

CAMPUS NOTES

A project submitted in partial fulfillment of requirements for degree of

BACHELOR OF TECHNOLOGY(CSE)

BY

ADITYA SHARMA

[Roll No: 2162017006]

[Session: 2021-25]

Under the Supervision of

Dr. Pramesh Srivastava

School of Engineering and Technology



IIMT UNIVERSITY

‘O’ Pocket, Ganga Nagar Colony,

Mawana Road, Meerut (U.P.), India

DECLARATION BY THE CANDIDATE

The Research work embodied in this project entitled “**CAMPUS NOTES**” has been carried out at the “**School of Engineering and Technology, IIMT University, Meerut, U.P.**”. The extent of information derived from the existing literature has been indicated in the body of the project at appropriate places along with the source of information. The work is original and has not been submitted in part or for any degree or diploma of this or any other University.

Date:

ADITYA SHARMA

Place: Meerut

[2162017006]

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project report entitled “**CAMPUS NOTES**” submitted by **ADITYA SHARMA** in partial fulfilment of requirements for degree of **Bachelor of Technology** at the “**School of Engineering and Technology, IIMT University, Meerut, U.P.**” is a record of the candidate own work carried out by him under my supervision. The matter embodied in this project is original and has not submitted for the award of any other degree.

Mrs. Shweta Garg
Assistant Professor

Dr. Pramesh Srivasatva
Associate Professor

Dr. Pankaj Singh
DEAN
IIMT University

Mrs. Archana Jain
HOD
IIMT University

ACKNOWLEDGEMENT

I am greatly thankful to **Dr. Pankaj Singh**, [Dean] School of Engineering and Technology IIMT University. Meerut, U.P. India for providing necessary infrastructure to carry out my project work at the university. I am express my sincere thanks and gratitude to **Mrs. Shweta Garg**, [Assistant Professor & Head] School of Computer Science & Applications IIMT University. Meerut, U.P. India for her moral support. valuable guidance and encouragement during the various stages of my work. I am feeling oblige in taking the opportunity to sincerely thanks to **Dr. Pramesh Srivastava**, [Associate Professor] School of Computer Science & Applications IIMT University. Meerut, U.P. India for his valuable guidance, valuable advice and whole heartedly co-operation. His sincerity, thoroughness, timely help, and constructive criticism has been a constant source of inspiration for me. I would also like to acknowledge my parents and friends for the whole hearted moral support and unending encouragement they provided me during my project work

Date:03/06/2025

Place: Meerut

Aditya Sharma

[2162017006]

CERTIFICATE OF PLAGIARISM CHECK

1	Name of Student/ Researchscholar	Aditya Sharma
2	Enrolment Number	IIMTU\2162085
3	Department Name	B.tech CSE
4	Subject	Major Project
5	Category	UG
6	Admission Year	2021
7	Title of the Dissertation/ Thesis	Campus Notes
8	Name of the Supervisor/ Guide	Dr.Pramesh Srivastava
9	Acceptable Maximum Limit	10%
10	% Similarity of content Identified	
11	Used Software Drillbit / Turnitin	
12	Date of Verification	

Report on Plagiarism check, Item with _____% of similarity is attached.

Signature of the Student

Signature of the Supervisor

Dean/Deputy Dean/Head of the Department

(Seal)

TABLE OF CONTENTS

Topic	Page
1. Abbreviations	1-3
2. Introduction	4-7
3. Objectives	8-10
4. Tools/Environment Used	11-13
5. Analysis/Observation Document	14-15
* Software Requirement Specification (SRS)	16-18
* Diagrams (ER, DFD)	19-21
* Data Dictionary	22-26
6. Design Document	27
* Modularization	27-29
* Database Design	30-31
* Procedural Design	32-33
7. Program Code	33-59
8. Input and Output Screens	60-62
9. Discussion	63-64
10. Bibliography	65-66



ABBREVIATIONS

This section lists the abbreviations and acronyms used throughout the report to ensure clarity and consistency for readers unfamiliar with technical terminology.

	Abbreviation	Full Form	
	-----	-----	
	UI	User Interface	
	UX	User Experience	
	DBMS	Database Management System	
	ER	Entity Relationship	
	DFD	Data Flow Diagram	
	SRS	Software Requirement Specification	

	HTML	HyperText Markup Language	
	CSS	Cascading Style Sheets	
	JS	JavaScript	
	SQL	Structured Query Language	
	CRUD	Create, Read, Update, Delete	
	HTTP	Hypertext Transfer Protocol	
	MVC	Model View Controller	
	IDE	Integrated Development Environment	

	API	Application Programming Interface	
	UI	User Interface	
	UX	User Experience	

INTRODUCTION

Campus Notes is a web-based platform that simplifies the process of sharing and accessing academic notes among students. It aims to streamline peer-to-peer educational support within an academic institution. The platform is designed to allow students to upload, download, and search for study materials based on subjects, courses, or semesters. It ensures a centralized repository of notes that can be accessed anytime, anywhere, thereby reducing dependency on physical distribution. It also promotes a collaborative academic environment where students contribute and benefit mutually. The system supports user authentication to maintain content integrity and prevent misuse. Features like subject-wise organization, PDF preview, user dashboards, and responsive design make the platform user-friendly and efficient. The project follows software engineering principles including modular design, maintainability, and data integrity. The target audience includes undergraduate and postgraduate students, faculty members, and administrators who want a structured, digital system to distribute and manage academic resources. Campus Notes helps reduce redundancy in study material and enhances academic efficiency. The system can be further enhanced with machine learning-based recommendations, notifications, and user performance analytics.

Key Features:

1. Centralized Repository

One of the core functionalities of Campus Notes is its dynamic centralized repository where students can effortlessly upload their notes and study materials categorized by subjects, specific courses, and semesters. The advanced search capabilities enable users to quickly locate the most relevant notes for their academic needs, thereby improving study efficiency. A thoughtfully designed user interface coupled with subject-wise organization ensures that students can browse through materials in a structured and systematic manner without any confusion or clutter.

2. User Authentication

To uphold the integrity and relevance of the resources shared, Campus Notes incorporates a

robust user authentication system. This security feature restricts access to registered and verified users, preventing unauthorized usage and minimizing the risks of misinformation or plagiarized content. Moreover, the authentication process helps maintain a trustworthy and collaborative academic community, where contributions are credited to their rightful authors.

3.PDF Preview

An important feature of the platform is the PDF preview functionality, which allows users to view the contents of notes before deciding to download them. This feature saves bandwidth and device storage by giving students the option to verify the usefulness of a document in advance.

4.User Dashboards

In addition, the platform provides personalized user dashboards, offering detailed summaries of each student's notes uploads, downloads, and interactions. This personalized space fosters greater engagement by enabling students to track their learning progress and contributions within the campus community.

5.Responsive Design

Campus Notes is built with a mobile-first, responsive design philosophy, ensuring optimal usability across desktops, tablets, and smartphones. This multipurpose accessibility guarantees that students can engage with the platform anytime and anywhere, bridging gaps caused by physical distance or limited access to printed materials. The platform's user-centric approach ensures it remains intuitive and straightforward enough to encourage active participation from a diverse range of users.

6.Benefits

- **Reduced Dependency on Physical Distribution:** By providing a digital platform, Campus Notes minimizes the need for physical copies of study materials, making it easier for students to access resources.
- **Collaborative Academic Environment:** The platform encourages students to contribute

their notes, fostering a sense of community and mutual benefit.

- **Enhanced Academic Efficiency:** By reducing redundancy in study materials, students can focus on their studies without the hassle of searching for resources.

7. Architecture and Design Principles

Architecturally, the system adheres to software engineering principles such as modular design, which promotes maintainability and scalability. By structuring the application into distinct modules like authentication, file management, and search, developers can efficiently manage and upgrade the system over time. Emphasis on data integrity through validation, encryption of sensitive information, and controlled permissions further enhances the reliability of the platform.

8. Target Audience

The primary users of Campus Notes include:

- **Undergraduate and Postgraduate Students:** Seeking a structured system to access and share academic resources.
- **Faculty Members:** Looking for a platform to distribute course materials and engage with students.
- **Administrators:** Interested in managing and overseeing the distribution of academic resources effectively.

9. Future Enhancements

Beyond its current capabilities, Campus Notes envisions integrating intelligent features powered by machine learning and analytics. For example, recommendation algorithms can analyze users' academic behavior and preferences to suggest personalized study materials, boosting their learning efficiency. Notification systems can alert students about new uploads, deadlines, or collaborative opportunities, keeping the academic community continuously informed and engaged. Furthermore, user performance analytics can provide valuable insights into individual

study habits, enabling students and educators to identify areas for improvement and tailor educational approaches.

OBJECTIVES

The primary objective of *Campus Notes* is to develop a centralized digital platform that facilitates seamless and structured note-sharing among students, fostering a collaborative academic environment. This initiative aims to enhance access to quality educational resources by allowing students to upload and retrieve notes categorized by semester, subject, or faculty, thereby streamlining the study process and promoting academic efficiency. By consolidating all study materials into a single, well-organized system, the platform minimizes reliance on unverified third-party sources and informal channels. A key focus is to maintain the quality and relevance of shared content through account-based submissions, user reviews, and administrative oversight. *Campus Notes* is also committed to inclusivity, ensuring that all students—including those from underprivileged backgrounds—have equal access to essential learning materials. Furthermore, the platform encourages both students and faculty to engage with digital tools, thereby supporting the development of digital literacy within the academic community. User privacy and data protection are prioritized through secure authentication systems and clear privacy policies. Designed with future scalability in mind, the platform can be expanded to include interactive features such as quizzes, discussion forums, or AI-powered content suggestions. Overall, the project aligns with broader institutional goals of promoting e-learning, supporting digital campus transformation, and democratizing access to knowledge for all learners.

Key objectives include:

1. Easy Access to Academic Content

One of the core objectives of the platform is to provide seamless access to notes and academic resources. Students can upload and download content based on filters like course semester, subject matter, or specific faculty members. This categorization system makes the process intuitive and user-friendly, eliminating the hassle of browsing through unrelated materials. By streamlining access to content, the platform saves time and helps students prepare more effectively for exams and assignments.

2. Centralized and Organized Repository

Campus Notes addresses the issue of fragmented and inconsistent study materials by offering a centralized repository. Instead of relying on social media groups, peer-to-peer sharing, or unofficial websites, students can turn to a single, reliable source for all their academic needs. This centralization ensures that the information is consistent, reduces duplication, and enhances the overall learning experience through better organization and structure.

3. Quality Control and Relevance

To maintain academic integrity and usefulness, the platform incorporates quality control mechanisms. All submissions are tied to user accounts, allowing for accountability. Faculty or administrators can monitor content to ensure it meets academic standards. In future versions, rating systems and peer reviews can further enhance quality by enabling users to flag outdated, inaccurate, or low-quality notes. This ensures that students rely on trustworthy materials for their studies.

4. Promoting Educational Inclusivity

Inclusivity is a major focus of the platform. Many students struggle to access quality study materials due to financial or logistical constraints. Campus Notes aims to bridge this gap by offering a free and open-access platform for academic content. Regardless of their background, all students can benefit equally from the resources, helping to create a more level academic playing field across the institution.

5. Enhancing Digital Literacy

Another key objective is to foster digital literacy among both students and faculty. By using the

platform, users become more comfortable with digital tools and online collaboration. This aligns with modern educational goals, where proficiency in digital platforms is increasingly essential. It also prepares students for professional environments where such skills are expected.

6. Ensuring User Privacy and Security

To build user trust and ensure safe use of the platform, strong privacy measures are essential. Campus Notes will use secure authentication methods, such as email or institutional login, to protect accounts. User data policies will be transparent, and sensitive information will be kept confidential. These measures ensure that users feel safe while contributing to or accessing the platform.

7. Scalability and Future Development

Campus Notes is designed with long-term scalability in mind. While the initial focus is on note-sharing, the platform's architecture supports future features such as discussion forums, collaborative group spaces, quizzes, and AI-powered study aids. This flexibility allows the platform to evolve with the needs of its users and the educational institution, supporting continuous improvement in digital learning environments.

Tools, Environment Used / Materials & Methods

Campus Notes is developed using the Django web framework due to its rapid development capabilities and robust support for database-driven applications. The frontend is built using HTML, CSS, and JavaScript for responsive design. Bootstrap is used to ensure UI consistency across devices. The backend logic is written in Python using Django, while the database is handled through SQLite (for development) and can be migrated to PostgreSQL or MySQL for production. Described below:

1. Framework:

- **Django:** Chosen for its rapid development capabilities and robust support for database-driven applications.

2. Frontend Technologies:

- **HTML:** For structuring the web pages.
- **CSS:** For styling and layout.
- **JavaScript:** For adding interactivity and dynamic content.
- **Bootstrap:** Utilized to ensure UI consistency and responsiveness across various devices.

3. Backend Technologies:

- **Python:** The primary programming language used for backend logic, leveraging Django's features.

4. Database:

- **SQLite:** Used during the development phase for its simplicity and ease of use.
- **PostgreSQL/MySQL:** Options for production deployment, allowing for scalability and robustness.

Methods

- **Development Approach:**

Agile methodology is likely employed to facilitate iterative development and continuous feedback.

- **Responsive Design:**

Implemented using Bootstrap to ensure that the application is accessible on various screen sizes and devices.

- **Database Management:**

Initial development with SQLite allows for quick prototyping, with the option to migrate to more robust databases like PostgreSQL or MySQL for production environments.

This structured approach ensures that Campus Notes is built efficiently, with a focus on user

experience and scalability

Tools & Technologies:

- * Frontend: HTML5, CSS3, JavaScript, Bootstrap
- * Backend: Python 3.x, Django
- * Database: SQLite (development), PostgreSQL (optional)
- * IDE: VS Code / PyCharm
- * Version Control: GitHub
- * Hosting/Deployment: Render / PythonAnywhere / Heroku
- * Diagrams: Draw\io / Lucid chart

The development followed Agile methodology with iterative cycles, each introducing a set of features followed by testing. Database schema was normalized to maintain integrity. Role-based access was implemented to separate users (students) from admin (faculty/staff).

Analysis / Observation Document

Software Requirements Specifications (SRS):

Introduction:

Campus Notes is a centralized platform designed for students to share academic resources by uploading and downloading notes categorized by subject and semester. The platform emphasizes secure access, efficient content management, and a responsive user experience.

Functional Requirements:

- **Student Registration and Login:** Users can create accounts and log in securely to access the platform.
- **Notes Upload with Categorization:** Students can upload their notes, organizing them by subject and semester for easy retrieval.
- **Search and Filter Notes:** A search feature enables users to quickly find specific notes, with options to filter results based on categories.
- **Admin Panel for Content Moderation:** An admin interface allows for the moderation of uploaded content, ensuring quality and relevance.

Non-Functional Requirements:

- **Cross-Platform Compatibility:** The application is designed to work on various devices and operating systems.
- **Secure Login System:** A secure authentication process protects user accounts and data.
- **User -Friendly Interface:** The platform features an intuitive design for easy navigation.
- **Scalability:** The system is built to accommodate a growing user base without compromising performance.

Assumptions and Constraints:

- Users must have internet access to utilize the platform.
- Uploaded files are required to be in PDF format for consistency.
- Admin approval is necessary for notes to be publicly accessible.

This project aims to enhance academic collaboration among students, providing a reliable and efficient way to share educational resources.

DIAGRAMS

Class Diagram

The Entity-Relationship (ER) diagram serves as a foundational blueprint for the Campus Notes system, illustrating the key entities involved and their interrelationships. This diagram is crucial for understanding how data is structured and how different components of the application interact with one another.

Entities:

1. User :

The User entity represents the students who will interact with the Campus Notes platform. Each user is uniquely identified by an ID, which serves as the primary key. Additional attributes include the user's name, email address, and password. The email address is particularly important as it is used for account verification and communication. The password is stored securely, typically hashed, to protect user data and ensure secure access to the platform.

2. Notes:

The Notes entity encapsulates the academic resources that users upload. Each note is assigned a unique ID, which acts as its primary key. The title and description provide context about the content of the note, while the file path indicates where the actual document is stored. The “ **uploaded_by** ” attribute links each note back to the user who uploaded it, establishing a clear

relationship between users and their contributions to the platform.

3. **Subject:**

The Subject entity categorizes the notes into specific academic disciplines. Each subject has a unique ID, serving as its primary key, and a name that describes the subject (e.g., "Mathematics," "Biology"). This categorization is essential for users to easily navigate and find relevant notes. The Subject entity also includes a foreign key that links it to the Semester entity, indicating which semester the subject is associated with.

4. **Semester:**

The Semester entity represents the academic periods during which subjects are taught. Each semester is uniquely identified by an ID, which is the primary key. The name or number of the semester (e.g., "Fall 2023") provides context for users. This entity is crucial for organizing subjects and, consequently, the notes associated with them, allowing users to filter content based on their current academic schedule.

Relationships:

- **User** **uploads** **Notes:**
This one-to-many relationship indicates that a single user can upload multiple notes, but each note is associated with only one user. This relationship is vital

for tracking contributions and ensuring that users can manage their own uploads effectively.

- **Notes belong to Subject:**

This many-to-one relationship signifies that multiple notes can be associated with a single subject, while each note is linked to only one subject. This structure allows for efficient categorization and retrieval of notes based on academic disciplines.

- **Subject assigned to Semester:**

This many-to-one relationship indicates that multiple subjects can be offered in a single semester, while each subject is tied to one specific semester. This relationship helps in organizing the curriculum and ensuring that students can find notes relevant to their current courses.

The ER diagram not only aids in the design of the database schema but also serves as a communication tool among developers, stakeholders, and users, ensuring a shared understanding of the system's structure.

DATA FLOW DIAGRAM

The Level 0 Data Flow Diagram (DFD) provides a high-level overview of the data flow within the Campus Notes system, illustrating how users interact with the system and how data is processed. This diagram is essential for understanding the system's functionality and the pathways through which data moves.

Components:

- **User:**

The User is the primary external entity that interacts with the Campus Notes platform. Users can perform various actions, including logging in, uploading notes, and searching for existing notes. This interaction is crucial as it drives the functionality of the application.

- **Login/Upload/Search**

Module:

This central process module handles user requests. When a user attempts to log in, their credentials are validated against the database. If successful, the user gains access to the platform's features. The module also facilitates the upload of notes, where users can submit their documents along with relevant metadata.

Additionally, it processes search queries, allowing users to find notes based on specific criteria such as subject or keywords.

- **Notes**

Module:

The Notes Module is responsible for managing the business logic related to

notes. It interacts with the database to store new uploads, retrieve existing notes, and categorize them according to subjects and semesters. This module ensures that the notes are organized and accessible to users, enhancing the overall user experience.

- **Database:**

The Database serves as the persistent storage for all data related to users, notes, subjects, and semesters. It is where all uploaded notes are stored, along with user information and metadata. The database ensures data integrity and security, allowing for efficient retrieval and management of information.

Flow Explanation:

1. **User**

Interaction:

Users initiate interactions by sending their login credentials to the Login/Upload/Search Module. This module processes the request and checks the credentials against the database.

2. **Authentication:**

Upon successful authentication, users can access the platform's features. They can upload notes by providing the necessary information, including the file and its associated metadata.

3. **Note**

Management:

When a user uploads a note, the Notes Module processes the upload, storing the note in the database and linking it to the appropriate subject and user. This ensures that the note is categorized correctly and can be retrieved later.

4. Search

Functionality:

Users can also perform search operations to find specific notes. The search queries are processed by the Notes Module, which retrieves relevant notes from the database based on the user's criteria.

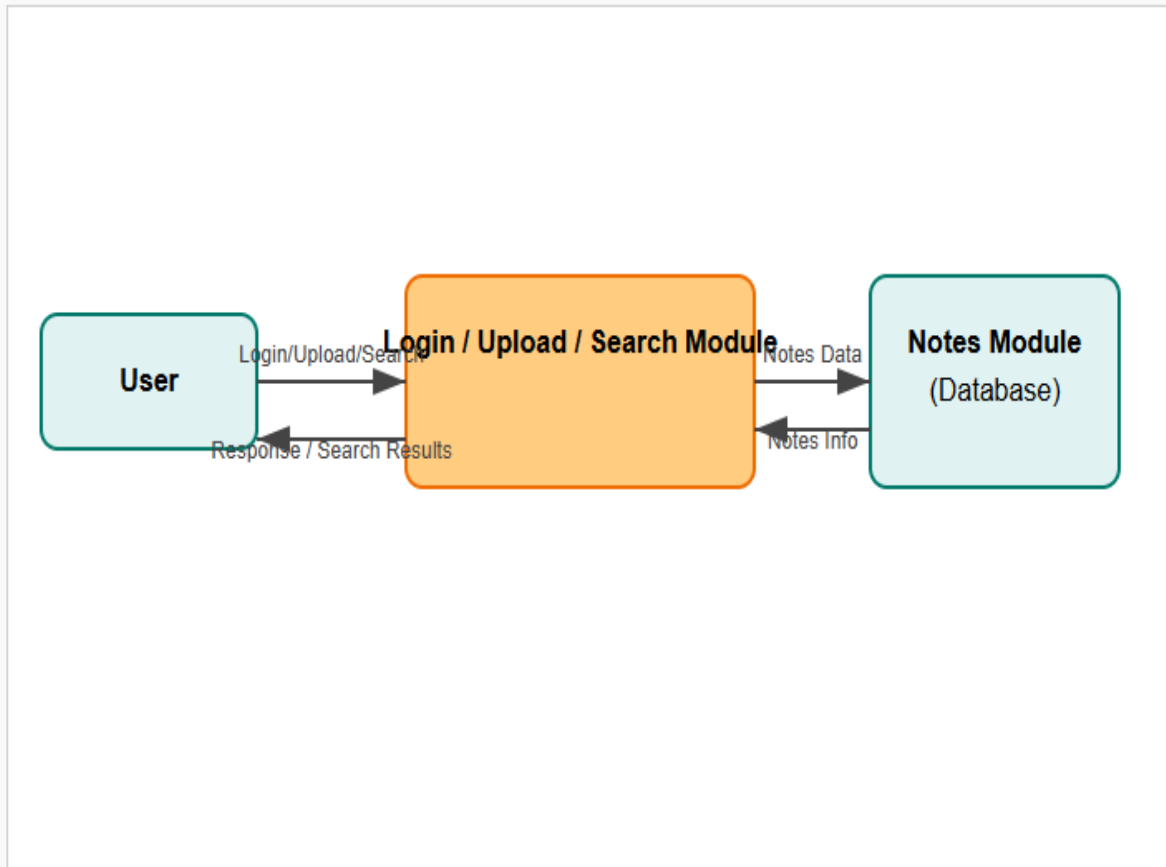
5. Feedback

Loop:

After processing user requests, the system returns results, such as confirmation of successful uploads or search results, back to the users. This feedback loop is essential for maintaining user engagement and satisfaction.

The Level 0 DFD provides a clear and concise overview of the system's functionality, helping stakeholders understand how data flows through the application and how user interactions are managed.

Campus Notes - Level 0 Data Flow Diagram



DATA DICTIONARY

To ensure a clear understanding of the system's structure and the role each data element plays within the *Campus Notes* platform, a data dictionary is provided below. The data dictionary defines each table in the database along with its respective fields, data types, and descriptions. This structured overview is essential for both database design and future system maintenance. It also serves as a foundational reference for developers, analysts, and stakeholders involved in the development and scaling of the platform. The tables outlined include core entities such as users, notes, subjects, and download history—each designed to support the platform's objective of organized, secure, and scalable academic content sharing.

The data dictionary serves as a critical component in the design and documentation of the *Campus Notes* platform. It provides a detailed description of the database structure, including the tables, fields, data types, and relationships that define how data is stored, accessed, and managed within the system. The goal of this platform is to facilitate seamless sharing of academic notes, and for that, a well-structured and normalized database is essential. The data dictionary not only helps in understanding the underlying data architecture but also supports developers, database administrators, and system analysts during development, testing, and future enhancements of the application.

Each table in the data dictionary represents a key entity within the system. For example, the User table manages data related to students and administrators who interact with the platform. The Notes table holds information about each uploaded academic note, including its metadata and file storage path. The Subject table helps categorize notes according to academic disciplines and semesters, allowing

for organized and efficient retrieval of content. Furthermore, the Downloads table captures download history, which can be used for usage analytics, tracking, and improving content relevance.

By outlining these tables and their attributes clearly, the data dictionary ensures transparency in the system design and provides a blueprint for implementing features such as search, filtering, permission control, and content analytics. It also supports scalability by offering a well-defined schema that can be expanded in the future to include additional functionalities like user ratings, AI-based recommendations, and collaborative study tools. In essence, the data dictionary acts as both a technical reference and a planning tool, aligning the database design with the overall objectives of the Campus Notes project.

This documentation serves multiple purposes:

- **Design Reference:** Assists in effective database planning and schema development.
- **System Maintenance:** Facilitates future updates and troubleshooting.
- **Collaboration Tool:** Acts as a shared reference for developers, data analysts, and project stakeholders.

The primary tables covered include key entities that support Campus Notes' mission of secure, organized, and scalable academic content sharing. These entities include:

- **Users:** Represents individuals registered on the platform.
- **Notes:** Stores uploaded academic materials.

- **Subjects:** Categorizes notes based on academic disciplines or courses.
- **Download History:** Tracks user interaction with downloadable content.

Each table is designed to ensure data integrity, streamline performance, and support future feature expansion.

Data Dictionary Overview

To ensure a clear understanding of the system's structure and the role data element plays within the Campus Notes platform, a data dictionary is provided below.

Users

Field	Data Type	Description
user_id	INTEGER	Primary key of the user
email	TEXT	Email address of the user
password	TEXT	Hashed password of the user
name	TEXT	Full name of the user

Subjects

Field	Data Type	Description
subject_id	INTEGER	Primary key of the subject
name	TEXT	Name of the subject
code	TEXT	Subject code

Notes

Field	Data Type	Description
note_id	INTEGER	Primary key of the note
user_id	INTEGER	Foreign key referencing users
subject_id	INTEGER	Foreign key referencing subjects
title	TEXT	Title of the note
content	TEXT	Content of the note

Download History

Field	Data Type	Description
download_id	INTEGER	Foreign key of referencing users
user_id	INTEGER	Foreign key of
download_date	DATETIME	Date and time of the download

DESIGN DOCUMENT

The design of the **Campus Notes platform** is built on a modular and scalable architecture that emphasizes maintainability, security, and user-centric functionality. Each component of the system has been carefully structured to ensure that the platform can efficiently manage academic content sharing while accommodating future enhancements. The core elements of the design include modularization, database structure, data integrity mechanisms, procedural workflows, and user interface design—all integrated seamlessly to provide a reliable and engaging user experience.

1. Modularization Details:

The architecture of the platform follows a modular approach, ensuring that each component handles a specific set of responsibilities without unnecessary coupling to others. This separation of concerns enhances code reusability, simplifies debugging, and accelerates development cycles.

□ Authentication Module:

This module is responsible for managing all user identity functions such as login, signup, and password encryption. It ensures that only authenticated users can access protected parts of the system. Additionally, it handles session management, secure storage of credentials, and validation mechanisms to prevent

unauthorized access.

□ **Notes_Module**

This is the heart of the platform where users can upload academic notes in PDF format, categorize them under specific subjects, and make them available for download. It supports metadata handling (like titles and descriptions), file storage, and version control if needed in the future.

□ **Admin_Module**

Designed for administrators, this module includes tools to monitor uploaded content, approve or reject notes before they go public, and remove content that violates guidelines. It ensures quality control and prevents the distribution of incorrect or inappropriate material.

□ **UI_Module**

This module governs the front-end interface and interaction flow. It is responsible for rendering views, managing navigation, handling user inputs, and providing responsive behavior across different devices. The UI module

communicates with back-end APIs to display dynamic data.

Data Integrity & Constraints:

Ensuring the correctness, validity, and security of the stored data is a top priority.

The following constraints are applied across the database and application logic:

- **Foreign** **Keys**
These enforce referential integrity, ensuring that every note is associated with an existing user and subject. Any deletion or update to parent records is handled gracefully to avoid orphaned data.
- **File** **Type** **Validation**
Only .pdf files are allowed for upload. This constraint simplifies file handling, previewing, and storage, while also promoting a standard format for academic content.
- **Unique** **Email** **Constraint**
To prevent duplicate user accounts and ensure each user is uniquely identifiable, the system enforces a uniqueness constraint on the email field in the user table.
- **Input** **Sanitization**
All inputs are validated and sanitized to prevent common security issues such as SQL injection, cross-site scripting (XSS), and malformed file uploads.

2.Database Design:

The platform's back-end is powered by a relational database schema, chosen for its ability to model structured relationships and enforce consistency. It includes the following key tables:

- User**

Table

Fields: user_id (primary key), name, email, password This table holds essential user information, ensuring secure and unique identity for each individual.
- Subject**

Table

Fields: subject_id (primary key), name, semester Each subject acts as a category under which notes are organized. The inclusion of semester data allows for contextual filtering.
- Note**

Table

Fields: note_id (primary key), title, file, user_id, subject_id This table captures metadata about the uploaded notes, including the relationship to the uploading user and subject category.

The database design also integrates **foreign key constraints** to preserve relationships between users, notes, and subjects, enabling efficient joins and queries across modules.

3.Procedural Design:

This section outlines the sequence of operations that define how the system processes major functions. These workflows are designed to be intuitive, efficient, and secure.

- Uploading**

Notes

When a user uploads a note, the system first validates the file type and input fields (title, subject). If validation passes, metadata is inserted into the database and the PDF file is stored in the designated file system or cloud storage. The upload is marked as pending until an admin reviews and approves it.
- Searching**

Notes

Users can browse or search for notes by filtering through academic subjects and semesters. The backend performs filtered queries and returns matching entries, which are then displayed in the user's interface with options to preview or download.
- Admin**

Approval

Workflow:

Notes uploaded by users enter a moderation queue. Admins access a dashboard that lists pending uploads. They can review the content, approve it (making it visible to all users), or reject it (with optional feedback to the uploader). This process ensures quality control and academic relevance of

shared content.

3.UI Design:

UI Design

The user interface is designed to be simple, responsive, and task-oriented, making it easy for both students and admins to navigate and complete actions efficiently.

- **Dashboard**

Layout

The main interface includes a tabbed navigation system, allowing users to switch between uploaded notes, approved notes, downloads, and their profile. Each tab is clearly labeled and provides easy access to the relevant actions.

- **File**

Preview

Feature

Before downloading, users can preview the content of a note using an embedded PDF viewer. This feature helps users confirm the usefulness of a note before saving it.

- **Real-time**

Notifications

The system provides feedback through notifications for major actions like successful uploads, admin approval status, or file errors. These alerts improve transparency and guide the user throughout their interactions with the platform.

PROGRAM CODE

```
import os
```

```
from django.core.asgi import get_asgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE',  
'Campusnotes.settings')
```

```
application = get_asgi_application()
```

```
asgi.py
```

```
from pathlib import Path
```

```
BASE_DIR= Path(_file_).resolve().parent.parent
```

```
import os
```

```
MEDIA_URL = '/media/'
```

```
MEDIA_ROOT = BASE_DIR / 'media'
```


Build paths inside the project like this: BASE_DIR / 'subdir'.

```
BASE_DIR = Path(_file_).resolve().parent.parent
```

Quick-start development settings - unsuitable for production

See

<https://docs.djangoproject.com/en/5.2/howto/deployment/checklist/>

SECURITY WARNING: keep the secret key used in production
secret!

```
SECRET_KEY = 'django-insecure-u$0zuih0vi(s=1cq&ut)bc&*=4ctp-  
-)8!#!u+1q7qax7u+)'
```

SECURITY WARNING: don't run with debug turned on in
production!

```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [
```

```
    'django.contrib.admin',
```

```
    'django.contrib.auth',
```

```
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
```

```
    'notes',
```

```
]
```

```
MIDDLEWARE = [
```

```
'django.middleware.security.SecurityMiddleware',  
  
'django.contrib.sessions.middleware.SessionMiddleware',  
  
'django.middleware.common.CommonMiddleware',  
  
'django.middleware.csrf.CsrfViewMiddleware',  
  
'django.contrib.auth.middleware.AuthenticationMiddleware',  
  
'django.contrib.messages.middleware.MessageMiddleware',  
  
'django.middleware.clickjacking.XFrameOptionsMiddleware',  
  
]
```

```
ROOT_URLCONF = 'Campusnotes.urls'
```

```
TEMPLATES = [  
  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
  
        'DIRS': [],  
  
        'APP_DIRS': True,
```

```
'OPTIONS': {  
  
    'context_processors': [  
  
        'django.template.context_processors.request',  
  
        'django.contrib.auth.context_processors.auth',  
  
        'django.contrib.messages.context_processors.messages',  
  
    ],  
  
    },  
  
    },  
  
]
```

```
WSGI_APPLICATION = 'Campusnotes.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/5.2/ref/settings/#databases
```

```
DATABASES = {
```

```
'default': {  
  
    'ENGINE': 'django.db.backends.sqlite3',  
  
    'NAME': BASE_DIR / 'db.sqlite3',  
  
}  
  
}
```

Password validation

<https://docs.djangoproject.com/en/5.2/ref/settings/#auth-password-validators>

AUTH_PASSWORD_VALIDATORS = [

```
{  
  
    'NAME':  
    'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',  
  
    },  
  
    {
```

```

        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',

    },

    {

        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',

    },

    {

        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },

]

```

Internationalization

<https://docs.djangoproject.com/en/5.2/topics/i18n/>

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/5.2/howto/static-files/
```

```
STATIC_URL = 'static/'
```

```
# Default primary key field type
```

```
# https://docs.djangoproject.com/en/5.2/ref/settings/#default-auto-field
```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

```
LOGIN_URL = 'login'
```

```
LOGIN_REDIRECT_URL = 'home'
```

```
settings.py
```

```
from django.contrib import admin
```

```
from django.urls import path, include
```

```
from django.conf import settings
```

```
from django.conf.urls.static import static
```

```
urlpatterns = [
```

```
    path('admin/', admin.site.urls),
```

```
    path("", include('notes.urls')),
```

```
]
```

```
if settings.DEBUG:
```

```
    urlpatterns += static(settings.MEDIA_URL,
```



```
document_root=settings.MEDIA_ROOT)
```

```
url.py
```

```
import os
```

```
from django.core.wsgi import get_wsgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'Campusnotes.settings')
```

```
application = get_wsgi_application()
```

```
qagi.py
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<h1>Campus Notes</h1>
```

```
<title>Campus Notes - Home</title>
```

```
<link
```

```
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.mi
```

```
n.css" rel="stylesheet">
```

```
</head>
```

```
<body class="container mt-5">
```

```
<h1 class="mb-4">Uploaded Notes</h1>
```

```
<a href="{ % url 'upload_note' % }" class="btn btn-primary mb-3">Upload New Note</a>
```

```
<a href="{ % url 'logout' % }" class="btn btn-danger mb-3 float-end">Logout</a>
```

```
<form method="get" class="mb-4">
```

```
<label for="subject" class="form-label">Filter by Subject:</label>
```

```
<select name="subject" id="subject" class="form-select"
onchange="this.form.submit()">
```

```
<option value="">All Subjects</option>
```

```
<option value="Math" { % if request.GET.subject == "Math"
% }selected{ % endif % }>Math</option>
```

```
<option value="Physics" { % if request.GET.subject == "Physics"
% }selected{ % endif % }>Physics</option>
```

```
<option value="Chemistry" {% if request.GET.subject ==  
"Chemistry" %}selected{% endif %}>Chemistry</option>
```

```
<!-- Add your subjects here -->
```

```
</select>  
</form>
```

```
{% for note in notes %}
```

```
<div class="card mb-3">
```

```
<div class="card-body">
```

```
<h5 class="card-title">{{ note.title }}</h5>
```

```
<p class="card-text">Subject: {{ note.subject }}</p>
```

```
<p class="card-text">Uploaded: {{ note.uploaded_at|date:"M  
d, Y H:i" }}</p>
```

```
<p class="card-text">Downloads: {{ note.download_count  
}}</p>
```

```
<a href="{% url 'download_note' note.id %}" class="btn btn-  
outline-success">Download</a>
```

</div>

</div>

{ % empty % }

<p>No notes uploaded yet.</p>

{ % endfor % }

</body>

</html>

home.html

<!DOCTYPE html>

<html>

<head>

<h1>CAMPUS NOTES</h1>

<title>Login</title>

<link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

```
<body class="container mt-5">
```

```
<h2 class="mb-4">Login</h2>
```

```
<form method="post">
```

```
{% csrf_token %}
```

```
{{ form.as_p }}
```

```
<button type="submit" class="btn btn-primary">Login</button>
```

```
</form>
```

```
<p class="mt-3">Don't have an account?
```

```
<a href="{% url 'signup' %}">Sign up here</a>.
```

```
</p>
```

```
</body>
```

```
</html>
```

login.html

```
<!DOCTYPE html>
```

```

<html>
<head>

  <title>Sign Up</title>

  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.mi
n.css" rel="stylesheet">

</head>

<body class="container mt-5" style="max-width: 400px;">

  <h2 class="mb-4">Register</h2>

  <form method="post" novalidate>

    { % csrf_token % }

    { { form.as_p } }

    <button type="submit" class="btn btn-success w-100">Sign
Up</button>

  </form>
  <p class="mt-3 text-center">

```

Already have an account? Login

</p>

</body>

</html>

sugnup.html

<!DOCTYPE html>

<html>

<head>

<title>Upload Note</title>

<link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body class="container mt-5">

<h2 class="mb-4">Upload a New Note</h2>

<form method="post" enctype="multipart/form-data" class="mb-3">

```

{% csrf_token %}

{{ form.as_p }}

<button type="submit" class="btn btn-success">Upload</button>

<a href="{% url 'home' %}" class="btn btn-secondary ms-
2">Back</a>

</form>

</body>

</html>

upload_note.html

from django.contrib import admin

from .models import Subject, Note

admin.site.register(Subject)
admin.site.register(Note)

admin.py

```



```
from django.apps import AppConfig
```

```
class NotesConfig(AppConfig):
```

```
    default_auto_field = 'django.db.models.BigAutoField'
```

```
    name = 'notes'
```

```
apps.py
```

```
from django import forms
```

```
from .models import Note
```

```
class NoteForm(forms.ModelForm):
```

```
    class Meta:
```

```
        model = Note
```

```
        fields = ['title', 'subject', 'file']
```

```
forms.py
```

```
from django.db import models
```

```
from django.contrib.auth.models import User
```

```
from django.utils import timezone
```

```
class Subject(models.Model):
```

```
    name = models.CharField(max_length=100)
```

```
    def str(self):
```

```
        return self.name
```

```
class Note(models.Model):
```

```
    title = models.CharField(max_length=200)
```

```
    description = models.TextField(blank=True)
```

```
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
```

```
    uploaded_by = models.ForeignKey(User,  
on_delete=models.CASCADE)
```

```
file = models.FileField(upload_to='notes/')
```

```
upload_date = models.DateTimeField(auto_now_add=True)
```

```
def str(self):
```

```
    return self.title
```

```
class Note(models.Model):
```

```
    title = models.CharField(max_length=100)
```

```
    subject = models.CharField(max_length=50)
```

```
    file = models.FileField(upload_to='notes/')
```

```
    uploaded_at = models.DateTimeField(default=timezone.now)
```

```
    download_count = models.PositiveIntegerField(default=0)
```

```
def str(self):
```

```
    return self.title
```

```
models.py
```

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
```

```
    path("", views.home, name='home'),
```

```
    path('upload/', views.upload_note, name='upload_note'),
```

```
    path('signup/', views.signup_view, name='signup'),
```

```
    path('login/', views.login_view, name='login'),
```

```
    path('logout/', views.logout_view, name='logout'),
```

```
    path('download/<int:note_id>', views.download_note,  
name='download_note'),
```

```
]
```

```
urls.py
```

```
from django.shortcuts import render, redirect
```

```
from .models import Note, Subject
```

```
from .forms import NoteForm
```

```
from django.contrib.auth.forms import UserCreationForm,
AuthenticationForm

from django.contrib.auth import login, authenticate, logout

from django.contrib.auth.decorators import login_required

from django.db.models import Q

from django.shortcuts import get_object_or_404

from django.http import FileResponse, HttpResponseRedirect
```

```
def home(request):

    notes = Note.objects.all()

    return render(request, 'notes/home.html', {'notes': notes})
@login_required

def upload_note(request):

    if request.method == 'POST':

        form = NoteForm(request.POST, request.FILES)
```

```
    if form.is_valid():

        note = form.save(commit=False)

        note.uploaded_by = request.user

        note.save()

        return redirect('home')

    else:

        form = NoteForm()

        return render(request, 'notes/upload_note.html', {'form': form})

def signup_view(request):

    if request.method == 'POST':

        form = UserCreationForm(request.POST)
        if form.is_valid():

            user = form.save()

            login(request, user)

            return redirect('home')

    else:
```

```

        form = UserCreationForm()

    return render(request, 'notes/signup.html', {'form': form})


def login_view(request):

    if request.method == 'POST':

        form = AuthenticationForm(data=request.POST)

        if form.is_valid():

            user = form.get_user()

            login(request, user)

            return redirect('home')

    else:

        form = AuthenticationForm()

    return render(request, 'notes/login.html', {'form': form})


def logout_view(request):

```

```
logout(request)
```

```
return redirect('login')
```

```
def home(request):
```

```
    subject = request.GET.get('subject')
```

```
    if subject:
```

```
        notes = Note.objects.filter(subject__icontains=subject)
```

```
    else:
```

```
        notes = Note.objects.all()
```

```
    return render(request, 'notes/home.html', {'notes': notes})
```

```
def download_note(request, note_id):
```

```
    note = get_object_or_404(Note, id=note_id)
```

```
    note.download_count += 1
```

```
    note.save()
```



```
return HttpResponseRedirect(note.file.url)
```

```
views.py
```

```
#!/usr/bin/env python
```

```
"""Django's command-line utility for administrative tasks."""
```

```
import os
```

```
import sys
```

```
def main():
```

```
    """Run administrative tasks."""
```

```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',  
'Campusnotes.settings')
```

```
    try:
```

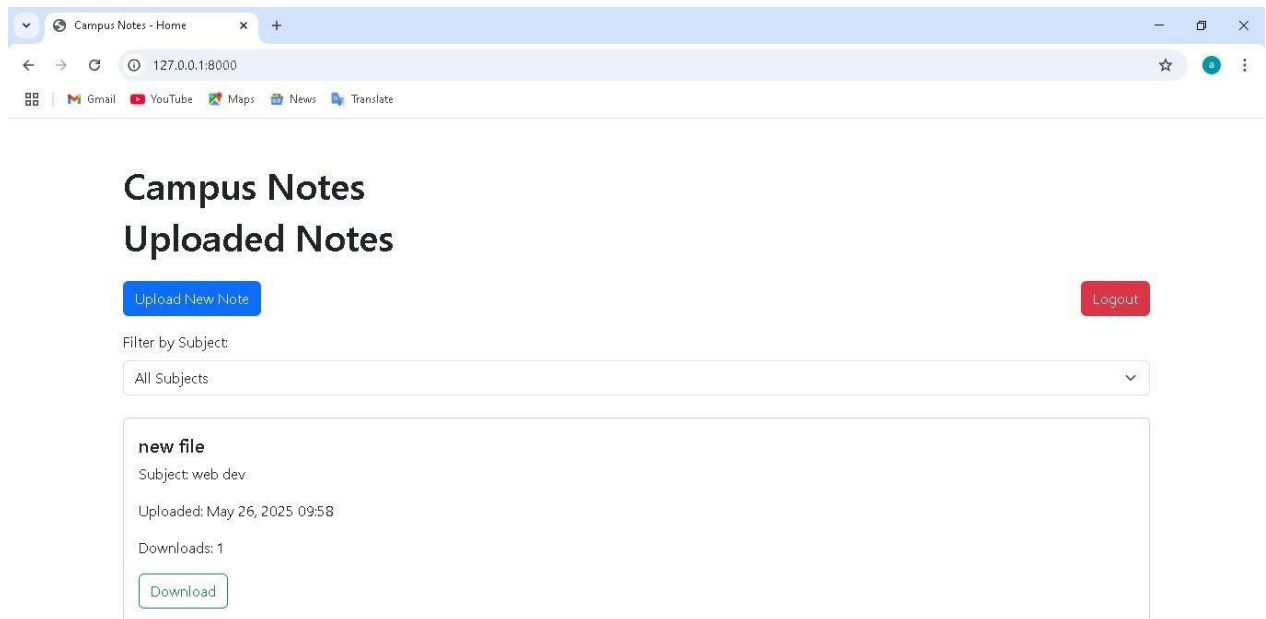
```
        from django.core.management import  
execute_from_command_line
```

```
    except ImportError as exc:
```

```
        raise ImportError(
```

```
"Couldn't import Django. Are you sure it's installed and "  
"available on your PYTHONPATH environment variable? Did  
you "  
"forget to activate a virtual environment?"  
  
) from exc  
  
execute_from_command_line(sys.argv)  
  
if __name__ == '__main__':  
  
    main()  
  
manage.py
```

Input and Output Screens / Results



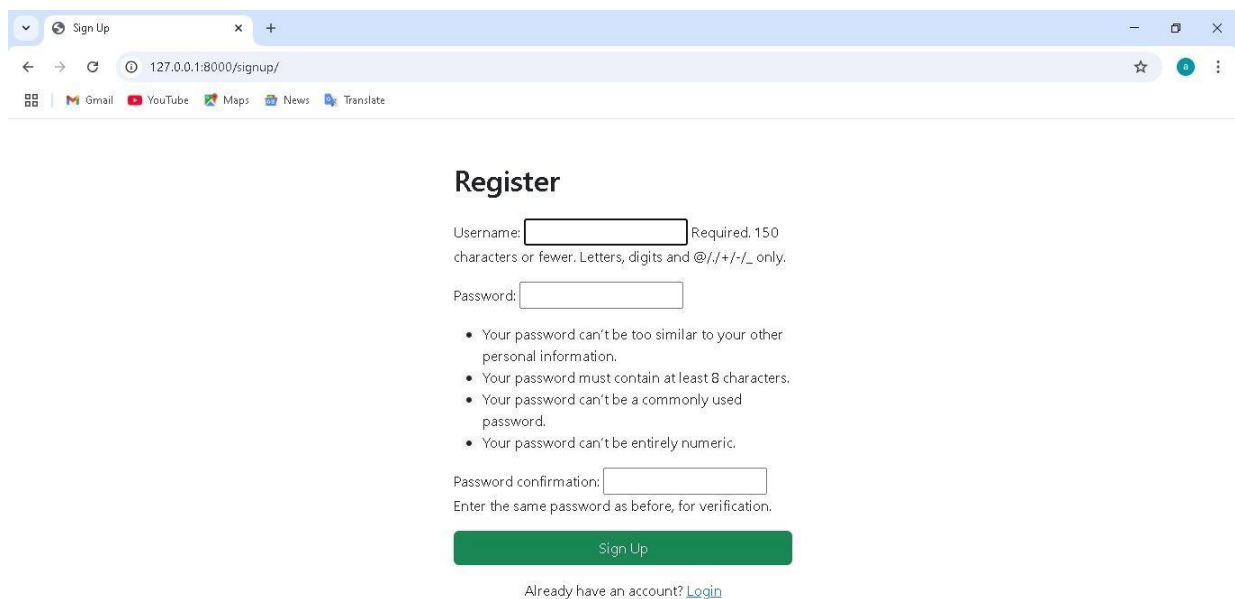
The screenshot shows a web browser window with the title "Campus Notes - Home". The address bar displays "127.0.0.1:8000". The page content includes a header with the title "Campus Notes" and a subtitle "Uploaded Notes". There are two buttons: "Upload New Note" (blue) and "Logout" (red). Below these is a "Filter by Subject:" dropdown menu set to "All Subjects". A card titled "new file" shows "Subject: web dev", "Uploaded: May 26, 2025 09:58", and "Downloads: 1", with a "Download" button.

Campus Notes
Uploaded Notes

[Upload New Note](#) [Logout](#)

Filter by Subject:
All Subjects

new file
Subject: web dev
Uploaded: May 26, 2025 09:58
Downloads: 1
[Download](#)



The screenshot shows a web browser window with the title "Sign Up". The address bar displays "127.0.0.1:8000/signup/". The page content includes a heading "Register" and a form with fields for "Username:" and "Password:". The "Username:" field has a note: "Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only." Below the "Password:" field is a list of password requirements. There is a "Password confirmation:" field with a note: "Enter the same password as before, for verification." At the bottom is a green "Sign Up" button and a link "Already have an account? [Login](#)".

Register

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:
Enter the same password as before, for verification.

[Sign Up](#)

Already have an account? [Login](#)



CAMPUS NOTES

Login

Username:

Password:

Login

Don't have an account? [Sign up here.](#)



Django administration

WELCOME, ADITYASHARMA. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#) [Change](#)

Users [+ Add](#) [Change](#)

NOTES

Notes [+ Add](#) [Change](#)

Subjects [+ Add](#) [Change](#)

Recent actions

My actions

[Note object \(2\)](#)

Note

[Note object \(3\)](#)

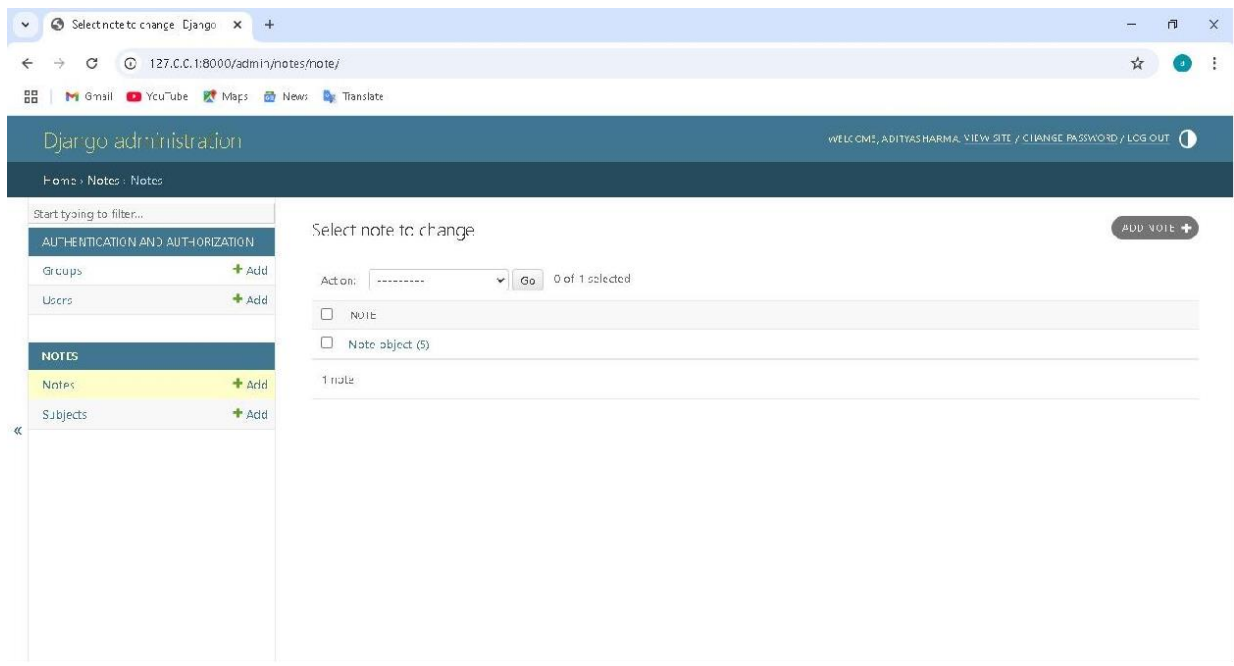
Note

[Note object \(4\)](#)

Note

[Note object \(1\)](#)

Note



DISCUSSION

Campus Notes effectively addresses the challenge of centralized academic note sharing in a digital format. By enabling students to upload and access notes efficiently, it fosters peer collaboration and reduces dependency on physical notes. Implementing secure authentication and admin moderation helped maintain content quality and prevent misuse.

The Campus Notes platform successfully addresses the long-standing challenge of providing a centralized, accessible, and digital repository for academic note sharing. By empowering students to upload, browse, and download notes with ease, the platform fosters a spirit of peer-to-peer collaboration and reduces the reliance on physical, informal methods of sharing academic resources. This not only streamlines access to learning materials but also democratizes the availability of content for students from diverse academic backgrounds.

A significant factor in maintaining the platform's integrity has been the implementation of secure authentication mechanisms and a robust admin moderation system. These features ensure that only verified users can contribute content, and that all shared notes meet established quality and relevance standards. As a result, the system effectively mitigates risks associated with spam, duplication, or the distribution of inappropriate material—thus preserving the platform's educational focus.

The development process was greatly enhanced by leveraging Django's built-in capabilities, particularly in areas such as user authentication, URL routing, and admin dashboard customization. This allowed for rapid prototyping and deployment, while adhering to strong security principles such as hashed password

storage and CSRF protection.

From an architectural standpoint, the platform was designed with scalability in mind. While the initial deployment uses a standard relational database (e.g., SQLite or PostgreSQL), there are plans to transition to more scalable and distributed database solutions as usage grows. Future enhancements will include interactive features such as note rating systems, comment sections, and AI-powered content recommendations, which will help personalize the user experience and surface high-quality content more efficiently.

Feedback collected during the beta testing phase was overwhelmingly positive. Test users appreciated the platform's intuitive interface, fast access to relevant notes, and the ability to preview content before downloading. These responses validated the core design choices and highlighted opportunities for further enhancements, such as improved search filters and mobile responsiveness.

The use of Django's built-in features accelerated development and ensured security. Scalability remains a focus, with future plans for migrating to more robust database systems and adding new features like note rating, comments, and AI-driven recommendations.

User feedback during beta testing was positive, highlighting improved access and usability. Overall, Campus Notes contributes to digital transformation in education by providing an intuitive platform that supports academic success and knowledge sharing.

BIBLIOGRAPHY

1. Django Documentation –

<https://docs.djangoproject.com>

2. W3Schools – <https://www.w3schools.com>

3. MDN Web Docs –

<https://developer.mozilla.org>

4. Draw\io for Diagramming – <https://draw.io>

5. Bootstrap Documentation –

<https://getbootstrap.com>

6. SQLite Documentation –

<https://www.sqlite.org/docs.html>

7. Stack Overflow – <https://stackoverflow.com>

8. GitHub – <https://github.com>

9. Lucidchart – <https://lucidchart.com>

10. Python Official Docs – <https://docs.python.org>