SUBMITTED BY:

ABHINAV R NAIR

ANUJA praveen

pavithra vijeshkumar

MINI PROJECT

oNLINE NOTES sHARING SYSTEM - ATHENA

1. **SYNOPSIS**

The primary objective of the Notes Sharing System is to create a seamless, centralized digital platform that enhances communication and collaboration among students and educators. In an increasingly digital academic environment, the system is designed to simplify the process of exchanging study materials, distributing assignments, and fostering healthy academic interaction within classrooms.

From the perspective of administrators, the goal is to provide an efficient tool for managing user data, monitoring platform usage, and maintaining oversight of all uploaded content. The admin panel will serve as the control centre, ensuring that the system operates smoothly and that users adhere to the intended usage guidelines.

For users, the platform offers an interactive and supportive learning space. Users will be able to access academic content shared by the web page, collaborate by sharing their own notes, and stay engaged through a group chat feature tailored for Classroom-wide discussions. The system is also intended to encourage peer-to-peer support, helping users stay updated and involved even outside traditional classroom settings.

1. **INTRODUCTION**

The **Online Note Sharing System** is a web-based platform developed to simplify and enhance the way students and learners exchange academic notes and resources. Designed for a wide range of users—from school students to competitive exam aspirants like JEE and NEET candidates—the system provides a centralized and efficient digital environment for sharing study materials.

Unlike traditional methods that rely on physical distribution or scattered messaging platforms, this system offers a streamlined, paperless solution for uploading, accessing, and organizing notes. Users can securely register, log in, and actively participate in sharing educational content across various study groups.

One of the platform’s standout features is the **reputation system**, which assigns users a score based on the number of quality notes they contribute. This encourages active participation and helps highlight top contributors within the community. The more helpful content a user shares, the higher their reputation grows—fostering a sense of recognition and trust among peers.

Additionally, the system includes a **personal To-Do feature** that allows users to set and manage academic tasks. This helps users stay organized, track their progress, and plan their studies more effectively within the same platform.

With a clean, user-friendly interface and real-time collaboration tools, the Online Note Sharing System empowers learners to support each other, stay organized, and access valuable resources anytime, from anywhere. It is a modern solution tailored for today’s digital learners who seek both convenience and collaboration in their academic journey.

1. **SYSTEM ANALYSIS**

**3.1. EXISTING SYSTEM**

Currently, platforms like **WhatsApp** and **Google Classroom** are commonly used for sharing academic notes. However, these tools are not specifically designed for structured note management, leading to several challenges. In WhatsApp, notes often get buried under unrelated conversations, making it difficult to retrieve older materials. The lack of proper organization and search features means students waste time scrolling through chat histories.

On the other hand, Google Classroom, while more formal, requires users to install additional apps or have specific file viewers (like Google Docs or Google Drive) to access and edit notes. This can be inconvenient, especially for users on limited devices or slower networks. Moreover, both platforms lack features like contributor recognition, task tracking, and centralized repositories—making collaboration and long-term resource management inefficient.

**Drawbacks of existing system:**

* Poor organization
* No version control
* Security issues
* Distractions
* Overload stream
* File format issues

**3.2 PROPOSED SYSTEM**

The proposed system is a next-generation **Online Note Sharing Platform** designed to overcome the limitations of existing tools and offer a more seamless, organized, and user-focused experience. It not only addresses issues such as content retrieval, lack of structure, and dependency on third-party apps, but also introduces enhanced features like user reputation scoring, group-based note organization, and personal task tracking. With a focus on simplicity, accessibility, and collaboration, the system aims to provide a smarter and more efficient solution for academic content sharing and management.

**Advantages**:

* Easy to use web-based system.
* Users can register online.
* Notes are accessible from anywhere.
* Centralized control by administrator.
* Managed notes in a secure manner.

**MODULE DISCRIPTION**

A module is a small, self-contained part which is the building block in software development

that does a specific job. They help in organizing complex programs by breaking them down

into smaller parts.

**Administration module**

The admin can approve creation of study groups and respond to flagged or reported items. They will be able to remove or suspend user accounts. Sends updates or announcements to study groups.

**User management module**

User can create study group and add other users. He can chat in the group and upload or download notes. User can request to join a group and manage downloaded notes. They can also create a "to do" list in the study group

**3.3 FEASIBILITY**

A feasibility study is undertaken to determine the possibility or probability of either improving the existing system or developing a completely new system.it helps to obtain an overview of the problem and to get rough assessment of whether feasible solution exists. This is essential to avoid committing large resources to a project.

**3.3.1 Technical Feasibility:**

**1.Infrastructure:** Cloud-based infrastructure can support large user bases and note storage.

**2.Security:** Implementing robust security measures (e.g., encryption, access controls) can protect user data.

**3.Scalability:** Scalable architecture can accommodate growing user bases and note volumes.

**3.3.2 Economic Feasibility:**

**1.Cost-effectiveness:** Online note sharing can reduce costs associated with physical notetaking and storage.

**2.Revenue streams:** Potential revenue streams include subscription models, advertising, and premium features.

**3.Market demand:** Growing demand for online collaboration and knowledge sharing tools.

**3.3.3 Social Feasibility:**

**1.User adoption:** User-friendly interfaces and intuitive design can facilitate adoption.

**2.Collaboration:** Online note sharing can foster collaboration and knowledge exchange among users.

**3.Community building:** Online platforms can create communities around shared interests and topics.

REQUIREMENT ANALYSIS

Requirement analysis task is a process of discovery,refinement,modelling and specification both the developers and customer take an activity role in requirement analysis can be divided into:

4.1Problem recognition

4.2Problem evaluation & synthesis

4.3Modeling

**4.1 Problem Recognition**

The primary problem is that existing platforms used for academic collaboration, such as WhatsApp and Google Classroom, are not specialized for structured note-sharing and management, leading to significant inefficiencies.

* **Poor Organization**
* **Dependency on External Apps**
* **Lack of Key Features**

**4.2 Problem Evaluation & Synthesis**

The proposed "Athena" system is designed to directly address the limitations of existing platforms by creating a centralized, feature-rich web application tailored for academic collaboration.

The system aims to provide a superior user experience through several key advantages:

* **Centralized and Accessible:** It offers a single, secure web-based platform where all notes are managed, making them accessible from anywhere without the need for third-party apps.
* **Structured and Organized:** By design, it overcomes the issue of content retrieval and provides a more structured environment for academic sharing.
* **User-Focused Features:** The platform introduces unique features to enhance engagement and organization:

**4.3 Modeling**

The system's architecture and data flow are defined through a detailed database schema and a set of Data Flow Diagrams (DFDs).

**Data Modeling (Database Tables):**

The system's data is organized across several tables to ensure integrity and logical structure. Key tables include:

* **Admin and Users:** These tables store login credentials and profile information for administrators and regular users, respectively.
* The Users table also tracks status (active, banned) and reputation\_score.
* **Study Group and Group Members:** Manages the groups themselves—including their name, creator, and approval status—and tracks which users belong to which group and their roles (member, moderator).
* **Group Messages, Notes, and Downloaded Notes:** These tables handle the core content.

Group Messages stores the chat history, Notes catalogs all uploaded files with their titles and paths, and Downloaded Notes keeps a record of which users have downloaded which notes.

* **Reports, Announcements, and To-Do:** These tables support key system features by managing user-submitted reports, admin-pushed announcements, and personal tasks for users.

DATA FLOW DIAGRAM(DFD)

A DFD also known as the bubble chart has the purpose of clarifying system requirements and identifying major information that will become programs in system design.A DFD is a pictorial representations of network that describes the flow of data through a system.The symbols used in Data Flow Diagram are:

* It represents a data source or destination
* It represents flow of data
* It represents a process that transforms data
* A black rectangle with a white background

  AI-generated content may be incorrect. It represents data storage(eg:table)

5**.**System Specification

5.1 HARDWARE REQUIREMENTS

|  |  |
| --- | --- |
| Component | Requirement |
| CPU Type | Pentium dual-core or above |
| RAM | 2 GB or above |
| Hard Disk Drive | 1. B or greater |

5.2SOFTWARE REQUIREMENTS

|  |  |
| --- | --- |
| Component | Requirement |
| Operating System | WINDOWS 7 or higher / Linux |
| Front-End | HTML, CSS, JavaScript |
| Back-End | PHP, MySQL Server |
| Web Server | Apache or similar |

6. SYSTEM DESIGN

System design is a process of developing specification for candidate system that meet the criteria established in the system analysis.Major step in design are the preparation of the input forms output reports in a form application to the user.

The main objective of the system design is to use the package easily by any computer operation.System design is the creative act of invention,developing new inputs,a database,offline files,method procedure and output for processing business to meet an organization objective.System design builds information gathered during the system analysis.

6.1 DATA DESIGN

Data design creates a model of data or information that is represented at a higher level of abstraction.The structure of data has always been an important part of software design.The software design activities translate this requirements model into data structure at software component level.Data design required to manage the large volume of information.In this system,normalization process,the redundant field will be eliminated finally produce the efficient table.

6.2 ARCHITECTURAL DESIGN

Architectural design is a comprehensive framework that describes its form and how they fit together.The properties of component interact with other components.Architectural design focuses on the representation of structure of the software.

6.3 PROCEURAL DESIGN

Procedural design or component level design occurs after data,architectural and interface design must be translate into operational software.The procedural design for each component,represented in graphical,tabular or text based notation,is primary work product produced during component level design.

6.4 INTERFACE DESIGN

Interface design creates and effective communication medium between a human and computer,Design identifies objects and action then creates a screen layout that forms the basis for user interface.

Interface design focus on:

1.The design of interfaced between software components and non-human procedures and consumer of information.

2.The design of interface between software components.

3.The design of interface between a human and computer.

7.CODING

The coding phase involves the translation of the system design into a machine-readable format. For the Athena project, this involved writing server-side scripts in PHP to handle logic and database operations, structuring the web pages with HTML, styling them with CSS for a responsive and visually appealing layout, and using JavaScript for client-side interactivity and dynamic updates.

The code is organized into a modular structure, separating concerns such as database connection (connect.php), user authentication (user\_login.php), and feature-specific logic (e.g., files within the dashboard/chat\_box/ directory). This approach ensures that the code is maintainable, scalable, and easier to debug. Emphasis was placed on writing secure code, particularly in handling user input and database queries to prevent vulnerabilities like SQL injection.

SYSTEM TESTING

Testing is the process of executing a program with the intent of finding errors. System testing is the stage of implementation aimed at ensuring that the system works accurately and efficiently before the live operation commences. A series of tests were performed for the proposed system.

8.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design—the module. For Athena, individual PHP scripts and JavaScript functions were tested in isolation. For example, the user\_register.php script was tested to ensure it correctly validates input, checks for duplicate users, and inserts new records into the database. On the front-end, JavaScript functions for features like group search were tested to ensure they correctly filter and display results.

8.2 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. In this project, modules were integrated incrementally. For instance, after testing the user login and group creation modules separately, they were tested together to ensure that a logged-in user could successfully create a group and that the group was correctly associated with their user ID in the database.

8.3 Validation Testing

Validation testing is done to ensure the complete assembly of an error-free software product. Validation is termed successful if the software functions in a manner that is reasonably expected by the customer. The fully integrated Athena platform was tested against the user requirements to ensure all features worked as specified, from user registration to real-time chat and administrative functions.

9.SYSTEM IMPLEMENTATION

A crucial phase in the system's life cycle is the successful implementation of the new system design. This involves deploying the application to a web server, setting up the database, and making the platform accessible to users. The implementation of Athena requires a server environment with PHP and a MySQL database installed.

The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications.

The implementation process includes:

* Setting up the athena database using the provided SQL script.
* Configuring the database connection details in connect.php.
* Deploying the file structure to a web server.
* Ensuring the server has the correct permissions for file uploads (e.g., for profile pictures and notes).

10. SOFTWARE MAINTENANCE

Software maintenance is an integral part of the software life cycle. Maintenance covers a wide range of activities, including correcting coding and design errors and updating user support. Maintenance can be classified as corrective, adaptive, and perfective.

* **Corrective Maintenance**: Repairing processing or performance failures or making changes because of previously uncorrected problems or false assumptions. For Athena, this could involve fixing a bug where chat messages do not load correctly.
* **Adaptive Maintenance**: Changing the program function to adapt to changes in the environment, such as upgrading to a new version of PHP or MySQL.
* **Perfective Maintenance**: Enhancing the software by adding new features based on user feedback, such as implementing a search function for chat messages or adding a calendar feature.

Any system developed should be secured and protected against possible hazards. Security measures are provided to prevent unauthorized access to the database at various levels. The system uses session management to ensure that only authenticated users can access protected areas.

11. CONCLUSION

This software planning is to provide a framework that enables students and administrators to manage academic collaboration effectively. It eliminates and reduces the hardships faced by the existing fragmented system. The Athena platform allows users to manage study groups, share notes, communicate in real-time, and organize tasks, all within a single, integrated environment.

The implementation of the system will reduce the time spent on administrative tasks and provide readily accessible information.

The goals achieved by this project are:

* Efficient management of collaborative records.
* Less processing time for finding information and resources.
* A user-friendly and intuitive interface.
* A platform that is portable and flexible for further enhancement.

**APPENDIX A** (Tables)

1. **Admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatypes** | **Constraints** | **Description** |
| admin\_id | INT | Primary key,  Auto increment | Unique admin  ID |
| user\_name | VARCHAR(100) | Not null | Username of  admin |
| password | VARCHAR(255) | Not null | Admin’s  password |
| email | VARCHAR(100) | Unique,  Not null | Admin’s  e-mail. |

**2. User**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **datatype** | **Constraints** | **description** |
| user\_id | INT | Primary key,  Auto increment | Unique user ID |
| user\_name | VARCHAR (100) | Unique,not null | User’s username |
| name | VARCHAR (100) | Not null | User’s name |
| email | VARCHAR (100) | Unique,not null | User’s email |
| phone\_no | BIGINT | Unique,not null | User’s phone no |
| password | VARCHAR (255) | Not null | Password of user |
| status | ENUM  (active,banned) | Default ‘active’ | User’s status changed from active If admin  decides |
| reputation\_score | INT | Default 0 | Contribution score |
| Profile\_pic | VARCHAR(255) | null | Profile Pic |
| created\_at | DATETIME | Default current\_time  stamp | Account creation date and time |

1. **Study group**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatype** | **Constraints** | **Description** |
| group\_id | INT | Primary key,  Auto increment | Unique group  ID |
| group\_name | VARCHAR(100) | Not null | Study groups  name |
| user\_id | INT | Not null,  Foreign key→  users | Creator of the groups |
| approved | BOOLEAN | Default false | Approval of  admin |
| created\_at | DATETIME | Default current  \_timestamp | Group creation timestamp |

1. **Group memebers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatype** | **Constraints** | **Description** |
| members\_id | INT | Primary key,  Auto increment | Unique ID of  group |
| group\_id | INT | Not null,foreign  key→study group | Associated group in  question |
| user\_id | INT | Not null,foreign  key→user | Member of the  group |
| role | ENUM  (members,mode-  Rator) | Default ‘member’ | Member or  moderator |
| status | ENUM(pending,  Joined) | Default  ‘pending’ | ‘pending’,  ‘joined’ |

1. **Group messages**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatype** | **Constarints** | **Description** |
| message\_id | INT | Primary\_key,  Auto increment | Message\_ID |
| group\_id | INT | Not null,foreign  key→study groups | Liked group |
| user\_id | INT | Not null,foreign  key→users | Sender of  message |
| content | TEXT | Not null | Message content |
| time\_stamp | DATETIME | Default current  \_timestamp | When it user  sent |

1. **Notes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatype** | **Constraints** | **Description** |
| note\_id | INT | Primary key,  Auto increment | Note ID |
| group\_id | INT | Not null,foreign  key→study group | Related group |
| user\_id | INT | Not null,foreign  key→users | Who uploaded |
| title | VARCHAR  (100) | Not null | Title of the  note |
| file\_path | VARCHAR  (255) | Not null | File location |
| upload\_time | DATETIME | Default current  \_timestamp | Time of upload |

1. **Reports**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatype** | **Constraints** | **Description** |
| report\_id | INT | Primary key,  Auto increment | Report ID |
| user\_by | INT | Not null,foreign  key→users | Who reported |
| target\_type | ENUM  (user,message,  note) | Not null | User,message or note |
| target\_id | INT | Not null | ID of reported item |
| reason | TEXT | Not null | Reason for  flopping |
| status | ENUM  (open,resolved,  review | Default ‘open’ | Open or  Resolved or  Under review |
| created\_at | DATETIME | Default current  \_timestamp | Report time |

1. **Announcements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatype** | **Constraints** | **Description** |
| annoucement\_id | INT | Primary key,  Auto increment | Announcement  \_ID |
| admin\_id | INT | Not null,foreign  key→admin | sender |
| title | VARCHAR  (150) | Not null | Title of  message |
| content | TEXT | Not null | Message content |
| group\_id | INT | Foreign key→  Study group | Group target  (null=all) |
| created\_at | DATETIME | Default current  \_timestamp | When it was  set |

1. **To Do**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatype** | **Constraints** | **Description** |
| to\_do\_id | INT | Primary key,  Auto increment | To do ID |
| group\_id | INT | Not null,foreign  key→study group | Associated group |
| user\_id | INT | Not null,foreign  key→users | Creator of the task |
| task | VARCHAR  (200) | Not null | Task description |
| assigned\_to | INT | Foreign key→  Users nulloble | Optional asssigned user |
| status | ENUM | Default ‘open’ | Open,in progress,done |

**10.Downloaded notes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Datatype** | **Constraints** | **Description** |
| download\_id | INT | Primary key,  Auto increment | Unique download record |
| note\_id | INT | Not null,foreign key→notes | The note that  the user  downloaded |
| user\_id | INT | Not null,foreign key→users | The user who  Downloaded it |
| downloaded\_at | DATETIME | Default current  \_timestamp | When the note  Was download |

**APPENDIX B** (DFD)

Context level DFD

A diagram of a diagram

AI-generated content may be incorrect.

Level 1 DFD for Admin

A diagram of a login

AI-generated content may be incorrect.

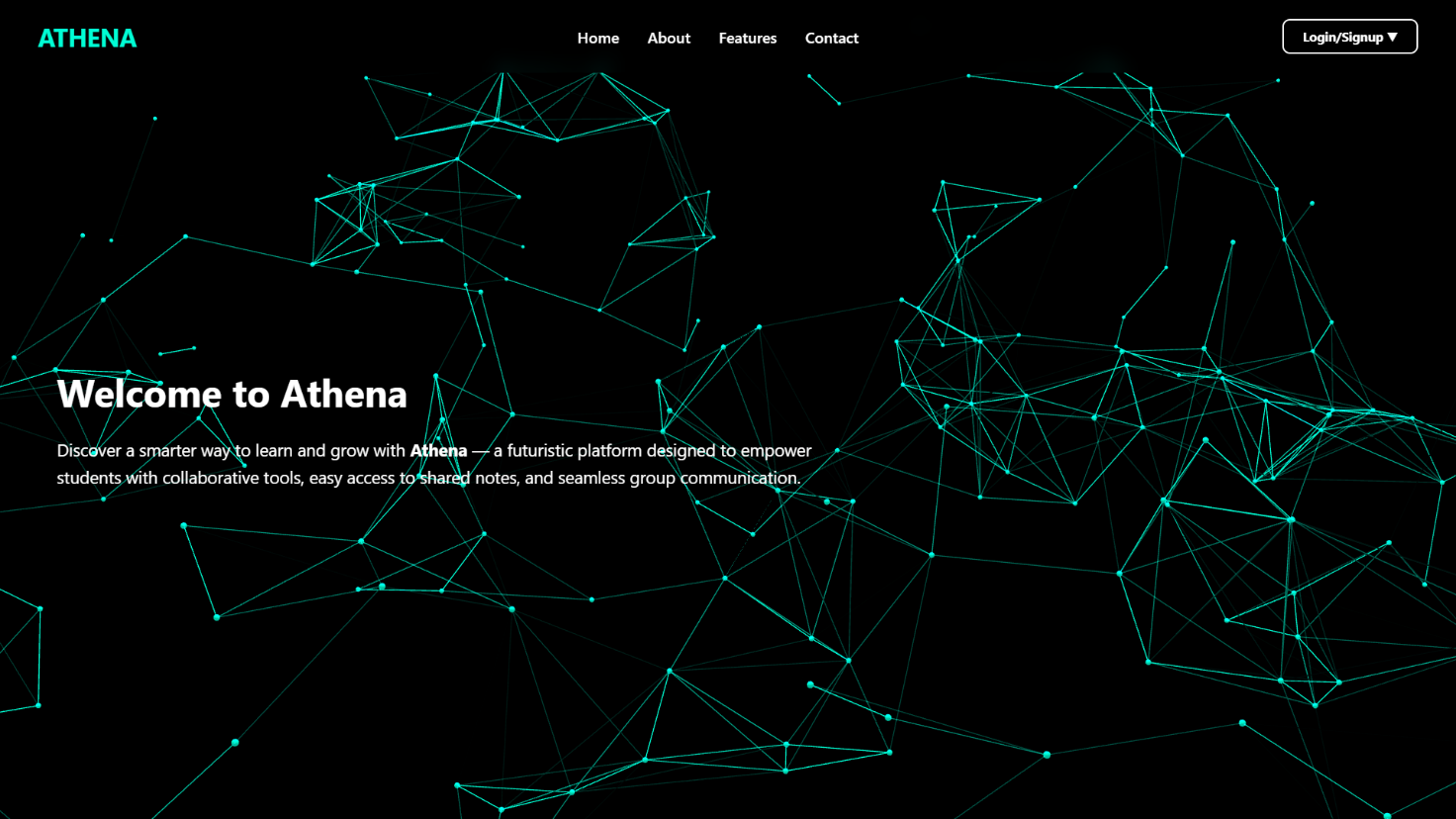
Level 1 DFD for User

A diagram of a group

AI-generated content may be incorrect.

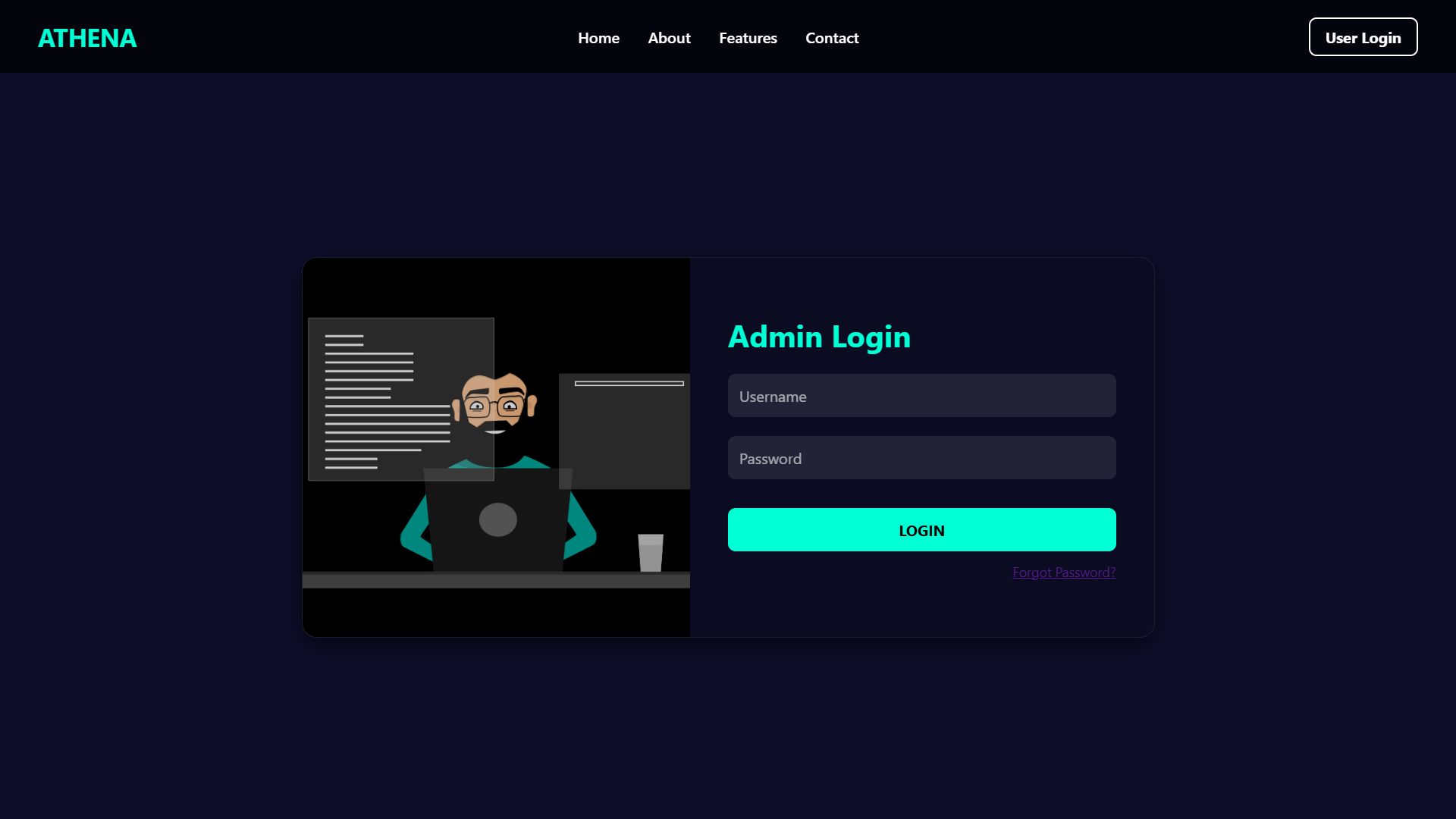
**APPENDIX C (INPUT AND OUTPUT FORMS)**

**Index Page**

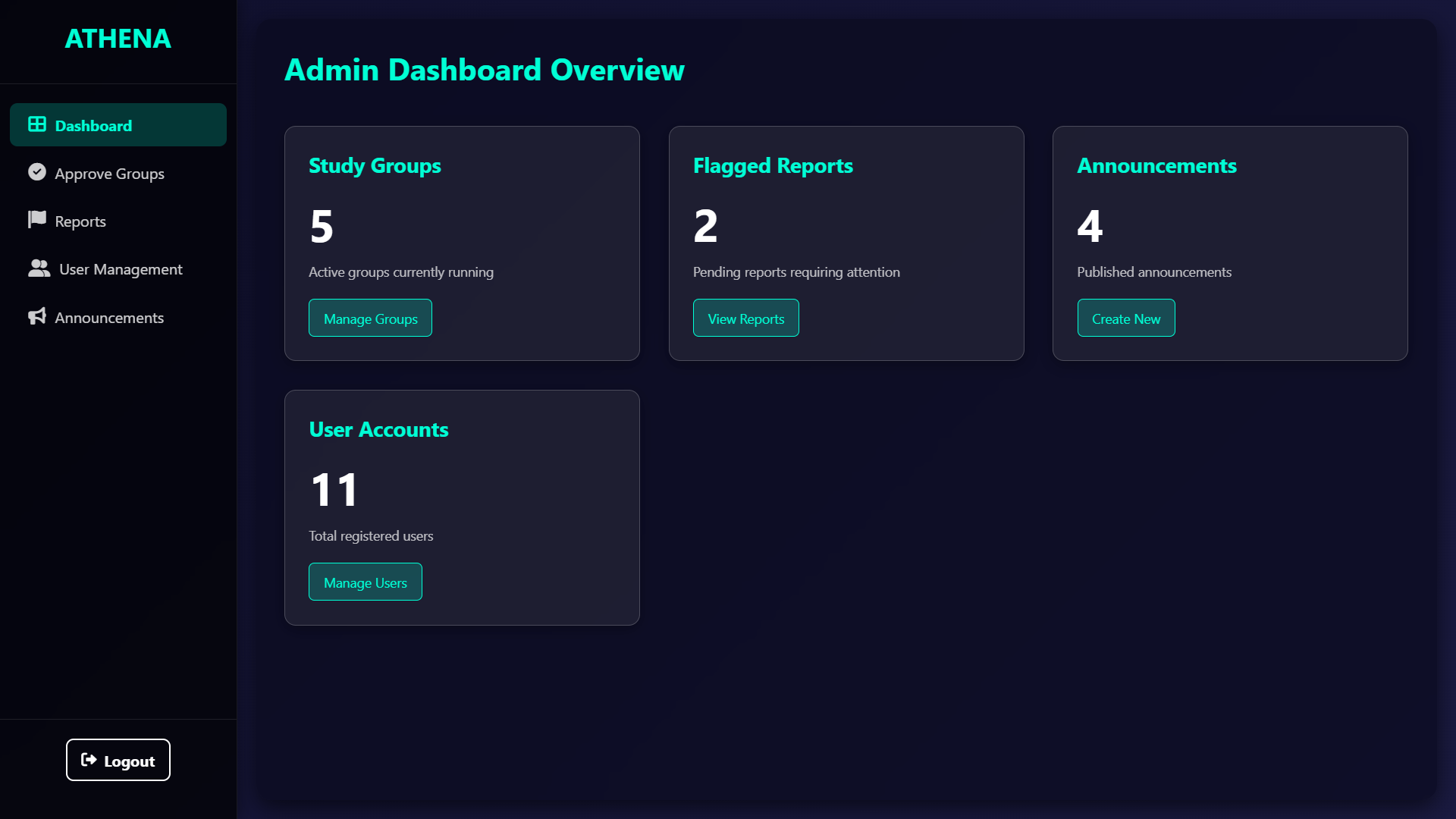


ADMIN

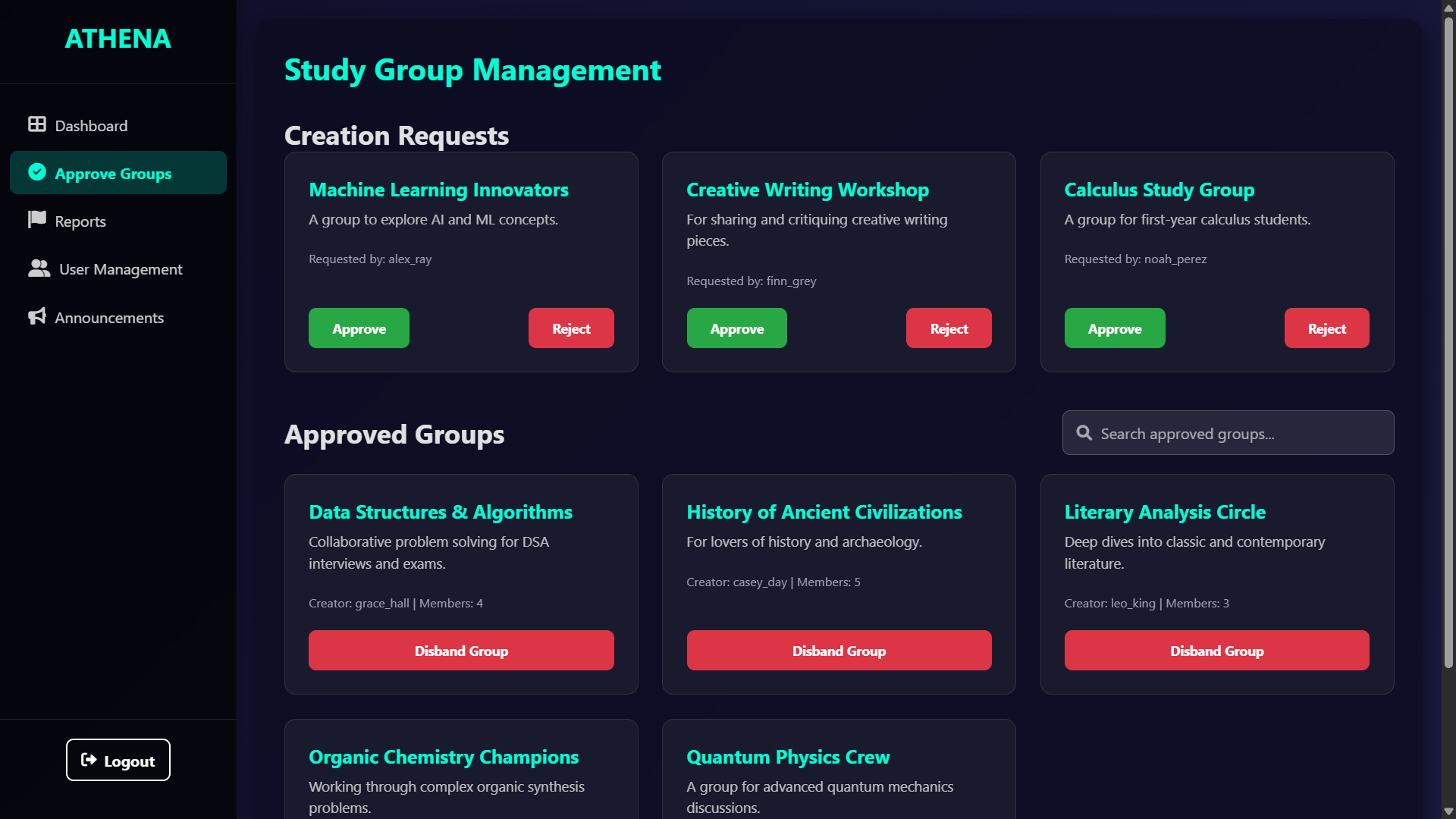
Admin Login



Dashboard



Approve groups



Reports

Screens screenshot of a computer screen

AI-generated content may be incorrect.

User Management

A screenshot of a computer

AI-generated content may be incorrect.

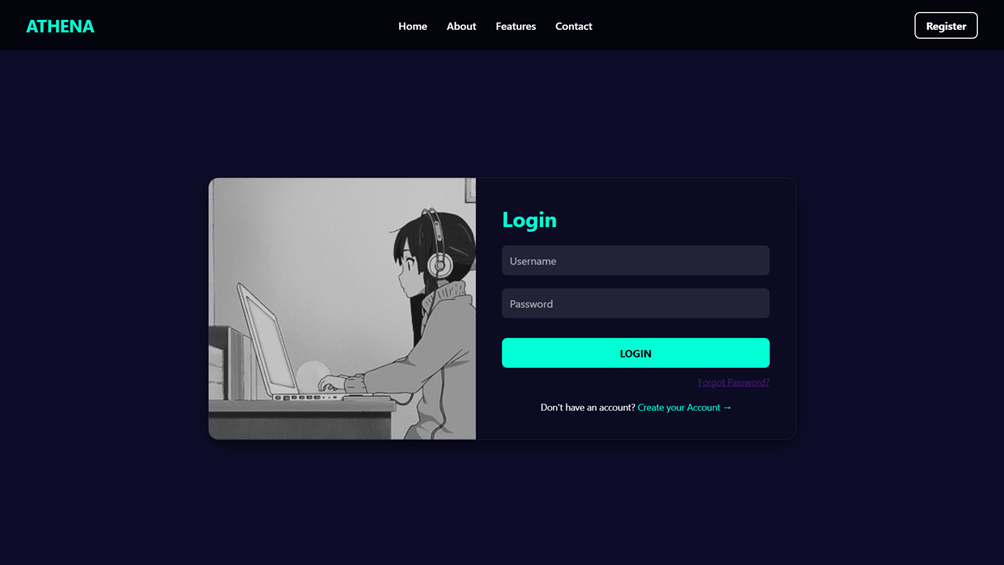
Announcements

A screenshot of a computer

AI-generated content may be incorrect.

USER

User Login

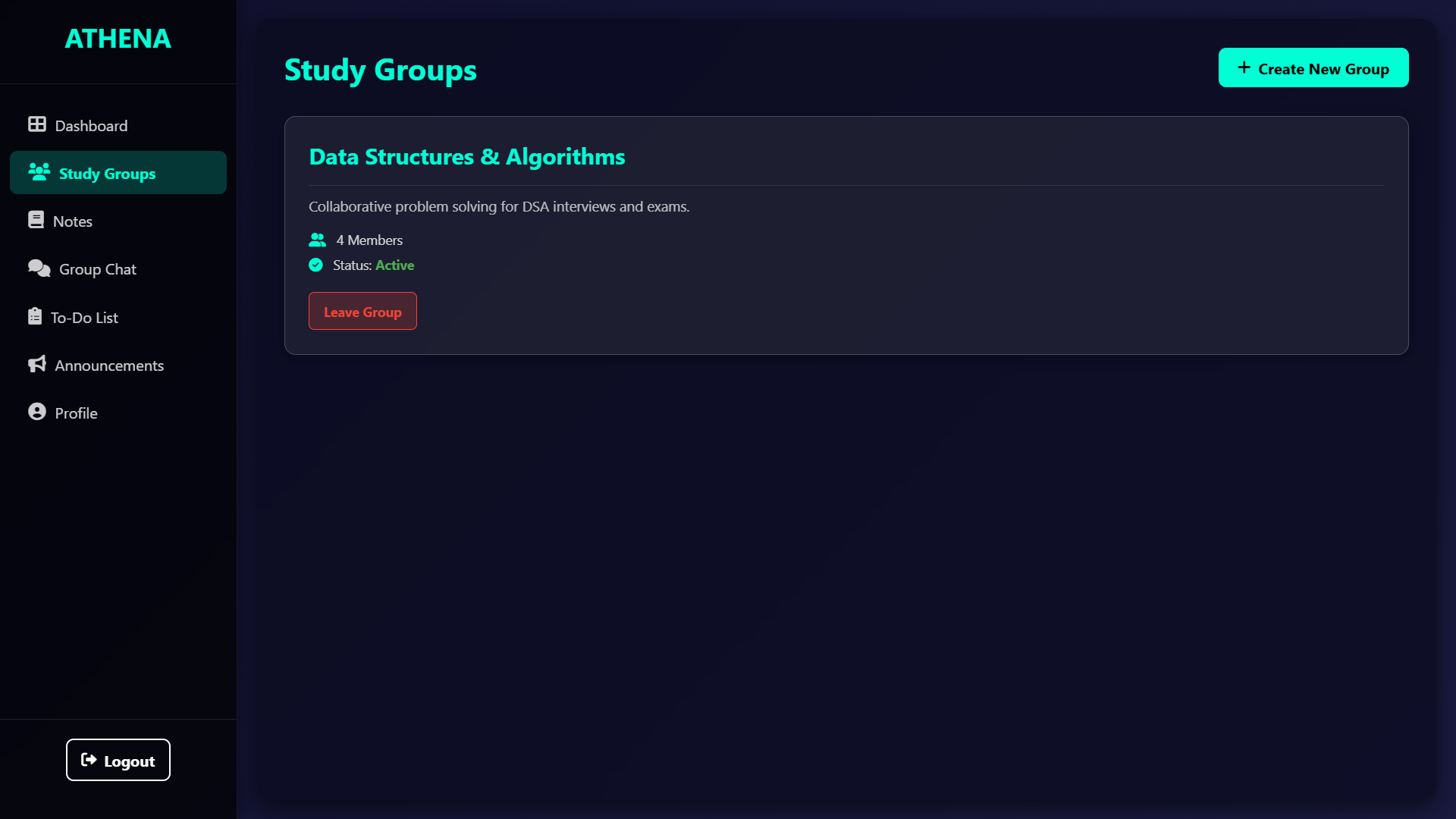


Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

Study groups



Notes

A screenshot of a computer

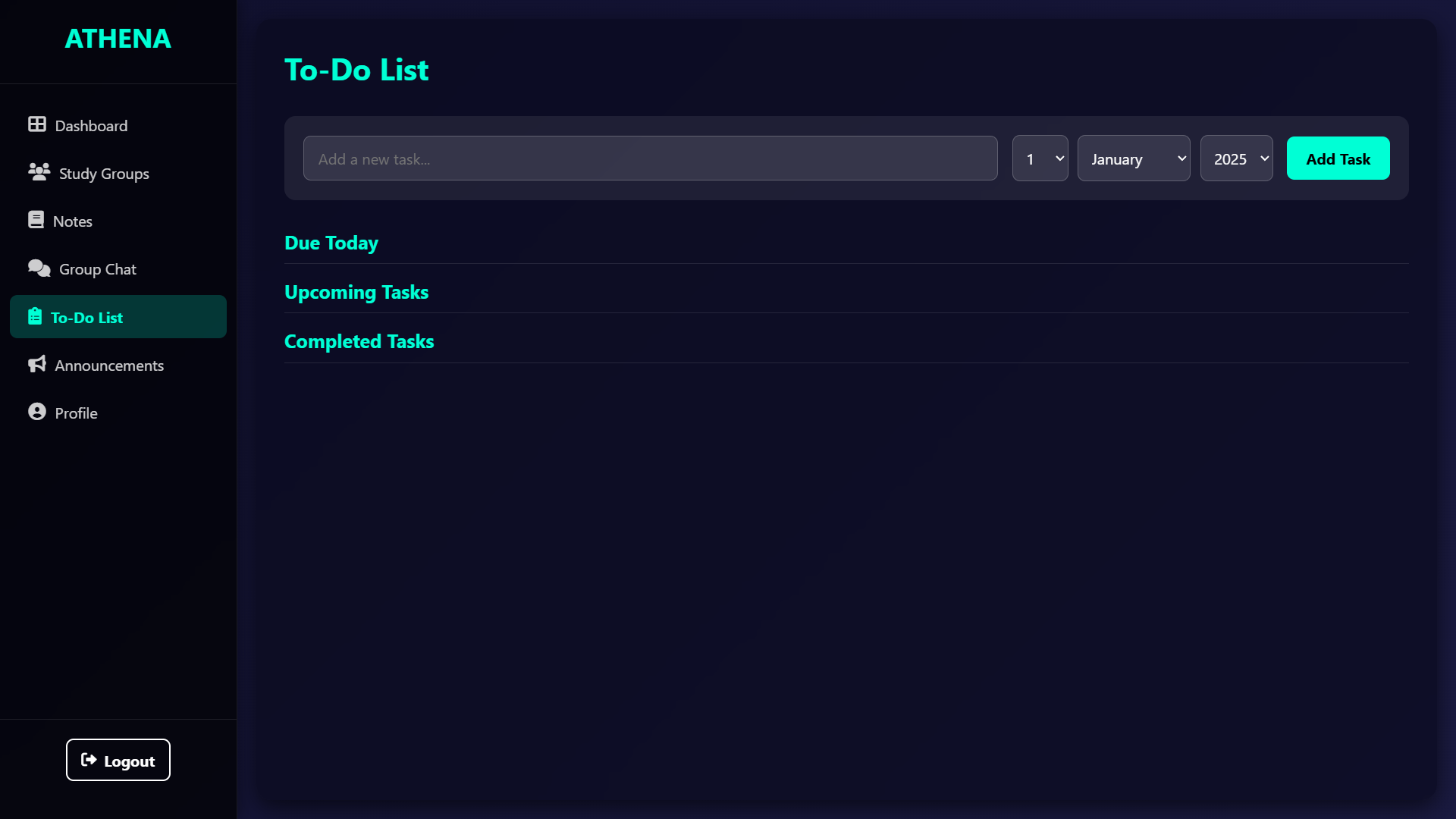
AI-generated content may be incorrect.

Group Chat

A screenshot of a computer

AI-generated content may be incorrect.

To-Do List



Announcements

A screenshot of a computer

AI-generated content may be incorrect.

Profile

A screenshot of a computer

AI-generated content may be incorrect.

**APPENDIX D (CODING)**

Admin

1.*login\_page.php*

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

  <title>Admin Login - Athena</title>

  <link rel="stylesheet" href="adminstyle.css" />

</head>

<body>

  <!-- Header section, similar to user login page -->

  <header class="navbar">

    <div class="logo"><a href="../index.php">ATHENA</a></div>

    <nav>

      <ul class="nav-links">

        <li><a href="../index.php">Home</a></li>

        <li><a href="../index.php#about">About</a></li>

        <li><a href="../index.php#services">Features</a></li>

        <li><a href="../index.php#contact">Contact</a></li>

      </ul>

    </nav>

    <!-- Admin login doesn't need a register button, so it's omitted -->

    <div class="nav-btn">

      <a href="../users/login\_page.php" class="login-btn">User Login</a>

    </div>

  </header>

  <!-- Login container to center the content -->

  <div class="login-container">

    <!-- Login box with glass effect -->

    <div class="login-box glass">

      <!-- Left side for illustration/image -->

      <div class="login-left">

        <!-- Using bgimage1.jpeg as it's a generic image and available in the jpeg\_files directory -->

        <img src="../jpeg\_files/bgimage3.gif" alt="Admin Illustration"/>

      </div>

      <!-- Right side for the login form -->

      <div class="login-right">

        <h2>Admin Login</h2>

        <!-- Form action points to admin\_login.php for processing -->

        <form action="admin\_login.php" method="POST">

          <!-- Input group for username -->

          <div class="input-group">

            <input type="text" name="username" placeholder="Username" required />

          </div>

          <!-- Input group for password -->

          <div class="input-group">

            <input type="password" name="password" placeholder="Password" required />

          </div>

          <!-- Submit button -->

          <button type="submit" class="login-btn">LOGIN</button>

          <!-- Extras section (e.g., Forgot Password) -->

          <div class="extras">

            <a href="#">Forgot Password?</a>

          </div>

          <!-- No registration link for admin login -->

        </form>

      </div>

    </div>

  </div>

</body>

</html>

2.*adminstyle.css*

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

  font-family: 'Segoe UI', sans-serif;

}

body {

background: linear-gradient(135deg, #0f0f2c, #1a1a40);

  color: white;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  justify-content: center;

  align-items: center;

}

.logo a {

  text-decoration: none;

  color: inherit;

}

.navbar {

  display: flex;

  justify-content: space-between;

  align-items: center;

  padding: 20px 40px;

  background-color: rgba(0, 0, 0, 0.7);

  position: fixed;

  width: 100%;

  top: 0;

  height: 77px;

  z-index: 1000;

  backdrop-filter: blur(10px);

}

.logo {

  font-size: 26px;

  font-weight: bold;

  color: #00ffd5;

}

.nav-links {

  list-style: none;

  display: flex;

  gap: 30px;

  margin-right: 9px;

}

.nav-links a {

  color: white;

  text-decoration: none;

  font-weight: 500;

  transition: color 0.3s ease;

}

.nav-links a:hover {

  color: #00ffd5;

}

.nav-btn .login-btn {

  background: transparent;

  border: 2px solid white;

  padding: 8px 16px;

  border-radius: 8px;

  color: white;

  font-weight: bold;

  transition: background 0.3s, color 0.3s;

}

.nav-btn .login-btn:hover {

  background: #00ffd5;

  color: black;

}

.login-container {

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100vh;

  padding-top: 80px;

  background-color: #0d0d2a;

  width: 100%;

}

.login-box {

  display: flex;

  width: 900px;

  max-width: 95%;

  background-color: rgba(10, 10, 30, 0.7);

  border-radius: 16px;

  overflow: hidden;

  box-shadow: 0 8px 24px rgba(0, 0, 0, 0.6);

  border: 1px solid rgba(255, 255, 255, 0.08);

  backdrop-filter: blur(20px);

}

.login-left {

  flex: 1;

  min-height: 400px;

}

.login-left img {

  width: 100%;

  height: 100%;

  object-fit: cover;

  filter: brightness(0.8);

}

.login-right {

  flex: 1;

  padding: 40px;

  display: flex;

  flex-direction: column;

  justify-content: center;

}

.login-right h2 {

  font-size: 32px;

  margin-bottom: 20px;

  color: #00ffd5;

}

.input-group {

  margin-bottom: 20px;

}

.input-group input {

  width: 100%;

  padding: 12px;

  background-color: rgba(255, 255, 255, 0.1);

  color: white;

  border: none;

  border-radius: 8px;

  font-size: 16px;

  outline: none;

}

.input-group input::placeholder {

  color: rgba(255, 255, 255, 0.6);

}

.login-btn {

  padding: 12px;

  background-color: #00ffd5;

  border: none;

  color: black;

  font-weight: bold;

  font-size: 16px;

  border-radius: 8px;

  cursor: pointer;

  width: 100%;

  margin-top: 10px;

  transition: background 0.3s ease;

}

.login-btn:hover {

  background-color: #00c8aa;

}

.extras {

  margin-top: 12px;

  font-size: 14px;

  text-align: right;

}

.register-link {

  margin-top: 20px;

  font-size: 14px;

  text-align: center;

}

.register-link a {

  color: #00ffd5;

  text-decoration: none;

}

.register-link a:hover {

  text-decoration: underline;

}

@media (max-width: 768px) {

  .login-box {

    flex-direction: column;

    width: 90%;

  }

  .login-left {

    min-height: 200px;

  }

  .login-right {

    padding: 30px;

  }

  .navbar {

    padding: 15px 20px;

  }

  .nav-links {

    gap: 15px;

  }

}

@media (max-width: 480px) {

  .login-right h2 {

    font-size: 28px;

  }

  .login-btn {

    font-size: 14px;

  }

}

.nav-btn .login-btn {

  background: transparent;

  border: 2px solid white;

  padding: 8px 16px;

  border-radius: 8px;

  color: white;

  font-weight: bold;

  transition: background 0.3s, color 0.3s;

  text-decoration: none;

}

3.*admin\_login.php*

<?php

session\_start();

require './connect.php';

$username = trim($\_POST['username']);

$password = trim($\_POST['password']);

$username = $conn->real\_escape\_string($username);

$password = $conn->real\_escape\_string($password);

$sql = "SELECT \* FROM Admin WHERE user\_name = '$username'";

$result = $conn->query($sql);

if ($result->num\_rows === 1) {

    $admin = $result->fetch\_assoc();

    if ($password === $admin['password']) {

        $\_SESSION['admin\_id'] = $admin['admin\_id'];

        $\_SESSION['admin\_name'] = $admin['user\_name'];

        header("Location: ./dashboard/admin\_panel.php");

        exit();

    } else {

        echo "<script>alert('❌ Incorrect password'); window.location='login\_page.php';</script>";

    }

} else {

    echo "<script>alert('❌ Admin not found'); window.location='login\_page.php';</script>";

}

$conn->close();

?>

4.*approve\_group.php*

<?php

session\_start();

require\_once '../../connect.php';

if (!isset($\_SESSION['admin\_id'])) {

    header("Location: ../../login\_page.php");

    exit();

}

$message = '';

$message\_type = '';

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['group\_id'])) {

    $group\_id = $\_POST['group\_id'];

    if (isset($\_POST['approve'])) {

        // --- Approve Logic ---

        $update\_query = "UPDATE study\_group SET approved = 1 WHERE group\_id = ?";

        $stmt = $conn->prepare($update\_query);

        $stmt->bind\_param("i", $group\_id);

        if ($stmt->execute()) {

            $creator\_query = "SELECT user\_id FROM study\_group WHERE group\_id = ?";

            $stmt\_creator = $conn->prepare($creator\_query);

            $stmt\_creator->bind\_param("i", $group\_id);

            $stmt\_creator->execute();

            $creator\_result = $stmt\_creator->get\_result();

            if($creator\_data = $creator\_result->fetch\_assoc()) {

                $creator\_id = $creator\_data['user\_id'];

                $insert\_member\_query = "INSERT INTO group\_members (group\_id, user\_id, role) VALUES (?, ?, 'moderator')";

                $stmt\_member = $conn->prepare($insert\_member\_query);

                $stmt\_member->bind\_param("ii", $group\_id, $creator\_id);

                $stmt\_member->execute();

                $stmt\_member->close();

            }

            $stmt\_creator->close();

            $message = "Group approved successfully.";

            $message\_type = 'success';

        } else {

            $message = "Error approving group: " . $conn->error;

            $message\_type = 'error';

        }

        $stmt->close();

    } elseif (isset($\_POST['reject'])) {

        // --- Reject Logic (for pending groups) ---

        $delete\_query = "DELETE FROM study\_group WHERE group\_id = ? AND approved = 0";

        $stmt = $conn->prepare($delete\_query);

        $stmt->bind\_param("i", $group\_id);

        if ($stmt->execute()) {

            $message = "Group request rejected and removed successfully.";

            $message\_type = 'success';

        } else {

            $message = "Error rejecting group: " . $conn->error;

            $message\_type = 'error';

        }

        $stmt->close();

    } elseif (isset($\_POST['disband'])) {

        // --- Disband Logic (for approved groups) ---

        $conn->begin\_transaction();

        try {

            $sql\_dn = "DELETE dn FROM downloaded\_notes dn JOIN notes n ON dn.note\_id = n.note\_id WHERE n.group\_id = ?";

            $stmt\_dn = $conn->prepare($sql\_dn);

            $stmt\_dn->bind\_param("i", $group\_id);

            $stmt\_dn->execute();

            $stmt\_dn->close();

            $sql\_notes = "DELETE FROM notes WHERE group\_id = ?";

            $stmt\_notes = $conn->prepare($sql\_notes);

            $stmt\_notes->bind\_param("i", $group\_id);

            $stmt\_notes->execute();

            $stmt\_notes->close();

            $sql\_msgs = "DELETE FROM group\_messages WHERE group\_id = ?";

            $stmt\_msgs = $conn->prepare($sql\_msgs);

            $stmt\_msgs->bind\_param("i", $group\_id);

            $stmt\_msgs->execute();

            $stmt\_msgs->close();

            $sql\_members = "DELETE FROM group\_members WHERE group\_id = ?";

            $stmt\_members = $conn->prepare($sql\_members);

            $stmt\_members->bind\_param("i", $group\_id);

            $stmt\_members->execute();

            $stmt\_members->close();

            $sql\_group = "DELETE FROM study\_group WHERE group\_id = ?";

            $stmt\_group = $conn->prepare($sql\_group);

            $stmt\_group->bind\_param("i", $group\_id);

            $stmt\_group->execute();

            $stmt\_group->close();

            $conn->commit();

            $message = "Group disbanded successfully. All associated data has been removed.";

            $message\_type = 'success';

        } catch (Exception $e) {

            $conn->rollback();

            $message = "Error disbanding group: " . $e->getMessage();

            $message\_type = 'error';

        }

    }

}

// --- Fetch Pending Group Requests into an array ---

$pending\_groups\_query = "

    SELECT sg.group\_id, sg.group\_name, sg.description, u.user\_name AS creator\_name

    FROM study\_group sg

    JOIN users u ON sg.user\_id = u.user\_id

    WHERE sg.approved = 0

    ORDER BY sg.created\_at ASC

";

$pending\_groups\_result = $conn->query($pending\_groups\_query);

$pending\_groups = [];

if ($pending\_groups\_result) {

    while ($row = $pending\_groups\_result->fetch\_assoc()) {

        $pending\_groups[] = $row;

    }

}

$num\_pending = count($pending\_groups);

// --- Fetch Approved Groups for Management ---

$approved\_groups\_query = "

    SELECT sg.group\_id, sg.group\_name, sg.description, u.user\_name AS creator\_name,

           (SELECT COUNT(\*) FROM group\_members gm WHERE gm.group\_id = sg.group\_id) AS member\_count

    FROM study\_group sg

    JOIN users u ON sg.user\_id = u.user\_id

    WHERE sg.approved = 1

    ORDER BY sg.group\_name ASC

";

$approved\_groups\_result = $conn->query($approved\_groups\_query);

$conn->close();

?>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

  <title>Approve Groups - Admin Dashboard</title>

  <link rel="stylesheet" href="../panelstyle.css">

  <link rel="stylesheet" href="approve\_group.css">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

</head>

<body>

  <div class="dashboard-container">

    <aside class="sidebar">

      <div class="sidebar-header">

        <div class="logo">ATHENA</div>

      </div>

      <nav class="sidebar-nav">

        <ul>

          <li><a href="../admin\_panel.php"><i class="fas fa-th-large"></i> Dashboard</a></li>

          <li class="active"><a href="approve\_group.php"><i class="fas fa-check-circle"></i> Approve Groups</a></li>

          <li><a href="../reports/reports.php"><i class="fas fa-flag"></i> Reports</a></li>

          <li><a href="../user\_management/user\_management.php"><i class="fas fa-user-friends"></i> User Management</a></li>

          <li><a href="../announcements/announcements.php"><i class="fas fa-bullhorn"></i> Announcements</a></li>

        </ul>

      </nav>

      <div class="sidebar-footer">

        <a href="../../admin\_logout.php" class="logout-btn"><i class="fas fa-sign-out-alt"></i> Logout</a>

      </div>

    </aside>

    <main class="main-content">

      <header class="main-header">

        <div class="welcome-section">

          <h2>Study Group Management</h2>

        </div>

      </header>

      <?php if ($message): ?>

        <div class="message <?php echo $message\_type; ?>"><?php echo htmlspecialchars($message); ?></div>

      <?php endif; ?>

      <div class="content-container">

        <section class="management-section">

          <h2>Creation Requests</h2>

          <div class="<?php echo ($num\_pending === 1) ? 'single-item-container' : 'card-grid'; ?>">

            <?php if ($num\_pending > 0): ?>

              <?php foreach ($pending\_groups as $group): ?>

                <div class="request-card">

                  <div class="card-content">

                    <h3><?php echo htmlspecialchars($group['group\_name']); ?></h3>

                    <p><?php echo htmlspecialchars($group['description'] ?: 'No description provided.'); ?></p>

                    <p style="font-size: 13px; color: #a0a0b0;">Requested by: <?php echo htmlspecialchars($group['creator\_name']); ?></p>

                  </div>

                  <div class="card-actions">

                    <form method="POST" action="">

                      <input type="hidden" name="group\_id" value="<?php echo $group['group\_id']; ?>">

                      <button type="submit" name="approve" class="action-btn approve-btn">Approve</button>

                    </form>

                    <form method="POST" action="">

                      <input type="hidden" name="group\_id" value="<?php echo $group['group\_id']; ?>">

                      <button type="submit" name="reject" class="action-btn reject-btn" onclick="return confirm('Are you sure you want to reject this group?');">Reject</button>

                    </form>

                  </div>

                </div>

              <?php endforeach; ?>

            <?php else: ?>

              <p class="no-requests">No pending group requests.</p>

            <?php endif; ?>

          </div>

        </section>

        <section class="management-section">

            <!-- ADDED: Header with search bar -->

            <div class="section-header">

                <h2>Approved Groups</h2>

                <div class="search-container">

                    <i class="fas fa-search"></i>

                    <input type="text" id="approved-group-search" placeholder="Search approved groups...">

                </div>

            </div>

            <!-- MODIFIED: Added an ID to the grid container -->

            <div id="approved-groups-grid" class="card-grid">

                <?php if ($approved\_groups\_result && $approved\_groups\_result->num\_rows > 0): ?>

                    <?php while($group = $approved\_groups\_result->fetch\_assoc()): ?>

                        <!-- MODIFIED: Added a specific class for JS targeting -->

                        <div class="request-card approved-group-card">

                            <div class="card-content">

                                <h3><?php echo htmlspecialchars($group['group\_name']); ?></h3>

                                <p><?php echo htmlspecialchars($group['description'] ?: 'No description provided.'); ?></p>

                                <p style="font-size: 13px; color: #a0a0b0;">

                                    Creator: <?php echo htmlspecialchars($group['creator\_name']); ?> |

                                    Members: <?php echo $group['member\_count']; ?>

                                </p>

                            </div>

                            <div class="card-actions">

                                <form method="POST" action="" style="width: 100%;">

                                    <input type="hidden" name="group\_id" value="<?php echo $group['group\_id']; ?>">

                                    <button type="submit" name="disband" class="action-btn reject-btn" style="width: 100%;" onclick="return confirm('Are you sure you want to PERMANENTLY disband this group? This cannot be undone.');">Disband Group</button>

                                </form>

                            </div>

                        </div>

                    <?php endwhile; ?>

                <?php else: ?>

                    <p class="no-requests">No approved groups yet.</p>

                <?php endif; ?>

            </div>

        </section>

      </div>

    </main>

  </div>

  <script src="approve\_group.js"></script>

</body>

</html>

5.*approve\_group.js*

document.addEventListener('DOMContentLoaded', () => {

    const searchInput = document.getElementById('approved-group-search');

    if (searchInput) {

        searchInput.addEventListener('keyup', () => {

            const filter = searchInput.value.toLowerCase();

            const cards = document.querySelectorAll('.approved-group-card');

            cards.forEach(card => {

                const title = card.querySelector('h3').textContent.toLowerCase();

                const description = card.querySelector('p').textContent.toLowerCase();

                if (title.includes(filter) || description.includes(filter)) {

                    card.style.display = ''; // Setting display to '' reverts it to the default (e.g., 'flex')

                } else {

                    card.style.display = 'none'; // Hide the card if it doesn't match

                }

            });

        });

    }

});

5.*approve\_group.css*

.content-container {

    display: flex;

    flex-direction: column;

    gap: 40px;

}

.management-section {

    width: 100%;

}

.section-header {

    display: flex;

    justify-content: space-between;

    align-items: center;

    flex-wrap: wrap;

    gap: 20px;

    margin-bottom: 20px;

}

.management-section h2 {

    font-size: 28px;

    color: #e0e0e0;

    margin-bottom: 0;

}

.search-container {

    position: relative;

}

.search-container i {

    position: absolute;

    left: 15px;

    top: 50%;

    transform: translateY(-50%);

    color: rgba(255, 255, 255, 0.6);

}

#approved-group-search {

    padding: 12px 15px 12px 40px;

    background-color: rgba(255, 255, 255, 0.1);

    border: 1px solid rgba(255, 255, 255, 0.2);

    border-radius: 8px;

    color: white;

    font-size: 16px;

    outline: none;

    width: 350px;

    transition: border-color 0.3s ease;

}

#approved-group-search::placeholder {

    color: rgba(255, 255, 255, 0.6);

}

#approved-group-search:focus {

    border-color: #00ffd5;

}

.card-grid {

    display: grid;

    grid-template-columns: repeat(auto-fit, minmax(320px, 1fr));

    gap: 25px;

}

.single-item-container .request-card {

    max-width: 450px;

}

.request-card {

    background-color: #1a1a2e;

    border-radius: 12px;

    padding: 25px;

    border: 1px solid rgba(255, 255, 255, 0.1);

    display: flex;

    flex-direction: column;

    justify-content: space-between;

    transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.request-card:hover {

    transform: translateY(-5px);

    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.3);

}

.card-content h3 {

    font-size: 20px;

    color: #00ffd5;

    margin-bottom: 8px;

}

.card-content p {

    font-size: 15px;

    color: rgba(255, 255, 255, 0.7);

    line-height: 1.5;

    margin-bottom: 20px;

    flex-grow: 1;

}

.card-actions {

    display: flex;

    justify-content: space-between;

    align-items: center;

    margin-top: auto;

}

.action-btn {

    padding: 12px 25px;

    border: none;

    border-radius: 8px;

    font-size: 14px;

    font-weight: bold;

    cursor: pointer;

    transition: all 0.2s ease-in-out;

    color: white;

}

.approve-btn {

    background-color: #28a745;

}

.approve-btn:hover {

    background-color: #218838;

    box-shadow: 0 0 10px rgba(40, 167, 69, 0.5);

}

.reject-btn {

    background-color: #dc3545;

}

.reject-btn:hover {

    background-color: #c82333;

    box-shadow: 0 0 10px rgba(220, 53, 69, 0.5);

}

.message {

    padding: 15px;

    margin-bottom: 20px;

    border-radius: 8px;

    font-weight: bold;

    text-align: center;

}

.success {

    background-color: rgba(76, 175, 80, 0.2);

    color: #4CAF50;

    border: 1px solid #4CAF50;

}

.error {

    background-color: rgba(244, 67, 54, 0.2);

    color: #f44336;

    border: 1px solid #f44336;

}

.no-requests {

    color: rgba(255, 255, 255, 0.7);

    text-align: center;

    padding: 20px;

    background-color: rgba(255, 255, 255, 0.05);

    border-radius: 8px;

}

User

1.*login\_page.php*

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

  <title>User Login - Athena</title>

  <link rel="stylesheet" href="userstyle.css" />

</head>

<body>

  <header class="navbar">

    <div class="logo"><a href="../index.php">ATHENA</a></div>

    <nav>

      <ul class="nav-links">

        <li><a href="../index.php">Home</a></li>

        <li><a href="../index.php#about">About</a></li>

        <li><a href="../index.php#services">Features</a></li>

        <li><a href="../index.php#contact">Contact</a></li>

      </ul>

    </nav>

    <div class="nav-btn">

      <a href="registration/registeration.php" class="login-btn">Register</a>

    </div>

  </header>

  <div class="login-container">

    <div class="login-box glass">

      <div class="login-left">

        <img src="../jpeg\_files/bgimage2.jpeg" alt="Illustration"/>

      </div>

      <div class="login-right">

        <h2>Login</h2>

        <form action="./user\_login.php" method="POST">

          <div class="input-group">

            <input type="text" name="username" placeholder="Username" required />

          </div>

          <div class="input-group">

            <input type="password" name="password" placeholder="Password" required />

          </div>

          <button type="submit" class="login-btn">LOGIN</button>

          <div class="extras">

            <a href="#">Forgot Password?</a>

          </div>

          <div class="register-link">

            <p>Don't have an account? <a href="./registration/registeration.php">Create your Account →</a></p>

          </div>

        </form>

      </div>

    </div>

  </div>

</body>

</html>

2.*user\_login.php*

<?php

session\_start();

require 'connect.php';

$username = trim($\_POST['username']);

$password = trim($\_POST['password']);

$username = $conn->real\_escape\_string($username);

$password = $conn->real\_escape\_string($password);

$sql = "SELECT user\_id, user\_name, name, password, status FROM users WHERE user\_name = '$username' OR email = '$username'";

$result = $conn->query($sql);

if ($result->num\_rows === 1) {

    $user = $result->fetch\_assoc();

    if ($password === $user['password']) {

        if ($user['status'] === 'banned') {

            echo "<script>alert('Your account is banned. Please contact admin.'); window.location='login\_page.php';</script>";

            exit();

        }

        $\_SESSION['user\_id'] = $user['user\_id'];

        $\_SESSION['user\_name'] = $user['user\_name'];

        $\_SESSION['status'] = $user['status'];

        $\_SESSION['name'] = $user['name'];

        $group\_check\_query = "SELECT COUNT(\*) as group\_count FROM group\_members WHERE user\_id = ?";

        $stmt\_group\_check = $conn->prepare($group\_check\_query);

        $stmt\_group\_check->bind\_param("i", $user['user\_id']);

        $stmt\_group\_check->execute();

        $group\_result = $stmt\_group\_check->get\_result()->fetch\_assoc();

        $stmt\_group\_check->close();

        if ($group\_result['group\_count'] == 0) {

            header("Location: ./first\_time/first\_time\_setup.php");

            exit();

        } else {

            header("Location: ./dashboard/user\_panel.php");

            exit();

        }

    } else {

        echo "<script>alert('❌ Incorrect password'); window.location='login\_page.php';</script>";

    }

} else {

    echo "<script>alert('❌ User not found'); window.location='login\_page.php';</script>";

}

$conn->close();

?>

3.*user\_login.php*

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

  font-family: 'Segoe UI', sans-serif;

}

body {

  background: linear-gradient(135deg, #0f0f2c, #1a1a40);

  color: white;

  min-height: 100vh;

}

.logo a {

  text-decoration: none;

  color: inherit;

}

.navbar {

  display: flex;

  justify-content: space-between;

  align-items: center;

  padding: 20px 40px;

  background-color: rgba(0, 0, 0, 0.7);

  position: fixed;

  width: 100%;

  top: 0;

  height: 77px;

  z-index: 1000;

  backdrop-filter: blur(10px);

}

.logo {

  font-size: 26px;

  font-weight: bold;

  color: #00ffd5;

}

.nav-links {

  list-style: none;

  display: flex;

  gap: 30px;

   margin-right: 28px;

}

.nav-links a {

  color: white;

  text-decoration: none;

  font-weight: 500;

  transition: color 0.3s ease;

}

.nav-links a:hover {

  color: #00ffd5;

}

.nav-btn .login-btn {

  background: transparent;

  border: 2px solid white;

  padding: 8px 16px;

  border-radius: 8px;

  color: white;

  font-weight: bold;

  transition: background 0.3s, color 0.3s;

}

.nav-btn .login-btn:hover {

  background: #00ffd5;

  color: black;

}

.login-container {

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100vh;

  padding-top: 80px;

  background-color: #0d0d2a;

}

.login-box {

  display: flex;

  width: 900px;

  max-width: 95%;

  background-color: rgba(10, 10, 30, 0.7);

  border-radius: 16px;

  overflow: hidden;

  box-shadow: 0 8px 24px rgba(0, 0, 0, 0.6);

  border: 1px solid rgba(255, 255, 255, 0.08);

  backdrop-filter: blur(20px);

}

.login-left {

  flex: 1;

  min-height: 400px;

}

.login-left img {

  width: 100%;

  height: 100%;

  object-fit: cover;

  filter: brightness(0.8);

}

.login-right {

  flex: 1;

  padding: 40px;

  display: flex;

  flex-direction: column;

  justify-content: center;

}

.login-right h2 {

  font-size: 32px;

  margin-bottom: 20px;

  color: #00ffd5;

}

.input-group {

  margin-bottom: 20px;

}

.input-group input {

  width: 100%;

  padding: 12px;

  background-color: rgba(255, 255, 255, 0.1);

  color: white;

  border: none;

  border-radius: 8px;

  font-size: 16px;

  outline: none;

}

.input-group input::placeholder {

  color: rgba(255, 255, 255, 0.6);

}

.login-btn {

  padding: 12px;

  background-color: #00ffd5;

  border: none;

  color: black;

  font-weight: bold;

  font-size: 16px;

  border-radius: 8px;

  cursor: pointer;

  width: 100%;

  margin-top: 10px;

  transition: background 0.3s ease;

}

.login-btn:hover {

  background-color: #00c8aa;

}

.extras {

  margin-top: 12px;

  font-size: 14px;

  text-align: right;

}

.register-link {

  margin-top: 20px;

  font-size: 14px;

  text-align: center;

}

.register-link a {

  color: #00ffd5;

  text-decoration: none;

}

.register-link a:hover {

  text-decoration: underline;

}

.nav-btn .login-btn {

  background: transparent;

  border: 2px solid white;

  padding: 8px 16px;

  border-radius: 8px;

  color: white;

  font-weight: bold;

  transition: background 0.3s, color 0.3s;

  text-decoration: none;

}

4.*profile\_page.php*

<?php

session\_start();

require '../../connect.php';

if (!isset($\_SESSION['user\_id'])) {

    header("Location: ../../login\_page.php");

    exit();

}

$user\_id = $\_SESSION['user\_id'];

$success\_message = '';

$error\_message = '';

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_FILES['profile\_pic\_upload'])) {

    if (!$conn || $conn->connect\_error) {

        require '../../connect.php';

    }

    $target\_dir = "../../../user\_files/profile\_pics/";

    if (!is\_dir($target\_dir)) {

        mkdir($target\_dir, 0777, true);

    }

    $file\_name = basename($\_FILES["profile\_pic\_upload"]["name"]);

    $uploadOk = 1;

    $imageFileType = strtolower(pathinfo($file\_name, PATHINFO\_EXTENSION));

    $unique\_file\_name = uniqid('profile\_') . '.' . $imageFileType;

    $target\_file\_unique = $target\_dir . $unique\_file\_name;

    $db\_file\_path = "user\_files/profile\_pics/" . $unique\_file\_name;

    $check = getimagesize($\_FILES["profile\_pic\_upload"]["tmp\_name"]);

    if($check === false) {

        $error\_message = "File is not an image.";

        $uploadOk = 0;

    }

    if ($\_FILES["profile\_pic\_upload"]["size"] > 5000000) {

        $error\_message = "Sorry, your file is too large (max 5MB).";

        $uploadOk = 0;

    }

    if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg" && $imageFileType != "gif" ) {

        $error\_message = "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";

        $uploadOk = 0;

    }

    if ($uploadOk == 1) {

        if (move\_uploaded\_file($\_FILES["profile\_pic\_upload"]["tmp\_name"], $target\_file\_unique)) {

            $update\_query = "UPDATE users SET profile\_pic = ? WHERE user\_id = ?";

            $stmt\_update = $conn->prepare($update\_query);

            $stmt\_update->bind\_param("si", $db\_file\_path, $user\_id);

            if ($stmt\_update->execute()) {

                $success\_message = "Profile picture updated successfully!";

            } else {

                $error\_message = "Error updating database: " . $stmt\_update->error;

            }

            $stmt\_update->close();

        } else {

            $error\_message = "Sorry, there was an error uploading your file.";

        }

    }

}

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['update\_personal\_info\_submit'])) {

    if (!$conn || $conn->connect\_error) {

        require '../../connect.php';

    }

    $new\_name = trim($\_POST['name'] ?? '');

    $new\_email = trim($\_POST['email'] ?? '');

    $new\_phone\_no = trim($\_POST['phone\_no'] ?? '');

    if (empty($new\_name) || empty($new\_email) || empty($new\_phone\_no)) {

        $error\_message = "All personal information fields are required.";

    } elseif (!filter\_var($new\_email, FILTER\_VALIDATE\_EMAIL)) {

        $error\_message = "Invalid email format.";

    } else {

        $check\_duplicate\_query = "SELECT user\_id FROM users WHERE (email = ? OR phone\_no = ?) AND user\_id != ?";

        $stmt\_check = $conn->prepare($check\_duplicate\_query);

        $stmt\_check->bind\_param("ssi", $new\_email, $new\_phone\_no, $user\_id);

        $stmt\_check->execute();

        $duplicate\_result = $stmt\_check->get\_result();

        if ($duplicate\_result->num\_rows > 0) {

            $error\_message = "Email or Phone number already in use by another account.";

        } else {

            $update\_query = "UPDATE users SET name = ?, email = ?, phone\_no = ? WHERE user\_id = ?";

            $stmt\_update = $conn->prepare($update\_query);

            $stmt\_update->bind\_param("sssi", $new\_name, $new\_email, $new\_phone\_no, $user\_id);

            if ($stmt\_update->execute()) {

                $success\_message = "Personal information updated successfully!";

                $\_SESSION['name'] = $new\_name;

            } else {

                $error\_message = "Error updating personal information: " . $stmt\_update->error;

            }

            $stmt\_update->close();

        }

        $stmt\_check->close();

    }

}

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['change\_password\_submit'])) {

    if (!$conn || $conn->connect\_error) {

        require '../../connect.php';

    }

    $current\_password = $\_POST['current\_password'] ?? '';

    $new\_password = $\_POST['new\_password'] ?? '';

    $confirm\_password = $\_POST['confirm\_password'] ?? '';

    $get\_password\_query = "SELECT password FROM users WHERE user\_id = ?";

    $stmt\_get\_password = $conn->prepare($get\_password\_query);

    $stmt\_get\_password->bind\_param("i", $user\_id);

    $stmt\_get\_password->execute();

    $password\_result = $stmt\_get\_password->get\_result();

    $user\_db\_password = $password\_result->fetch\_assoc()['password'] ?? '';

    $stmt\_get\_password->close();

    if ($current\_password !== $user\_db\_password) {

        $error\_message = "Current password is incorrect.";

    } elseif (empty($new\_password) || empty($confirm\_password)) {

        $error\_message = "New password and confirm password fields cannot be empty.";

    } elseif ($new\_password !== $confirm\_password) {

        $error\_message = "New password and confirm password do not match.";

    } elseif (strlen($new\_password) < 6) {

        $error\_message = "New password must be at least 6 characters long.";

    } else {

        $hashed\_new\_password = $new\_password;

        $update\_password\_query = "UPDATE users SET password = ? WHERE user\_id = ?";

        $stmt\_update\_password = $conn->prepare($update\_password\_query);

        $stmt\_update\_password->bind\_param("si", $hashed\_new\_password, $user\_id);

        if ($stmt\_update\_password->execute()) {

            $success\_message = "Password updated successfully!";

        } else {

            $error\_message = "Error updating password: " . $stmt\_update\_password->error;

        }

        $stmt\_update\_password->close();

    }

}

if (!$conn || $conn->connect\_error) {

    require '../../connect.php';

}

$user\_query = "SELECT user\_name, name, email, phone\_no, status, reputation\_score, profile\_pic, created\_at FROM users WHERE user\_id = ?";

$stmt\_user = $conn->prepare($user\_query);

$stmt\_user->bind\_param("i", $user\_id);

$stmt\_user->execute();

$user\_result = $stmt\_user->get\_result();

$user\_data = $user\_result->fetch\_assoc();

$stmt\_user->close();

if ($conn && !$conn->connect\_error) {

    $conn->close();

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

  <title>User Profile - Athena</title>

  <link rel="stylesheet" href="../panelstyle.css" />

  <link rel="stylesheet" href="profile\_style.css" />

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

</head>

<body>

  <div class="dashboard-wrapper">

    <aside class="sidebar">

      <div class="sidebar-header">

        <div class="logo">ATHENA</div>

      </div>

      <nav class="sidebar-nav">

        <ul>

          <li><a href="../user\_panel.php"><i class="fas fa-th-large"></i> Dashboard</a></li>

          <li><a href="../study\_groups/study\_group.php"><i class="fas fa-users"></i> Study Groups</a></li>

          <li><a href="../notes/notes.php"><i class="fas fa-book"></i> Notes</a></li>

          <li><a href="../chat\_box/group\_chat.php"><i class="fas fa-comments"></i> Group Chat</a></li>

          <li><a href="../to-do/to-do.php"><i class="fas fa-clipboard-list"></i> To-Do List</a></li>

          <li><a href="../announcements/u\_announcements.php"><i class="fas fa-bullhorn"></i> Announcements</a></li>

          <li class="active"><a href="profile\_page.php"><i class="fas fa-user-circle"></i> Profile</a></li>

        </ul>

      </nav>

      <div class="sidebar-footer">

        <a href="../../user\_logout.php" class="logout-btn"><i class="fas fa-sign-out-alt"></i> Logout</a>

      </div>

    </aside>

    <main class="main-content">

      <header class="main-header">

        <div class="welcome-section">

          <h2>Profile Settings</h2>

        </div>

        <!-- NEW: Header actions container for the delete button -->

        <div class="header-actions">

            <button id="delete-account-trigger-btn" class="delete-account-btn-header"><i class="fas fa-trash-alt"></i> Delete Account</button>

        </div>

      </header>

      <section class="profile-grid">

        <?php if ($success\_message): ?>

            <div class="message success-message"><?= htmlspecialchars($success\_message) ?></div>

        <?php endif; ?>

        <?php if ($error\_message): ?>

            <div class="message error-message"><?= htmlspecialchars($error\_message) ?></div>

        <?php endif; ?>

        <div class="card personal-info-card">

          <h3>Personal Information</h3>

          <div class="card-content">

            <div class="profile-avatar">

              <img src="../../../<?= htmlspecialchars($user\_data['profile\_pic'] ?? 'https://placehold.co/100x100/0f0f2c/00ffd5?text=DP') ?>" alt="Profile Picture" class="avatar-img">

              <form action="" method="POST" enctype="multipart/form-data" id="profile-pic-form" style="display: none;">

                <input type="file" name="profile\_pic\_upload" id="profile\_pic\_input" accept="image/\*">

                <input type="submit" value="Upload" style="display: none;">

              </form>

              <button class="change-avatar-btn" id="trigger-pic-upload">Change Avatar</button>

            </div>

            <form action="" method="POST" id="personal-info-form">

              <div class="input-group">

                <label for="name">Full Name</label>

                <input type="text" id="name" name="name" value="<?= htmlspecialchars($user\_data['name'] ?? '') ?>" placeholder="Enter full name" required>

              </div>

              <div class="input-group">

                <label for="email">Email Address</label>

                <input type="email" id="email" name="email" value="<?= htmlspecialchars($user\_data['email'] ?? '') ?>" placeholder="Enter email address" required>

              </div>

              <div class="input-group">

                <label for="phone\_no">Phone No.</label>

                <input type="tel" id="phone\_no" name="phone\_no" value="<?= htmlspecialchars($user\_data['phone\_no'] ?? '') ?>" placeholder="Enter phone number" required>

              </div>

              <button type="submit" name="update\_personal\_info\_submit" class="save-changes-btn-local"><i class="fas fa-save"></i> Save Personal Info</button>

            </form>

          </div>

        </div>

        <div class="card password-settings-card">

          <h3>Password Settings</h3>

          <div class="card-content">

            <form action="" method="POST" id="password-change-form">

              <div class="input-group">

                <label for="current-password">Current Password</label>

                <input type="password" id="current-password" name="current\_password" placeholder="Enter current password">

              </div>

              <div class="input-group">

                <label for="new-password">New Password</label>

                <input type="password" id="new-password" name="new\_password" placeholder="Enter new password">

              </div>

              <div class="input-group">

                <label for="confirm-password">Confirm New Password</label>

                <input type="password" id="confirm-password" name="confirm\_password" placeholder="Confirm new password">

              </div>

              <button type="submit" name="change\_password\_submit" class="save-changes-btn-local" style="margin-top: auto;"><i class="fas fa-key"></i> Change Password</button>

            </form>

          </div>

        </div>

        <div class="card account-overview-card">

          <h3>Account Overview</h3>

          <div class="card-content">

            <div class="reputation-score-container">

                <div class="reputation-score-label">Reputation Score</div>

                <div class="reputation-score-value"><?= htmlspecialchars($user\_data['reputation\_score'] ?? '0') ?></div>

            </div>

            <div class="info-item">

              <span class="info-label">Status</span>

              <span class="info-value"><?= htmlspecialchars($user\_data['status'] ?? 'N/A') ?></span>

            </div>

            <div class="info-item">

              <span class="info-label">Created At</span>

              <span class="info-value"><?= htmlspecialchars(date('Y-m-d', strtotime($user\_data['created\_at'] ?? ''))) ?></span>

            </div>

          </div>

        </div>

      </section>

    </main>

  </div>

  <div id="delete-confirm-modal" class="modal-overlay" style="display: none;">

      <div class="modal-content">

          <h3>Confirm Account Deletion</h3>

          <p>Are you sure you want to permanently delete your account? This action cannot be undone.</p>

          <div class="modal-actions">

              <button id="cancel-delete-btn" class="modal-btn cancel-btn">Cancel</button>

              <form action="delete\_account.php" method="POST" style="display: inline;">

                  <button type="submit" id="confirm-delete-btn" class="modal-btn confirm-btn">Delete Account</button>

              </form>

          </div>

      </div>

  </div>

  <script src="../dscript.js"></script>

</body>

</html>

4.*delete\_account.php*

<?php

session\_start();

require '../../connect.php';

if (!isset($\_SESSION['user\_id']) || $\_SERVER['REQUEST\_METHOD'] !== 'POST') {

    header("Location: ../../login\_page.php");

    exit();

}

$user\_id = $\_SESSION['user\_id'];

// Begin a transaction to ensure all or nothing is deleted.

$conn->begin\_transaction();

try {

    $created\_groups\_query = "SELECT group\_id FROM study\_group WHERE user\_id = ?";

    $stmt\_created\_groups = $conn->prepare($created\_groups\_query);

    $stmt\_created\_groups->bind\_param("i", $user\_id);

    $stmt\_created\_groups->execute();

    $created\_groups\_result = $stmt\_created\_groups->get\_result();

    while ($group = $created\_groups\_result->fetch\_assoc()) {

        $group\_id = $group['group\_id'];

        $conn->query("DELETE FROM downloaded\_notes WHERE note\_id IN (SELECT note\_id FROM notes WHERE group\_id = $group\_id)");

        $conn->query("DELETE FROM notes WHERE group\_id = $group\_id");

        $conn->query("DELETE FROM group\_messages WHERE group\_id = $group\_id");

        $conn->query("DELETE FROM group\_members WHERE group\_id = $group\_id");

        $conn->query("DELETE FROM reports WHERE group\_id = $group\_id");

        $conn->query("DELETE FROM study\_group WHERE group\_id = $group\_id");

    }

    $stmt\_created\_groups->close();

    $conn->query("DELETE FROM downloaded\_notes WHERE user\_id = $user\_id");

    $conn->query("DELETE FROM reports WHERE user\_id = $user\_id OR target\_id = $user\_id");

    $conn->query("DELETE FROM group\_messages WHERE user\_id = $user\_id");

    $conn->query("DELETE FROM notes WHERE user\_id = $user\_id");

    $conn->query("DELETE FROM to\_do WHERE user\_id = $user\_id");

    $conn->query("DELETE FROM group\_members WHERE user\_id = $user\_id");

    $stmt\_delete\_user = $conn->prepare("DELETE FROM users WHERE user\_id = ?");

    $stmt\_delete\_user->bind\_param("i", $user\_id);

    $stmt\_delete\_user->execute();

    $stmt\_delete\_user->close();

    $conn->commit();

    session\_destroy();

    header("Location: ../../login\_page.php?message=account\_deleted");

    exit();

} catch (Exception $e) {

    $conn->rollback();

    $\_SESSION['error\_message'] = "Could not delete account. An error occurred: " . $e->getMessage();

    header("Location: profile\_page.php");

    exit();

}

$conn->close();?>

5.*delete\_account.php*

.profile-grid {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr)); /\* Creates a responsive 3-column grid \*/

  gap: 30px;

  margin-top: 30px;

}

.profile-grid .card {

  background-color: rgba(255, 255, 255, 0.08);

  border-radius: 12px;

  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);

  padding: 25px;

  display: flex;

  flex-direction: column;

  border: 1px solid rgba(255, 255, 255, 0.2);

  backdrop-filter: blur(10px);

  transition: transform 0.3s ease, box-shadow 0.3s ease;

  color: white;

}

.profile-grid .card:hover {

  transform: translateY(-8px);

  box-shadow: 0 12px 28px rgba(0, 0, 0, 0.4);

}

.profile-grid .card h3 {

  font-size: 22px;

  color: #00ffd5;

  margin-bottom: 20px;

  border-bottom: 1px solid rgba(255, 255, 255, 0.1);

  padding-bottom: 15px;

}

.profile-grid .card .card-content {

    flex-grow: 1;

    display: flex;

    flex-direction: column;

}

.info-item {

  display: flex;

  justify-content: space-between;

  align-items: center;

  padding: 10px 0;

  border-bottom: 1px solid rgba(255, 255, 255, 0.08);

}

.info-item:last-child {

  border-bottom: none;

}

.info-label {

  font-weight: bold;

  color: rgba(255, 255, 255, 0.8);

}

.info-value {

  color: rgba(255, 255, 255, 0.7);

}

.profile-avatar {

  display: flex;

  flex-direction: column;

  align-items: center;

  margin-bottom: 20px;

}

.avatar-img {

  width: 120px;

  height: 120px;

  border-radius: 50%;

  object-fit: cover;

  border: 3px solid #00ffd5;

  margin-bottom: 15px;

  box-shadow: 0 0 10px rgba(0, 255, 213, 0.3);

}

.change-avatar-btn {

  background-color: rgba(0, 255, 213, 0.2);

  color: #00ffd5;

  padding: 8px 15px;

  border: 1px solid #00ffd5;

  border-radius: 6px;

  font-size: 14px;

  cursor: pointer;

  transition: background-color 0.2s ease, color 0.2s ease, transform 0.2s ease;

}

.change-avatar-btn:hover {

  background-color: #00ffd5;

  color: black;

  transform: translateY(-2px);

}

.input-group {

  margin-bottom: 15px;

}

.input-group label {

  display: block;

  margin-bottom: 8px;

  font-weight: bold;

  color: rgba(255, 255, 255, 0.8);

}

.input-group input[type="password"],

.input-group input[type="text"],

.input-group input[type="email"],

.input-group input[type="tel"] {

  width: 100%;

  padding: 10px 15px;

  background-color: rgba(255, 255, 255, 0.1);

  color: white;

  border: 1px solid rgba(255, 255, 255, 0.2);

  border-radius: 8px;

  font-size: 16px;

  outline: none;

  transition: border-color 0.2s ease;

}

.input-group input::placeholder {

  color: rgba(255, 255, 255, 0.6);

}

.input-group input:focus {

  border-color: #00ffd5;

}

.reputation-score-container {

    text-align: center;

    padding: 20px 0;

    flex-grow: 1;

    display: flex;

    flex-direction: column;

    justify-content: center;

    border-bottom: 1px solid rgba(255, 255, 255, 0.1);

    margin-bottom: 10px;

}

.reputation-score-value {

    font-size: 72px;

    font-weight: bold;

    color: #00ffd5;

    line-height: 1;

    text-shadow: 0 0 20px rgba(0, 255, 213, 0.5);

}

.reputation-score-label {

    font-size: 16px;

    color: rgba(255, 255, 255, 0.8);

    margin-top: 5px;

    text-transform: uppercase;

    letter-spacing: 1px;

}

.save-changes-btn-local {

  background-color: #00ffd5;

  color: black;

  padding: 10px 20px;

  border: none;

  border-radius: 8px;

  font-size: 16px;

  cursor: pointer;

  display: inline-flex;

  align-items: center;

  gap: 8px;

  font-weight: bold;

  transition: background-color 0.3s ease, transform 0.2s ease;

  margin-top: 10px;

}

.save-changes-btn-local:hover {

  background-color: #00c8aa;

  transform: translateY(-2px);

}

.save-changes-btn-local i {

    margin-right: 5px;

}

.message {

    padding: 10px;

    margin-bottom: 15px;

    border-radius: 8px;

    font-weight: bold;

    text-align: center;

    grid-column: 1 / -1;

}

.success-message {

    background-color: rgba(76, 175, 80, 0.2);

    color: #4CAF50;

    border: 1px solid #4CAF50;

}

.error-message {

    background-color: rgba(244, 67, 54, 0.2);

    color: #f44336;

    border: 1px solid #f44336;

}

@media (max-width: 1200px) { /\* Adjust breakpoint for when cards stack \*/

  .profile-grid {

    grid-template-columns: repeat(auto-fit, minmax(350px, 1fr));

  }

}

.delete-account-btn-header {

    background-color: #dc3545;

    color: white;

    padding: 8px 15px;

    border: none;

    border-radius: 8px;

    font-size: 14px;

    font-weight: bold;

    cursor: pointer;

    transition: background-color 0.3s ease, transform 0.2s ease;

    display: inline-flex;

    align-items: center;

    gap: 8px;

}

.delete-account-btn-header:hover {

    background-color: #c82333;

    transform: translateY(-2px);

}

.modal-overlay {

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background-color: rgba(0, 0, 0, 0.7);

    backdrop-filter: blur(5px);

    display: flex;

    justify-content: center;

    align-items: center;

    z-index: 2000;

}

.modal-content {

    background-color: #1a1a2e;

    padding: 30px;

    border-radius: 12px;

    border: 1px solid rgba(255, 255, 255, 0.1);

    box-shadow: 0 8px 24px rgba(0, 0, 0, 0.5);

    width: 90%;

    max-width: 450px;

    text-align: center;

}

.modal-content h3 {

    color: #f44336; /\* Red color for warning \*/

    margin-top: 0;

    margin-bottom: 15px;

    font-size: 22px;

}

.modal-content p {

    color: rgba(255, 255, 255, 0.8);

    margin-bottom: 25px;

    line-height: 1.6;

}

.modal-actions {

    display: flex;

    justify-content: center;

    gap: 15px;

}

.modal-btn {

    padding: 10px 25px;

    border: none;

    border-radius: 8px;

    font-size: 16px;

    font-weight: bold;

    cursor: pointer;

    transition: background-color 0.2s ease;

}

.cancel-btn {

    background-color: #4a4c68;

    color: white;

}

.cancel-btn:hover {

    background-color: #5c5e81;

}

.confirm-btn {

    background-color: #dc3545;

    color: white;

}

.confirm-btn:hover {

    background-color: #c82333;

}

**BIBLIOGRAPHY**

Books Used :-

Software Engineering – RS Pressman

Website references :-

<https://openai.com>

<https://roadmap.sh/frontend/projects>

<https://www.guvi.in>

<https://miro.com/app/dashboard>