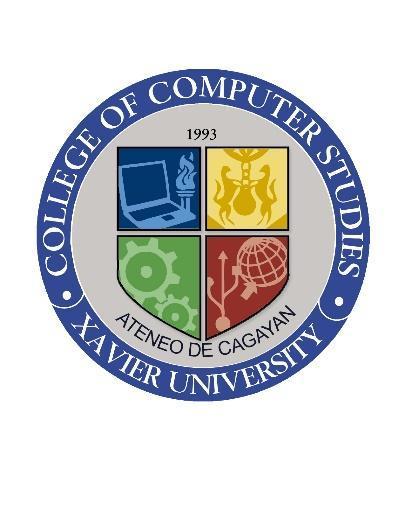
**XAVIER UNIVERSITY – ATENEO DE CAGAYAN**

**COLLEGE OF COMPUTER STUDIES**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**Final Project Documentation**

In Partial Fulfillment of the Requirement for the Course

ITCC 16.1 – Systems Administration And Maintenance

by:

Rubio, Emilio Rafael A.

Enguio, Nathaniel R.

Espiritu, Brett Rainiel

Hiponia, Karlos Miguel R.

May 14, 2025

**Introduction**

**Background of Client**

The Xavier University Athletics Office (UAO) plays a vital role in overseeing the scheduling and management of sports and event facilities on campus. These facilities, which include the gymnasium, covered court, field, and table tennis dug-out, are frequently used by students, athletes, guests, and university personnel for various academic, recreational, and athletic events.

At present, the reservation process is conducted manually. This traditional approach has led to several operational challenges, including double bookings, delays in processing, lack of centralized records, and inefficient communication between users and administrators. Such inefficiencies compromise the quality of service provided to stakeholders and increase the workload for both the Athletics Office staff and facility users.

In response to these challenges, the proposed Xavier University Athletics Office Reservation System aims to automate and streamline the reservation process through a secure, web-based platform. This system will serve as a centralized hub for managing reservations, checking real-time availability, and facilitating communication between users and administrators. By leveraging modern web technologies and adopting industry-standard development practices, the system aspires to enhance the efficiency, accuracy, and transparency of facility management at Xavier University.

The UAO Reservation System is designed to support the university’s mission of operational excellence by ensuring that resources are utilized optimally and user satisfaction is significantly improved. It also aligns with the university’s broader goal of embracing digital transformation across administrative processes.

**Scope and Limitation**

**Scope**

The Xavier University Athletics Office Reservation System encompasses the design, development, and deployment of a web-based application that automates the reservation of university sports and event facilities.

The main features and components of the system include:

* User Authentication: Implementation of a secure login system with role-based access control, differentiating between general users (students and guests), administrators, and superusers.
* Reservation Management: Allowing users to submit, update, and cancel reservation requests for designated facilities.
* Real-Time Availability: Displaying an interactive calendar showing current availability of facilities to prevent scheduling conflicts.
* Admin Controls: Providing administrators the tools to approve or reject reservations, manage scheduling, verify uploaded payment proofs, and issue security passes for outsider guests.
* Automated Notifications: Sending email or SMS alerts for booking confirmations, updates, and reminders.
* Technology Stack: Utilizing Django for backend development, SQLite as the database solution, and HTML, CSS, and JavaScript for frontend development.
* Cloud Deployment: Hosting the system on a cloud platform to ensure accessibility and scalability.

**Limitation**

While the system is comprehensive in scope, certain limitations are recognized in its initial implementation:

* Facility Coverage: The system is limited to managing reservations for the gymnasium, covered court, field, and table tennis dug-out only.
* Manual Payment Verification: Payment transactions are not processed online. Users are required to upload proof of payment, which must be verified manually by administrators.
* Limited External Integration: The system does not currently interface with other university systems such as the student portal, finance, or security management platforms.
* Security Pass Handling: The process for handling outsider guest access, including coordination with the university’s Security Office, remains partly manual.
* Data Architecture: SQLite is used for storing system data, which may require additional configuration if integration with relational systems or migration is needed in the future.
* Scope of Automation: While the system automates many processes, certain tasks such as the approval of documents or coordination with third-party offices remain dependent on human intervention.

**Planning Phase**

**Methodology used**

The methodology integrates enterprise architecture frameworks and structured development planning:

**Zachman Framework:**

* Used to identify stakeholders and perspectives across different domains: Planner, Owner, Designer, Builder, Subcontractor, and User.
* A 6x6 matrix addressed What, How, Where, Who, When, Why aspects of the system for different roles.

**Technology Stack:**

* Frontend: HTML, CSS, JavaScript
* Backend: Django, Django REST Framework
* Database: MongoDB
* Version Control: Git + GitHub
* IDE: Visual Studio Code

**Project Timeline**

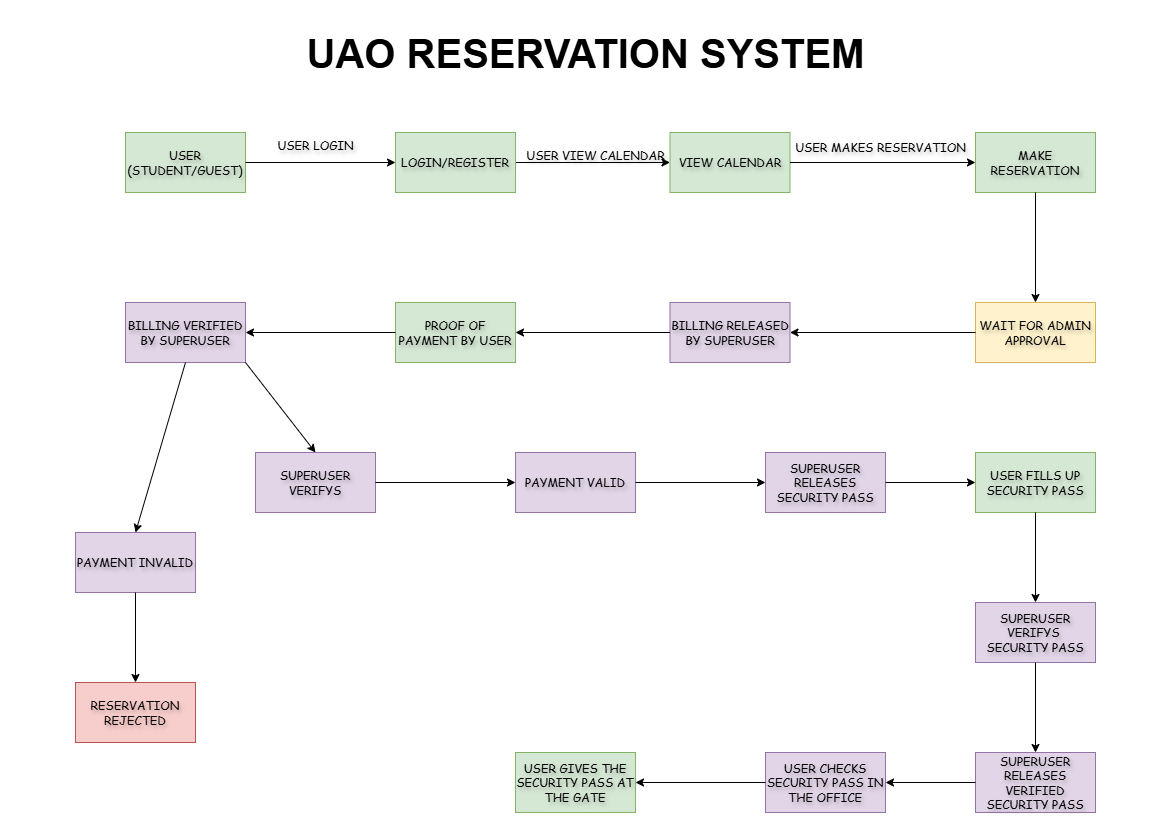
The project will follow a structured timeline to ensure organized and efficient development. During Week 1 to 2, the team will focus on Project Planning, establishing objectives, scope, and initial strategies. Week 3 to 4 will be dedicated to Requirements Gathering, where the team will identify and document user needs, system features, and functional requirements. In Week 5 to 6, the team will work on Mock-up Design, creating interface prototypes and visual layouts to guide frontend development. Week 7 to 10 will involve Frontend Development, where the user interface will be built using web technologies like HTML, CSS, and JavaScript. From Week 11 to 13, the focus will shift to Backend Development, implementing the system logic and database integration using Django and SQLite. Week 14 to 15 will cover Integration Testing, ensuring that all modules function correctly together and meet the system requirements. Finally, Week 16 to 17 will be allocated for Deployment.

**Roles & Responsibility**

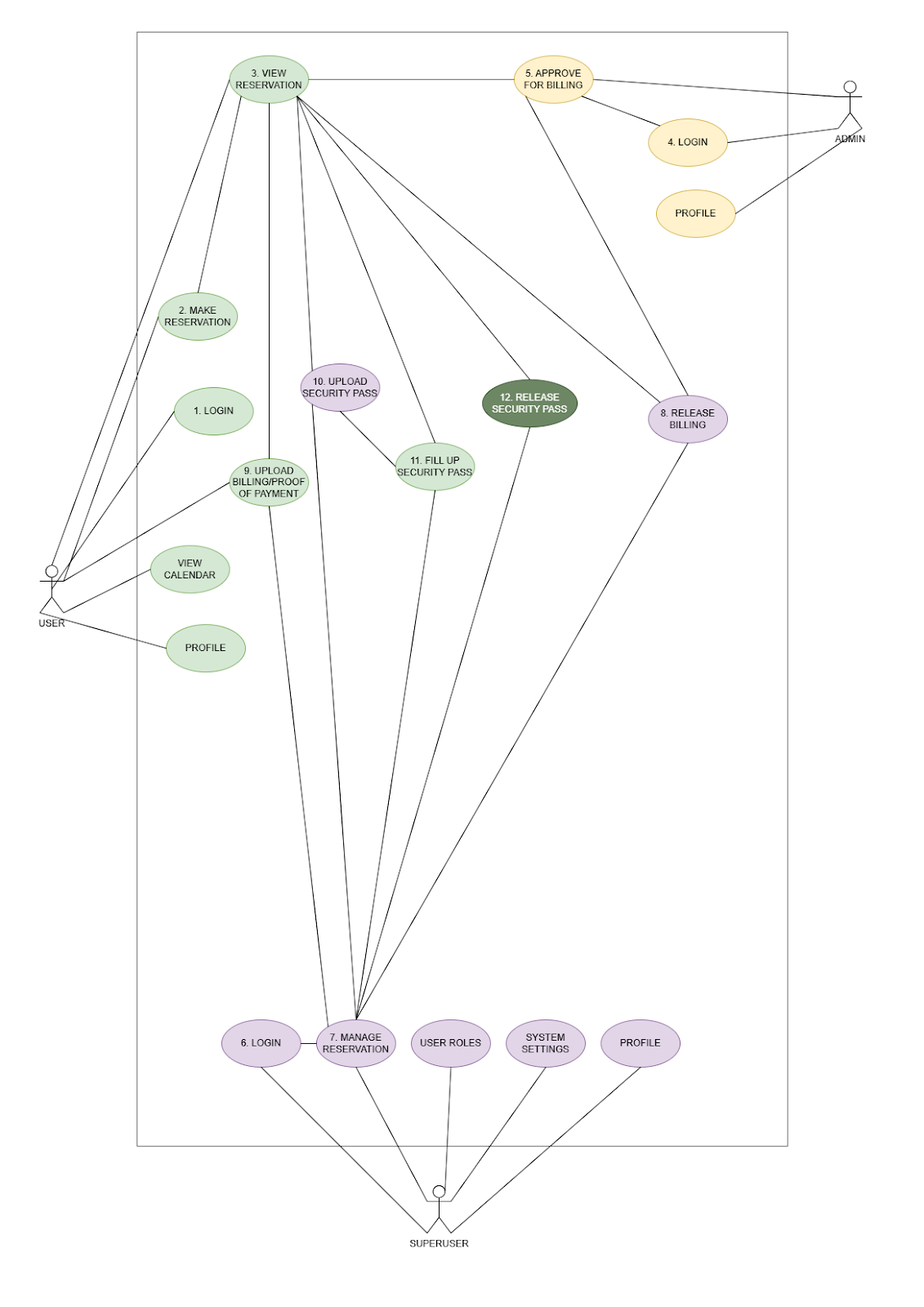
For the project, the primary function of each team member is full-stack development, with each member responsible for at least one integration module in the proposed system. Emilio Rafael A. Rubio is in charge of the Authentication Module, tasked with implementing a secure login system and role-based access control using Django’s built-in authentication framework. Brett Rainiel Espiritu is responsible for the Reservation Interface Module, focusing on the development of the interactive calendar and user interface for checking facility availability and submitting reservations. Nathaniel R. Enguio will handle the Scheduling Conflict Detection Module, developing the logic to prevent overlapping bookings through real-time validation and database queries. Karlos Miguel R. Hiponia is assigned the Notification Module, integrating automated email and SMS alerts for booking confirmations, updates, and reminders. In addition to their individual roles, all members will collaborate on testing, debugging, and system deployment to ensure the UAO Reservation System is reliable, efficient, and user-friendly.

**Analysis Phase**

**DFD**

****

**USE CASE**

****

**Infrastructure Setup**

**Frontend**

**Technologies: HTML, CSS, JavaScript**

* Login Interface:
  + - Role-based login: USER (Student/Guest), ADMIN, SUPERUSER
    - Basic session-based authentication with Django template rendering
* Forms & UI Components:
  + - Reservation Form (with status display: Pending, Approved, etc.)
    - Upload buttons for Billing Statement, Payment Receipt, and Security Pass Form
* Status progress bar or indicators for users
  + - Admin dashboard for approvals
    - Superuser interface for billing, receipt validation, and form review
* Feedback & Notifications:
  + - Online form validation messages
    - Success/error messages
    - Notification panel or alerts (simple banners or modal popups)

Backend

Framework: Django  
Database: SQLite

* User Management:
  + - Django User model extended with roles (Student, Admin, Superuser)
    - Login/Logout via Django Auth system
* Core Django Apps:
  + - reservations: Handles reservation submissions and status tracking
    - billing: Manages billing statement uploads and payment receipt verification
    - security\_pass: Handles downloadable pass form logic and file uploads
* Status Management Logic:  
   Implemented through Django views with conditional logic:
  + - Only allow next step if previous status is complete

**Functional and User Requirements**1. Submit reservation requests for sports facilities, including gymnasium, covered court, field, and table tennis dug-out with facility availability check.

a. Input: Facility name, date of use, start time, end time, organizer name, contact number, event type, number of participants (insiders and outsiders), additional equipment and manpower requested.

b. Output: Confirmation message and pending reservation record if the facility is available; error message if already booked.

2. Update or cancel submitted reservations before approval.

a. Input: Reservation ID, updated reservation details (date, time, facility, etc.) or cancellation request.

b. Output: Confirmation message with updated reservation record or cancellation notice.

3. View list of reserved facilities and reservation statuses (approved, pending,

rejected).

a. Input: User ID (to view user-specific reservations) or admin access credentials (to view all reservations).

b. Output:  List of reservation records with details and current statuses.

4. Approve or reject reservation requests by the administrator.

a. Input: Reservation ID, admin decision (approve/reject), optional admin remarks.

b. Output: Updated reservation status (approved/rejected) and notification sent to the user.

**Design and Development phase**

**Design Plan**

The design of the UAO Reservation System is based on a modular, layered architecture using enterprise architecture frameworks (Zachman and TOGAF). The design includes:

**1. Front-End Design**

* Developed using HTML, CSS, and JavaScript.
* Features a user-friendly UI for:
  + Viewing availability (interactive calendar).
  + Making reservations.
  + Admin access for reviewing and approving requests.

**2. Back-End Design**

* Powered by Django and Django REST Framework.
* Handles user authentication, booking logic, notification, and admin functions.
* Communicates with the database and manages API endpoints.

**System Specification**

1. Submit reservation requests for sports facilities, including gymnasium, covered court, field, and table tennis dug-out with facility availability check.

a. Input: Facility name, date of use, start time, end time, organizer name, contact number, event type, number of participants (insiders and outsiders), additional equipment and manpower requested.

b. Output: Confirmation message and pending reservation record if the facility is available; error message if already booked.

2. Update or cancel submitted reservations before approval.

a. Input: Reservation ID, updated reservation details (date, time, facility, etc.) or cancellation request.

b. Output: Confirmation message with updated reservation record or cancellation notice.

3. View list of reserved facilities and reservation statuses (approved, pending,

rejected).

a. Input: User ID (to view user-specific reservations) or admin access credentials (to view all reservations).

b. Output:  List of reservation records with details and current statuses.

4. Approve or reject reservation requests by the administrator.

a. Input: Reservation ID, admin decision (approve/reject), optional admin remarks.

b. Output: Updated reservation status (approved/rejected) and notification sent to the user.

**Program Layout**

|  |  |
| --- | --- |
| **Component** | **Description** |
| Facility User Interface (UI) | User-facing module for viewing and booking facilities |
| Admin Interface | Admin dashboard for managing and approving requests |
| Reservation Management Service | Core backend service that processes requests and enforces rules |
| Authentication Module | Handles login/logout and role-based access control |
| Scheduling Conflict Detection | Prevents double bookings using real-time checks |
| Notification System | Sends booking confirmations, reminders via email/SMS |
| Security Pass Issuance | For outsider access, approved by admin and chief security |
| Invoice Upload | Allows facility users to upload payment proofs |

**Testing And Evaluation Phase**

**Maintenance And Support Phase**

**Manual of Policies & Procedure**

**Purpose**

This manual outlines the official policies and procedures governing the operation, usage, and maintenance of the UAO Reservation System at Xavier University Ateneo de Cagayan.

**Scope**

**Applies to all users of the system including:**

* Students
* Guests
* Superusers
* Administrators

**Policies**

1. **Booking Policy**
   * Only authenticated users may book facilities.
   * All bookings require admin approval.
   * Reservations must be made at least 24 hours in advance.
2. **User Roles and Access**
   * **Students/Guests**: Can submit and view reservations.
   * **Administrators**: Can approve, reject, and manage all reservations.
   * **Superuser**: Handles everything.
3. **Data Handling**
   * All user and booking data must be securely stored and transmitted.
   * Regular data backups are mandatory.
4. **Change Management**
   * Any system update or change must be reviewed by the development and QA teams.
   * All changes must be logged and documented.
5. **Incident Reporting**
   * System bugs or booking conflicts must be reported immediately via the support portal.

**Maintenance Plan**

**System Overview**

The UAO Reservation System is a web-based system, which makes it easy for users particularly students and athletes to reserve facilities. Currently, booking is manual, and this may result in scheduling conflicts, delays, and additional work for the instructors. This project seeks to correct that by offering a simple, automated reservation system that enhances efficiency and enhances the overall experience. With a Django based system, having a simple to use interface, real time updates for bookings, and an approval mechanism for administrators. It uses Django, HTML, CSS, JavaScript.

**Maintenance Types and Tasks**

**Preventive Maintenance**

* Scheduled backups, system updates, and regular security scans to prevent future issues.

**Corrective Maintenance**

* Bug fixing and resolution of operational issues that arise during system use.

**Adaptive Maintenance**

* Adjustments to the system based on new regulatory requirements, technologies, or user devices.

**Perfective Maintenance**

* Refinements based on user feedback to enhance performance, usability, or appearance.

**Maintenance Schedule**

* Weekly: Backup and review logs
* Monthly: System performance monitoring and minor updates
* Quarterly: Evaluate user feedback and perform UI/UX enhancements
* Annually: Full system review, upgrade planning, and audit

**Tools and Resources**

* Visual Studio Code (IDE)
* Git & GitHub (Version Control)
* DB Browser

**Roles and Responsibilities**

* Emilio Rafael A. Rubio: Monitors uptime, security, and performance.
* Brett Rainiel Espiritu: Handles bug fixes, logic upgrades, and backend performance.
* Nathaniel R. Enguio: Updates UI/UX and ensures responsiveness.
* Karlos Miguel R. Hiponia: Conducts testing before and after updates.
* Franz Fortunado: Responds to user queries and logs issues.

**Risk Assessment and Mitigation**

**Potential risks include:**

* System Downtime: If maintenance is ignored, users may be unable to book facilities.
* Data Breaches: Delayed security updates can expose sensitive data.

**Mitigation strategies:**

* Implement regular patching schedules
* Maintain error logging and monitor system health
* Backup data routinely and test recovery plans

**Appendices**

**Sample checklists**

* Weekly backup completed?
* Latest patches installed?
* System logs reviewed?
* New issues resolved?

**Example log files**

**LOG\_2025\_05\_01:**

* Booking created: Field 2, 2PM-4PM
* Name: Rafi Rubio
* Admin Approval: Ready for Billing.
* Superuser: Security Pass released.

**Screenshots or script snippets**

@login\_required

def user\_makereservation(request):

today = timezone.now().date()

if request.method == 'POST':

# Debug: Print all POST data

print("\n=== Form Submission Data ===")

for key, value in request.POST.items():

print(f"{key}: {value}")

print("===========================\n")

organization = request.POST.get('organization')

representative = request.POST.get('representative')

contact\_number = request.POST.get('contact\_number')

date\_reserved = request.POST.get('date\_reserved')

insider\_count = request.POST.get('insider\_count', 0)

outsider\_count = request.POST.get('outsider\_count', 0)

date = request.POST.get('date')

start\_time = request.POST.get('start\_time')

end\_time = request.POST.get('end\_time')

reasons = request.POST.get('reasons', '').strip()

# Facility (single selection)

facility = request.POST.get('facility')

facility\_use = facility

# Event Types (checkboxes, allow multiple)

event\_types = request.POST.getlist('event\_type')

if not event\_types:

messages.error(request, 'Please select at least one event type')

return redirect('user\_portal:user\_makereservation')

event\_type = ', '.join(event\_types)

# Validate required fields

required\_fields = [facility, organization, representative, date, start\_time, end\_time, event\_type]

if not all(required\_fields):

messages.error(request, 'Please fill in all required fields.')

return redirect('user\_portal:user\_makereservation')

# Facilities Needed

facility\_keys = [

'long\_tables', 'mono\_block\_chairs', 'narra\_chairs', 'podium', 'xu\_seal', 'xu\_logo',

'sound\_system', 'bulletin\_board', 'scaffolding', 'flag', 'philippine\_flag', 'xu\_flag',

'ceiling\_fans', 'stand\_fans', 'iwata\_fans', 'stage\_non\_acrylic', 'digital\_clock',

'others'

]

facilities\_needed = {

key.replace('\_', ' ').title(): int(request.POST.get(f"{key}\_quantity", 0))

for key in facility\_keys

if request.POST.get(f"{key}\_quantity")

}

# Manpower Needed

manpower\_keys = [

'security', 'janitor', 'electrician', 'technician',

'assistant\_technician', 'digital\_clock\_operator', 'plumber', 'other\_manpower'

]

manpower\_needed = {

key.replace('\_', ' ').title(): int(request.POST.get(f"{key}\_quantity", 0))

for key in manpower\_keys

if request.POST.get(f"{key}\_quantity")

}

try:

# Debug: Print reservation data before creation

print("\n=== Creating Reservation ===")

print(f"User: {request.user.username}")

print(f"Date: {date}")

print(f"Start Time: {start\_time}")

print(f"End Time: {end\_time}")

print(f"Facility: {facility}")

print(f"Status: Pending")

print("==========================\n")

# Save the reservation

reservation = Reservation.objects.create(

user=request.user,

organization=organization,

representative=representative,

contact\_number=contact\_number,

date\_reserved=date\_reserved,

date=date,

insider\_count=insider\_count,

outsider\_count=outsider\_count,

start\_time=start\_time,

end\_time=end\_time,

reasons=reasons,

facility=facility,

facility\_use=facility\_use,

event\_type=event\_type,

facilities\_needed=facilities\_needed,

manpower\_needed=manpower\_needed,

status="Pending",

)

# Debug: Print success message

print(f"\n=== Reservation Created Successfully ===")

print(f"Reservation ID: {reservation.id}")

print("=====================================\n")

messages.success(request, 'Reservation created successfully!')

except Exception as e:

# Debug: Print error details

print(f"\n=== Error Creating Reservation ===")

print(f"Error: {str(e)}")

print("===============================\n")

messages.error(request, f'Error creating reservation: {str(e)}')

return redirect('user\_portal:user\_makereservation')

return redirect('user\_portal:user\_myreservation')

return render(request, 'user\_portal/user\_makereservation.html', {

'today': today,

})

**Any diagrams not placed earlier**