Research Portal Management System

A Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Technology

In

Computer Science And Engineering

Of

Tezpur University

2025

Submitted By

NABAJIT PAUL – CSB22034

Under The Guidance of

DR. JYOTISMITA TALUKDAR

ASSISTANT PROFESSOR

TEZPUR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

TEZPUR UNIVERSITY, TEZPUR, ASSAM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I **NABAJIT PAUL**, Roll Number **CSB22034**, a student of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** at **TEZPUR UNIVERSITY**, hereby declare that the project report entitled **"Research Project Management System"** is an original work carried out by me. The information and data presented in this report are true and authentic to the best of my knowledge. This work has not been submitted in part or full to any other University or Institute for the award of any degree or diploma.

DATE: 29/05/2025 **NABAJIT PAUL**

PLACE: TEZPUR UNIVERSITY CSB22034

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,

TEZPUR UNIVERSITY

2025

Certificate From the Project Guide

This is to certify that the project report entitled **"Research Project Management System"** submitted by **NABAJIT PAUL**, Roll No: **CSB22034** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** is a Bonafide record of the work carried out by him under my supervision and guidance. The work reported herein is original and has not been submitted, in part or full, to any other University or Institute for the award of any degree or diploma.

DATE:29/05/2025

PLACE: TEZPUR UNIVERSITY

DR. JYOTISMITA TALUKDAR

ASSISTANT PROFESSOR

PROJECT GUIDE

ABSTRACT

The management of research publications, associated data, and researcher profiles is a critical task in academic and research institutions. Traditional methods often involve manual tracking, disparate systems, and a lack of integration with authoritative researcher identification platforms like ORCID (Open Researcher and Contributor ID). This project, "Research Project Management System," aims to address these challenges by developing a web-based application that facilitates efficient management of research papers and integrates seamlessly with the ORCID public API.
The primary objective of this system is to provide researchers (students and faculty) with a centralized platform to register, upload their research papers, and automatically fetch and display their publication records from their ORCID profiles. The system also includes an administrative module for overseeing user activity and managing the platform's content.

The methodology involved designing and implementing a Flask-based web application using Python for the backend logic, SQLAlchemy as the Object-Relational Mapper (ORM) for database interactions with a MySQL database, and HTML, CSS, and JavaScript for the front-end user interface. Key features include user registration with ORCID ID validation, secure login, paper upload functionality with metadata (title, authors, category, publication name, date), automated fetching of public works from ORCID upon registration and login, and distinct dashboards for regular users and administrators. The system ensures data persistence by storing user information, uploaded paper details, and fetched ORCID works in a structured relational database.
The results demonstrate a functional system capable of user management, paper submission, and ORCID data synchronization. Users can view a consolidated list of their manually uploaded papers and ORCID-linked publications. The admin dashboard provides an overview of system usage, including total users and all uploaded papers. The integration with ORCID automates the process of keeping publication records up-to-date, reducing manual effort and enhancing data accuracy. This project successfully delivers a streamlined solution for research management, highlighting the benefits of integrating external authoritative data sources like ORCID. Future enhancements could include advanced search functionalities, collaborative features, and more detailed analytics.

TABLE OF CONTENTS -

Chapter 1: Introduction

- **1.1 Background of the Problem**
    - Discuss the challenges researchers and academic institutions face in managing research output (publications, articles, conference papers).
    - Mention issues like scattered data, difficulty in tracking publications, manual effort in maintaining profiles, lack of standardized researcher identifiers before systems like ORCID.
    - Briefly touch upon how digital solutions are becoming essential.
- **1.2 Project Motivation**
    - The desire to learn specific technologies (Flask, ORCID API)?
    - The need for a simplified, integrated system for individual researchers or small groups.
    - The importance of ORCID and the benefit of automating data synchronization.
- **1.3 Scope of the Project**
    - User registration (student/faculty) with ORCID ID.
    - Secure user login.
    - Fetching and displaying public works from ORCID.
    - Manual uploading of research papers with metadata.
    - User dashboard to view own ORCID works and uploaded papers.
    - Admin user with an overview dashboard (total users, all papers).
- **1.4 Objectives**
    1. To design and develop a web application for user registration and authentication, incorporating ORCID ID.
    2. To implement functionality to fetch and store public research works from a user's ORCID profile using the ORCID API.
    3. To enable users to upload their research papers along with relevant metadata.
    4. To create a user-specific dashboard displaying both ORCID-linked and manually uploaded publications.
    5. To develop an administrative interface for monitoring system usage and viewing all uploaded papers.
    6. To utilize Flask, SQLAlchemy, and MySQL for the backend and database, and HTML/CSS/JS for the frontend.

Chapter 2: Problem Statement

- **2.1 Defining the Problem**
    - "Researchers, particularly students and faculty in academic settings, often struggle with efficiently managing and showcasing their diverse research outputs. This includes keeping track of publications listed on ORCID and those not yet indexed, as well as manually submitted works."
    - The Sub-Problems:
        - **Data Silos:** Research information stored in multiple places (local drives, different profile sites).
        - **Manual Updates:** Time-consuming to keep profiles like ORCID and institutional lists manually updated.
        - **Lack of Centralization:** No single point of view for a researcher's complete output within a local context.
        - **Discovery & Visibility:** While ORCID helps, a local system can provide context within an institution.
        - **Administrative Oversight:** Difficulty for departments/admins to get an overview of research activity.
- **2.2 Significance of the Problem**
        - **Saves Time and Effort:** Automation reduces manual data entry.
        - **Improves Data Accuracy:** Syncing with authoritative sources like ORCID.
        - **Enhances Visibility:** A consolidated view of research output.
        - **Supports Researchers:** Especially early-career researchers and students in organizing their work.
        - **Facilitates Institutional Reporting (Potential):** Although basic in this version, it's a step towards better institutional tracking.
        - Provides a practical learning experience in web development, API integration, and database management.

Chapter 3: System Requirements Specification (SRS)

- **3.1 Functional Requirements**
    - **3.1.1 User Registration & Authentication Module**
        - FR1: The system shall allow new users (Student/Faculty) to register by providing a username, email, password, ORCID ID, and user type.
        - FR2: The system shall validate the ORCID ID format during registration.
        - FR3: The system shall prevent registration with duplicate usernames, emails, or ORCID IDs.
        - FR4: The system shall securely store user credentials, hashing passwords.
        - FR5: The system shall allow registered users to log in using their username and password.
        - FR6: The system shall allow users to log out, clearing their session.
        - FR7: The system shall provide a dedicated login mechanism for an 'admin' user.
    - **3.1.2 ORCID Integration Module**
        - FR8: The system shall fetch public work details (title, type, year, journal, DOI) and personal details (name) from the ORCID API for a given ORCID ID.
        - FR9: The system shall store fetched ORCID works in the database, linked to the respective user.
        - FR10: The system shall update a user's ORCID data upon login if an ORCID ID is provided or already associated with the account.
        - FR11: The system shall handle cases where ORCID data cannot be fetched (e.g., invalid ID, API errors) gracefully.
    - **3.1.3 Paper Management Module**
        - FR12: The system shall allow logged-in users to upload research paper files (e.g., PDF).
        - FR13: The system shall allow users to provide metadata for uploaded papers: title, author name(s), category (Journal, Conference, Book Chapter), publication name (journal/conference/book title), and publication date.
        - FR14: The system shall store uploaded paper files in a designated server location, organized by user.
        - FR15: The system shall store paper metadata in the database, linked to the uploading user.

- **3.1.4 User Dashboard Module**
  - FR16: The system shall display a personalized dashboard for logged-in users.
  - FR17: The user dashboard shall list the user's ORCID-linked publications.
  - FR18: The user dashboard shall list the user's manually uploaded papers.
  - FR19: The dashboard shall provide a form for uploading new paper

    3.1.5 Admin Dashboard Module
  - FR20: The system shall provide a dedicated dashboard for the admin user.
  - FR21: The admin dashboard shall display the total number of registered users (excluding admin).
  - FR22: The admin dashboard shall display a table of all papers uploaded by all users, including uploader details.
  - FR23: The admin dashboard shall display ORCID-linked data for all registered users.

- **3.2 Non-Functional Requirements**
  - **3.2.1 Usability**
    - NFR1: The user interface shall be intuitive and easy to navigate for both regular users and administrators.
    - NFR2: Forms for registration, login, and paper upload shall be clear and provide appropriate feedback/validation messages.
    - NFR3: The system shall provide clear visual feedback for successful operations or errors (flash messages).
  - **3.2.2 Performance**
    - NFR4: User registration and login should complete within 3-5 seconds under normal load.
    - NFR5: ORCID API calls should have a reasonable timeout (e.g., 10-15 seconds) to prevent indefinite waiting.
    - NFR6: Dashboard loading times should be acceptable (e.g., under 5-7 seconds for typical data loads).
    - NFR7: Paper uploads should be efficient, with file size limits considered (though not explicitly implemented, mention as a consideration).
  - **3.2.3 Security**
    - NFR8: User passwords shall be stored securely using strong hashing algorithms (e.g., Werkzeug's generate_password_hash).

        - NFR9: The system shall use session management to maintain user login states securely.
        - NFR10: File uploads shall use secure_filename to prevent path traversal or malicious filenames.
        - NFR11: Access to admin functionalities shall be restricted to the designated admin user.
        - NFR12: The SECRET_KEY for session management shall be kept confidential.
    - **3.2.4 Reliability**
        - NFR13: The system shall handle common errors gracefully (e.g., database connection issues, ORCID API unavailability) and provide informative messages.
        - NFR14: Data integrity shall be maintained in the database (e.g., foreign key constraints).
    - **3.2.5 Maintainability**
        - NFR15: The codebase shall be well-structured (e.g., separation of concerns in app.py, models.py, config.py).
        - NFR16: Code shall be commented where necessary to explain complex logic.
- **3.3 Use Case Diagrams**
    - **3.3.1 Use Case Diagram for User**
        - Create a UML Use Case diagram.
        - **Actors:** User (Student/Faculty)
        - **Use Cases:** Register, Login, Logout, View Dashboard, Fetch ORCID Data, Upload Paper, View ORCID Works, View Uploaded Papers.
        - Show relationships (e.g., Login <includes> Verify Credentials).
    - **3.3.2 Use Case Diagram for Administrator**
        - Create a UML Use Case diagram.
        - **Actors:** Administrator
        - **Use Cases:** Login (as Admin), Logout, View Admin Dashboard, View Total Users, View All Uploaded Papers, View All Users' ORCID Data.
- **3.4 Constraints and Assumptions**
    - **Constraints:**
        - The system relies on the availability and public accessibility of the ORCID API.

- The admin credentials are hardcoded in the current version (a known limitation for development, not production).
- The system requires a modern web browser for the frontend.
- The system is designed for deployment on a server with Python, Flask, and MySQL installed.

- **Assumptions:**
  - Users will have valid ORCID IDs and their ORCID profiles will have public works.
  - Users have a stable internet connection to access the system and for ORCID API calls.
  - The UPLOAD_FOLDER is writable by the application server process.

Chapter 4: Literature Survey / Related Work

- **4.1 Existing Research Management Systems**

  Existing Tools:

  - **Reference Managers:** Zotero, Mendeley, EndNote (focus on citation management, some have profile features).
  - **Institutional Repositories:** DSpace, EPrints (focus on archiving and disseminating institutional research).
  - **Researcher Networking Sites:** ResearchGate, Academia.edu (focus on profiles, paper sharing, social networking).
  - **Commercial CRIS Systems:** (Current Research Information Systems) like Pure, Symplectic Elements (comprehensive institutional solutions, often expensive).

- **4.2 Role of ORCID in Research Ecosystem**

  - Provides a unique and persistent researcher identifier (ORCID iD).
  - Maintains a centralized and verifiable research profile.
  - Enhances visibility and recognition of research work.
  - Integrates with journals, funders, and institutions for seamless workflows.
  - Promotes trust, transparency, and accurate attribution in research.

- **4.3 Comparison with Proposed System**

  - **Focus:** The system has a narrower focus on individual/small group management and tight ORCID integration for personal tracking and simple institutional overview via admin.
  - **Complexity:** Likely simpler than large CRIS systems or feature-rich reference managers.
  - **Cost:** Open source (yours) vs. commercial or subscription models.
  - **Key Differentiator:** The direct ORCID sync as a core part of registration/login for updating a local cache, combined with manual uploads, tailored for a simpler use case.

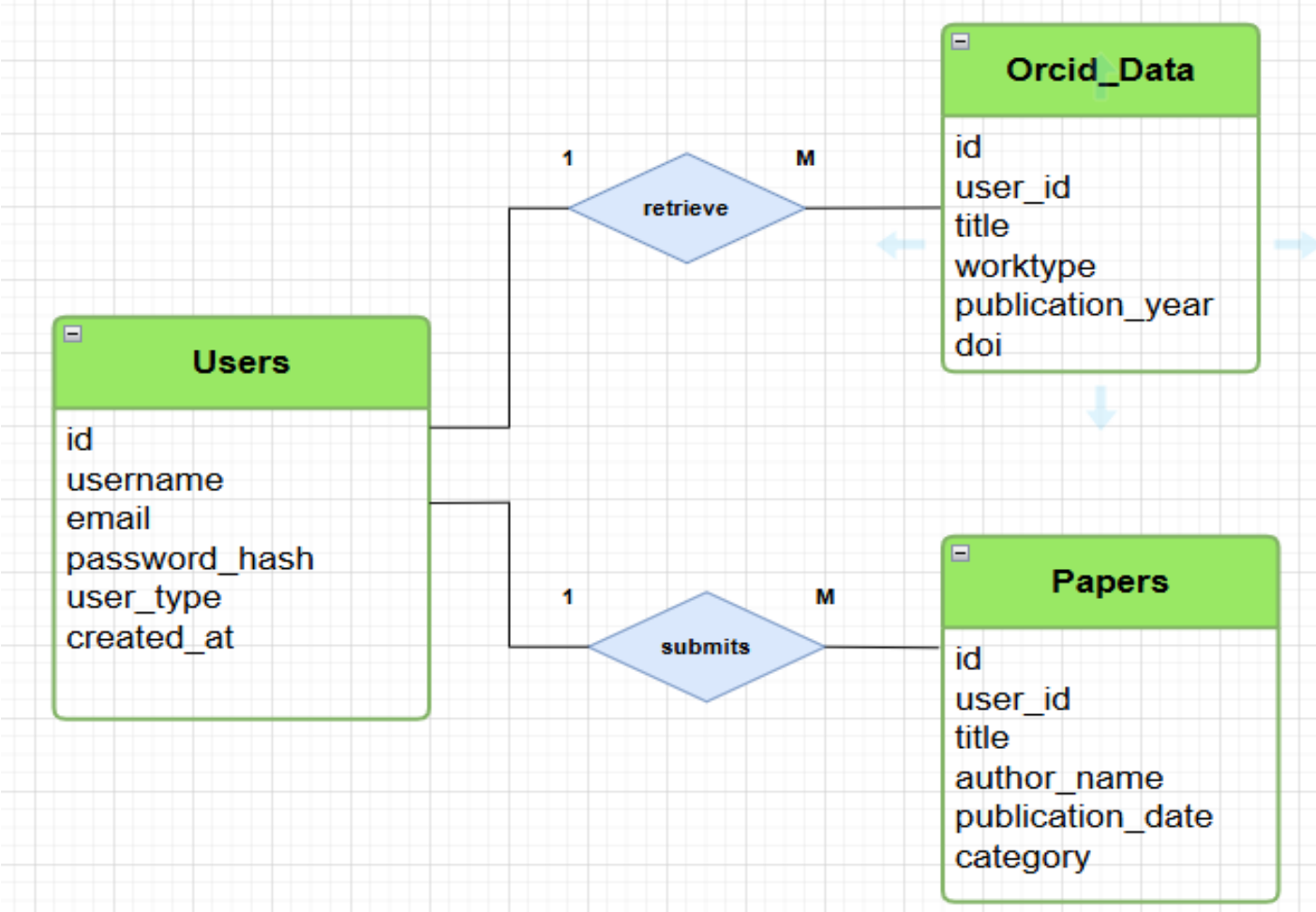
- **4.4 Gaps Addressed by This Project**

  - A lightweight, easy-to-use system for individuals to consolidate ORCID data and manual uploads.
    - A simplified admin view for basic institutional tracking without the overhead of a full CRIS.

- Providing a hands-on tool that integrates directly with ORCID for data fetching in a straightforward manner.
- A learning platform to understand the practicalities of API integration for research data.

Chapter 5: System Design

- **5.1 System Architecture** -
    - **ER DIAGRAM** –



Entities and Attributes

1. Users

Attributes:- **id, username, email, password_hash, user_type, created_at .**

2. Orcid_Data

Attributes:- **id, user_id, title, worktype, publication_year, doi .**

3. Papers

Attributes:- **id, user_id, title, author_name, publication_date, category .**

Relationships

- **Users → Papers**

  ➤ A single user can submit multiple papers, but each paper is submitted by only one user.

  ➤ Relationship: One-to-Many (1 : M)

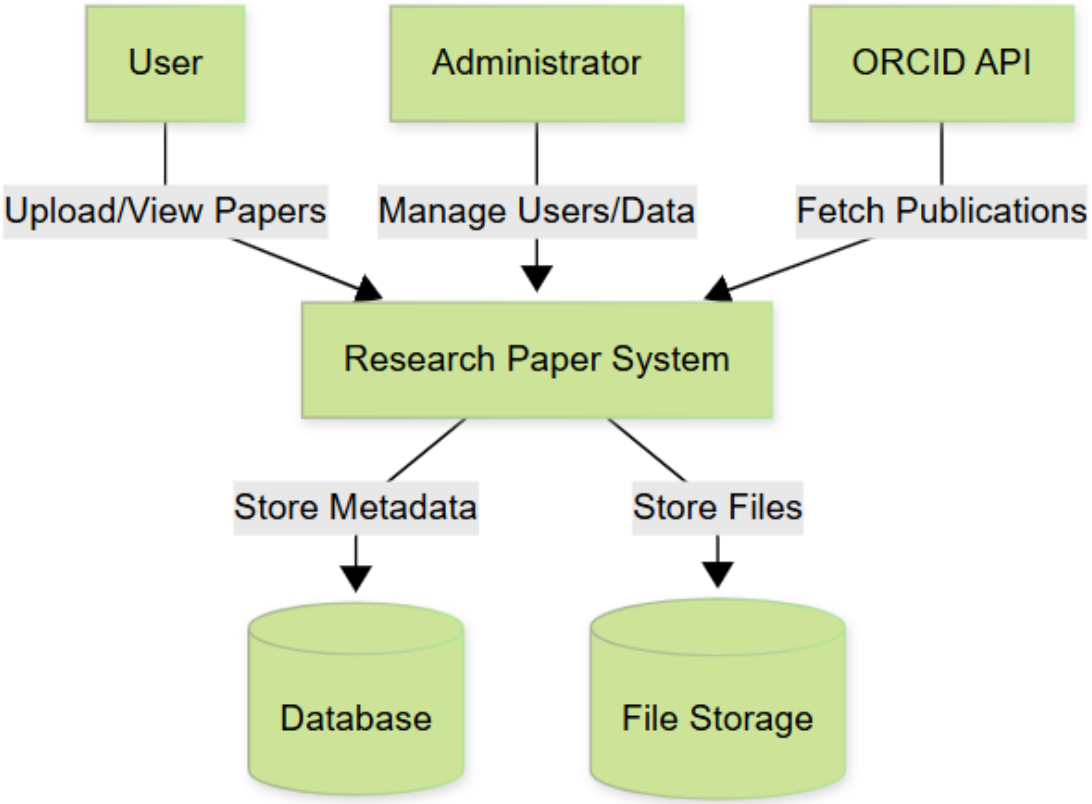
- **Users → Orcid_Data**

  ➤ A single user can retrieve multiple ORCID records, but each ORCID record is tied to only one user.

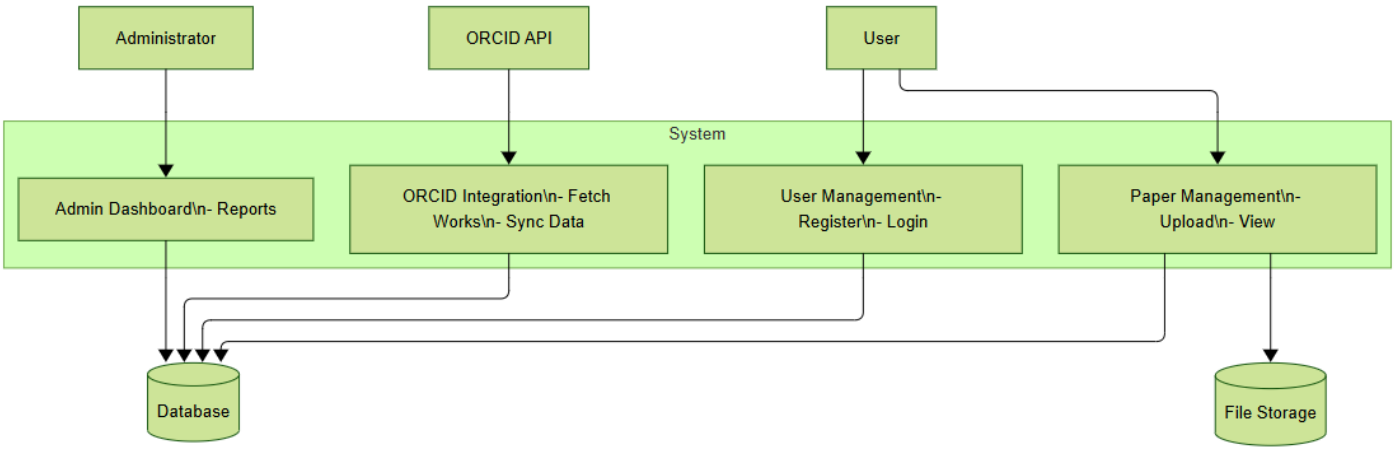
  ➤ Relationship: One-to-Many (1: M)

DFD (DATA FLOW DIAGRAM) –

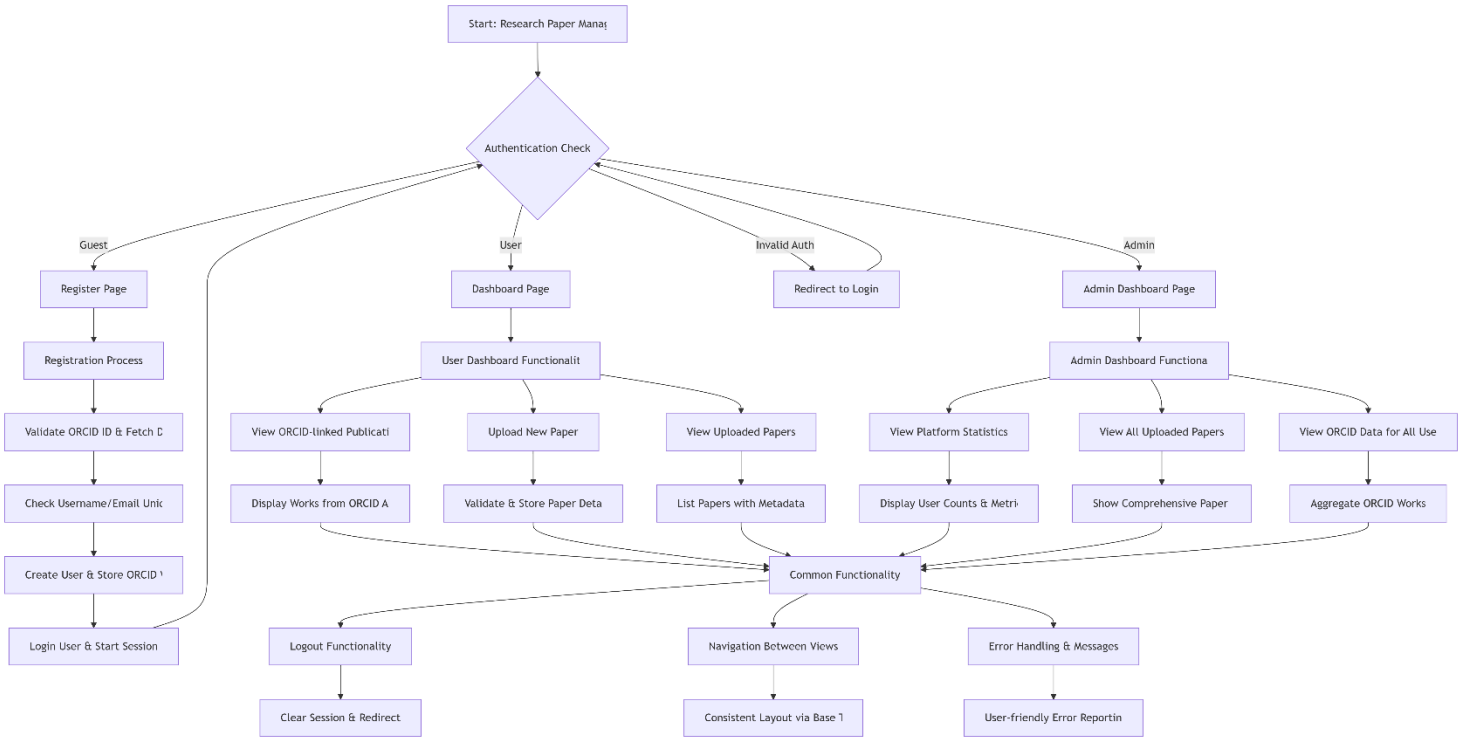
LEVEL 0 –



LEVEL 1 –



FLOWCHART -


Start: Research Paper Manag
Authentication Check
Guest
User
Invalid Auth
Admin
Register Page
Dashboard Page
Redirect to Login
Admin Dashboard Page
Registration Process
User Dashboard Functionalit
Admin Dashboard Functiona
Validate ORCID ID & Fetch D
View ORCID-linked Publicati
Upload New Paper
View Uploaded Papers
View Platform Statistics
View All Uploaded Papers
View ORCID Data for All Use
Check Username/Email Unic
Display Works from ORCID A
Validate & Store Paper Deta
List Papers with Metadata
Display User Counts & Metri
Show Comprehensive Paper
Aggregate ORCID Works
Create User & Store ORCID
Common Functionality
Login User & Start Session
Logout Functionality
Navigation Between Views
Error Handling & Messages
Clear Session & Redirect
Consistent Layout via Base T
User-friendly Error Reportin

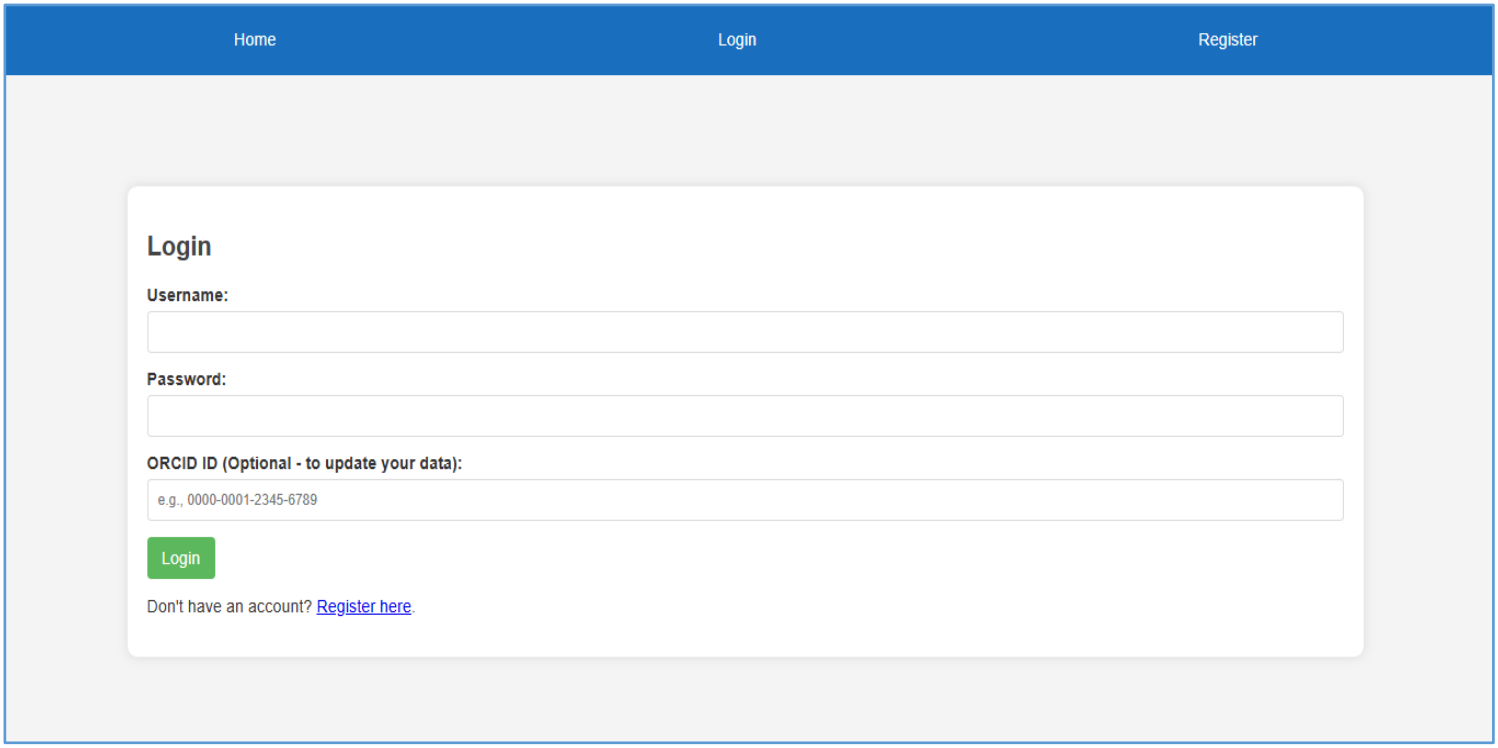
Chapter 6: Technology Stack

- **6.1 Backend Technologies**
    - **6.1.1 Python**
        - Version used (e.g., Python 3.8+).
        - Why Python? Because Readability, large ecosystem of libraries, suitability for web development, ease of learning.
    - **6.1.2 Flask Framework**
        - Version used.
        - Why Flask? (Micro-framework, lightweight, flexible, Jinja2 templating, Werkzeug for WSGI, good for smaller to medium projects, easy to get started).
    - **6.1.3 SQLAlchemy (ORM)**
        - Version used.
        - Why SQLAlchemy? Powerful ORM, database agnostic (though you used MySQL), facilitates database interactions in Pythonic way, handles schema management.
    - **6.1.4 Requests Library**
        - Version used.
        - Why Requests? Simple and elegant HTTP library for making API calls to ORCID.
- **6.2 Frontend Technologies**
    - **6.2.1 HTML, CSS, JavaScript**
    - HTML5 for structure.
    - CSS3 for styling
    - JavaScript for client-side interactions
- **6.3 Database**
    - **6.3.1 MySQL**
        - Version used.
        - Why MySQL? Popular open-source RDBMS, reliable, good performance for web applications, SQLAlchemy support via mysqlconnector.

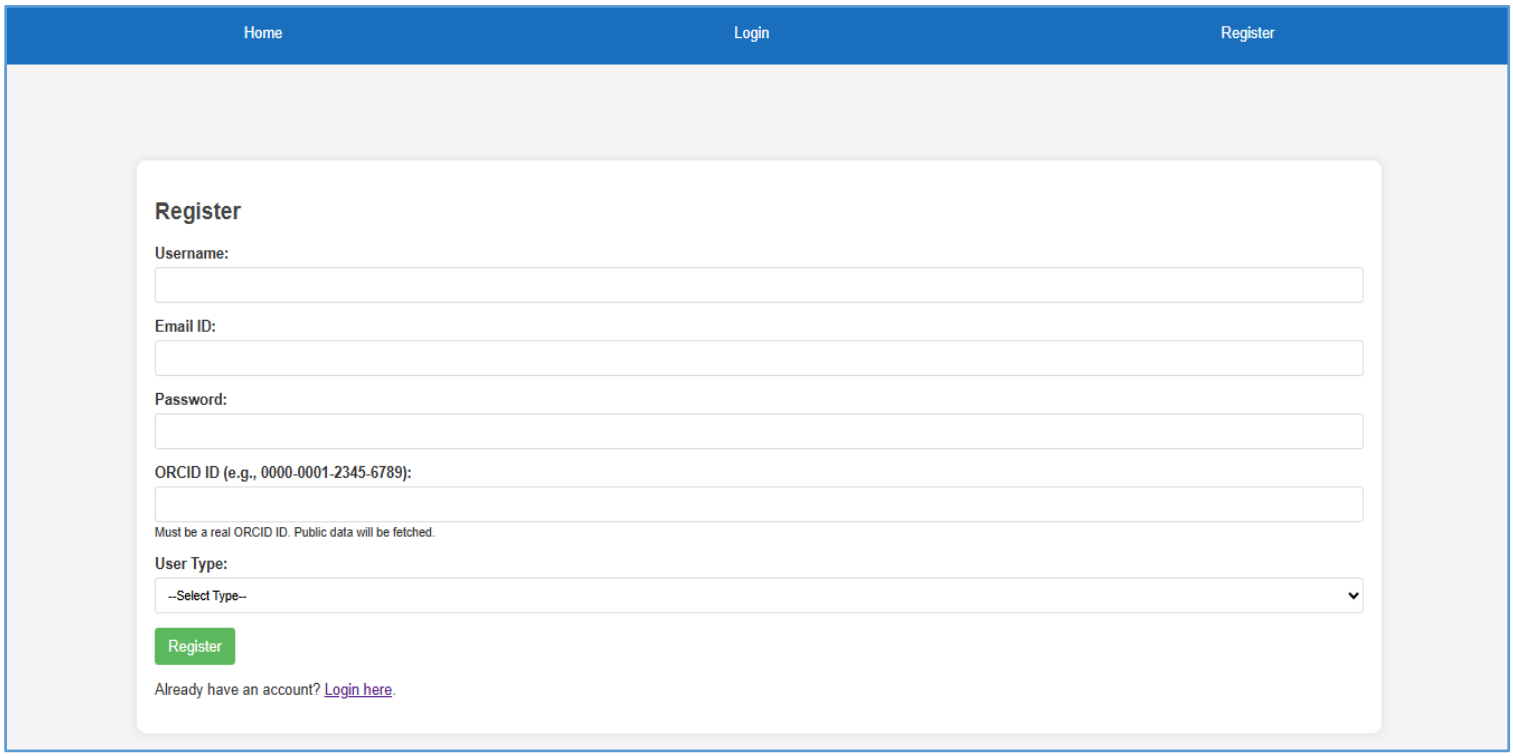
- **6.4 Development Tools & Environment**
    - **6.4.1 IDE, Version Control, etc.**
        - **IDE:** (e.g., Visual Studio Code, PyCharm).
        - **Version Control:** Git, GitHub/GitLab (if used).
        - **Operating System:** (e.g., Windows, Linux, macOS).
        - **Python Virtual Environment:** (e.g., venv) for managing dependencies.

Chapter 7: Implementation

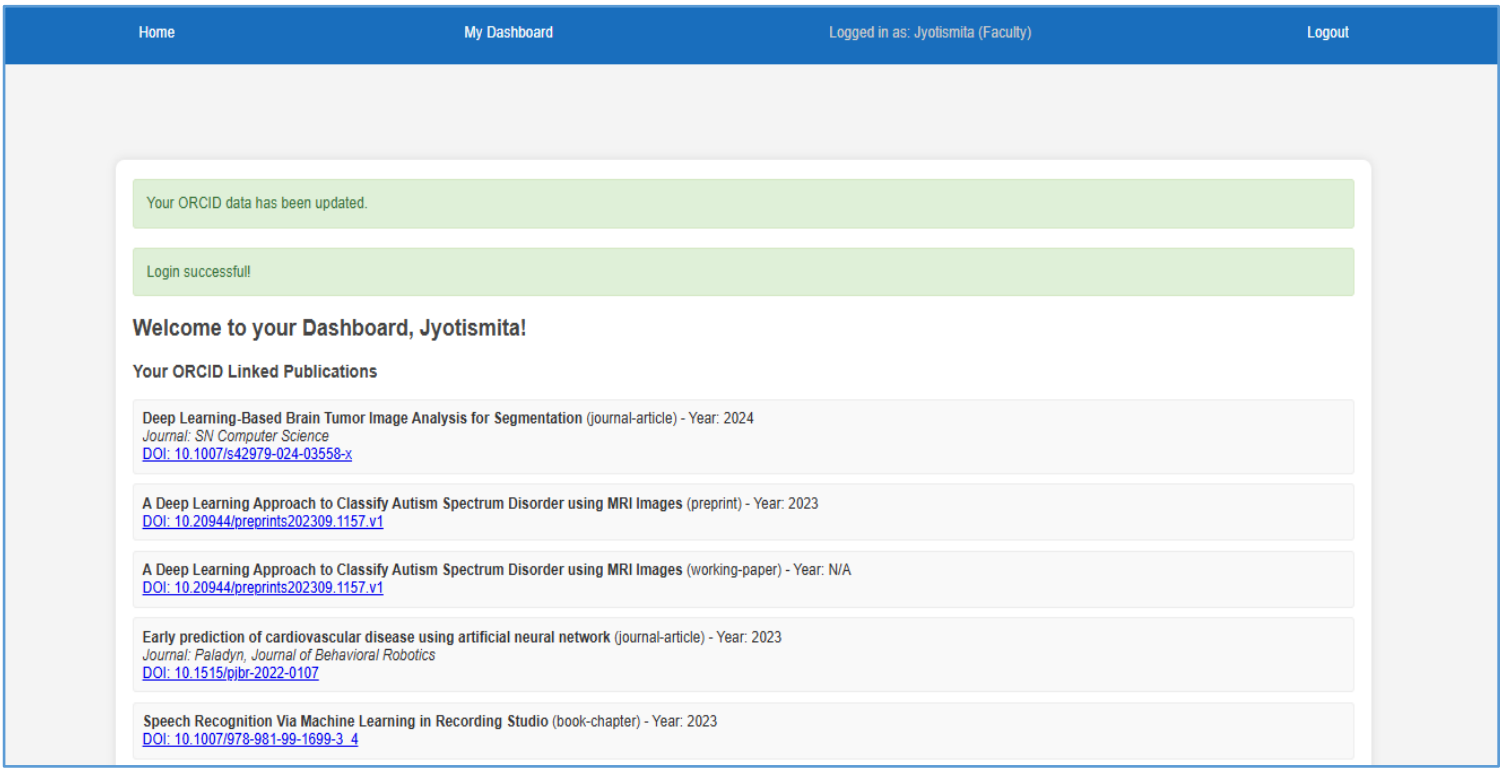
LOGIN DASHBOARD –



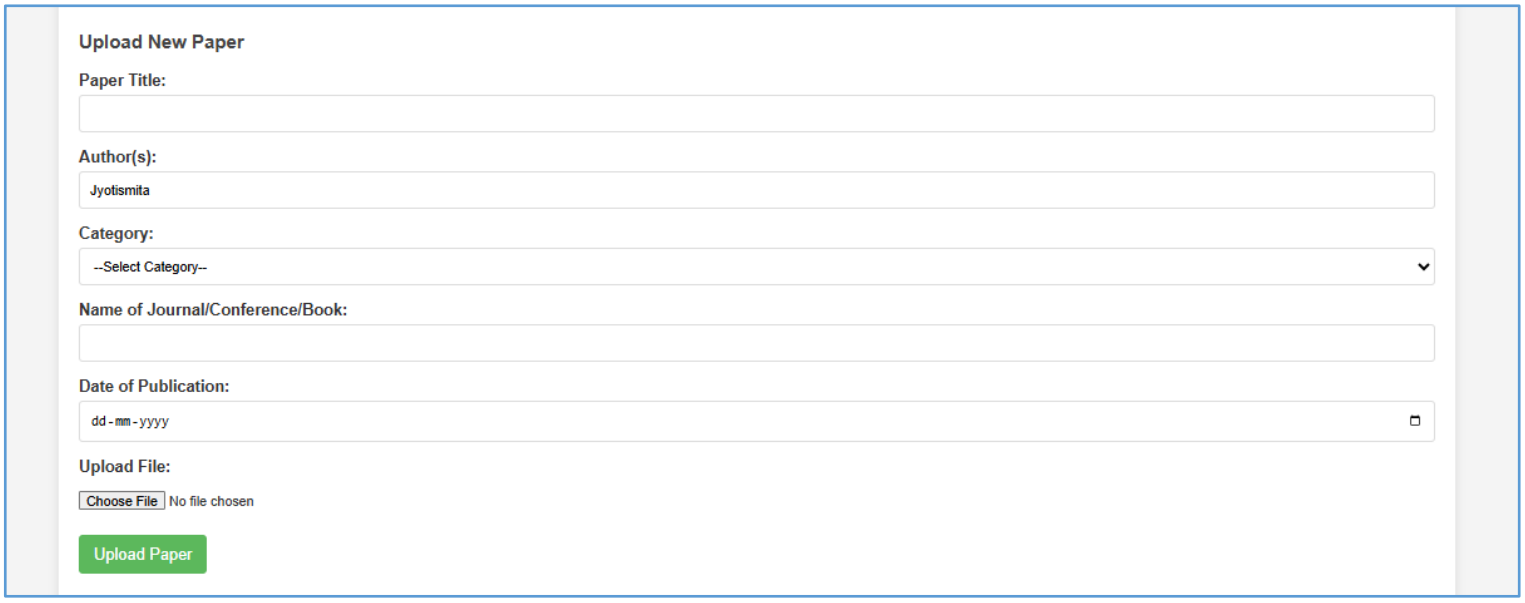
REGISTER DASHBOARD -



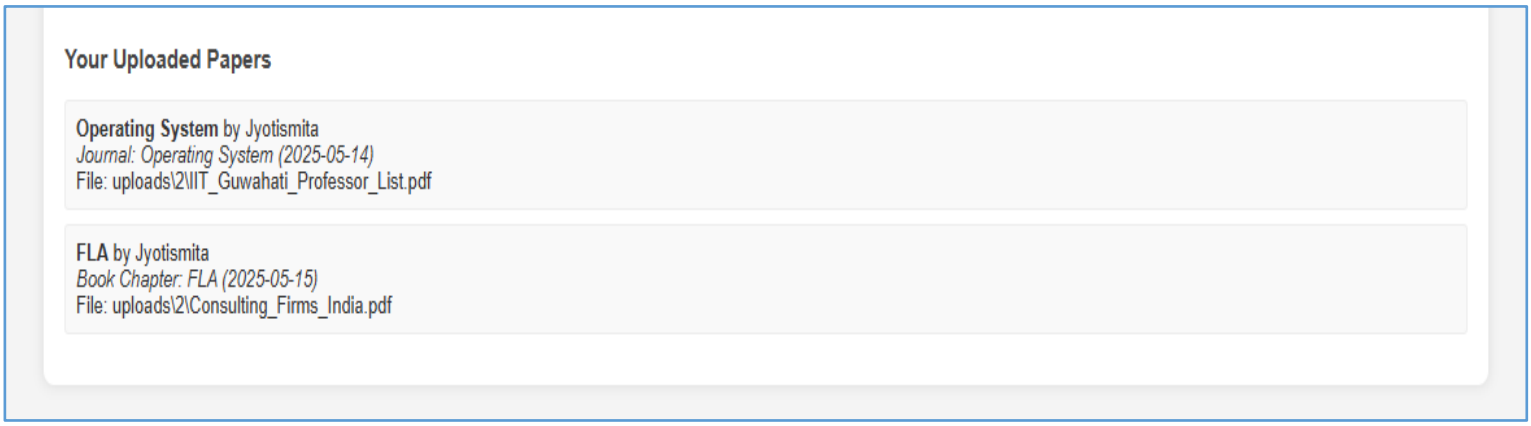
USER DASHBOARD –



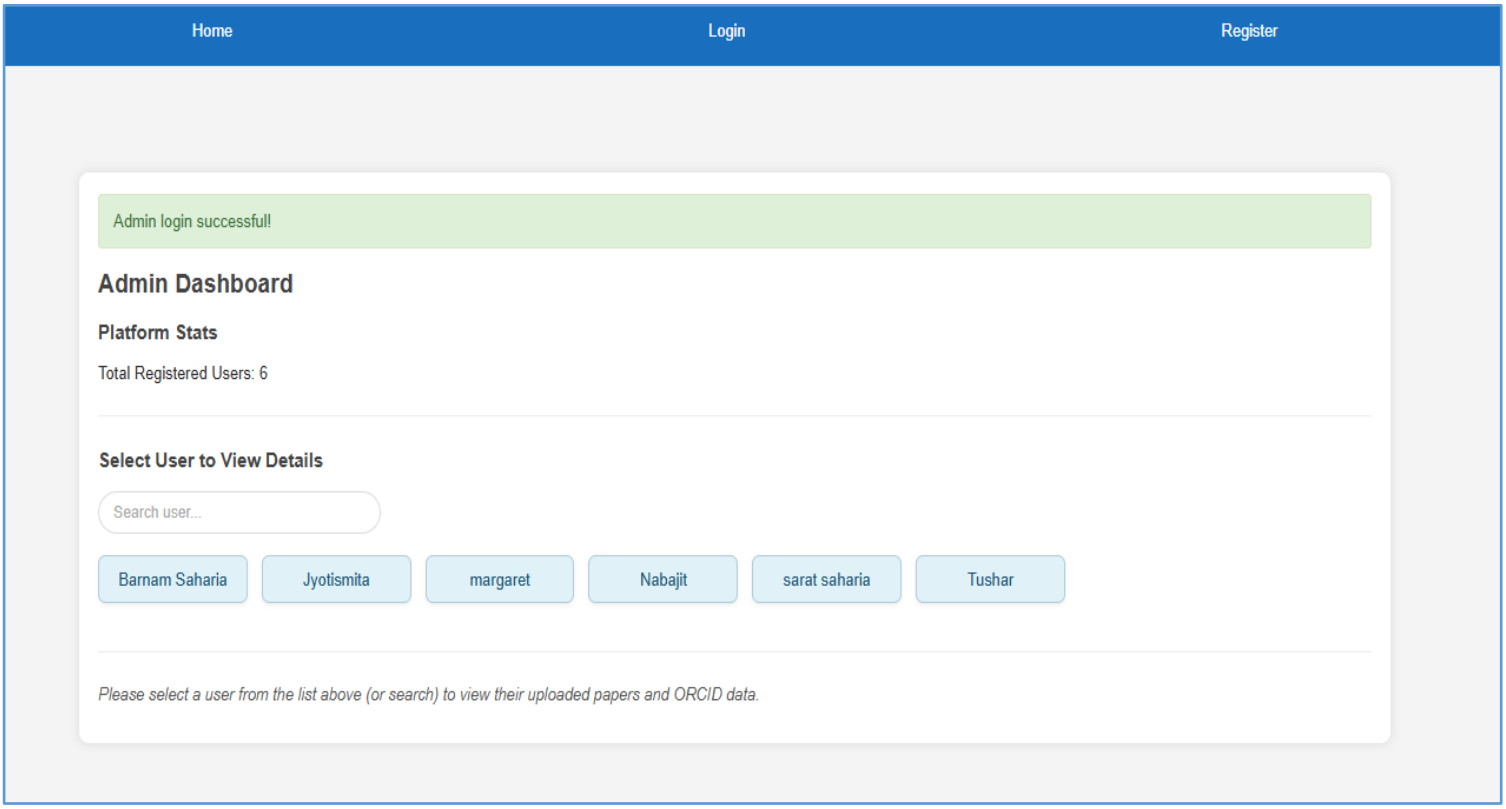
PAPER UPLOAD SECTION DASHBOARD -



UPLOADED PAPER SECTION –



ADMIN DASHBOARD –



ADMIN DASHBOARD (2) –

Select User to View Details

Search user...

Barnam Saharia | Jyotismita | margaret | Nabajit | sarat saharia | Tushar

Details for User: Jyotismita

Uploaded Papers by Jyotismita

Title	Author(s)	Category	Publication Name	Date	File Path
Operating System	Jyotismita	Journal	Operating System	2025-05-14	uploads\2\IIT_Guwahati_Professor_List.pdf
FLA	Jyotismita	Book Chapter	FLA	2025-05-15	uploads\2\Consulting_Firms_India.pdf
SSCD	Jyotismita	Journal	SSCD	2025-05-27	uploads\2\Banks_India.pdf
DBMS	Jyotismita	Book Chapter	DATABASE MANAGEMENT SYSTEM	2025-05-28	uploads\2\Profile.pdf
DBMS	Jyotismita	Book Chapter	DATABASE MANAGEMENT SYSTEM	2025-05-28	uploads\2\Profile.pdf

ORCID-Linked Data for Jyotismita (0000-0002-5167-7500)

Deep Learning-Based Brain Tumor Image Analysis for Segmentation (journal-article) - Year: 2024
Journal: SN Computer Science
DOI: 10.1007/s42979-024-03558-x

A Deep Learning Approach to Classify Autism Spectrum Disorder using MRI Images (preprint) - Year: 2023
DOI: 10.20944/preprints202309.1157.v1

A Deep Learning Approach to Classify Autism Spectrum Disorder using MRI Images (working-paper) - Year: N/A
DOI: 10.20944/preprints202309.1157.v1

Chapter 8: Testing

- **8.1 Testing Strategy**
    - Mentioned that testing was primarily manual,

      focusing on user workflows and critical

      functionalities.
    - Types of testing performed: Unit (conceptual, for small functions), Integration (module interactions), System (end-to-end).
- **8.2 Unit Testing**

  8.2.1 Test Cases for Helper Functions

  - Example: For fetch_orcid_public_data:
  - Input: Valid ORCID, Invalid format ORCID, Non-existent ORCID, ORCID with no public works.
  - Expected Output: Correct data, None, specific error handling.
  - Example: For secure_filename: Tested with various filenames.

  8.2.2 Test Cases for Model Methods

  - Example: User.set_password() and User.check_password(): Tested if password hashing and checking work correctly.
- **8.3 Integration Testing**

  8.3.1 Testing User Registration with ORCID Fetch

  - Scenario: User registers -> register route calls fetch_orcid_public_data -> data saved to User and OrcidWork tables.
  - Verification: Check database entries, check session, check flashed messages.

  8.3.2 Testing Paper Upload and Database Save

  - Scenario: User uploads paper -> upload_paper route saves file -> metadata saved to Paper table.
  - Verification: Check file system, check database entry.

- **8.4 System Testing**
    - **8.4.1 End-to-End User Workflow Testing**
        - Scenario 1: New user registration -> login -> view dashboard -> upload paper -> view updated dashboard -> logout.
        - Scenario 2: Existing user login -> (optional ORCID update) -> view dashboard.
    - **8.4.2 Admin Workflow Testing**
        - Scenario: Admin login -> view admin dashboard -> check user count -> check all papers list -> check user ORCID data.
- **8.5 Bug Fixes and Issues Encountered**
    - Initial issues with parsing deeply nested ORCID JSON (e.g., publicationdate being null or year object missing). Solution: Added more robust .get() checks and isinstance.
    - File path issues for uploads (e.g., ensuring UPLOAD_FOLDER exists, userspecific subdirectories).
    - Session not persisting or decorators not working as expected (common Flask learning curve issues).
    - Database constraint violations (e.g., UNIQUE constraints for email/username/orcid_id).

Chapter 9: Result and Discussion

- **9.1 Overview of Achieved Functionality**
    - "The developed Research Project Management System successfully allows users to register, link their ORCID profiles, upload research papers, and view a consolidated list of their publications. Administrators have an overview of system activity."
- **9.2 System Performance Analysis**
    - **9.2.1 Response Times for Key Operations**
        - General feel: "Page loads for dashboards and forms were generally responsive under single-user testing conditions."
        - "User registration and login, excluding ORCID API calls, were nearinstantaneous."
    - **9.2.2 ORCID API Call Latency**
        - "The most significant factor impacting performance was the ORCID API call during registration and login. Fetching data from the external API typically took [e.g., 2-5 seconds] depending on network conditions and the amount of data on the ORCID profile."
- **9.3 User Feedback**
    - "Informal feedback indicated the UI was straightforward to use"
    - "Users appreciated the automatic fetching of ORCID data."
- **9.4 Comparison with Objectives**
        - Objective 1 (User reg/auth with ORCID): Achieved.
        - Objective 2 (Fetch/store ORCID works): Achieved.
        - Objective 3 (Enable paper uploads): Achieved.
        - ...and so on.

Chapter 10: Limitations

- **10.1 Scope Limitations**
  - No advanced search or filtering of papers/works.
  - No collaborative features (e.g., multiple authors on one paper entry within the system).
  - Limited user roles (only regular user and one super admin). No departmental admins, etc.
  - Does not handle different versions of uploaded papers.
  - No public-facing portal for browsing research (it's an internal management tool).
- **10.2 Technical Limitations**
  - **Security:**
    - Hardcoded admin credentials (major one for a real app, acceptable for dev project if stated).
    - Lack of CSRF protection.
    - Basic file type validation (relies on secure_filename, no deep content inspection).
  - **Scalability:**
    - The admin dashboard fetching all ORCID data for all users could be slow with many users/works.
    - Single server deployment model.
  - **ORCID API:**
    - Relies on public ORCID API; does not use authenticated member API for more features.
    - Only fetches works; doesn't push data back to ORCID (which would require OAuth).
- **10.3 Challenges Faced During Development**
  - Understanding the ORCID API JSON structure and its variability.
  - Implementing robust error handling for external API calls.
  - Managing Flask sessions and authentication decorators correctly.
  - Ensuring secure file uploads and path management.
  - Debugging database interaction issues with SQLAlchemy.

Chapter 11: Future Work / Enhancements

- **11.1 Advanced Search and Filtering**
    - Implement search by title, author, year, keyword, category for both uploaded papers and ORCID works.
    - Allow filtering by publication year, user type, etc.
- **11.2 Collaborative Features**
    - Allow multiple system users to be associated with a single uploaded paper.
    - Version control for uploaded documents.
- **11.3 Enhanced Admin Capabilities**
    - User management (edit/delete users, reset passwords).
    - Ability to edit/delete any uploaded paper.
    - More granular admin roles.
- **11.4 File Preview and Download Management**
    - Securely serve uploaded files for download.
    - Inline preview for common file types (e.g., PDFs).
- **11.5 Email Notifications**
    - Notifications for new user registration (to admin), successful paper upload (to user).
- **11.6 Improved Security Measures (CSRF, etc.)**
    - Implement CSRF protection (e.g., Flask-WTF).
    - More robust input validation on all forms.
    - Regular security audits.
- **11.7 Analytics and Reporting**
    - Generate reports on research output (e.g., papers per department/user type, trends over time).
    - Visualizations (charts, graphs) of research activity.

Chapter 12: Conclusion

- **12.1 Summary of Achievements**
    - "The Research Project Management System was developed to provide an efficient platform for researchers to manage their publications by integrating manual uploads with automated ORCID data fetching. The system successfully meets its core objectives of user management, paper submission, and ORCID synchronization, offering distinct functionalities for regular users and administrators."
- **12.2 Overall Project Impact**

  "This project serves as a practical demonstration of how web technologies can be leveraged to streamline research management tasks. It provides a foundation upon which more comprehensive institutional research information systems can be built, and highlights the value of integrating with authoritative external data sources like ORCID."

References / Bibliography

- **Websites:**
  - Flask Documentation: https://flask.palletsprojects.com/
  - SQLAlchemy Documentation: https://www.sqlalchemy.org/
  - ORCID Public API Documentation: https://info.orcid.org/documentation/apitutorials/api-tutorial-read-data-on-a-record/ (or the specific v3.0 docs)
  - Requests Library Documentation: https://requests.readthedocs.io/
  - Jinja2 Documentation: https://jinja.palletsprojects.com/
  - MySQL Documentation: https://dev.mysql.com/doc/