SUNWAY COLLEGE KATHMANDU

CREATING AI LEADERS

IN ACADEMIC PARTNERSHIP WITH

BIRMINGHAM CITY University

PROFESSIONAL PLACEMENT YEAR REPORT

Spring (March 2023) Batch

Sunway International Business School

Maitidevi, Kathmandu, Nepal

Bibek Shah

23189619

BSc(Hons) Computer and Data Science

Faculty of Computing, Engineering and The Built Environment

Declaration

This report is prepared as a partial requirement for the Professional Placement Year for BSc(Hons) Computer and Data Science.

.....

STUDENT NAME: Bibek Shah

BCU ID: 23189619

DATE: 2025/06/01

ENDORSED BY

.....	.....
ACADEMIC SUPERVISOR	FIELD SUPERVISOR
Date:	Date: .....
Stamp :	Stamp : 

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prixa Technologies, located in Jawalakhel, Lalitpur, for providing me the opportunity to join their team as a Django Intern and complete my Professional Placement Year within such a dynamic and forward-thinking environment. This internship has been an incredibly valuable experience, significantly enhancing my understanding of the tech industry and allowing me to refine my technical and design skills in real-world applications.

A special thanks to my field supervisor, Amrita Khuswaha, whose consistent guidance, insightful feedback, and mentorship have been instrumental in my professional growth. Her support helped me overcome challenges and expand my knowledge in areas such as user experience and interface design. I would also like to thank my academic supervisor for their continuous encouragement and constructive input throughout both the placement and the report-writing process. My heartfelt appreciation goes to the entire team at Prixa Technologies, especially the Django and backend teams, for their collaboration, warmth, and for fostering a truly supportive work culture. Finally, I am deeply thankful to Sunway International Business School and Birmingham City University for organizing and facilitating this placement program, which has provided me with an invaluable opportunity to bridge academic learning with real-world experience.

Contents

1. Introduction

1.1 Introduction about the company

Prixa Technologies is a forward-thinking technology company headquartered in Jawalakhel, Lalitpur, Nepal, renowned for its expertise in digital media solutions, artificial intelligence (AI), and large-scale content delivery systems. Since its inception in 2015, Prixa has been a driving force in Nepal's digital transformation, with a strong focus on enhancing the online presence of media organizations and delivering scalable, high-impact IT solutions tailored to the needs of the local market.

With a solid foundation in developing AI-powered applications, cloud-based platforms, and custom enterprise solutions, Prixa serves a diverse client base across multiple industries, with a particular emphasis on the media sector. Its flagship product, Snowberry, is a robust content management system used by more than 50 media companies, enabling efficient handling of high-volume traffic and seamless content delivery.

Expanding beyond traditional services, Prixa has also ventured into cutting-edge AI-driven platforms such as News Talk, an application that converts written news articles into spoken content in Nepali, powered by its proprietary AI voice technology, Riri. This innovation exemplifies the company's commitment to making information more accessible and engaging for Nepali audiences.

Currently, Prixa Technologies offers a comprehensive range of services, including:

- AI-powered content delivery and management systems
- Full-stack web and mobile application development
- UI/UX design and enterprise-level system architecture
- Cloud hosting solutions and IT consultation
- Social media marketing and advertisement analytics

With a mission rooted in innovation and local impact, Prixa Technologies continues to lead the way in building Nepal's digital future.

2. Organization's Background

2.1 History

Established in 2015, Prixa Technologies is a leading technology company based in Kathmandu, Nepal. The company began its journey focusing on web and mobile application development, IT consulting, and business process automation. Over the years, it has significantly broadened its expertise to include cutting-edge technologies such as artificial intelligence (AI), machine

learning (ML), big data analytics, and content management systems. Operating with a team of over 70 skilled professionals, Prixa has positioned itself at the forefront of Nepal's tech industry by delivering scalable, secure, and innovative digital solutions tailored to a wide range of clients.

Prixa Technologies is known for its groundbreaking work in AI and digital infrastructure, with notable innovations like RiRi, Nepal's first AI-powered Nepali language conversational platform, and Snowberry, a high-performance media management system capable of handling giga-scale traffic for news organizations. These innovations demonstrate Prixa's commitment to solving real-world problems through advanced technology. In recognition of its achievements, the company was awarded the National ICT Innovation Award in 2023 by the Ministry of Information and Communications, acknowledging its outstanding contribution to Nepal's technology sector and its leadership in AI innovation.

2.2 Mission

Prixa Technologies is committed to delivering top-quality web development services designed to help businesses grow and thrive in the digital space. The company places a strong emphasis on creating user-friendly, reliable, and innovative websites and applications that are customized to meet the specific needs of each client. By focusing on intuitive design and robust functionality, Prixa ensures that its digital products not only look appealing but also perform seamlessly across all devices.

The core mission of Prixa Technologies is to provide effective digital solutions that enhance overall business performance and improve the customer experience. By combining technical expertise with a deep understanding of market trends, the company helps clients achieve their goals through scalable, high-impact web solutions. Whether it's a startup or an established enterprise, Prixa aims to be a trusted partner in driving digital transformation and long-term success.

2.3 Objectives

- To deliver high-quality, responsive, and scalable web development solutions for clients across various industries.
- To continuously adopt and integrate the latest technologies and best practices in web development.
- To provide excellent customer service by understanding client needs and offering tailored digital solutions.
- To support businesses in enhancing their online presence and digital transformation.
- To build long-term partnerships with clients based on trust, innovation, and reliability.
- To invest in the professional growth of the team and fostering a creative, collaborative work environment.

2.4 Product/Services

- Prixa Technologies offers a wide range of digital solutions focused on helping businesses grow through technology.
- Their core services include web development, mobile application development, and custom software solutions tailored to client needs.
- They also specialize in IT consulting, system integration, UI/UX design, and business process automation.
- Prixa has developed innovative products such as RiRi, an AI-powered Nepali language conversational platform, and Snowberry, a high-traffic news media management system.

3. Description on projects

During my professional placement at Prixa Technologies, which began on March 17 and completed on June 17, I have actively worked on several self-initiated projects alongside assigned tasks. This opportunity has enabled me to contribute to real-time development processes while gaining practical, hands-on experience in a professional and collaborative environment. The placement has significantly enhanced my technical skills, problem-solving abilities, and understanding of industry-standard practices in software development.

3.1 Real-Time Chat App

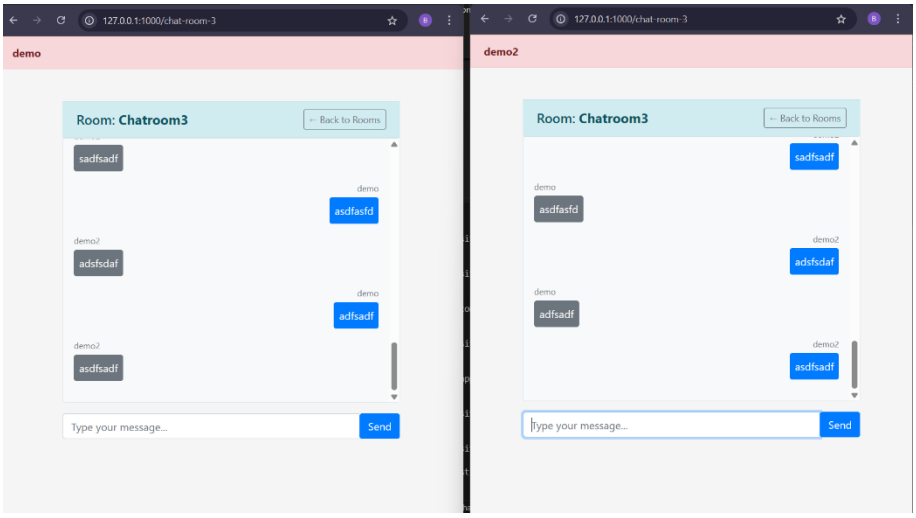


Fig1. Real time chat app

This Chat Web Application is a robust, real-time messaging platform developed using Django for the backend and Django Channels to enable asynchronous communication via WebSockets.

Designed to provide a modern and responsive user experience, the application allows users to register or log in securely and access a list of available chat rooms. Each chat room is dynamically generated and uniquely identified using a URL-friendly slug, allowing for clean navigation and user-friendly URLs. Upon entering a room, a persistent WebSocket connection is established between the client and the server, facilitating real-time, bidirectional communication without requiring page refreshes.

Messages sent by users are transmitted through the WebSocket, processed server-side using asynchronous consumers, and then broadcasted instantly to all participants in the same room. This real-time interaction creates a seamless and engaging user experience similar to contemporary chat platforms. Additionally, all messages are stored in a relational database, ensuring chat history persistence and allowing users to revisit previous conversations at any time. The application also implements user authentication to ensure secure access and personalized interaction within chat rooms.

On the frontend, the interface is built using Bootstrap, offering a clean, responsive, and mobile-friendly design. Visual cues are provided to distinguish messages sent by the logged-in user from those sent by others, improving the clarity and readability of the conversation. The system architecture follows best practices in modern web development by combining synchronous HTTP views for page rendering with asynchronous WebSocket handling for real-time features. This project showcases the power of combining Python, Django, and Django Channels to build scalable, real-time web applications, and reflects a strong grasp of both traditional backend development and modern asynchronous programming techniques.

3.2 Room Finder Web App

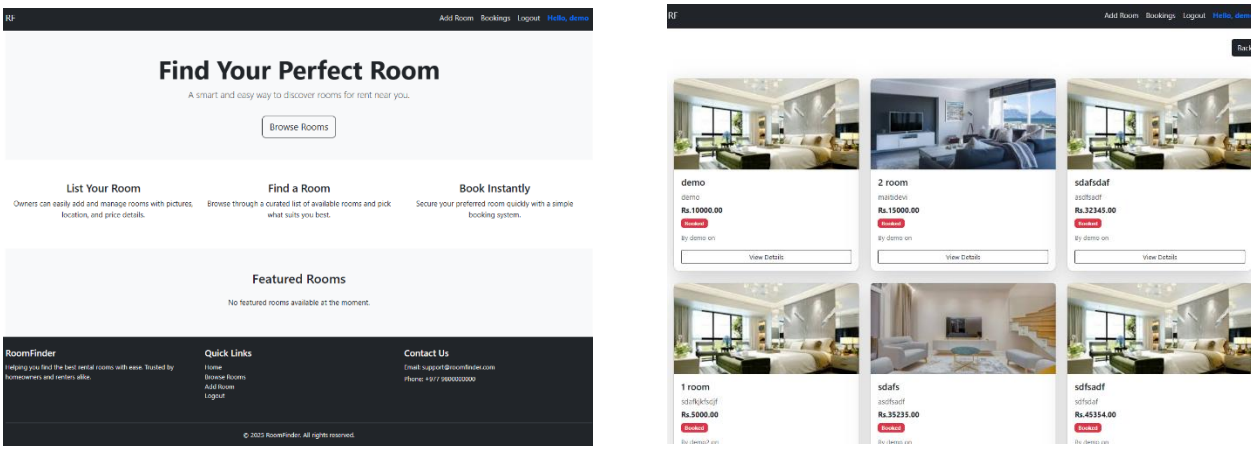


Fig 2. Room finder web app

This Room Finder Web Application is a full-featured, Django-based platform designed to streamline the process of discovering and booking rental rooms. Developed with a focus on usability and real-world practicality, the platform acts as a digital bridge between house owners seeking to rent out their rooms and tenants in search of available accommodations. The application features a secure user authentication system, allowing both room owners and renters to register, log in, and manage their activities through an intuitive dashboard.

For property owners, the platform offers the ability to create detailed room listings by submitting essential information such as the room's title, description, location, price, availability status, and images. These listings are stored in the backend database and displayed in a well-organized frontend interface. On the other side, users who are searching for accommodations can browse a list of available rooms, view individual room details, and initiate a booking with a single click. To ensure reliability, the application automatically marks a room as unavailable once it has been booked, preventing double bookings and maintaining accurate availability in real time.

The system incorporates role-based permissions to ensure that users only have access to actions appropriate to their role—owners can only manage their own listings, while renters can only view and book rooms. Built with a modern, responsive design using Bootstrap, the frontend offers a seamless and accessible experience across devices. The booking logic, room visibility, and user flow are handled securely and efficiently using Django's robust architecture and ORM. Overall, this Room Finder application offers a transparent, efficient, and scalable solution for managing short- or long-term room rentals, providing real value to both property owners and seekers.

3.3 Restful APIs

I have utilized Django REST Framework (DRF) to efficiently and effectively build RESTful APIs that power the backend of modern web and mobile applications. DRF significantly simplifies the process of transforming Django models into JSON-formatted responses, making it easier for frontend clients to consume and interact with backend data. At the core of DRF is the concept of serializers, which are used to convert complex data types—such as Django model instances—into native Python datatypes that can then be rendered into JSON. These serializers not only handle output formatting but also perform input validation and deserialization, ensuring that incoming data meets the necessary requirements before it is saved to the database.

The standard development process I follow begins with defining Django models, followed by creating serializers for those models, and then building API views using either APIView classes, viewsets, or generic class-based views. These components collectively handle various HTTP methods such as GET, POST, PUT, PATCH, and DELETE, enabling clients to perform CRUD operations through intuitive and structured API endpoints. DRF’s robust support for authentication and permission classes allows me to implement secure access control, ensuring that only authorized users can access or modify specific resources. Moreover, the built-in Browsable API interface is particularly useful during development and debugging, providing a visual and interactive environment to test endpoints directly from the browser. Overall, DRF

allows me to create scalable, modular, and secure APIs while accelerating development and maintaining clean, maintainable code architecture.

3.4 Blog Web App

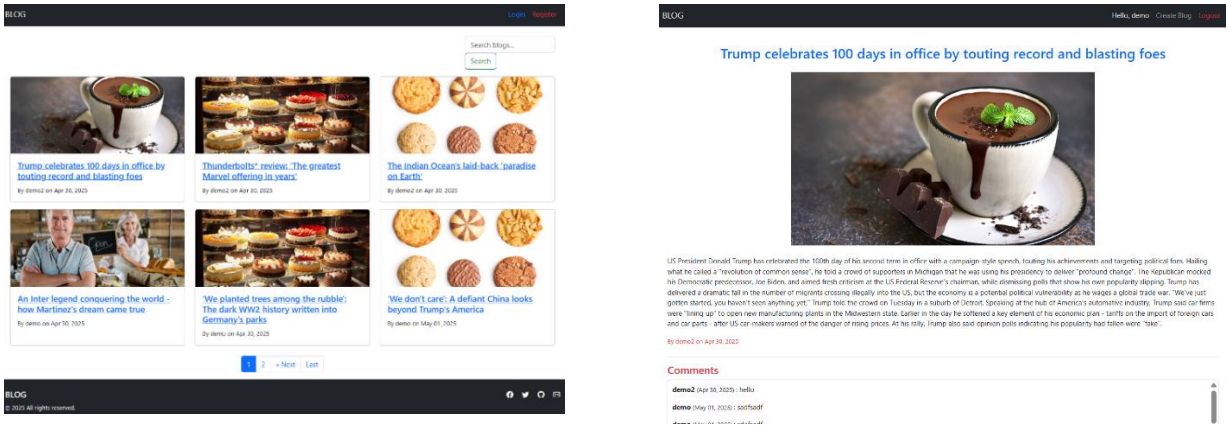


Fig 3. Blog web app

This Blog App is a comprehensive content management platform built with Django and Django REST Framework, designed to provide a seamless experience for both readers and content creators. It includes robust user authentication with registration and login functionality, supporting JWT token-based authentication for secure API access.

The app implements role-based permissions, allowing admins full control over content, staff members to manage posts, and regular users to engage by viewing and commenting. Users can create, update, and delete blog posts, which are presented in a responsive, paginated card layout with search functionality for easy navigation. The commenting system supports both authenticated and anonymous users, displaying comments in a clean, scrollable format alongside usernames and timestamps. Additionally, a like and dislike feature encourages user interaction, with safeguards to prevent multiple votes per post. The API provides endpoints for managing posts and users, protected by JWT and permission classes to ensure security.

Custom Django signals track key user actions like registration and login, enabling extensibility and monitoring. The user interface is enhanced with Bootstrap styling, featuring a dynamic navbar that shows the logged-in user's name, a fixed footer with social links, and popup messages for user feedback on login and registration. The app also incorporates caching mechanisms to improve performance by reducing database queries, and custom Django management commands facilitate administrative tasks such as batch creation of posts. Overall, this Blog App offers a scalable, secure, and user-friendly platform ready for further expansion and integration.

3.5 Portfolio

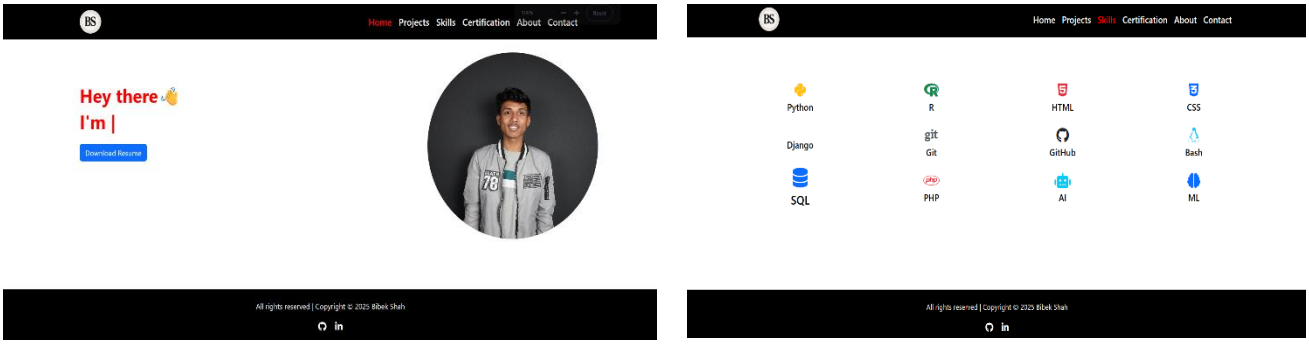


Fig4. Personal portfolio

This portfolio website is developed using Django, a robust and versatile Python web framework known for its rapid development capabilities and clean, pragmatic design principles. Leveraging Django's powerful backend infrastructure, the application efficiently handles all server-side operations, including URL routing, database interactions, user authentication, and processing of form submissions, ensuring a smooth and secure user experience.

3.5.1 Frontend

The user interface is meticulously crafted using Bootstrap 5, a popular front-end framework that guarantees a fully responsive and mobile-first design. This ensures that the portfolio adapts seamlessly across a wide range of devices, from large desktop monitors to small smartphone screens, providing an optimal viewing experience for every visitor. The design incorporates dynamic navigation elements that improve site usability, stylish project showcase cards that highlight key work and achievements, and interactive forms that facilitate easy user engagement. Django's templating system is utilized to dynamically render HTML pages, allowing for reusable components and efficient loading of static assets such as CSS, JavaScript, and images. This approach reduces redundancy and enhances maintainability of the frontend code.

3.5.2 Backend

On the backend, Django orchestrates the core functionalities vital to the website's operation. It serves web pages dynamically based on user requests and manages data flow securely through its Object-Relational Mapping (ORM) layer, which interacts with the database to store and retrieve information. One of the critical features of this portfolio is the contact form, which allows visitors to send messages directly through the website. When a user fills out the contact form, the data is sent securely to the server, where Django performs validation checks to ensure that all required fields are properly completed and that the data conforms to expected formats.

Upon successful validation, the backend uses Django's built-in email framework to send the submitted message to a predefined email address. This requires configuring SMTP (Simple Mail

Transfer Protocol) settings, such as Gmail SMTP, to facilitate sending emails programmatically from the server. Depending on the application's configuration, the email sending process can be handled synchronously or asynchronously to optimize performance and user experience.

The email sending workflow is structured as follows:

1. The user completes and submits the contact form.
2. Django validates the form input and invokes the email sending function.
3. The message is transmitted to the designated email inbox, either immediately or via background task processing.
4. The frontend then provides real-time feedback, displaying a confirmation message to inform the user that their inquiry has been successfully sent.

3.5.3 Overall Integration

This seamless integration between the frontend and backend creates a smooth, interactive user experience that encourages visitor engagement while enabling the site owner to receive direct communications efficiently. The design and functionality together ensure the portfolio is not only visually appealing but also practical and fully functional. Additionally, the use of Django's secure framework provides confidence that user data is handled responsibly, following best practices for web security.

By combining modern front-end technologies with a robust backend architecture, this portfolio website serves as a comprehensive digital showcase of skills and projects, reflecting both technical proficiency and thoughtful design considerations. It exemplifies how Python and Django can be leveraged to build scalable, maintainable, and user-centric web applications.

3.6 Hospital Management System

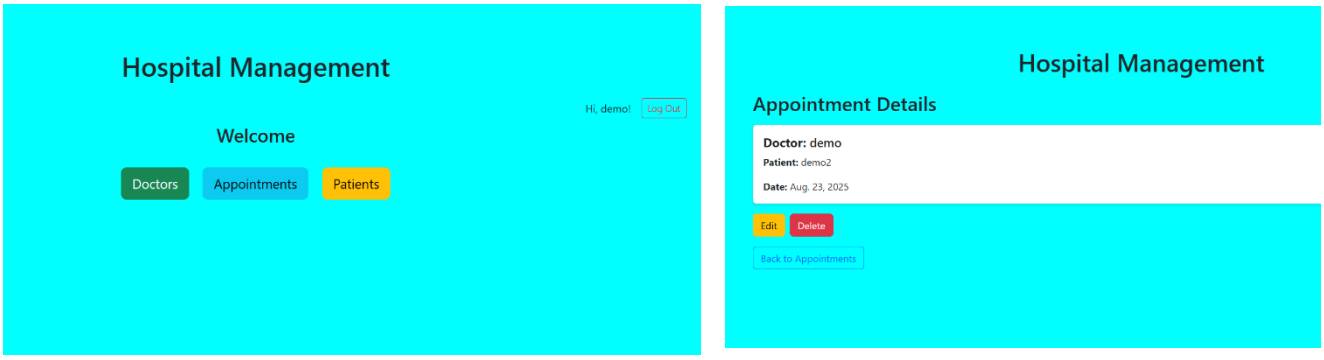
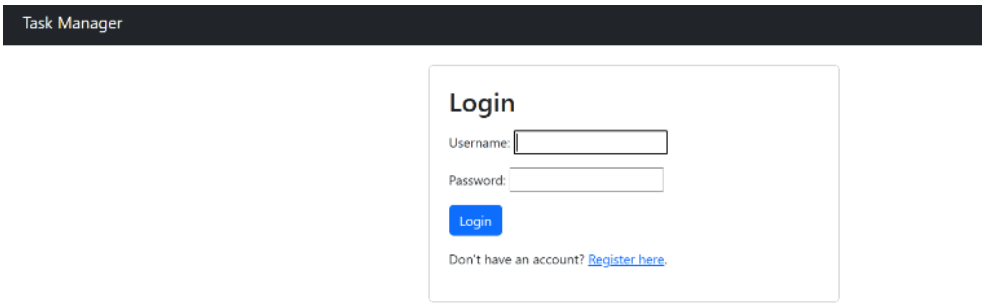


Fig5. Hospital management web app

The Hospital Management System is a Django-based web application designed to streamline and manage hospital operations such as doctor and patient records, and appointment scheduling. The system allows authenticated users to add, update, view, and delete both doctor and patient profiles. Once a doctor and a patient are registered in the system, an appointment can be scheduled by selecting both parties along with a specific date and time. This makes it easier for hospitals to manage appointments and ensure organized interaction between healthcare providers and patients. The structure is built with a focus on efficiency, ensuring that essential data is accessible while being securely maintained.

To ensure the privacy and integrity of hospital data, the application implements user authentication and role-based access using Django's built-in authentication system. Only authenticated users can log in to the system and perform any operations. To enforce this, the system uses LoginRequiredMixin in all views that handle Create, Read, Update, and Delete (CRUD) operations. This ensures that unauthorized users cannot access or manipulate sensitive data. The design emphasizes modularity, scalability, and ease of use, making it an ideal solution for small to medium-sized healthcare institutions looking for a simple and secure way to manage internal workflows.

3.7 Task Manager



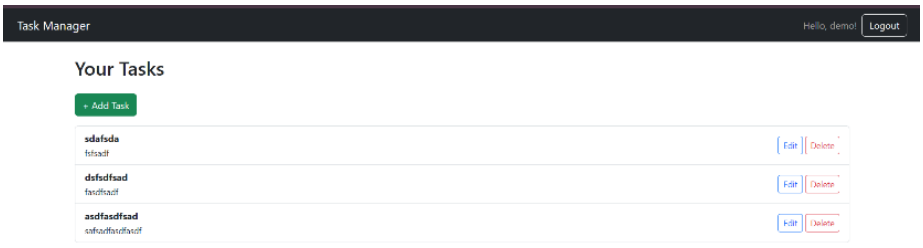


Fig6. Task manager webapp

The Task Manager is a Django-powered web application that enables users to efficiently manage their daily tasks through a secure and user-friendly interface. Upon successful authentication, users gain access to their personalized task dashboard where they can add new tasks, update existing ones, and delete completed or irrelevant entries. Each task is associated with the logged-in user, ensuring a private and personalized experience. This system promotes productivity by offering a simple way to organize tasks and keep track of pending and completed actions in a structured format.

To maintain security and user-specific access, the application enforces authentication using Django's built-in user system. Users must log in to gain access to task-related operations, and views that handle task creation, editing, and deletion are protected using Django's LoginRequiredMixin. This prevents unauthorized access and ensures that one user cannot interfere with another's data. The application serves as an excellent example of how authentication and authorization can be implemented in a real-world scenario while providing practical functionality for everyday task management.

3.8 Ukaalo News Website

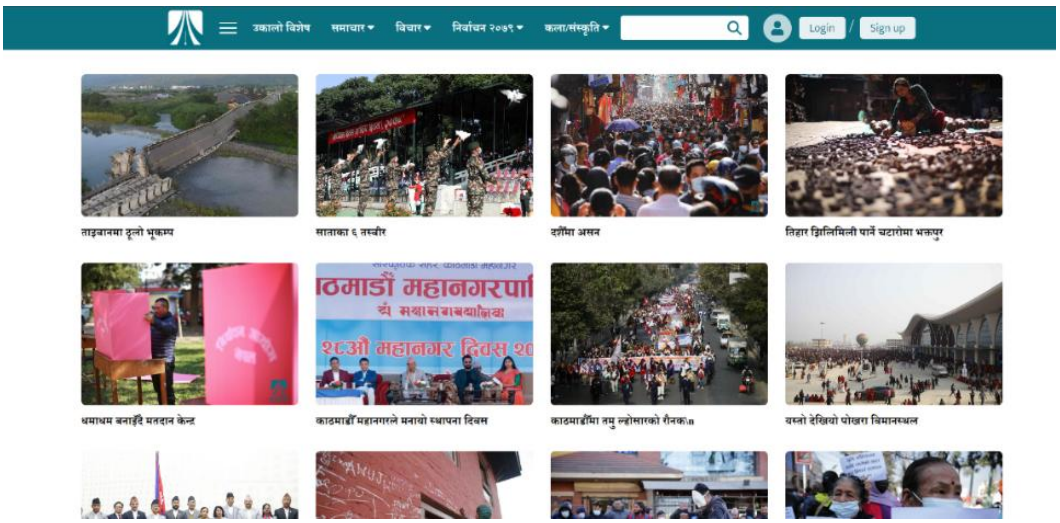


Fig7. Ukaalo news website

During my internship at Prixa Technologies, I had the opportunity to work on a significant feature enhancement for the Ukaalo project, one of the company's prominent content platforms. My primary responsibility involved implementing a functionality within the Django framework that allowed specific authors to be assigned to their respective images using the Django admin interface. This task required an in-depth understanding of Django's model relationships,

including foreign key associations and model-level validations, to ensure data integrity between authors and their uploaded content.

To accomplish this, I customized the Django admin panel to enhance the usability and efficiency of the image-author linking process for internal content managers. Each image uploaded to the platform was carefully mapped to its corresponding author, providing a structured system for content ownership and editorial transparency. By accurately linking content to its source, the system allowed the editorial team to maintain clear attribution records, improve workflow accountability, and reduce content management errors. This feature added significant value by making the backend system more reliable and logically organized for daily operations.

Furthermore, I extended this functionality to the custom Ukaalo admin-panel, built specifically for the live Ukaalo Prixa production environment. I was responsible for ensuring the seamless publishing of author-linked images, validating both backend logic and frontend behavior within the internal CMS. This required close attention to user experience, error handling, and form processing logic to make sure the publishing interface operated intuitively for content managers. I conducted testing to verify the integrity of data associations and smooth integration within the platform's existing content workflow.

The feature was completed according to the specifications provided by my mentor, meeting all functional requirements and delivery timelines. Through this task, I gained valuable hands-on experience in Django admin customization, practical applications of content workflows, and collaborative development in a live production setting. The experience not only deepened my technical proficiency in Django but also taught me the importance of delivering clean, maintainable, and user-oriented solutions in a real-world team environment.

4. Summary

This placement at Prixa Technologies significantly enhanced my skills in real-world software development, particularly in backend engineering, API development, and collaborative project workflows.

Key Skills Learned and Refined:

- Django Framework & Django REST Framework (DRF)
- API design, development, and integration
- PostgreSQL and database modeling using Django ORM
- Authentication and authorization best practices
- Version control with Git and collaborative development
- Admin panel customization and backend logic implementation
- Sprint-based development and technical documentation

This internship has been a foundational step in preparing me for a full-time role as a backend developer. It strengthened my technical capabilities, improved my problem-solving approach, and provided the professional discipline needed to contribute meaningfully in a dynamic, tech-driven environment like Prixa Technologies.