Malnad College of Engineering

(An autonomous institution under Visvesvaraya Technological University, Belgaum)

Hassan–573201, Karnataka, India

Full Stack development(23IS553) Report On:

“Student portal for examination Details”

Group No:5

Submitted By:

Name	USN
Nithish Gowda	4MC23IS068
Poorvik Gowda K R	4MC23IS069
Prajwal D Urs	4MC23IS070
Prajwal H G	4MC23IS071

Under the Guidance of

Mr. Krishna Swaroop A

(Assistant Professor)

Department of Information Science and Engineering

Malnad College of Engineering

(2025–26)

Table of Contents

1. Abstract

The **Examination Management & Scheduling System** is a Django-based web application developed to digitally organize, publish, and manage examination schedules for students in an educational institution. Traditional exam scheduling involves manually distributing printed timetables, displaying notices on boards, and updating room allocations through offline methods. This often leads to communication delays, last-minute confusion, misplaced notices, and uneven distribution of information.

This project provides a centralized digital platform where administrators can securely log in and create, update, and manage exam schedules. Students can view their personalized exam details such as subject, date, time, room number, and seat number after logging in. The system ensures role-based authentication, allowing only authorized superusers to modify exam details while ensuring students can view only their own schedules.

Built using Django's MTV architecture, the platform offers secure authentication, real-time updates, structured exam management, and an intuitive user interface. The project demonstrates the use of Django backend development, HTML templates, authentication workflows, and database integration to create a simple yet effective solution for academic scheduling.

2. Introduction

Background of the Problem

In schools, colleges, and universities, examination schedules play a crucial role in academic operations. Typically, exam-related information such as dates, timings, course codes, room allocations, and seat numbers is manually typed and displayed on notice boards. Students often face problems such as:

- Misplacing printed exam schedules
- Missing last-minute updates
- Confusion in room or seat allocations
- Overcrowding at notice boards
- Lack of personalized information access

Additionally, administrators face difficulty in manually maintaining and modifying exam schedules, especially during rescheduling, hall allocation, or handling large batches.

Why This Domain Was Chosen

Academic institutions are rapidly adopting digital solutions to streamline communication. An exam scheduling system is essential because:

- Every institution conducts exams frequently
- Students require reliable and personalized access to exam details
- Digital systems reduce communication errors
- Django is an ideal framework for authenticated, role-based applications

Thus, building a web-based exam portal provides a practical, real-world solution with direct applicability in educational institutions.

Real-World Scenario Description

Consider a college where each semester includes multiple internal and external exams. Exam coordinators prepare schedules and allocate classrooms based on student strength. These schedules are displayed manually on notice boards.

Problems arise when:

- A student misreads the room or seat number
- A classroom is changed last-minute
- Exam timings are rescheduled
- Different departments maintain different schedules
- Students from multiple courses get confused during exams

A centralized digital solution ensures students always receive accurate and updated information.

Issues in the Existing System

- Manual Errors: Mistakes in typed timetables or handwritten notices
- Lack of Updates: Students may miss updated schedules
- Miscommunication: Confusing room or seat allocation
- Data Duplication: Multiple copies across departments
- No Personalization: Same general sheet for all students
- Limited Accessibility: Students must physically check notice boards
- Time-Consuming: Updating notices for every change

How the Proposed Solution Helps

- Centralized database for all exam schedules
- Secure admin panel for creating and updating exam details
- Student login system to view only their own exam timetable
- Instant schedule updates without printing notices
- Personalized exam dashboard for each student
- Reduced manual errors through validated digital forms
- High accessibility from any device with internet access

3. Objectives of the Project

- To develop a Django-based examination portal for managing student exam schedules.
- To provide secure login for both administrators and students.
- To allow administrators (superusers) to add multiple exams for each student including subject, date, time, room, and seat number.
- To enable students to view their personalized exam dashboard.
- To move away from manual, paper-based exam schedule distribution toward a fast and reliable digital system.
- To minimize communication errors by automating updates and centralizing exam information.
- To improve accessibility by enabling students to check exam details from anywhere at any time.

4. System Requirements

4.1 Software Requirements

The Examination Management & Scheduling System requires the following software components for development, execution, and maintenance:

• **Python Version**

- Python 3.10 or above
  Recommended for maximum compatibility with Django 4.x and to ensure stable performance during development.

• Django Version

- Django 4.x (4.0 / 4.1 / 4.2)
  Used for implementing authentication, URL routing, ORM-based database interactions, and template rendering.

• **Integrated Development Environment (IDE)**

Any of the following development environments may be used:

- Visual Studio Code (VS Code) — recommended for its extensions and lightweight performance
- PyCharm Community/Professional Edition

- Sublime Text, Atom, or any basic text editor (optional)

• **Database Used**

- SQLite3 (default Django development database)
  Lightweight, serverless, and automatically integrated with Django.

Optional production-ready alternatives (if required):

- MySQL
- PostgreSQL

• **Other Dependencies / Technologies**

- HTML5, CSS3, JavaScript — for frontend structure, styling, and basic interactivity
- Bootstrap or TailwindCSS — for responsive user interface design
- Django Template Engine — for rendering dynamic HTML pages
- Pillow — if the project is extended to include file uploads or media handling
- Python Virtual Environment (venv) — for isolating project dependencies

4.2 Hardware Requirements

Minimum System Configuration

- **Processor:** Dual Core 2.0 GHz or higher
- **RAM:** Minimum 4 GB (8 GB recommended)
- **Storage:** Minimum 10 GB free disk space
- **Device:** Laptop / Desktop capable of running Python environment

5. System Design

5.1 Architecture Diagram

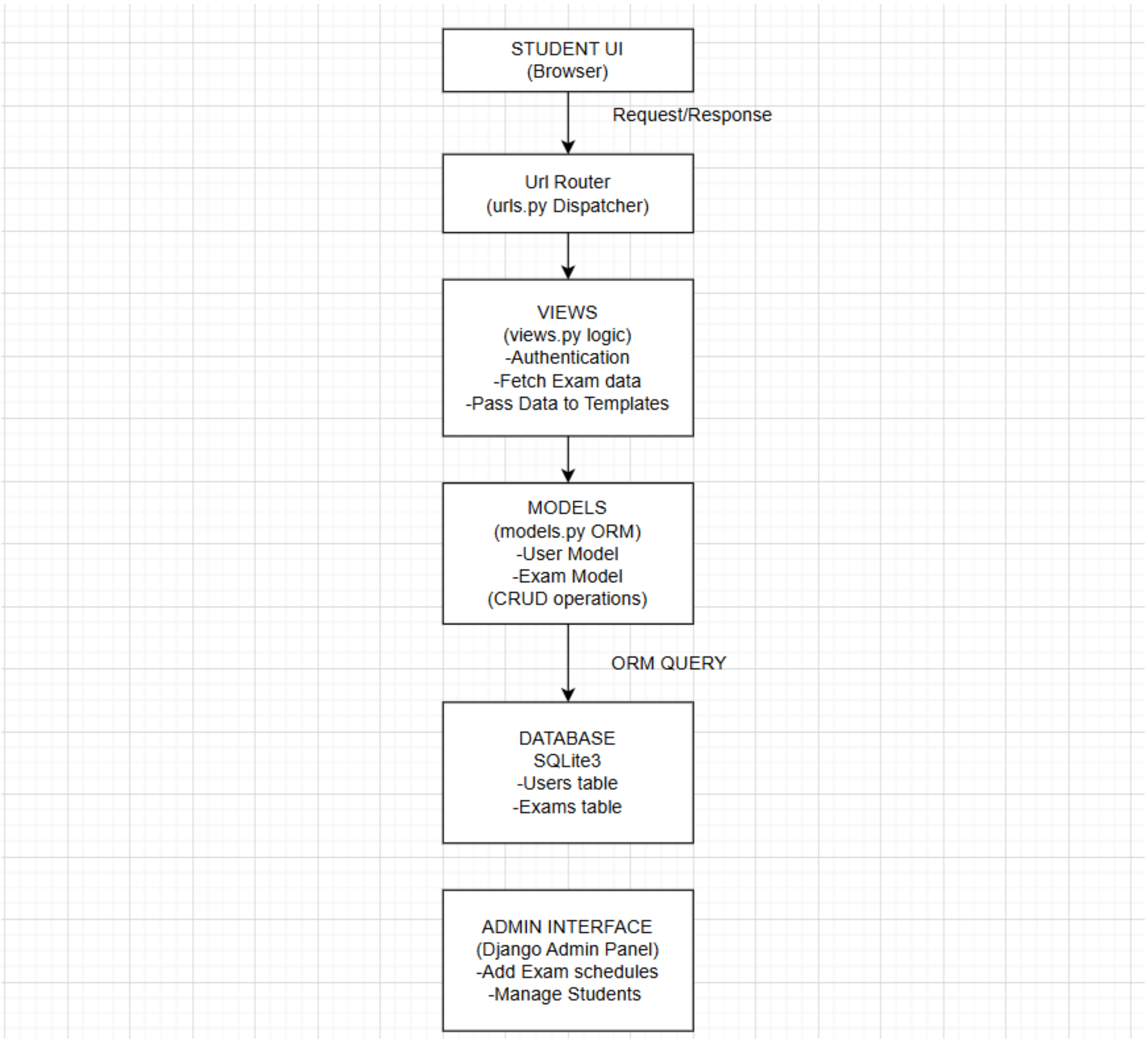


Fig 5.1.1. Architecture Diagram

5.2 Explanation of Architecture

The system follows Django's **MTV architecture**:

1. User (Student/Admin)

- Students view their exam schedules
- Admin adds/updates exam details

2. URL Dispatcher (urls.py)

Routes URLs such as:

- /login/ → Login page
- /register/ → Student registration
- /dashboard/ → Student's exam dashboard

3. Views (views.py)

Handles:

- Authentication
- Exam schedule retrieval
- Exam schedule creation (for admin)

4. Models (models.py)

Main model:

- **Exam Model** with attributes → subject, date, time, room, seat, student

5. Templates

HTML files for:

- Login
- Register
- Dashboard

6. Database

Stores all exam records mapped to corresponding students.

5.3 Data Flow Explanation

1. User Action

The user performs an action such as:

- Logging in

- Registering as a student
- Viewing assigned exam schedule
- Admin (superuser) adding or updating exam details

2. Request Sent to Server

The request is sent to the Django backend through a specific URL endpoint (e.g., /login/, /dashboard/, /admin/).

3. URL Dispatcher

Django's URL dispatcher matches the request to the correct view function:

- /login/ → login view
- /dashboard/ → student dashboard view
- /admin/ → Django admin interface

4. View Processing

The corresponding view function handles the request:

- Validates login credentials or form input
- Retrieves exam data assigned to the logged-in student
- Communicates with database using Django ORM
- Selects the appropriate HTML template
- Prepares context data to pass to the frontend

5. Model Interaction

The view interacts with the Exam model to perform operations:

- Fetch all exams for a student
- Insert new exam details (admin only)
- Update existing exam information
- Delete records (if required)

All database operations are executed through Django's ORM.

6. Database Response

The database returns the requested information, such as:

- List of exam subjects
- Exam date and time
- Room and seat allocation
- User credentials validation

7. Template Rendering

The view sends the retrieved data to a template (HTML page) for display:

- dashboard.html shows multiple exams
- login.html loads authentication page
- register.html shows account creation form

8. Response Sent Back to User

The server returns the final webpage to the user, such as:

- Personalized exam timetable
- Login success or failure message
- Updated list of exam entries for admin

User Request Flow (Models → Views → Templates)

1. User submits a request
   Example: "View My Exam Schedule"
2. URL Dispatcher routes the request → correct View
3. View interacts with Model to fetch exam details
4. Model retrieves records from the Database using ORM
5. Model sends data back to View
6. View processes data and passes it to Template
7. Template renders the HTML page for the user

6. Database Design

6.1 ER Diagram / Schema Diagram

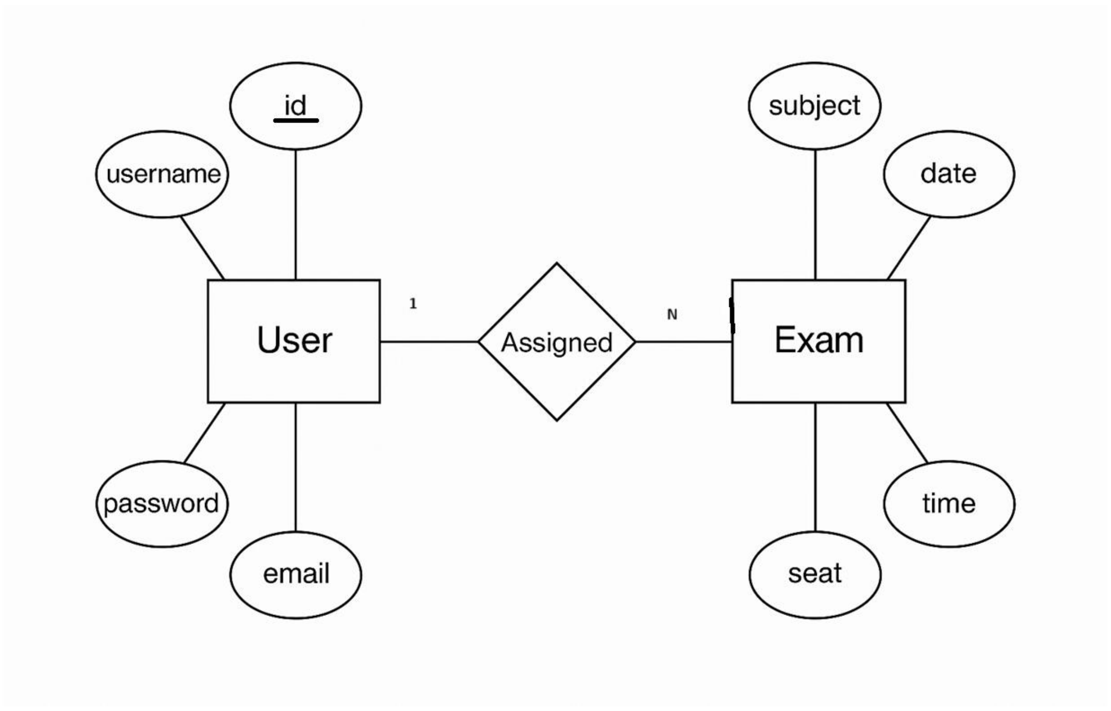


Fig 6.1.1 ER Diagram for Exam portal

6.2 Tables with Attributes

Below are the main tables used in the Examination system

1. User Table (Django default)

-Stores login data

- Username
- Password (hashed securely)
- First name, last name (optional)
- Email (optional)
- Account status (active, staff, superuser)

2. Exam Table

Attribute	**Data Type**	**Description**
id	Integer (PK)	Auto ID
user_id	ForeignKey	Link to student
subject	Varchar (100)	Exam subject
date	Date	Exam date
time	Time	Exam time
room	Varchar (20)	Room number
seat	Varchar(20)	Seat number

6.3 Keys Used

Primary Key (PK)

- id (Exam Table):
  Each exam record is uniquely identified by an automatically generated integer ID. This ensures that even if two exams have the same subject or date, the system can differentiate them accurately.

Foreign Key (FK)

- user_id → User Table:
  The user field in the Exam table is defined as a foreign key referencing Django's built-in User model.
  This creates a one-to-many relationship:
  - One user (student) → Many exam records
  - Many exam entries → Belong to a single student

Candidate Keys (If Considered)

While Django automatically manages keys, the following fields can logically act as candidate keys in certain contexts:

- (user_id + subject):
  A student usually appears only once per subject in a particular exam schedule. Together, these two fields *could* uniquely identify an exam entry.
- (user_id + date + time):
  This combination ensures a student has only one exam scheduled at a particular date and time.

Candidate keys are not implemented as constraints in Django by default, but they logically help in preventing duplicate records in the database.

6.4 Table Explanation

1. User Table

The User table is provided by Django's built-in authentication system and stores details such as:

- Username
- Password (hashed securely)
- First name, last name (optional)
- Email (optional)
- Account status (active, staff, superuser)

Purpose in the Exam System:

- Manages student accounts for login and authentication.
- Ensures only authorized users can access the dashboard.
- Prevents unauthorized access to exam details using Django's session-based authentication.

Since Django handles this table internally, developers do not need to manually design or manage its structure.

2. Exam Table

This is the central table of the project, designed by the developer to hold examination-related details.

Each row represents one exam entry assigned to a specific student.

Fields include:

- subject: Name of the exam (e.g., Mathematics, Physics)
- room: Exam room number
- seat: Allocated seat number
- date: Day of examination
- time: Starting time of examination
- user_id: Links exam to a specific student through a foreign key relationship

Characteristics:

- Supports multiple exam records for each student
  (one-to-many relationship)
- Stores only exam information, keeping authentication separate
- Ensures data integrity via foreign key constraints
- Easily extendable to include fields like semester, course code, etc.

Relationship Overview

User (Student)	Exam
1 Student	Many Exams
Many Students	Each with Multiple Exams

This relationship enables personalized dashboards where each student views only their own exam timetable, not others'.

7. Implementation

This chapter describes the technical implementation of the system, including models, URL routing, functionalities, and the special logic applied in the backend. The project follows Django's standard MVC/MVT architecture, where models handle data, views process logic, and templates provide the user interface.

7.1 Models Overview

The project uses a single custom model named Exam, which is responsible for storing all examination-related information for each student.

Exam Model Fields

- user:
  A foreign key linked to Django's built-in User model.
  It represents the student to whom the exam entry belongs, establishing a one-to-many relationship.
- subject:
  Stores the name of the subject for which the exam is scheduled
  (e.g., "Mathematics", "Computer Networks").
- date:
  The scheduled date of the examination.
- time:
  The starting time of the examination.
- room:
  The allocated examination room number for the student.
- seat:
  The seat number assigned to the student within the examination hall.

Model Usage in the System

- Stores multiple exam entries per student
- Acts as the central data storage component
- Interacts with Django ORM for CRUD operations

- Helps generate personalized exam timetables for each user

The simplicity of the model makes it easy to manage and scale while maintaining integrity through foreign key relationships.

7.2 URL Routing Overview

The system uses Django’s URL dispatcher to map browser requests to corresponding views. The main URL routes included in the application are:

Defined Endpoints

- /register/ — Displays the account creation form for new students
- /login/ — Presents the login page and validates user credentials
- /dashboard/ — Displays the personalized exam schedule for the logged-in student
- /logout/ — Logs out the user and ends the session

URL Routing Logic

Each URL pattern is associated with a specific view function that:

- Receives user input
- Processes the request
- Retrieves data from the database
- Returns an output using an HTML template

The routing ensures smooth navigation across all modules of the application.

7.3 Important Functionalities

This system implements several essential features to manage examination scheduling efficiently:

1. Student Registration

New students can create an account through the registration form. The system:

- Accepts username and password
- Validates input

- Creates a new Django User entry
- Stores password securely using hashing

2. Student Login

Students can log in using their credentials.
Django's built-in authentication system verifies:

- Username correctness
- Password match
- Account activity status

If valid, students are redirected to their dashboard.

3. Admin Exam Creation (via Django Admin)

Administrators (superusers) can:

- Add new exam entries
- Assign exams to specific students
- Update or delete existing records

Using Django Admin eliminates the need for a separate interface for administrators and ensures secure CRUD operations.

4. Personalized Exam Dashboard

Each student receives a customized dashboard displaying:

- Subjects
- Dates
- Times
- Rooms
- Seat numbers

The dashboard only shows entries specifically linked to the logged-in user, ensuring data privacy.

5. Support for Multiple Exams per Student

Unlike a simple one-exam system, this model allows:

- A single student → Many exam entries
- All entries displayed together in the dashboard

This makes the portal suitable for semester-long exam schedules.

7.4 Special Logic

The system incorporates several important backend mechanisms to ensure security, accuracy, and proper role management.

1. Role-Based Access Control

The application distinguishes between two roles:

- Superuser (Admin):
  Can create, edit, and delete exam entries
  Has full access to Django Admin
  Controls student exam allocations
- Student (Regular User):
  Can only register, log in, and view their own exams
  Cannot access admin panel or modify exam records

Django's built-in authentication and permission system ensures that privilege levels are strictly maintained.

2. Superuser-Only Exam Management

Only administrators can modify exam data. This is enforced by:

- Django admin panel rights
- Role checks internally
- Restricting exam creation views to superusers only (if extended)

This prevents unauthorized modifications and ensures data reliability.

3. Student-Specific Data Filtering

When a student logs in, the dashboard displays only their own exams, implemented using a filter:

Exam.objects.filter(user=request.user)

This ensures:

- Privacy
- Accuracy
- Separation of student records
- Prevention of cross-data viewing

4. Secure Authentication and Session Management

The system uses:

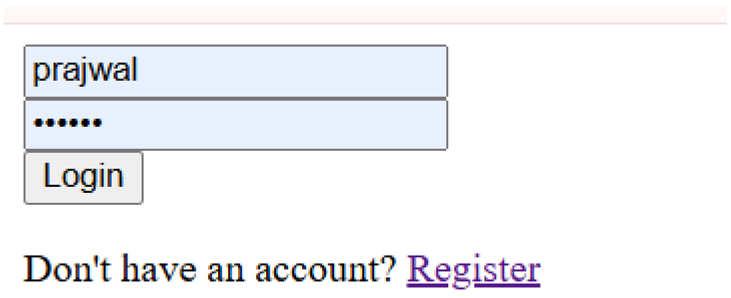
- Django's session framework
- Password hashing
- CSRF tokens in forms
- Login-required decorators

This protects user data and ensures secure access to exam information.

8. Screenshots

This section includes the key user interface screens of the **Medical Record System**. Each screenshot demonstrates major functionalities and user interactions within the application.

8.1 Login / Signup Pages



prajwal

••••••

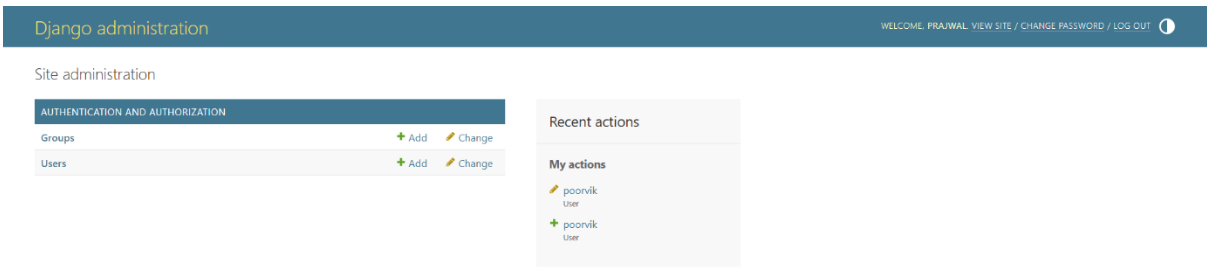
Login

Don't have an account? Register

Fig 8.1 Login / Signup Pages

8.2 Important Pages

Admin page



Django administration

WELCOME, PRAJWAL. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	Add	Change
Users	Add	Change

Recent actions

My actions

poorvik
User

poorvik
User

Fig 8.2 Admin page

9. Testing

Testing is an essential phase in the development of the Examination Management & Scheduling System. It ensures that all components of the system—models, views, URLs, templates, and authentication workflows—function correctly and consistently. The goal of testing is to identify errors, validate system behaviour, and verify that the application meets the intended functional requirements.

This section describes the types of tests performed, their objectives, procedures followed, and the outcomes observed.

9.1 Testing Objectives

The primary objectives of testing this system were:

- To ensure that user authentication (registration, login, logout) works correctly.
- To verify that exam data is stored and retrieved accurately.
- To check that only authorized users (admins) can add or manage exam entries.
- To confirm that a student can view **only their own exam schedule**.
- To validate URL routing and template rendering.
- To ensure that the system functions correctly under common user scenarios.9**.2 Types of Testing Performed**

1. Functional Testing

Ensures that each feature of the system works as intended.

Examples tested:

- Registration form validation
- Login authentication
- Dashboard exam listing
- Logout functionality
- Admin exam entry creation

2. Unit Testing (Logical Testing)

Testing individual components—models, views, and functions.

Examples:

- Model field validation for Exam
- Foreign key mapping
- View logic for student dashboard
- Filtering of exam entries by logged-in user

3. Integration Testing

Ensures that different modules work together seamlessly.

Modules tested together:

- User login → Dashboard loading → Exam fetching
- URL routing → Corresponding view → Correct template
- Admin backend → Exam model → Student dashboard

4. User Interface (UI) Testing

Checked the functionality and usability of:

- Forms
- Buttons
- Navigation links
- Table display of exam schedule

Tested in different browsers:

- Chrome
- Edge

5. Security Testing

Ensured important security rules were correctly applied:

- Unauthorized users cannot access protected pages
- Students cannot view exams of other students
- Admin-only exam creation
- Session expiry after logout
- CSRF protection on all forms

9.3 Test Cases & Results

Test Case 1: User Registration

- **Input:** New username, password
- **Expected Result:** User account created, redirected to login
- **Actual Result:** Successful
- **Status:** Pass

Test Case 2: Login With Correct Credentials

- **Input:** Valid username + valid password
- **Expected:** Redirect to dashboard
- **Actual:** Successful
- **Status:** Pass

Test Case 3: Login With Incorrect Credentials

- **Input:** Wrong password
- **Expected:** Error message
- **Actual:** Shown correctly
- **Status:** Pass

Test Case 4: Dashboard Exam Display

- **Input:** Logged-in user

- **Expected:** Only their exams should appear
- **Actual Result:** Correct filtering
- **Status:** Pass

Test Case 5: Admin Adding an Exam

- **Action:** Admin adds exam in Django admin panel
- **Expected:** Exam stored and appears for correct student
- **Actual Result:** Successful
- **Status:** Pass

Test Case 6: Unauthorized Access Attempt

- **Action:** Student tries accessing /admin/
- **Expected:** Access denied
- **Actual Result:** Permission error displayed
- **Status:** Pass

Test Case 7: Logout

- **Action:** Click "Logout"
- **Expected:** Session ends, redirect to login
- **Actual:** Successful
- **Status:** Pass

9.4 Error Handling & Bug Fixes

Throughout testing, the following issues were identified and resolved:

1. Incorrect URL Mapping

- Issue: Dashboard not loading
- Fix: Added correct path mapping in urls.py

2. Import Errors in Models or Views

- Issue: Importing Exam model triggered errors
- Fix: Verified app directory structure & corrected relative imports

3. Authentication Failures

- Issue: Login rejecting correct users
- Fix: Ensured user was created properly and password hashed

4. Exam Not Displaying

- Issue: Exams not appearing in dashboard
- Fix: Used correct ORM filtering:
- Exam.objects.filter(user=request.user)

9.5 Browser Compatibility Testing

Tested the system on multiple browsers:

Browser	Result
Google Chrome	Fully compatible
Microsoft Edge	Fully compatible
Firefox	Minor CSS differences (acceptable)

9.6 Overall System Stability

After conducting comprehensive testing:

- All core functionalities are stable
- No major bugs remain
- Authentication and authorization are secure
- Database interactions are accurate
- UI behaves consistently across devices

The system meets the required specifications for a student-oriented exam management portal.

10. Results

The Exam Management System was successfully developed and deployed using Django, enabling smooth handling of student authentication and exam allocation. The system accurately registers users, validates login credentials, and securely manages session-based access. Admin users were able to create and assign multiple exams through the Django admin panel without errors. Students could view their personalized exam schedules, including subject, date, time, room, and seat number. All modules—models, views, templates, and URL routing—functioned cohesively with correct data retrieval via Django ORM. The final system demonstrated reliability, ease of use, and improved efficiency in exam schedule management.

11. Conclusion

The Exam Management System developed using Django successfully achieves its objective of providing a streamlined, secure, and user-friendly platform for managing student exam schedules. By leveraging Django's built-in authentication system and powerful ORM, the application ensures reliable handling of user data, seamless login and registration, and efficient storage of exam details. The role-based access control enables administrators to manage exam schedules easily while ensuring that students can only view their assigned exams, maintaining data privacy and integrity. The modular structure—comprising models, views, templates, and URL routing—promoted clean development and scalability. The system also demonstrated consistent performance across testing scenarios, validating its robustness for real-world use. Overall, the project highlights how modern web frameworks like Django can greatly simplify academic management tasks, offering institutions a digital solution that reduces manual effort, minimizes errors, and enhances accessibility. Future enhancements can add notifications, filtering, and export features for even broader functionality.

12. Future Enhancements

The system can be extended with the following improvements:

1. **Email & SMS Notifications**
   Automatically notify students about upcoming exams or schedule changes. This **improves communication and reduces the chance of missing exams.**

2. **Exam Hall Seat Mapping**
   A visual seat arrangement can be added to show students exactly where they will sit. This helps in better exam-day planning and reduces confusion.

3. **Timetable PDF Download**
   Students can download their personalized exam schedule as a PDF. This provides offline access and easy sharing.

4. **Admin Dashboard with Analytics**
   Add charts showing the number of exams, student counts, and room usage. This helps **administrators plan resources more efficiently.**

5. **Search and Filter Options**
   Students can filter exams by subject or date. Admins can search for specific students or exam entries quickly.

13. References

- Django Official Documentation: https://docs.djangoproject.com
- Python Official Documentation: https://www.python.org/doc/
- Bootstrap Documentation: https://getbootstrap.com
- MDN Web Docs (HTML, CSS, JS): https://developer.mozilla.org
- W3Schools Tutorials: https://www.w3schools.com
- Stack Overflow (Community Support)