CHAPTER FOUR: IMPLEMENTATION

4.1 Introduction

This chapter details the implementation process of the Computing Research Projects Repository (CRPR) project. The implementation phase involved setting up the development environment, translating the system design into actual code, integrating the various modules, and deploying the application for use. The implementation was guided by the Agile methodology, which allowed for iterative development, continuous feedback, and adaptability throughout the project lifecycle. Best practices were followed to ensure the system was functional, efficient, and user-friendly.

4.2 Development Environment

The CRPR project was developed in a Windows environment using the following tools and technologies

Components	**Technology Used**
Operating System	Windows 10
Frontend	HTML, CSS , Javascript
Backend	Django
Database	PostgreSQL
Version Control	Github
API testing	Postman
Code editor	Visual Studio Code
Continous integration	Github
Deployment	PythonAnywhere
UI enhancements	Bootstrap

· **IDE/Code Editor:**

Used **VSCode** for writing code, which provide features like syntax highlighting, autocompletion, and integrated terminals to enhance productivity.

· **Version Control:**

Implemented **Git** and **GitHub** to manage code versions. This allows for version tracking, collaboration, and easy rollback of changes if necessary.

· **Environment Management:**

Used **virtual environments** to isolate dependencies.

Frameworks:

Used **Django** for backend development to handle the business logic, models, and database interactions.

Developed the frontend using **HTML**, **CSS**, and **JavaScript** to create an interactive and responsive UI.

Chose **PostgreSQL** for the production-ready database solution.

The choice of Django as the backend framework allowed for rapid development, built-in admin functionality, and robust security features.

2. Testing Setup

Unit Testing:

Used **Django's built-in testing framework** for unit testing various application components such as models, views, and templates.

· **Test Coverage:**

Incorporated **coverage.py** to check test coverage across the codebase. This helps ensure all critical areas are tested.

· **Frontend Testing:**

Used **Jest** for unit testing frontend components, forms, and UI elements.

· **Continuous Integration (CI):**

Integrated CI tools like **GitHub Actions**, **Travis CI** to automatically run tests upon each code push. This ensured new code doesn't break existing functionality.

3. Deployment Setup

Deployment Platforms:

Deployed the app on **Heroku** for simplicity and ease of setup.

Also considered using **AWS EC2** and **DigitalOcean** for more customization and scalability in the future.

Used **Docker** for containerization, ensuring consistent application behavior across different environments (development, testing, and production).

Database Configuration:

Configured **PostgreSQL** for production use.

Managed sensitive data, including database credentials and secret keys, through environment variables directly through Heroku’s dashboard.

Static and Media Files:

Configured Cloudflare for serving static files (CSS, JavaScript) and media files (images, documents) in the production environment.

CI/CD Pipeline:

Set up a CI/CD pipeline using **GitHub Actions** to automatically deploy changes to production whenever code is pushed to the main branch..

4.3 System Implementation

Purpose

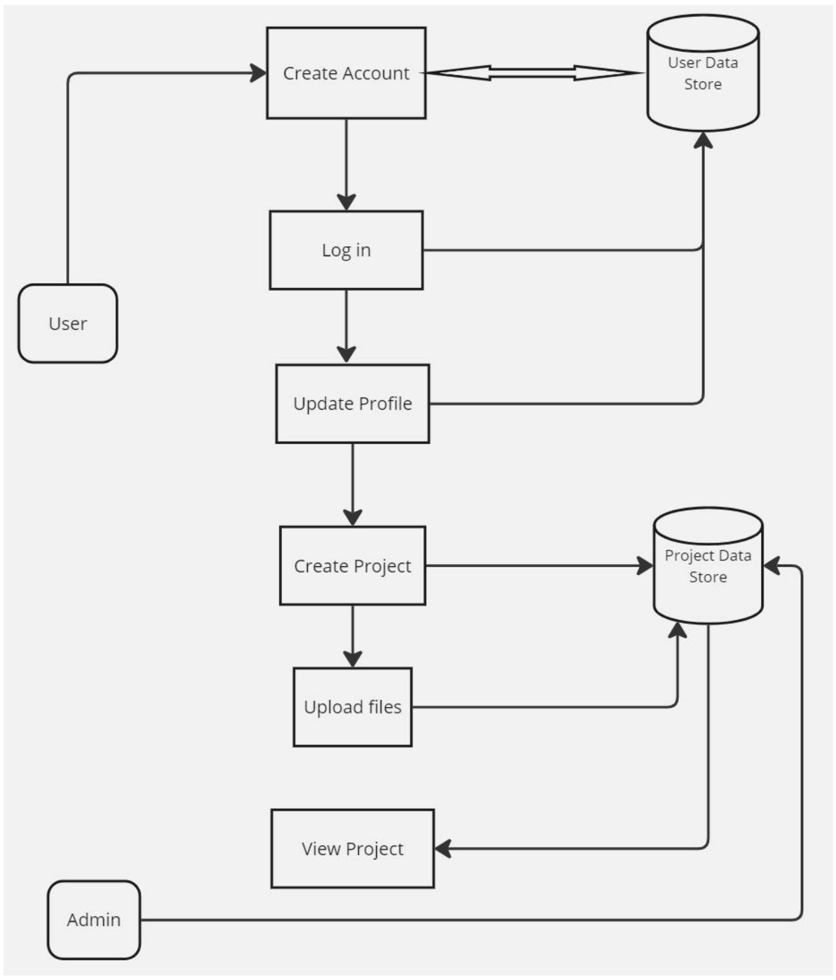
To illustrate how the CRPR project team planned and executed the system using the Agile methodology, organized into well-defined sprints with clear roles, timelines, and deliverables.

Agile Methodology Overview

The CRPR project followed the **Agile methodology**, promoting adaptability, iterative progress, and team collaboration. The development process was broken into **five sprints**, each one week long, focusing on key milestones. Regular retrospectives allowed the team to evaluate and refine the project incrementally.

Sprint Breakdown

Sprint	Timeline	Key Activities	Deliverables	Team roles
Sprint 1	(Jan 20 – Jan 31)	- Setup GitHub repo - Design wireframes - Create UI mockups	Wireframes Mockups GitHub repo	UI/UX Designer, Git Manager
Sprint 2	(Feb 3 – 14)	- Develop login/logout system - Create dashboard UI	Functional Auth Module Dashboard UI	Backend Developer, Frontend Developer
Sprint 3	(Feb 17 – Feb 28)	- Implement models/views - Add catalog features - CRUD operations	Backend functionality Project Catalog	Backend Developer, Database Designer
Sprint 4	(Mar 3– Mar 14)	- Write unit tests - Debug issues - Set up CI	Test Results CI Pipeline Resolved Bugs	Test Engineer, DevOps
Sprint 5	(Mar 17 – Apr 10)	- Final integration - Deploy on Heroku - Map domain	Live Application Deployment Docs Demo Link	Full Stack Developer, Deployment Lead



Context Diagram

4.4 Module-by-Module Implementation

a) User Authentication Module

The user authentication module ensures secure access to the system. The following implementations were carried out:

JWT-Based Login:
Upon login, users receive a JSON Web Token (JWT), which is used to authenticate future API requests. The token is securely stored on the client side (e.g., localStorage) and attached to HTTP headers during requests.

Role-Based Access Control:
Users are assigned roles (e.g., Admin, Supervisor, Researcher). Role-based middleware was implemented to restrict access to specific routes and functionalities based on the user's role.

Example:

/admin/* routes are accessible only to users with the Admin role.

/projects/my-projects is available to authenticated Researchers.

b) Core Business Logic

The CRPR platform enables project registration, supervision tracking, and status management. Key functionalities were handled through Django's backend services.

Project Lifecycle Management:
Users can submit new project proposals, which are then approved or rejected by supervisors/admins.

Backend Views & Services:

create_project_view: Handles new project submissions.

project_status_update_view: Allows status tracking and feedback from supervisors

supervisor_assignment_service: Automatically maps projects to available supervisors.

Modules Covered:

Project Registration

Supervisor Feedback and Comment

Department-wide Catalog Browsing

Advanced Search and Filters by Project Title, Year, and Status

c) API Implementation

The system follows RESTful API principles to promote modularity and scalability.

Django REST Framework (DRF) was used to build and expose endpoints for both frontend consumption and third-party integration.

Authentication Handling:
All protected API endpoints require a valid JWT passed in the Authorization header.

Tools Used for Testing:

Postman: Used for manual testing and simulating role-based access.

Swagger : Auto-generated API documentation for developers to understand available routes and payload formats

4.5 Frontend Development

Purpose

The frontend of the Computing Research Projects Repository (CRPR) was designed to ensure intuitive navigation, responsiveness, and accessibility for users including students, supervisors, and administrators.

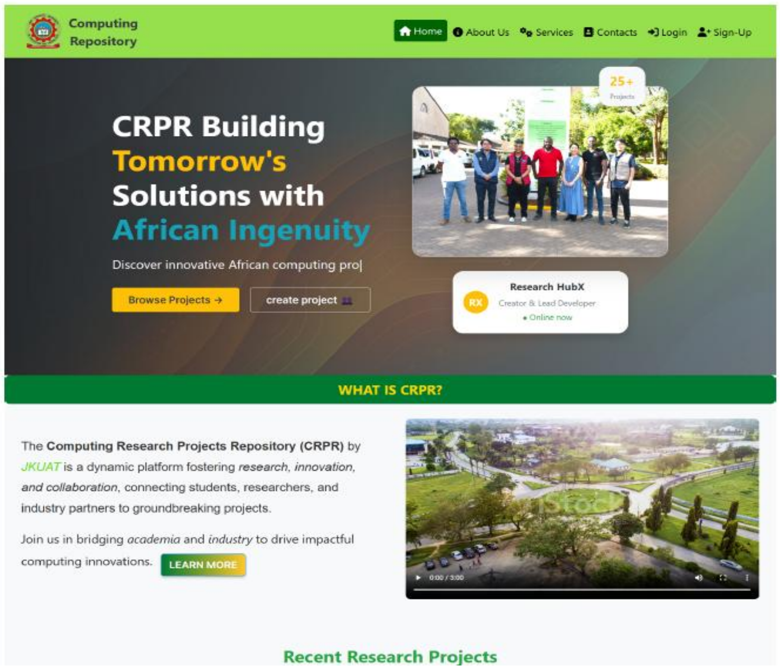
Tools & Technologies Used

- ✓ **HTML5 & CSS3** – Structured the content and styled the UI components.
- ✓ **JavaScript (ES6)** – Provided interactivity and dynamic behavior across the UI.
- ✓ **Bootstrap 5** – Utilized for responsive design and layout consistency.
- ✓ **Django Templates** – Used in combination with Django's views to serve dynamic content

Page Development Design

1. Landing Page

Introduces the platform with key features and navigation links. It includes a call-to-action for login or registration.

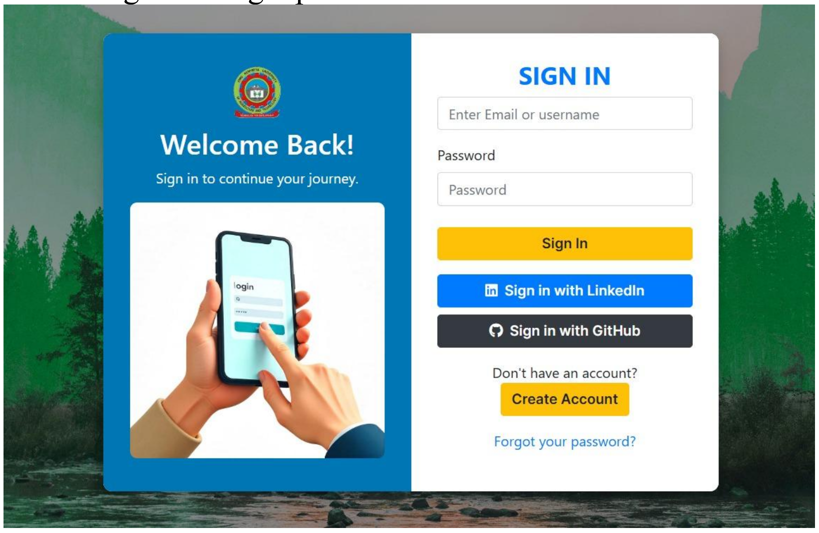


The **Computing Research Projects Repository (CRPR)** by *JKUAT* is a dynamic platform fostering *research, innovation, and collaboration*, connecting students, researchers, and industry partners to groundbreaking projects.

Join us in bridging *academia* and *industry* to drive impactful computing innovations.

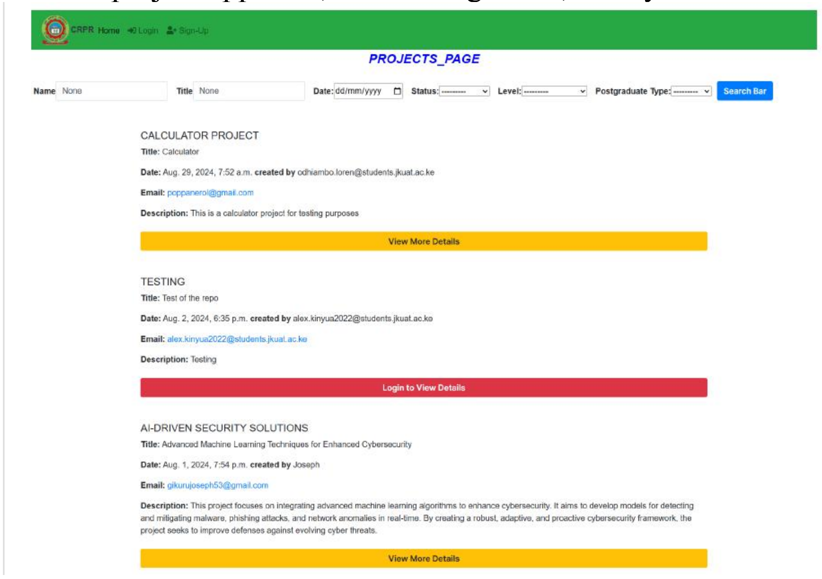
2. Login/Registration

Secure login and signup forms with validation and error feedback.



3. Admin Panel

Enables project approval, user management, and system monitoring for administrators.



PROJECTS_PAGE

Name None **Title** None **Date:** dd/mm/yyyy **Status:** --------- **Level:** --------- **Postgraduate Type:** --------- Search Bar

CALCULATOR PROJECT

Title: Calculator

Date: Aug. 29, 2024, 7:52 a.m. **created by** odhiambo.loren@students.jkuat.ac.ke

Email: poppanerol@gmail.com

Description: This is a calculator project for testing purposes

View More Details

TESTING

Title: Test of the repo

Date: Aug. 2, 2024, 6:35 p.m. **created by** alex.kinyua2022@students.jkuat.ac.ke

Email: alex.kinyua2022@students.jkuat.ac.ke

Description: Testing

Login to View Details

AI-DRIVEN SECURITY SOLUTIONS

Title: Advanced Machine Learning Techniques for Enhanced Cybersecurity

Date: Aug. 1, 2024, 7:54 p.m. **created by** Joseph

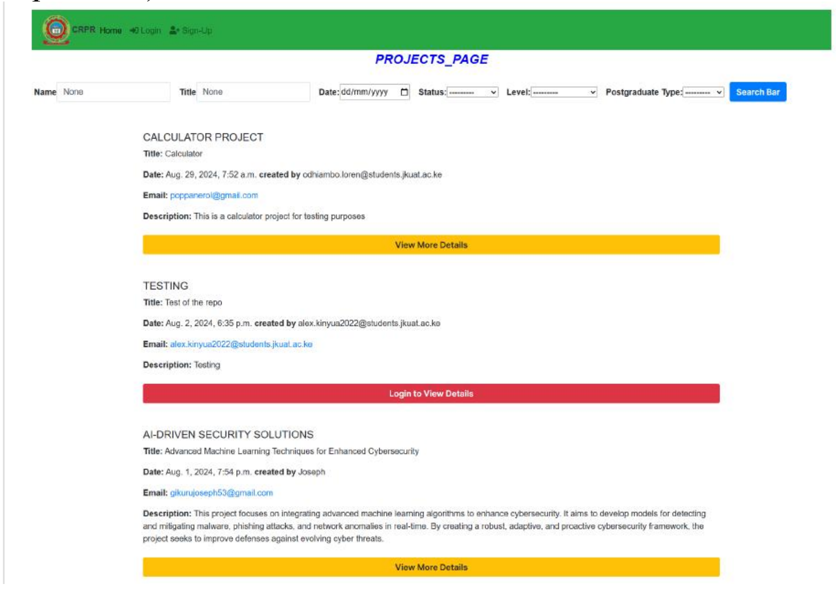
Email: gikurujoseph53@gmail.com

Description: This project focuses on integrating advanced machine learning algorithms to enhance cybersecurity. It aims to develop models for detecting and mitigating malware, phishing attacks, and network anomalies in real-time. By creating a robust, adaptive, and proactive cybersecurity framework, the project seeks to improve defenses against evolving cyber threats.

View More Details

4. Project Catalog

Displays all submitted projects with filtering options (e.g., by supervisor, year, department).



PROJECTS_PAGE

Name None Title None Date: dd/mm/yyyy Status: --------- Level: --------- Postgraduate Type: --------- Search Bar

CALCULATOR PROJECT

Title: Calculator

Date: Aug. 29, 2024, 7:52 a.m. **created by** odhiambo.loren@students.jkuat.ac.ke

Email: poppanerol@gmail.com

Description: This is a calculator project for testing purposes

View More Details

TESTING

Title: Test of the repo

Date: Aug. 2, 2024, 6:35 p.m. **created by** alex.kinyua2022@students.jkuat.ac.ke

Email: alex.kinyua2022@students.jkuat.ac.ke

Description: Testing

Login to View Details

AI-DRIVEN SECURITY SOLUTIONS

Title: Advanced Machine Learning Techniques for Enhanced Cybersecurity

Date: Aug. 1, 2024, 7:54 p.m. **created by** Joseph

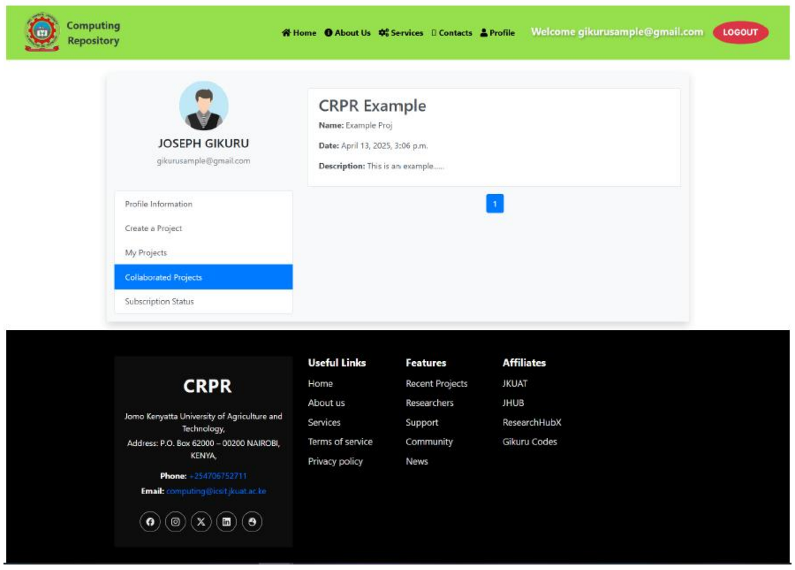
Email: gikurujoseph53@gmail.com

Description: This project focuses on integrating advanced machine learning algorithms to enhance cybersecurity. It aims to develop models for detecting and mitigating malware, phishing attacks, and network anomalies in real-time. By creating a robust, adaptive, and proactive cybersecurity framework, the project seeks to improve defenses against evolving cyber threats.

View More Details

5. Project Submission Form

Allows researchers to submit new project proposals with fields like title, abstract, description , etc.



JOSEPH GIKURU

gikurusample@gmail.com

CRPR Example

Name: Example Proj

Date: April 13, 2025, 3:06 p.m.

Description: This is an example.....

CRPR

Jomo Kenyatta University of Agriculture and Technology,

Address: P.O. Box 62000 – 00200 NAIROBI, KENYA,

Phone: +254706752711

Email: computing@icsit.jkuat.ac.ke

Useful Links	Features	Affiliates
Home	Recent Projects	JKUAT
About us	Researchers	JHUB
Services	Support	ResearchHubX
Terms of service	Community	Gikuru Codes
Privacy policy	News	

Responsive Design

The UI was built mobile-first, ensuring optimal usability across devices.

Media queries and Bootstrap’s grid system ensured content adapts to screen sizes.

Forms and tables were tested for usability on both desktop and mobile interfaces.

Notable UX Features

Dark Mode Compatibility (optional toggle added for accessibility).

Tooltips and **form validation** for better user guidance

Navigation Bar dynamically adapts based on user role (Student, Supervisor, Admin).

4.6 Database Implementation

Purpose

The CRPR (Computing Research Projects Repository) system required a structured, reliable, and relational data store to manage users, projects, and roles. The backend logic was tightly coupled with the database design to ensure smooth querying and efficient data retrieval.

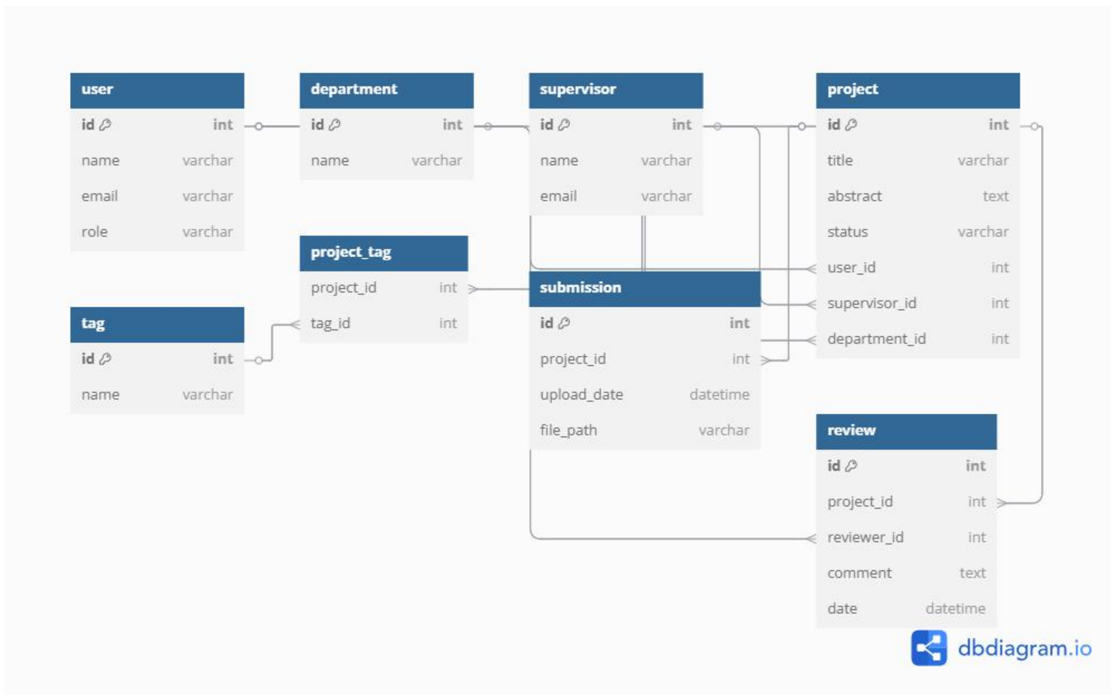
Database Type

PostgreSQL was used as the primary database.

It is a robust, open-source **relational SQL database** suitable for handling structured data with relations and constraints.

Key Tables

Table/Model	Description
Users	Stores registered user data including name, email, password (hashed), and role.
Projects	Contains project details like title, abstract, department, status, and submission date.
Supervisors	Maintains info on faculty who can review and supervise projects.
Feedback	Stores comments, ratings, or decisions by supervisors on submitted projects.
Departments	Defines the structure for organizing projects under specific academic units.



Optimization & Indexing

Primary Keys: Auto-incremented id fields on all tables.

Foreign Keys: Used to enforce referential integrity (e.g., project.user_id, feedback.project_id).

Indexes:

- Indexes were created on commonly queried fields like user_id, project_status, and submission_date to improve search performance.
- Composite indexes were used on (department_id, status) for dashboard filtering.

4.7 Testing and Debugging

Purpose:
To ensure the system functions as intended, identify bugs, and maintain quality throughout development.

What was done:

1. **Unit Testing:**
   Used Django's built-in testing framework to test models, views, and custom logic.
   Example command: `python manage.py test`
2. **Integration Testing:**
   Ensured different modules (e.g., login, dashboard, catalog) work together correctly.
3. **End-to-End Testing:**
   Emulated real user scenarios using **Cypress**, e.g., login, project creation, catalog filtering.
4. **Manual Testing:**
   Performed by team members for UI/UX validation and exploratory bug identification.
5. **Bug Tracking:**
   Bugs were tracked using GitHub Issues and categorized by severity (e.g., critical, moderate, low).
6. **Debugging Strategy:**

- ➢ Used Django Debug Toolbar and browser dev tools.
- ➢ Applied `print()` and logging statements during local testing.
- ➢ Sprint 3 included major debugging of project listing logic and date validation.

4.8 Security Implementation

Purpose:
To ensure data protection and secure interaction between users and the system through various best practices and technologies.

Key Strategies and Implementations:

1. Secure Login

- **JWT (JSON Web Tokens):**
  Used for maintaining user sessions securely. After a successful login, a token is issued and stored on the client side for authenticated requests.
- **Bcrypt:**
  All user passwords are hashed using bcrypt before being stored in the database, ensuring password confidentiality even if the database is compromised

2. Attack Prevention

- **XSS (Cross-Site Scripting) Mitigation:**
  All form inputs are validated and sanitized on both client and server sides to block malicious scripts.
- **SQL Injection Protection:**
  Django ORM is used for database interactions, which inherently protects against raw SQL injection.

3. HTTPS and Secure Communication

- The application enforces **HTTPS** in production to encrypt data between clients and the server.
- SSL certificates are set up through the deployment platform (e.g., Heroku/Cloudflare).

4. Role-Based Access Control

Users are assigned roles (Admin, Reviewer, Contributor, etc.).

Each route/view is protected with decorators/middleware to ensure only users with the appropriate roles can access sensitive actions (e.g., modifying records, viewing certain dashboards).

5. Input Validation

Form fields are validated using both **frontend (JavaScript)** and **backend (Django Forms/Serializers)** to enforce proper data structure and prevent invalid/malicious inputs.

4.9 Performance Optimization

Purpose:
To enhance the responsiveness, scalability, and efficiency of the CRPR system by optimizing both frontend and backend components.

Frontend Optimization

- **Lazy Loading:**
  Images, scripts, and certain components (e.g., project detail pages) are loaded only when needed, reducing initial page load time and improving perceived speed.
- **Minification & Compression:**
  JavaScript, CSS, and HTML files are minified using build tools (like Django Compressor or Webpack), and static assets are GZIP-compressed to reduce bandwidth usage.
- **Responsive Design Performance:**
  Lightweight CSS frameworks and efficient media queries ensure the site runs smoothly on all screen sizes with minimal reflows.

Backend Optimization

- **Database Query Optimization:**
  ORM queries are optimized using select_related and prefetch_related to minimize database hits during page loads and data-intensive operations.
- **Pagination:**
  List views (e.g., research projects, user submissions) implement pagination to reduce load times and memory usage.
- **Caching:**
  Frequently accessed data (such as category listings or featured projects) are cached using Django's cache framework (e.g., in-memory caching with Redis or Memcached).

Monitoring and Diagnostics

- **Performance Monitoring Tools:**
  Although CRPR is a small-scale system, performance profiling was done using:

  Django Debug Toolbar for development profiling

  Simple logging to track response times and query execution.

- **Scalability Readiness:**
  Views and services are structured to allow easy integration with external tools like:

  Grafana and **Prometheus** (for future monitoring in production).

  Sentry (for real-time error tracking and alerting).

Example:

> Pagination was applied on the /projects route to load only 10 entries per page. Project images are lazy-loaded to reduce initial data transfer. Django templates are cached to improve server response times during peak usage.

4.10 CI/CD and Deployment

Purpose:
This section outlines how the CRPR project was deployed using modern CI/CD practiCI/CD Pipeline

We used **GitHub Actions** for Continuous Integration and Continuous Deployment. The pipeline was configured to:

- ✓ Automatically run tests on every push or pull request.
- ✓ Build the Django project.
- ✓ Deploy to Heroku upon a successful push to the `main` branch.

Hosting Platform

We hosted CRPR on **Heroku**, taking advantage of its free tier for development, PostgreSQL integration, and easy deployment from GitHub.

Deployment Steps

1. **Connect GitHub to Heroku:**
   Linked the CRPR GitHub repository to the Heroku dashboard for automatic deploys.
2. **Configure environment variables:**
   Set secret keys and database credentials via Heroku's Config Vars.
3. **Push to `main` branch:**
   This triggers GitHub Actions, which runs the test suite and deploys if tests pass.
4. **Monitor deployment status:**
   Deployment logs were checked on both GitHub and Heroku for issues.

4.11 Collaboration and Workflow

Purpose:
This section highlights the team's collaborative approach, tools used, and development workflow to build and manage the CRPR project effectively.

Collaboration Tools

To ensure smooth coordination and transparency throughout the project, the following tools were used;

GitHub:
Used for version control, issue tracking, and code reviews. GitHub Issues were linked to specific tasks and sprints.

WhatsApp:
For real-time communication, team discussions, and sharing quick updates.

Workflow Process

The team followed an **Agile** methodology, broken into 1-week sprints. Here's how we managed the workflow:

Sprint Planning:
At the start of each week, the team reviewed pending tasks and assigned new ones based on priorities.

Daily Check-ins (Async/Chat-Based):
Developers updated task status and flagged blockers in Whatsapp.

Sprint Review & Retrospective:
At the end of each sprint, the team reviewed completed features, bugs, and suggested improvements for the next sprint.

Version Control Strategy (GitHub)

To maintain clean and collaborative code development, the following Git workflow was adopted:

Branch Naming:
Feature branches followed the format: feature/login, fix/bug-auth, or chore/setup.

Pull Requests (PRs):
All code was merged via pull requests. PRs required reviews from at least one team member to ensure code quality.

Merge Strategy:

main: Stable and production-ready code.

dev: Integration branch for completed features.

feature/*: Short-lived branches for specific tasks.

4.12 Challenges and Solutions

Purpose:
This section reflects on some of the key challenges faced during the development of the **Computing Research Projects Repository (CRPR)** and the strategies used to overcome them. Each obstacle contributed to valuable learning experiences that helped improve team coordination, technical problem-solving, and workflow optimization.

1. Development Delays

Challenge:
Some sprint tasks took longer than expected due to complex backend logic and dependency on design mockups.

Solution:
The team adjusted sprint scopes, prioritized core features first, and implemented buffer time between sprints to handle overflow tasks. Agile's flexibility helped in quick rescheduling without blocking progress.

2. Merge Conflicts

Challenge:
Simultaneous development on the same files led to merge conflicts, especially in models and templates.

Solution:
The team adopted stricter **Git branching practices**, clearly assigning files/modules to specific developers. Daily commits and smaller, more frequent pull requests reduced the likelihood of major conflicts.

3. Bugs in User Authentication

Challenge:
During Sprint 2, users were not being properly redirected after login/logout due to session mishandling.

Solution:
The issue was resolved by carefully debugging session storage and improving

token validation logic. Unit tests were added to cover edge cases in login/logout flows.

4. CI/CD Pipeline Misconfiguration

Challenge:
Initial GitHub Actions workflows failed to deploy to Heroku due to missing environment variables and incorrect build paths.

Solution:
The pipeline was restructured step-by-step. Secrets were securely added in GitHub, and deployment was tested in a staging branch before pushing to production.

5. Time Management

Challenge:
Some team members had overlapping academic or personal commitments that affected daily progress.

Solution:
The team used Trello to break down tasks into subtasks and estimated effort levels (high, medium, low). This allowed better planning around available working hours and more realistic timelines.

6. Frontend Styling Inconsistencies

Challenge:
Inconsistent styling and layout mismatches occurred when multiple developers worked on the UI.

Solution:
A basic style guide was introduced with reusable CSS components. A single developer was assigned to consolidate frontend styles during the final integration sprint.

4.13 Summary

The implementation phase of the Computing Research Projects Repository (CRPR) showcased a well-structured and collaborative development approach. Leveraging the Agile methodology, the project was broken down into five sprints, each with clearly defined goals and deliverables—from initial setup and UI mockups to core functionality, testing, and final deployment.

Key modules such as user authentication (secured via JWT and bcrypt), role-based access control, and project catalog management were successfully implemented. RESTful APIs were built and tested using Postman, ensuring seamless integration between the frontend and backend. The database design, backed by a relational model, facilitated efficient data organization and querying, while indexing improved performance.

The user interface was crafted with responsive design principles, ensuring accessibility across devices. Continuous Integration/Continuous Deployment (CI/CD) pipelines using GitHub Actions enabled smooth testing and deployment to Heroku, with domain mapping finalized in the last sprint.

With all core modules integrated, the application is now ready for extensive testing, stakeholder review, and user feedback. The team remains committed to iterating based on real-world use to enhance usability and functionality.