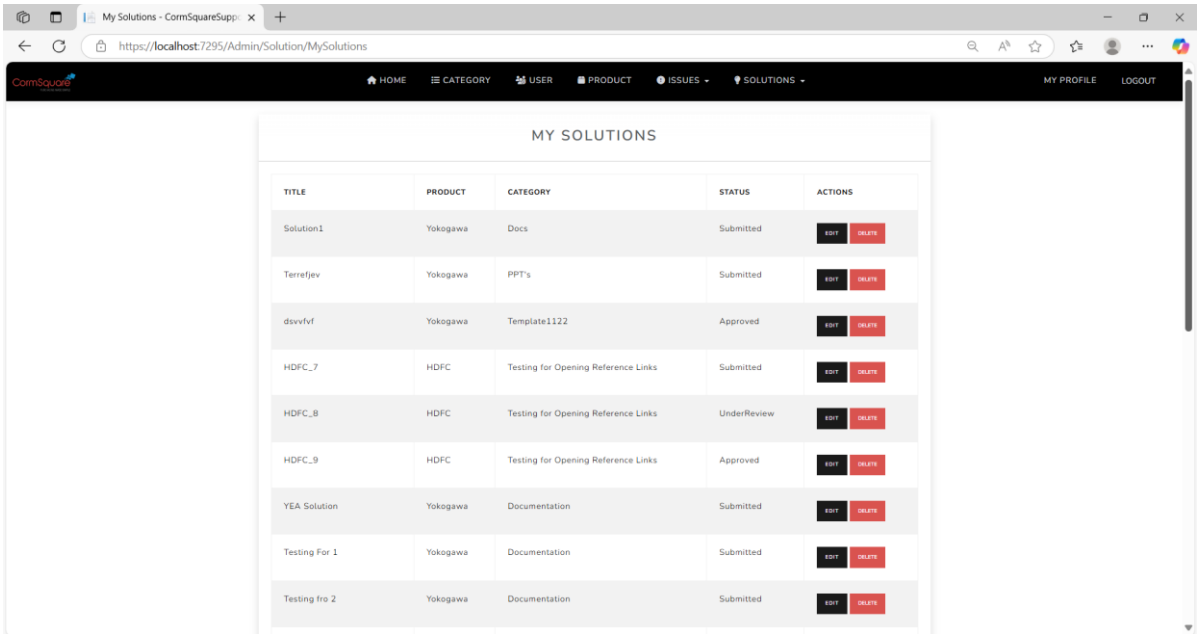




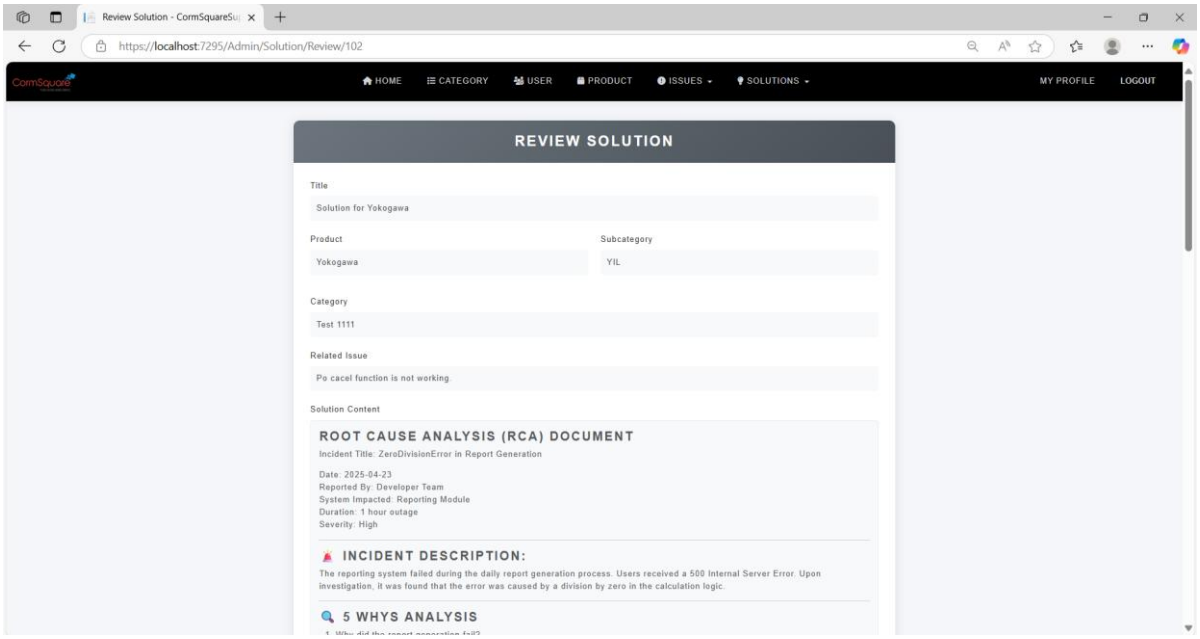
Creating a Solution:

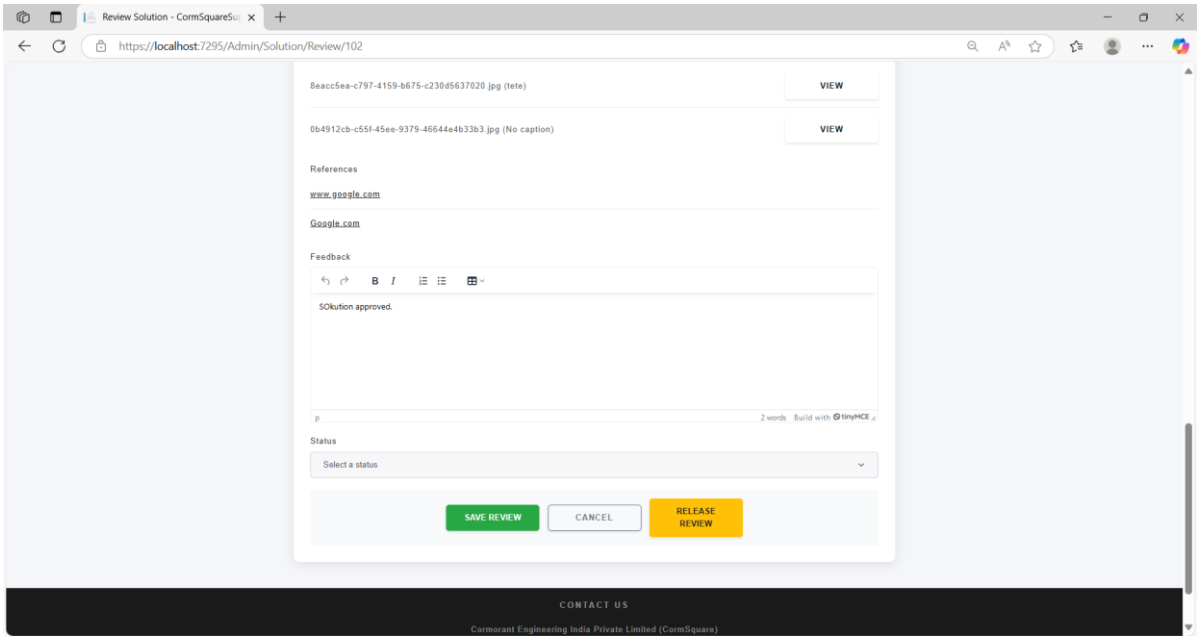


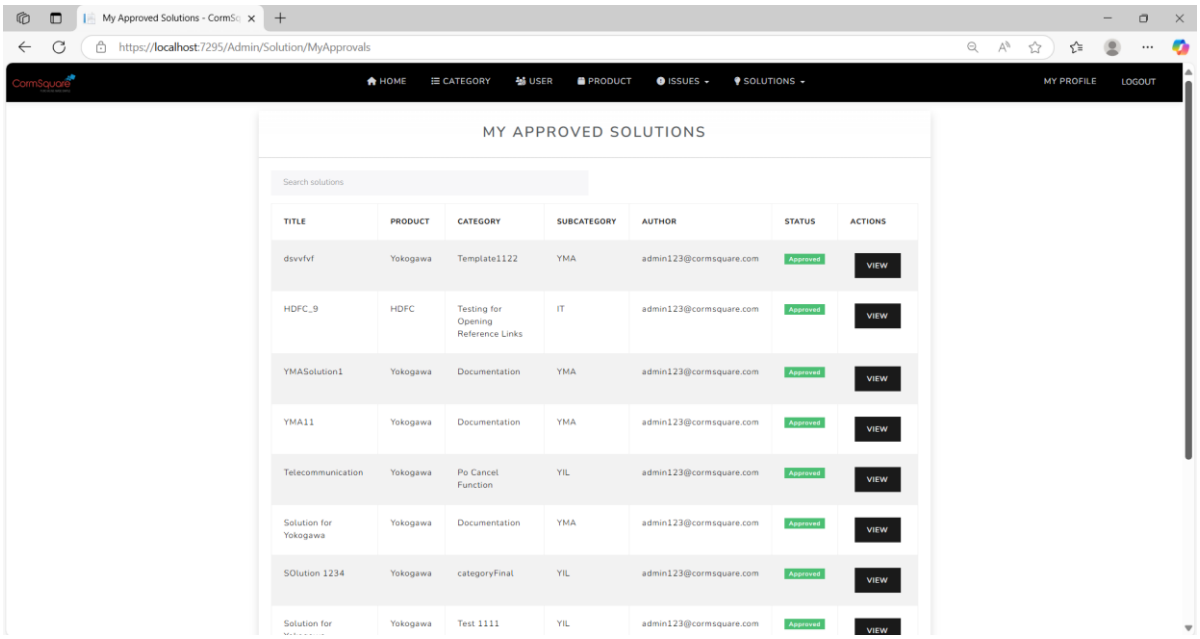




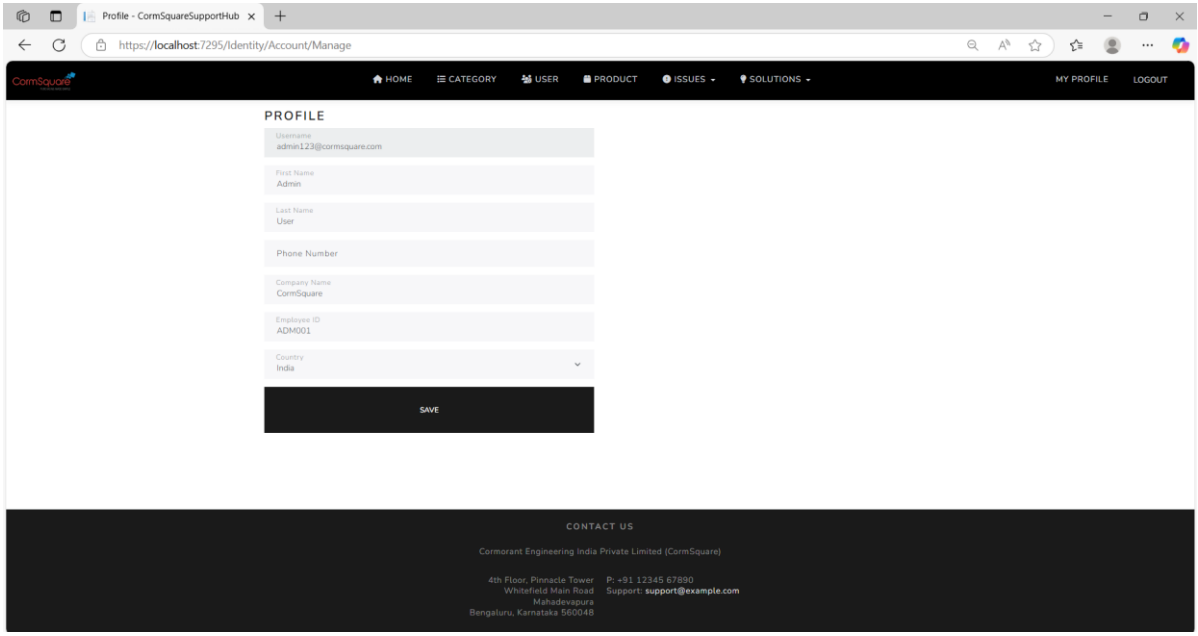
Reviewing a Solution:




Review Solution - CormSquareSu
https://localhost:7295/Admin/Solution/Review/102
8eacc5ea-c797-4159-b675-c230d5637020.jpg (tete)
VIEW
0b4912cb-c55f-45ee-9379-46644e4b33b3.jpg (No caption)
VIEW
References
www.google.com
Google.com
Feedback
SOkution approved.
2 words Build with tinyMCE
Status
Select a status
SAVE REVIEW
CANCEL
RELEASE REVIEW
CONTACT US
Cormorant Engineering India Private Limited (CormSquare)


My Approved Solutions - CormSq
https://localhost:7295/Admin/Solution/MyApprovals
CormSquare
HOME
CATEGORY
USER
PRODUCT
ISSUES
SOLUTIONS
MY PROFILE
LOGOUT
MY APPROVED SOLUTIONS
Search solutions
TITLE
PRODUCT
CATEGORY
SUBCATEGORY
AUTHOR
STATUS
ACTIONS
dsvvfvf
Yokogawa
Template1122
YMA
admin123@cormsquare.com
Approved
VIEW
HDFC_9
HDFC
Testing for Opening Reference Links
IT
admin123@cormsquare.com
Approved
VIEW
YMASolution1
Yokogawa
Documentation
YMA
admin123@cormsquare.com
Approved
VIEW
YMA11
Yokogawa
Documentation
YMA
admin123@cormsquare.com
Approved
VIEW
Telecommunication
Yokogawa
Po Cancel Function
YIL
admin123@cormsquare.com
Approved
VIEW
Solution for Yokogawa
Yokogawa
Documentation
YMA
admin123@cormsquare.com
Approved
VIEW
SOlution 1234
Yokogawa
categoryFinal
YIL
admin123@cormsquare.com
Approved
VIEW
Solution for
Yokogawa
Test 1111
YIL
admin123@cormsquare.com
Approved
VIEW

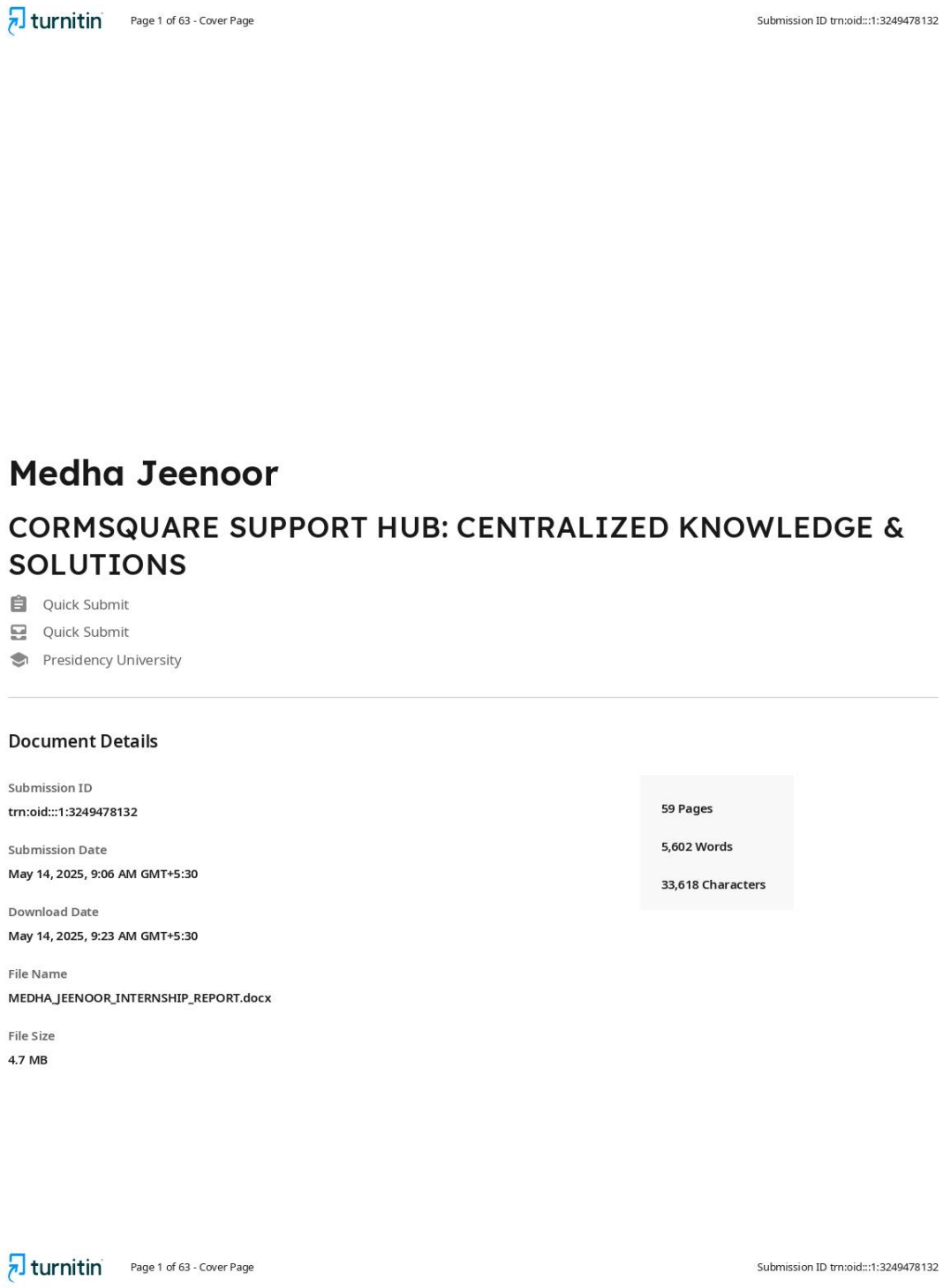
My Profile:


Profile - CormSquareSupportHub
https://localhost:7295/Identity/Account/Manage
CormSquare
HOME
CATEGORY
USER
PRODUCT
ISSUES
SOLUTIONS
MY PROFILE
LOGOUT
PROFILE
Username
admin123@cormsquare.com
First Name
Admin
Last Name
User
Phone Number
Company Name
CormSquare
Employee ID
ADM001
Country
India
SAVE
CONTACT US
Cormorant Engineering India Private Limited (CormSquare)
4th Floor, Pinnacle Tower
Whitefield Main Road
Mahadevapura
Bengaluru, Karnataka 560048
P: +91 12345 67890
Support: support@example.com

APPENDIX-C

ENCLOSURES

Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need for a page-wise explanation.



turnitin Page 1 of 63 - Cover Page Submission ID trn:oid:::1:3249478132

Medha Jeenoor

CORMSQUARE SUPPORT HUB: CENTRALIZED KNOWLEDGE & SOLUTIONS

Quick Submit

Quick Submit

Presidency University

Document Details

Submission ID
trn:oid:::1:3249478132

Submission Date
May 14, 2025, 9:06 AM GMT+5:30

Download Date
May 14, 2025, 9:23 AM GMT+5:30

File Name
MEDHA_JEENOOR_INTERNSHIP_REPORT.docx

File Size
4.7 MB

59 Pages

5,602 Words

33,618 Characters

turnitin Page 1 of 63 - Cover Page Submission ID trn:oid:::1:3249478132

14% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Cited Text

Match Groups

25 Not Cited or Quoted 14%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

9% Internet sources
7% Publications
13% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

turnitin Page 2 of 63 - Integrity Overview Submission ID trn:oid:::1:3249478132

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

0 AI-generated only 0%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

turnitin Page 2 of 61 - AI Writing Overview Submission ID trn:oid:::1:3249478132

SUSTAINABLE DEVELOPMENT GOALS

Sustainable Development Goal 9: Industry, Innovation, and Infrastructure

The CormSquare Knowledge Management System (KMS) is my way of pitching in on Sustainable Development Goal 9 (SDG 9)—you know, the one about building solid infrastructure, pushing sustainable industry, and sparking new ideas. This project is all about using digital tools to help teams share knowledge, tackle problems together, and keep things running smoothly.

Think of the KMS as a super-organized digital bookshelf for a company's know-how. It stores solutions, documents, and expertise in a way that's easy to find and reuse. With templates, file attachments, and searchable tags, it saves time and keeps work flowing without the usual mess.

I used some neat tech to make it happen: TinyMCE for writing rich content, SQL Server's

full-text search to quickly find stuff, and Chart.js to show data in a visual, easy-to-grasp way. These keep the system fresh and ready for whatever a company throws at it. I also tossed in a scoring and trending feature to spotlight the most useful info, so the good stuff stays front and center.

The system's setup—with roles like KM Creators, KM Champions, and Admins—lets everyone contribute ideas while keeping things orderly. It tracks files and guides content through clear workflows, creating a space where innovation can bloom without turning into chaos.

Down the road, I've got plans to make it even better with interactive dashboards, sharper analytics, and ways to pull in outside issues. This way, the system can keep growing and stay useful for years to come.
In a nutshell, this KMS isn't just a tool for right now—it's my shot at helping companies work smarter, come up with new ideas, and stay strong with a rock-solid digital backbone.