COMPUTING RESEARCH PROJECTS REPOSITORY

Innovation meets Collaboration

SCIT

Computer Technology

Contents

[**COMPUTING RESEARCH PROJECTS REPOSITORY** 4](#_Toc173234866)

[1.Introduction 4](#_Toc173234867)

[**1.1 Project Overview** 4](#_Toc173234868)

[**1.2Background** 4](#_Toc173234869)

[**1.3.Scope** 5](#_Toc173234870)

[**1.4 Objectives** 6](#_Toc173234871)

[**1.5 Stakeholders** 7](#_Toc173234872)

[2. Requirements for the Computing Research Projects Repository 9](#_Toc173234873)

[**2.1 Functional Requirements** 9](#_Toc173234874)

[**2.1.1 User Registration and Authentication** 9](#_Toc173234875)

[2.1.2 **Project Cataloging** 9](#_Toc173234876)

[**2.1.3 Search and Filtering** 10](#_Toc173234877)

[**2.1.4 Project Details Page** 10](#_Toc173234878)

[**2.1.5 User Roles and Permissions** 11](#_Toc173234879)

[**2.2 Non-Functional Requirements** 12](#_Toc173234880)

[**2.2.1 Performance** 12](#_Toc173234881)

[**2.2.2 Security** 12](#_Toc173234882)

[**2.2.3 Usability** 13](#_Toc173234883)

[**2.2.4 Scalability** 13](#_Toc173234884)

[**2.2.5 Reliability** 13](#_Toc173234885)

[**2.2.6 Compliance** 13](#_Toc173234886)

[3.Project Plan for the CRPR Project 14](#_Toc173234887)

[**3.1Timeline** 14](#_Toc173234888)

[**Month 1: Planning and Design** 14](#_Toc173234889)

[**Month 2: Development and Implementation** 14](#_Toc173234890)

[**Month 3: Testing, Deployment, and Documentation** 14](#_Toc173234891)

[**3.2 Work Breakdown Structure (WBS)** 15](#_Toc173234892)

[**3.2.1. Project Initiation** 15](#_Toc173234893)

[**3.2.2. Requirements Gathering and Analysis** 15](#_Toc173234894)

[**3.2.3. System Design** 15](#_Toc173234895)

[**3.2.4. Development Phase** 16](#_Toc173234896)

[**3.2.5. Testing Phase** 16](#_Toc173234897)

[**3.2.6. Deployment and Launch Preparation** 17](#_Toc173234898)

[**3.2.7. Documentation and Training** 17](#_Toc173234899)

[4.Design and Architecture 18](#_Toc173234900)

[**4.1 System Architecture** 18](#_Toc173234901)

[**4.2** **Design Diagrams** 19](#_Toc173234902)

[**4.2.1Context Diagram (Level 0 DFD)** 19](#_Toc173234903)

[**4.3 Technical Specifications** 22](#_Toc173234904)

[5. User Guide 23](#_Toc173234905)

[**5.1Accessing the Platform** 23](#_Toc173234906)

[5.1.1 **Logging** In 23](#_Toc173234907)

[**5.1.2 Registering an Account** 23](#_Toc173234908)

[**5.1.3. Recovering Your Password** 23](#_Toc173234909)

[**5.2 Using the Repository** 23](#_Toc173234910)

[**5.2.1. Creating a New Project** 23](#_Toc173234911)

[**5.2.2. Adding a Collaborator (Optional)** 24](#_Toc173234912)

[**5.2.3. Uploading Project files** 24](#_Toc173234913)

[**5.2.4. Viewing Projects** 24](#_Toc173234914)

[**5.2.5. Editing a Project** 24](#_Toc173234915)

[**5.2.6. Deleting a Project** 24](#_Toc173234916)

[**5.2.7. Downloading Project** 25](#_Toc173234917)

[**5.3. User Roles and Permissions** 25](#_Toc173234918)

[**5.3.1. Admins** 25](#_Toc173234919)

[**5.3.2. Researchers** 25](#_Toc173234920)

[**5.3.3. Guests** 25](#_Toc173234921)

[6.Deployment Plan for CRPR Project 26](#_Toc173234922)

[**6.1. Preparation** 26](#_Toc173234923)

[**6.2 Environment Setup** 26](#_Toc173234924)

[**6.3. Deployment** 27](#_Toc173234925)

[**6.4. Post-Deployment** 27](#_Toc173234926)

[**6.5 Rollback Plan** 28](#_Toc173234927)

[**6.6 Troubleshooting** 28](#_Toc173234928)

[7. Important Notice: Credentials for Password Reset and Payment Processing 30](#_Toc173234929)

[8.Maintenance and Support 31](#_Toc173234930)

[**8.1Maintenance Plan** 31](#_Toc173234931)

[**8.2.Support Plan** 31](#_Toc173234932)

[9.Appendices 32](#_Toc173234933)

[**Glossary** 32](#_Toc173234934)

[**References** 32](#_Toc173234935)

# **COMPUTING RESEARCH PROJECTS REPOSITORY**

# 1.Introduction

## **1.1 Project Overview**

The Computing Research Projects Repository (CRPR) is a centralized platform designed to catalogue and showcase ongoing research projects within the Department of Computing. This repository is designed to serve as a comprehensive and accessible database, offering detailed information on research initiatives across various domains including artificial intelligence, cybersecurity, software engineering, data science, and more.

CRPR aims to bridge the gap between researchers, students, and external stakeholders by providing a transparent and collaborative environment. By centralizing project information, CRPR facilitates easy access to resources, outcomes, and ongoing collaborations. This platform is not only a tool for knowledge sharing and fostering interdisciplinary cooperation within the department but also a valuable resource for prospective students, funding agencies, and industry partners who are interested in engaging with cutting-edge computing research. Through regular updates and maintenance, CRPR ensures that all information remains current and relevant, providing a reliable and up-to-date resource for all users.

## **1.2Background**

The Department of Computing at JKUAT is known for its pioneering research across a broad spectrum of computing domains, including artificial intelligence, cybersecurity, software engineering, and data science. Over the years, the department has undertaken numerous groundbreaking projects that have significantly contributed to the advancement of technology and knowledge in these fields. However, the diversity and volume of these research initiatives have presented challenges in terms of tracking, managing, and sharing information effectively.

Traditionally, information about ongoing research projects has been dispersed across various platforms and formats, making it difficult for researchers, students, and stakeholders to access comprehensive and up-to-date project details. This fragmentation has hindered collaboration and knowledge sharing, limiting the potential for interdisciplinary innovation and the ability to leverage existing research for new initiatives.

Recognizing these challenges, the Department of Computing identified the need for a centralized platform that could serve as a single source of truth for all ongoing research projects. The idea was to create a repository that would not only consolidate project information but also provide tools to facilitate collaboration, track project progress, and showcase research outcomes to a broader audience, including prospective students, funding agencies, and industry partners.

The Computing Research Projects Repository (CRPR) is the response to this need. It aims to streamline the process of managing and disseminating research information, providing a robust infrastructure that supports the department’s mission of fostering innovation, enhancing collaboration, and promoting excellence in computing research.

## **1.3.Scope**

The CRPR project will include the following key components:

1. **Comprehensive Database of Ongoing Research Projects**

The database is going to store detailed information on each research project. It includes fields for project title, description, collaborators, methodologies and outcomes.

The database supports various research domains, including artificial intelligence, cybersecurity, software engineering and data science.

1. **User-Friendly Interface for Accessing Project Information**

The CRPR will have a visually appealing web interface accessible via desktop and mobile devices. It will allow easy navigation and accessibility of projects and resources for all users, including researchers, students, department administrators, and external stakeholders.

Implement advanced search and filtering capabilities to quickly locate specific projects based on keywords, categories, and tags.

1. **Collaboration and Knowledge Sharing Features**

Enable project tagging and categorization to organize projects effectively and enhance discoverability.

Provide features for commenting, sharing resources, and collaborating on project documents in real-time.

1. **User Profiles and Personalization**

The project will entail a personalized user profiles for researchers, students, and stakeholders to customize their experience. It will have a dashboard for users to track their involvement and monitor project progress.

1. **Security and Access Control**

Define access control mechanisms to restrict access based on user roles and permissions, ensuring appropriate data privacy and security.

1. **Regular Updates and Maintenance**

The project will have a comprehensive plan for regular maintenance, updates, and improvements to ensure the platform remains current, functional, and secure. Implement procedures for data verification and quality control to maintain the accuracy and reliability of project information. Provide ongoing technical support through user manuals and FAQs.

## **1.4 Objectives**

1. **Centralized Research Information**

Develop a comprehensive database to store and manage detailed information about all ongoing research projects within the Department of Computing.

Ensure that the database includes key project details such as project title, description, lead researchers, team members, objectives, methodologies, and outcomes.

1. **Enhance Accessibility**

Create a user-friendly interface that allows easy navigation and access to project information for researchers, students, and external stakeholders.

Implement advanced search and filtering features to help users quickly locate specific projects based on criteria such as keywords, categories, and tags.

1. **Support Personalization**

Develop personalized user profiles that allow researchers, students, and stakeholders to customize their experience on the platform.

Implement features for users to follow projects, receive updates, and manage their contributions.

1. **Ensure Data Security**

Implement robust security measures, including encryption, user authentication, and data protection protocols, to safeguard sensitive information and maintain data privacy.

Define access control mechanisms to ensure appropriate data access based on user roles and permissions.

## **1.5 Stakeholders**

**1.Project Team**

Front-end Developer: Alex Maina

Designs and implement the user interface, ensure a seamless user experience, work on client-side scripting, and optimize the platform for various devices and browsers.

Back-end Developer: Gikuru Joseph

Develop and maintain the server-side logic, manage the database, ensure the performance and responsiveness of the platform, and integrate front-end elements.

Web Designer: Loren Odhiambo

Create and implement visually appealing and user-friendly designs for the CRPR platform, ensuring a cohesive look and feel across all pages.

Software Developer: Boniface Mwangi

Architect the platform, make high-level design decisions, lead the development team, review code, and ensure coding standards are maintained.

System Analyst: Felix Ouma

Develop detailed system specifications and design documents, outlining the architecture, components, and data flow of the platform.

2. **Researchers**

They will be faculty members and researchers who are involved in various research projects within the Department of Computing.

Their purpose is to provide detailed information about their research projects, including objectives, methodologies, outcomes, and updates. They will also use the platform to track and manage their projects.

3.**Students**

Undergraduate and postgraduate students will also be engaged in research activities or those who are interested in learning more about ongoing research.

Theirs role will be to use the platform for research purposes, learning, and academic development. They may also provide feedback on the platform’s usability and features.

**4. Department Administrators**

The Department heads, administrative staff, and other individuals who oversee the department’s activities and strategic goals will play part in the project.

They ensure the CRPR project aligns with departmental objectives, provide strategic oversight, and support the project’s integration with departmental operations.

5. **Industry Partners**

External organizations, companies, and industry professionals who may collaborate with the department on research projects or provide funding.

They will engage with the platform to identify potential collaboration opportunities, access project information for partnership considerations, and provide funding or resources.

# 

# 2. Requirements for the Computing Research Projects Repository

These are the functional and non-functional requirements for the CRPR project;

## **2.1 Functional Requirements**

### **2.1.1 User Registration and Authentication**

**FR-001: User Registration**

* Description**:** The system shall allow users to create an account using their school email address.
* Priority**:** High
* Acceptance Criteria**:**
  + Users can register by providing their email, password, and confirming their password.
  + An email confirmation is sent after successful registration.
  + Duplicate registrations using the same email address are prevented.
  + Passwords must be at least 8 characters long and include one uppercase letter, one number, and one special character.

**FR-002: User Login**

* Description**:** The system shall allow users to log in using their email and password.
* Priority**:** High
* Acceptance Criteria**:**
  + Users can log in with a registered email and password.
  + Invalid login attempts display appropriate error messages.

**FR-003: Password Recovery**

* Description**:** The system shall provide password recovery options for users who forget their passwords.
* Priority**:** Medium
* Acceptance Criteria**:**
  + Users can request password recovery by providing their email address.
  + A password reset link is sent to the user's email.
  + Users can reset their password using the link provided.

### 2.1.2 **Project Cataloging**

**FR-004: Add New Research Projects**

* Description**:** The system shall allow users to add new research projects, including title, abstract, authors, and relevant keywords.
* Priorit**y:** High
* Acceptance Criteria**:**
  + Users can input project details and submit them.
  + The system saves the project details in the database.

**FR-005: Update and Delete Project Details**

* Descriptio**n:** The system shall enable users to update and delete project details.
* Priority**:** High
* Acceptance Criteria**:**
  + Users can modify existing project details.
  + Users can delete projects, and the system removes them from the database.

### **2.1.3 Search and Filtering**

**FR-006: Search Projects**

* Description**:** The system shall allow users to search for projects by title, name, or date.
* Priority**:** High
* Acceptance Criteria**:**
  + Users can enter search terms and view matching projects.

**FR-007: Filter Projects**

* Description**:** The system shall enable users to filter projects by domain, date, and status (active, completed, cancelled).
* Priority**:** High
* Acceptance Criteria**:**
  + Users can apply filters to narrow down project results.

### **2.1.4 Project Details Page**

**FR-008: View Project Details**

* Description**:** The system shall provide a detailed view of each project, including title, name, authors, email, description and related document.
* Priority: High
* Acceptance Criteria**:**
  + Users can view comprehensive project information on a dedicated page.

**FR-009: Display Project Status and Outcomes**

* Description**:** The system shall display the project's current status (ongoing, completed) and outcomes if applicable.
* Priority**:** Medium
* Acceptance Criteria**:**
  + Users can see the project's status and related outcomes.

### **2.1.5 User Roles and Permissions**

**FR-010: Admin Management**

* Description**:** The system shall allow admins to manage user accounts and project entries.
* Priority**:** High
* Acceptance Criteria:
  + Admins can add, update, or delete user accounts and projects.

**FR-011: Researcher Management**

* Description**:** The system shall allow researchers to manage their own project entries.
* Priority: High
* Acceptance Criteria:
  + Researchers can add, update, or delete their own projects.

## **2.2 Non-Functional Requirements**

### **2.2.1 Performance**

**NFR-001: Concurrent Users**

* Description**:** The system shall support at least 200 concurrent users without performance degradation.
* Priority**:** High
* Acceptance Criteria:
  + Performance tests show the system can handle 200 concurrent users.

**NFR-002: Page Load Time**

* Description: The system shall load the project details page within 2 seconds under normal load conditions.
* Priority**:** High
* Acceptance Criteria:
  + Page load times are measured to be under 2 seconds during testing.

### **2.2.2 Security**

**NFR-003: Data Encryption**

* Description**:** The system shall use HTTPs for all communications to ensure data encryption.
* Priority: High
* Acceptance Criteria**:**
  + All data transmissions are encrypted using HTTPS.

**NFR-004: Role-Based Access Control**

* Description: The system shall implement role-based access control to restrict access to sensitive data.
* Priority**:** High
* Acceptance Criteria:
  + Access control mechanisms are in place and verified through security testing.

**NFR-005: Password Strength**

* Description: The system shall require strong passwords (at least 8 characters, including one uppercase letter, one number, and one special character).
* Priority**:** Medium
* Acceptance Criteria:
  + Password policies are enforced during user registration and password changes.

### **2.2.3 Usability**

**NFR-006: User-Friendly Interface**

* Description**:** The system shall have a user-friendly interface accessible to users with varying levels of technical expertise.
* Priority: High
* Acceptance Criteria:
  + User interface designs are tested and refined based on user feedback.

### **2.2.4 Scalability**

**NFR-008: System Scalability**

* Description: The system shall be scalable to accommodate increasing numbers of projects and users without significant performance loss.
* Priority**:** High
* Acceptance Criteria:
  + Scalability tests show the system performs well under increasing loads.

**NFR-009: Modular Architecture**

* Description: The system architecture shall support easy addition of new features and modules.
* Priority: Medium
* Acceptance Criteria:
  + New features can be added with minimal impact on existing functionalities.

### **2.2.5 Reliability**

**NFR-010: System Uptime**

* Description**:** The system shall have an uptime of 99.9% to ensure availability.
* Priority: High
* Acceptance Criteria**:**
  + System uptime is monitored and meets the specified target.

### **2.2.6 Compliance**

**NFR-011: Data Protection Regulations**

* Description: The system shall comply with relevant data protection regulations to ensure user data privacy.
* Priority**:** High
* Acceptance Criteria:
  + Compliance checks are performed, and the system meets requirements.

# 3.Project Plan for the CRPR Project

## **3.1Timeline**

### **Month 1: Planning and Design**

* **Week 1-2: Project Kickoff and Initial Planning**

Kick-off Meeting**:** Define project scope, objectives, and deliverables.

Assign Roles and Responsibilities**:** Finalize the project team and their roles.

* **Week 3-4: Requirements Gathering and Analysis**

Functional and Non-Functional Requirements: Gather and document detailed requirements.

### **Month 2: Development and Implementation**

* **Week 1-2: System Design**

Architecture Design**:** Create the system architecture and design the database schema.

User Interface Design: Develop wireframes and design the user interface (UI) in HTML/CSS.

Approval of Design Documents**:** Get feedback and approval from stakeholders.

* **Week 3-4: Development Phase**

Backend Development: Implement Django models, views, and APIs.

Frontend Development: Develop the frontend using HTML/CSS and integrate with Django backend.

Middleware Development: Implement and test custom middleware for uptime and other features.

Integration: Connect frontend and backend components, ensuring proper data flow and functionality.

### **Month 3: Testing, Deployment, and Documentation**

* **Week 1-2: Testing Phase**

Unit Testing: Test individual components and functionalities.

Integration Testing: Test the integration of frontend and backend components.

* **Week 3: Deployment and Launch Preparation**

Prepare Deployment Environment**:** Set up production environment and perform final configurations.

Deployment**:** Deploy the CRPR application to the production server.

Final Testing in Production: Conduct final testing to ensure everything is functioning correctly.

* **Week 4: Documentation and Training**

Create User Documentation**:** Develop user manuals and help documentation.

Create Technical Documentation**:** Document the system architecture, codebase, and setup instructions.

Project Review Meeting: Review the project with stakeholders and gather feedback.

## **3.2 Work Breakdown Structure (WBS)**

### **3.2.1. Project Initiation**

Kickoff Meeting

Define Scope and Objectives

Assign Roles and Responsibilities

### **3.2.2. Requirements Gathering and Analysis**

Functional Requirements

Non-Functional Requirements

Stakeholder Interviews

Approval of Requirements Document

### **3.2.3. System Design**

Architecture Design

* Define System Architecture
* Design Database Schema

User Interface Design

* + - Develop Wireframes
    - Create UI Designs

### **3.2.4. Development Phase**

Backend Development

* + - Set Up Django Environment
    - Develop Models and Views
    - Implement APIs

Frontend Development

* + - Develop HTML/CSS
    - Integrate with Backend

Middleware Development

* + - Implement Custom Middleware
    - Test Middleware

Integration

* + - Connect Frontend and Backend
    - Test Data Flow

### **3.2.5. Testing Phase**

Unit Testing

* + - Test Individual Components

Integration Testing

* + - Test Component Integration

User Acceptance Testing (UAT)

* + - Conduct Testing with Stakeholders

### **3.2.6. Deployment and Launch Preparation**

Prepare Deployment Environment

* + - Set Up Production Server
    - Configure Final Settings

Deployment

* + - Deploy to Production

Final Testing in Production

* + - Verify Functionality

### **3.2.7. Documentation and Training**

* + Create User Documentation
  + Create Technical Documentation

# 4.Design and Architecture

## **4.1 System Architecture**

The system architecture consists of several key components, each responsible for different aspects of the system's functionality. Below is an overview of the primary components:

1. **Frontend**

* Web Interface: The user-facing part of the CRPR, allowing users to browse and search for research projects, view detailed information, and interact with various features.
* Responsive Design: Ensures compatibility across different devices and screen sizes.

1. **Backend**

* Web Server: Handles incoming requests, processes them, and sends appropriate responses to the client. Typically built using a Django framework (Python)
* Application Logic: Contains the core functionality of the CRPR, including handling user authentication, managing project data, and serving the appropriate views.

1. **Database**

* Database Management System: Stores all data related to research projects, users, and other relevant information. MySQL is the relational database used.

1. **APIs**

* RESTful APIs: Enable communication between the frontend and backend and facilitate integration with other systems and services.

1. **Security**

* Authentication and Authorization: Ensures that only authorized users can access certain features and data.
* Data Encryption: Protects sensitive information during transmission and storage.

1. **Deployment and Hosting**

* Cloud Services: Platforms like AWS, Azure, or Google Cloud are used for hosting the web server, database, and other components.
* CI/CD Pipeline: Automated tools and processes for continuous integration and deployment to ensure smooth updates and maintenance.

## **Design Diagrams**

### **4.2.1Context Diagram (Level 0 DFD)**

The Context Diagram provides an overview of the entire system, showing the system boundaries and interactions with external entities.

**Entities:**

User: A researcher or student interacting with the CRPR.

Admin: The system administrator managing the CRPR.

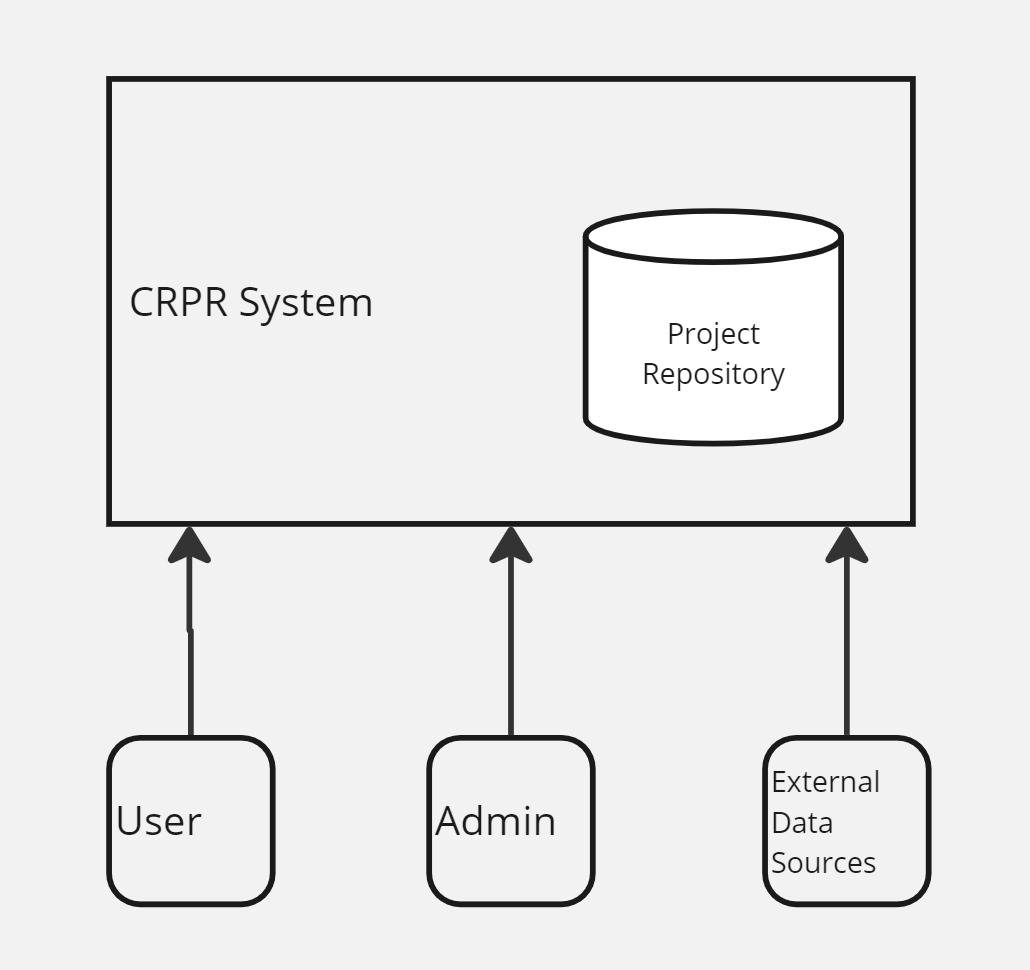
External Data Sources: Sources from which data is imported into the CRPR.

**Processes:**

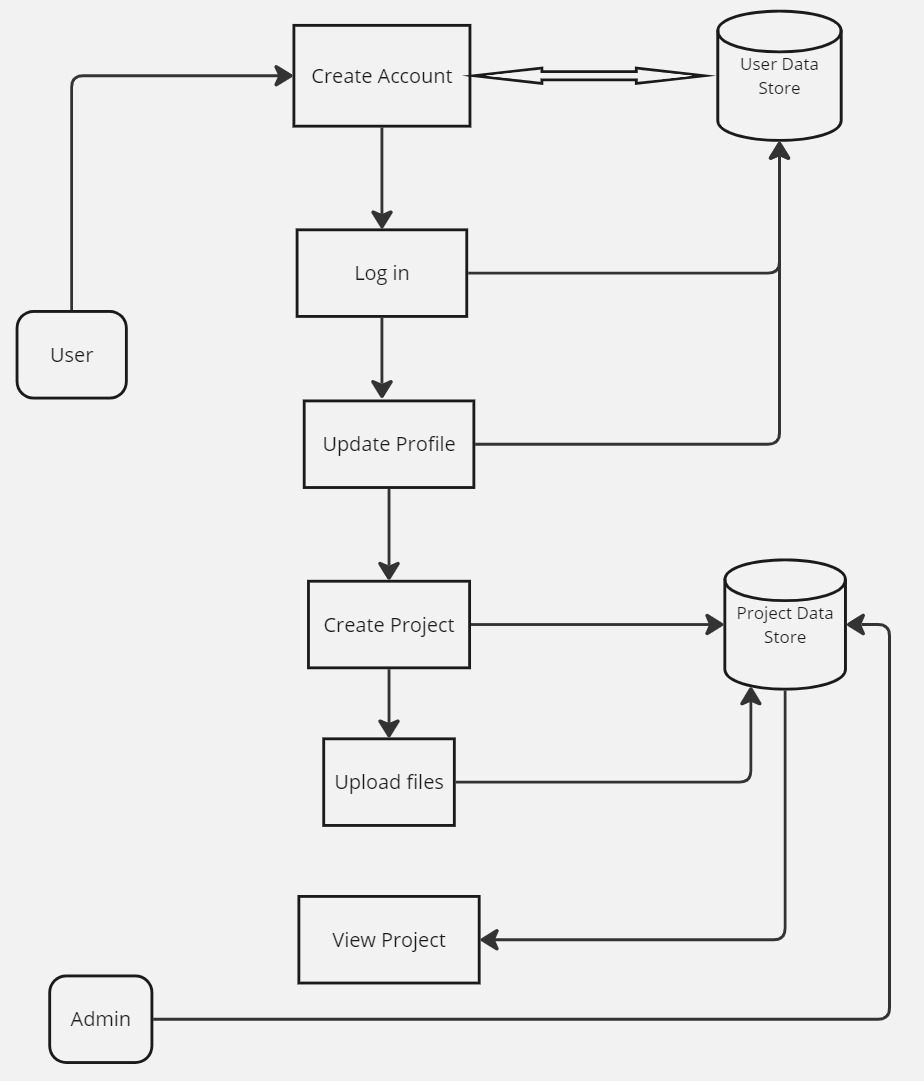
**CRPR System:** The main system processing user and admin requests.

**Data Stores:**

**Project Repository:** The central database storing all research project information.



Created using Miro app

Context Diagram

Created Using Miro App

**Description of Processes in the Diagram**

1. **Create Account** 
   * The user creates a new account, which updates the User Data Store.
2. **Log In** 
   * The user logs in to the system, validating credentials against the User Data Store.
3. **Update Profile** 
   * The user updates their profile information, which is saved in the User Data Store.
4. **Create Project** 
   * The user creates a new project, which is saved in the Project Data Store..
5. **Upload File** 
   * The user uploads a file related to a project, which is stored in the Project Data Store.
6. **View Project** 
   * The user views the details of a project, retrieving information from the Project Data Store.

## **4.3 Technical Specifications**

1. **Frontend**

* Languages: HTML5, CSS, JavaScript
* Frameworks/Libraries: Bootstrap (for styling)
* Responsive Design: Media queries, flexbox, grid layout

1. **Backend**

* Languages: Python
* Frameworks: Django (Python)
* API Standards: RESTful

1. **Database**

* DBMS: MySQL
* Schema:
  + Users Table: Stores user information (id, name, email, hashed password, role)
  + Projects Table: Stores project details (id, title, description, start date, end date, status, owner id)
  + Categories Table: Stores categories of projects (id, name)
  + Project Category Table: Many-to-many relationship between projects and categories (project, category id)

1. **APIs**

* **Endpoints:**
  + /api/projects: CRUD operations for projects
  + /api/users: CRUD operations for users
  + /api/categories: CRUD operations for categories
* **Authentication:** JWT (JSON Web Tokens)

1. **Security**

* Authentication: OAuth 2.0, JWT
* Data Encryption: TLS/SSL for data in transit, AES-256 for data at rest
* Input Validation: Server-side and client-side validation to prevent SQL injection, XSS, CSRF

1. **Deployment and Hosting**

* Cloud Provider: AWS
* Services:
  + EC2: for servers
  + RDS: for databases
  + S3: for static assets
* CI/CD Tools: GitHub Actions

# User Guide

Here we will guide you through the features of the CRPR and show you how to use and navigate through repository.

**Let us get Started.**

## **5.1Accessing the Platform**

* Open your web browser and navigate to the CRPR website: http://www.crpr.example.com.

### 5.1.1 **Logging** In

* Click on the "Login" link in the navigation bar.
* Enter your email and password.
* Click "Sign In" to access your account.

### **5.1.2 Registering an Account**

* Click on the "Create Account" button on the login page.
* Fill in the required fields (name, email, password).
* Click "Sign up" to create your account.

### **5.1.3. Recovering Your Password**

* Click on the "Forgot Password" link on the login page.
* Enter your email address.
* Follow the instructions sent to your email to reset your password.

## **5.2 Using the Repository**

### **5.2.1. Creating a New Project**

* Navigate to the "Create a Project" page from the Home page.
* Fill in the project details (level of study, postgraduate type, name, title, description, GitHub links etc.).
* Click "Next" to submit your project details.

### **5.2.2. Adding a Collaborator (Optional)**

After Submitting the project details, you will be directed to Collaborators page;

* Input the name and email of the Collaborator.
* Upload the image of the Collaborator.
* Click “Next” button.

### **5.2.3. Uploading Project files**

* In the Upload file page, click “Choose File” button to select the project file from your local machine.
* Enter the file description.
* Click “Submit” button to upload the project files.

### **5.2.4. Viewing Projects**

* Click the “View Projects” button from the Home page.
* Use the search bar to find projects by title and name.
* Apply filters to narrow down results by date, status or level of study.

### **5.2.5. Editing a Project**

* Locate your project in "My Projects" section in the Profiles page.
* Click the "Project box” under “Active Projects”.
* Click the “Update” button under the project details.
* Update the project details as needed.
* Delete unnecessary files by checking the box in the “Current files” section to upload new files.
* Click "Update" to save the changes.

### **5.2.6. Deleting a Project**

* Go to the "My Projects" section.
* Click the "Project box” under “Active Projects”.
* Click the "Cancel" button under the project details you want to remove.
* Confirm the deletion when prompted.

### 

### **5.2.7. Downloading Project**

* Click the “View Project” button from Homepage.
* Locate the needed project in the Projects Page.
* Click the “View More Details” button under the located project.
* Scroll down and move to the uploaded files section.
* Click the “Download” to install the project files.

## **5.3. User Roles and Permissions**

### **5.3.1. Admins**

* Manage user accounts.
* Access all projects and user information.

### **5.3.2. Researchers**

* Add, update, and delete their own projects.
* View and search the project catalog.

### **5.3.3. Guests**

* View and search the project catalog.
* No access to add or modify projects.

# 6.Deployment Plan for CRPR Project

## **6.1. Preparation**

**6.1.1 Code Review and Testing**

* Code Review: Ensure all code changes are reviewed and approved.
* Automated Testing: Run all unit tests and integration tests to verify code functionality.
* Manual Testing: Perform manual testing in a staging environment to catch any issues not covered by automated tests.

**6.1.2 Update Configuration Files**

* Production Settings: Update settings.py with production configurations:
  + Set DEBUG = False.
  + Configure ALLOWED\_HOSTS with the production domain or IP address.
  + Set up database settings with production credentials.
  + Configure static and media files settings.
* Environment Variables: Use environment variables for sensitive information (e.g., database credentials, secret key).

**6.1.3 Backup**

* Database Backup: Create a backup of the current production database.
* Application Backup: Backup the current version of the application files.

## **6.2 Environment Setup**

**6.2.1 Provision Server**

* Server Setup: Ensure the production server is up and running with appropriate specifications.
* Required Software:
  + Install Python from the official website.
  + Install Git from the official website.
  + Install and configure a web server (e.g., IIS or Apache).
  + Install a database server (MySQL).

**6.2.2 Clone Repository**

* Command Prompt: Open Command Prompt and navigate to the desired directory: cd path\to\directory
* Clone Repository: Clone the project repository: git clone <repository-url>

**6.2.3 Setup Virtual Environment**

* Navigate to Project Directory: cd <project-directory>
* Create Virtual Environment: python -m venv venv
* Activate Virtual Environment: venv\Scripts\activate

**6.2.4 Install Dependencies**

* Install Dependencies: pip install -r requirements.txt.

## **6.3. Deployment**

**6.3.1. Pull Latest Code**

* Update Repository: Pull the latest code from the repository: git pull origin main.

**6.3.2 Run Migrations**

* Apply Migrations: Apply database migrations: python manage.py migrate.

**6.3.3 Collect Static Files**

* Collect Static Files: python manage.py collectstatic.

**6.3.4 Configure Web Server**

**For IIS:**

* Install wfastcgi: pip install wfastcgi.
* Configure IIS:
  + Set up a new site in IIS and point it to the project directory.
  + Configure Fast CGI settings to point to the Django application.
  + Add a handler mapping for Fast CGI to handle Django requests.

**For Apache:**

* Install mod\_wsgi: pip install mod\_wsgi
* Update Apache Configuration

**6.3.5 Restart Web Server**

* Restart Web Server: Restart IIS or Apache to apply the changes.

## **6.4. Post-Deployment**

**6.4.1 Verify Deployment**

* Check Website: Access the website to ensure it is functioning correctly.
* Monitor Logs: Check server logs for any errors or issues.

**6.4.2 Set Up Monitoring**

* Monitoring Tools: Configure monitoring tools (e.g., New Relic, Sentry) to track application performance and uptime.

**6.4.3 Performance Testing**

* Load Testing: Conduct performance tests to ensure the application handles load effectively.

**6.4.4 Security Check**

* Security Settings: Verify that all security settings are correctly configured (e.g., HTTPS, secure cookies).

## **6.5 Rollback Plan**

**6.5.1 Restore Backups**

* Database Restoration: If issues are detected, restore the backup of the production database.
* Application Files Restoration: Restore the backup of the application files.

**6.5.2 Revert Code**

* Version Control: Use version control to revert to the previous stable version of the code.

**6.5.3 Restart Web Server**

* Restart Web Server: Restart IIS or Apache to apply the reverted changes.

**6.5.4 Notify Stakeholders**

* Communication: Inform stakeholders about the rollback and any potential impact on users.

## **6.6 Troubleshooting**

**6.6.1 Common Issues**

* Application Fails to Start:
  + Ensure correct Python version and dependencies.
  + Check settings.py and database connection.
* Static Files Not Loading:
  + Ensure static files are collected and configurations are correct.
  + Configure the web server to serve static files.
* Database Errors:
  + Run python manage.py migrate.
  + Check database settings and permissions.
* 500 Internal Server Error:
  + Check server logs for stack traces and error messages.
  + Verify web server configuration and environment variables.
* Permission Denied Errors:
  + Check file permissions and web server user permissions.

**6.6.2 Logging and Debugging**

* Enable Logging:
  + Update LOGGING configuration in settings.py to enable logging to a file.
* Monitor Logs:
  + Tail the log file to monitor real-time logs.
* Debugging Tools:
  + Use Django Debug Toolbar for development.
  + Insert breakpoints using import pdb; pdb.set\_trace ().
* Check Server Logs:
  + Review web server logs for errors or warnings.
* Validate Environment Variables:
  + Ensure all required environment variables are set correctly.

# 7. Important Notice: Credentials for Password Reset and Payment Processing

As we approach the hosting phase of the Computing Department Projects Repository website, it is important to address the current use of temporary developer credentials for critical functionalities.

**Current Configuration:**

1. Password Reset Emails: Currently configured with a developer's email account.

2. Subscription Payments: Integrated using a developer's PayPal Secret Key and ID.

Required Actions:

To ensure seamless and secure operation of the website, we urgently require the following official credentials from the university:

- University Email Account: This will be used for sending password reset emails to users.

- PayPal Business Account Details: Specifically, the PayPal Secret Key and Client ID associated with the university’s PayPal account for processing subscription payments.

**Disclaimer:**

While the website is operating with these temporary developer credentials, any issues, security breaches, financial losses, or damages arising from their use are outside our responsibility and control. It is imperative that these credentials be replaced with the official university-provided information as soon as possible to mitigate any risks.

Next Steps:

1. Provision of Credentials: Please provide the official university email and PayPal account credentials at the earliest convenience.

2. Credential Update: We will promptly update the website configuration with the provided credentials to ensure compliance and security.

We appreciate your immediate attention to this matter to ensure the successful deployment and operation of the Computing Department Projects Repository website.

For any questions or further assistance, please contact [+254705516200].

This version is more detailed and professional, clearly outlining the current situation, required actions, and the disclaimer regarding the use of developer credentials.

# 8.Maintenance and Support

## **8.1Maintenance Plan**

* **Schedule Regular Updates and Patches:**
  + Ensure the system is up-to-date with the latest security patches and feature enhancements.
  + Plan and implement a bi-monthly update schedule to address any emerging issues or improvements.
* **Monitor System Performance and User Feedback:**
  + Continuously monitor system logs, performance metrics, and user feedback to identify potential problems early.
  + Utilize tools like New Relic, Sentry, or custom monitoring scripts to track the application's health and performance.
* **Address Issues Promptly and Efficiently:**
  + Maintain a dedicated team to handle bug reports and issues raised by users.
  + Implement a ticketing system (such as JIRA or GitHub Issues) to track, prioritize, and resolve issues in a timely manner.

## **8.2.Support Plan**

* **Provide User Manuals and FAQs:**
  + Develop comprehensive user manuals that cover all aspects of using the CRPR system.
  + Create a Frequently Asked Questions (FAQ) section on the project's website to help users find quick answers to common questions.
* **Offer Dedicated Support Channels (Email, Chat, Phone):**
  + Set up dedicated support channels to assist users with their queries and issues.
  + Provide support via email at researchhubx@gmail.com, through a live chat feature on the website, and via phone at +254-705516200.

# 9.Appendices

## **Glossary**

* **API (Application Programming Interface):** A set of tools and protocols for building software applications. It defines how different software components should interact.
* **Bug:** An error or flaw in software that produces an incorrect or unexpected result.
* **CRPR (Computing Research Projects Repository):** A centralized platform for cataloguing and showcasing ongoing research projects within the Department of Computing at JKUAT.
* **CSS (Cascading Style Sheets):** A style sheet language used for describing the presentation of a document written in HTML or XML.
* **Django:** A high-level Python web framework that encourages rapid development and clean, pragmatic design.
* **HTML (Hypertext Markup Language):** The standard markup language for creating web pages and web applications.
* **JavaScript:** A programming language commonly used in web development to create interactive effects within web browsers.
* **Patch:** A piece of software designed to update or fix problems with a computer program or its supporting data.
* **User Manual:** A document that provides instructions or guidance on how to use a product or service.

## **References**

1. Django Documentation: <https://docs.djangoproject.com>
2. HTML Living Standard: https://html.spec.whatwg.org
3. CSS Documentation: <https://www.w3.org/Style/CSS/Overview.en.html>
4. JavaScript Guide: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
5. JKUAT Repository : <http://ir.jkuat.ac.ke/>
6. GitHub Guides: [Managing Repositories](https://guides.github.com/introduction/getting-your-project-on-github/)
7. Dryad FAQs: <https://datadryad.org/stash/requirements>

Contact Information For further support or questions, you can reach us at: -

**Email**: researchhubx@gmail.com

**Phone**: +254-705-516-200

Address: Department of Computing, Jomo Kenyatta University of Agriculture and Technology (JKUAT), Nairobi, Kenya

Feel free to reach out with any inquiries or feedback regarding the Computing Research Projects Repository (CRPR).