Tribhuvan University

Faculty of Humanities and Social Sciences

Final Year Internship Report

On

Python with Django
Intern

At

Sipalaya InfoTech Pvt.Ltd

In partial fulfillment of the requirements for the degree of Bachelor in Computer Application

Submitted to:

Department of Computer Application Kathmandu

College of Technology

Submitted by:

Umesh Tamang
Symbol number: 110502050

Under the Supervision of

Mr. Susan Risal

March 2025

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this internship report prepared under my supervision by **Umesh Tamang** for the partial fulfillment of the requirements for the degree of Bachelor in Computer Application. During his internship at **Sipalaya InfoTech Pvt. Ltd.**, he displayed strong skills in backend development using Django, building robust and efficient server-side applications.

.....

Mr. Susan Risal

Kathmandu College of Technology

CERTIFICATE OF APPROVAL

This is to certify that, this internship report prepared by **Umesh Tamang** entitled **"Internship on Python with Django"** at **Sipalaya InfoTech Pvt Ltd.** in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been studied. In our opinion, it is satisfactory in scope and quality as a project for the required degree.

……	……
Signature of Supervisor	**Signature of Coordinator**
Susan Risal (Supervisor)	Santosh Rijal (Coordinator)
Kathmandu College of Technology	Kathmandu College of Technology
Lokanthali-16, Bhaktapur	Lokanthali-16, Bhaktapur
……	……
Signature of Internal Examiner	**Signature of External Examiner**

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my mentor, **Er. Sujan Thadarai**, and supervisor, **Mr. Susan Risal** and our coordinator, **Mr. Santosh Rijal** for their continuous guidance and support throughout my internship. Their valuable feedback and insights have greatly contributed to my professional growth and have been instrumental in the success of the project I worked on.

I would also like to thank the entire development team for providing me with the opportunity to learn and collaborate. Their cooperation and expertise have enhanced my learning experience.

Finally, I am grateful to my family and friends for their unwavering support and encouragement throughout this journey. This internship has been an enriching experience, and I am truly thankful to everyone who made it possible.

Thanking you,

Umesh Tamang

ABSTRACT

The report presents results of a three-month internship program where Mr. Umesh Tamang worked on the Python with Django of the “Blue Caller” project at Sipalaya InfoTech Pvt. Ltd. The “Blue Caller”-Gig Economy Platform is a system, a play on the name: Blue Caller, dedicated to all the blue-caller workers, is an innovative gig economy platform designed to streamline the process of connecting customers with skilled blue-caller workers. Utilizing modern web technologies including Django, JavaScript and Tailwind, the system provides an intuitive and efficient user experiences. The platform features three main user entities: customers, workers and administrators. Customers are able to sign up or log in, report hands-on issues such as fixing wires or pipes or electricals, and specify a convenient date and time for the service. Workers are able to sign up, log in by administrators to ensure reliability and trustworthiness. Upon logging in, customers are able to view a list of available workers matching their specified needs and schedule, sorted according to the distance between them. They are able to then select a worker and send an appointment request. Workers, upon logging in, are able to view and manage incoming requests, accepting or rejecting them based on their availability. The system keeps all parties informed of the appointment status. The “Blue Caller” platform enhances the efficiency and convenience of booking skilled labor, benefiting both customers seeking reliable service and workers seeking opportunities.

Keywords: Blue-caller workers, Appointment, Cdn, Django

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CDN	Content Delivery Network
CSS	Cascading Style Sheets
Django	Python's Web Framework
HTML	Hypertext Markup Language
JS	JavaScript
NAPEC	National Association of Planning and Electrical Contractors
OOD	Object Oriented Design
SDLC	Software Development Life Cycle

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

1.1 Introduction

As my three months internship as a Python with Django Developer at a dynamic technology firm, I had the opportunity to work on an innovative project titled “Blue Caller.” The “Blue Caller” system is an innovative gig economy platform designed to streamline the process of connecting customers with skilled blue-caller workers.

Built using modern web technologies like JavaScript, Django, and Tailwind, the platform ensures a smooth and user-friendly experience. It features three main user roles: customers, workers, and administrators.

Customers can sign up or log in to report issues such as fixing electrical wires or plumbing, specifying a convenient date and time for service. Workers can also register, log in, and provide their credentials, including their expertise and location. To ensure reliability, administrators verify these details before allowing workers, to take job requests.

Once logged in, customers can browse a list of available workers matching their needs and schedule an appointment. Workers, upon logging in, can view and manage job requests, choosing to accept or decline based on their availability.

This project significantly enhanced my understanding of backend development, database management, and software security practices. It provided hands-on experience in integrating various technologies to create a functional and scalable gig economy system, making it a key milestone in my professional development.

1.2 Problem Statement

In Nepal, the demand for skilled blue-caller services such as electrical repairs, plumbing, carpentry, and maintenance work is growing rapidly due to increasing urbanization and in-

frastruture development. However, the market faces significant challenges, with 60% of customers reporting dissatisfaction with the reliability of services providers, according to survey by the National Association of Plumbing and Electrical Contractors, USA. A similar trend can be observed in Nepal, where the absence of a centralized platform has resulted in a fragmented market, leading to inefficiencies and inconsistent service quality.

Therefore, this lack of regulation not only affects service quality but also discourages, potential customers from hiring independent workers. A structured, digital platform could bridge this gap, ensuring reliability for customers while providing verified workers with better access to job opportunities.

1.3 Objectives

The primary objectives are:

- To build features like user authentication using Django-all auth, role-based access control (Admin, Worker, Customer), and job posting mechanisms using Django framework.
- To streamline the connection between customers and skilled workers through features like job list, appointment management, and worker verification.
- To integrate backend logic with dynamic frontend templates using Django template Language (DTL), ensuring that data is rendered efficiently and user-specific content is displayed accurately across the platform.

This project aims to streamline the connection between customers and skilled blue-caller workers, enhancing both service efficiently and job opportunities.

1.4 Scope and Limitations

1.4.1 Scope

i. Seamless Customer-Worker Connection: The platform enables customers to find and hire skilled blue-caller workers efficiently.

ii. Intuitive Request & Profile Management: Customers can submit service requests, while workers can manage their profiles and job opportunities.

iii. Enhanced Reliability & Trust: A verification process ensures trustworthy and skilled professionals, improving customer satisfaction.

iv. Better Job Visibility for Workers: Skilled workers gain more job opportunities and can build a professional reputation through the platform.

1.4.2 Limitation

i. Payment management has not been integrated, limiting real-world transaction capabilities.
ii. Comprehensive profile management is absent for both customers and workers was partially implemented.
iii. No integration with external APIs (e.g., location or SMS services) to enhance functionality.
iv. Real-time features like chat or notifications were not included in the system and limited time frame (3 months) restricted implementation of advanced features.

1.5 Report Organization

The report is broken up into six isolated chapters, each of which describes a different stage of development. The following is a summary of the chapters:

Chapter 1 is the overview of the project including the introduction, problem statement, objectives, scope, limitations, and development methodology.

Chapter 2 focuses on the background study and literature review related to the project for reference in project development.

Chapter 3 involves system analysis, including requirement analysis and feasibility analysis.

Chapter 4 includes the project design and modeling.

Chapter 5 focuses on the implementation and testing of the system.

Chapter 6 is the conclusion of the report and includes future recommendations. Furthermore, the reference section includes a list of all the sources cited in the report.

Chapter 2: Introduction to Organization

2.1 Organization Details

The internship was completed at Sipalaya InfoTech Pvt.Ltd. is a leading It institution in Nepal dedicated providing high-quality technical training and professional development services. Situated in Koteshwor, Kathmandu, it serves as a hub for students, fresh graduates, and IT enthusiasts who aspire to build a strong career in the tech industry. The organization's primary mission is to empower individuals with in-demand technical skills, offer practical and project-based learning.

The institute offers a wide variety of courses tailored to meet current industry demands. These include Python with Django, Python with Data Science, MERN Stack Development, UI/UX Design, Graphics Design and Digital Marketing etc. Each course blends theory with hands-on learning, enabling students to work on real-time projects and gain valuable practical experiences.

With over 5,000 students trained, more than 30 expert instructors, and partnerships with 5o+ hiring companies, Sipalaya has played vital role in bridging the gap between academic knowledge and industry needs. The leadership team-comprising CEO and Founder Er. Himal Rawal, CTO Er. Suman Tamang, Manager Madan Bist, and other experienced professionals-ensures that the courses are aligned with real-world expectations and technological advancements. Their commitment to excellence inspires students to build confidence, enhance their skills, and thrive competitive environments. The organization features modern infrastructure with well-equipped training labs, collaborate learning environments, and options for both online and offline classes. Sipalaya Info Tech Pvt.Ltd. stands out as a future-oriented institute focused on producing capable and confident tech professionals. With its comprehensive training programs, expert mentorship, and career support, it continues to shape the future of IT education and workforce development in Nepal.

2.2 Organization Hierarchy

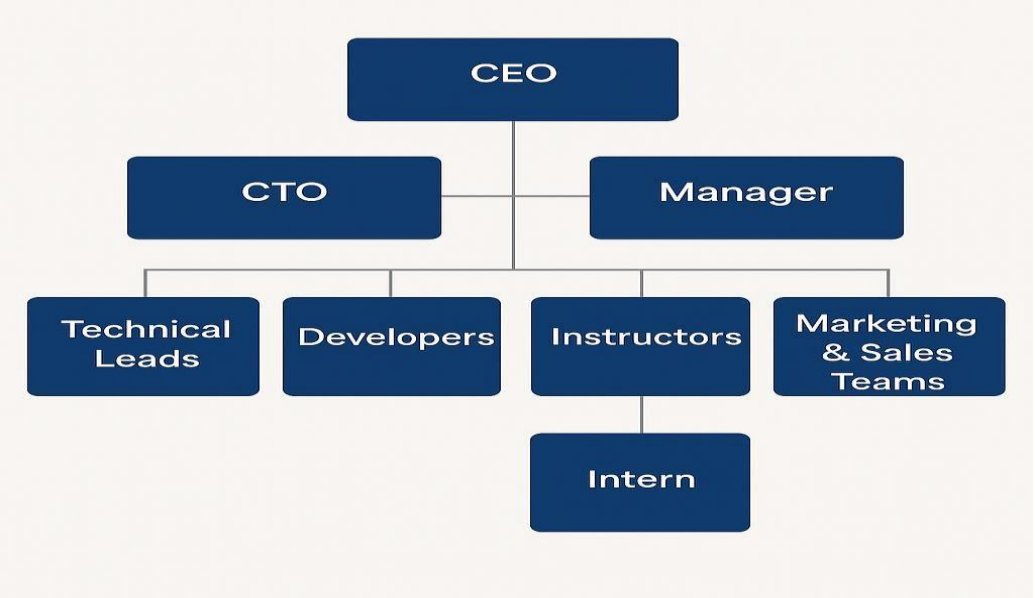


Figure 2.1 Organization Hierarchy

Sipalaya Info Tech follows a structured and collaborative organizational hierarchy that ensures efficient operations and smooth communication across all departments. At the top of the hierarchy is the chief Executive Officer (CEO), Er. Himal Rawal, who leads the company by setting the strategic direction and overseeing major organizational decisions. He is supported by the Chief Technology Officer (CTO), Er. Suman Tamang, who supervises technical departments and ensures the implementation of modern tools and technologies. The manager, Madan Bist, is responsible for managing daily operations, coordinating between teams, and maintaining overall workflow. Beneath this leadership level are various departments including Technical Leads, Developers, Instructors, Marketing and Sales Teams, Administrative Staff, and Interns. Interns are considered an important part of the structure and are assigned to different teams where they learn under mentorship of experienced professional. They assist in projects, attend team meetings, and gradually build real-world technical and soft skills.

2.3 Working Domains of Organization

The organization operates across a multitude of working domains, each contributing to its reputation as a versatile and innovative technology leader. The expertise and offerings cover the following working domains:

Technical Education and Skill Development:

Sipalaya Info Tech's primary focus is on delivery high quality IT training programs tailored to meet current industry demands. It offers courses such as Python with Django, Python with Data Science, MERN, UI/UX Design, Graphics Design and Digital Marketing etc. These courses are designed to equip students with both theoretical knowledge and practical skills, enabling them to build real-world applications and become job-ready professionals.

Project-Based learning:

Alongside regular training, Sipalaya emphasizes project-based learning where students work on real-time projects. This domain allows to apply their knowledge in practical scenarios, solve problems, collaborative in teams, and understand the complete software development life cycle, boosting their confidence and technical expertise.

Career Counselling and Job Placement:

Sipalaya supports students in building their professional careers by offering career counselling sessions, interview preparation, CV building workshops, and connecting them with hiring partners with over 50+ hiring companies, Sipalaya has successfully placed many of its traineess in reputed IT firms across Nepal. Sipalaya actively connects skilled traineess with job opportunities in roles such as developers, designers, digital marketors, and more.

Community Building and Tech Awareness:

Beyond classroom learning, Sipalaya actively organizes workshops, hackathons, seminars, and tech talks to foster a strong IT community. This domain focuses on encouraging collaboration, continuous learning, and innovation among aspiring tech enthusiasts. It helps students stay updated with the latest trends while promoting with the latest trends while promoting networking and peer learning opportunities.

2.4 Description of Intern Department/Unit

The intern department is a dynamic and integral segment of the organization, seamlessly integrated within the larger Web Development unit. This department is designed to serve as a nurturing ground for emerging talent, providing interns with comprehensive exposure to real-world projects while under the mentorship of experienced professionals. Interns are encouraged to participate actively in all phases of the project lifecycle—from the initial brainstorming and design sessions to coding, testing, and final deployment ensuring that they gain a holistic understanding of modern web development practices.

Table 2.1 Internship description

START DATE:	13 December 2025
END DATE:	13 March 2025
TOTAL DURATION:	3 MONTHS
INTERN POSITION:	PYTHON WITH DJANGO (Intern)
MENTOR:	Er. Sujan Thadarai
OFFICE HOUR:	10:30 am TO 6:00 pm

Chapter 3: Background Study and Literature Review

3.1 Background Study

The demand for skilled blue-caller workers has always been an integral part of modern society. In US, more than a third (35%) of the fastest-growing jobs are blue-caller, with 1.7 million new jobs expected between now and 2032. However, the process of finding reliable workers and efficiently and unstructured. Traditional methods, such as world-of-mouth referrals or classified advertisements, lack transparency, scalability and accessibility, leaving both customers and workers dissatisfied.

In recent years, the rise of gig economy platforms has demonstrated the potential of technology to bridge the gap between service providers and consumers. Platforms like Iber and TaskRabbit have streamline the process of connecting users with service providers in various domains.

The “Blue Caller” system aims to address this gap by providing a reliable, efficient, and user-friendly solution. By focusing on proximity-based worker suggestions, the system ensures convenience for customers while also offering job opportunities to skilled workers.

Additionally, the system builds trust through an administrative verification process, ensuring the reliability and professionalism of listed workers. This project not only addesses a pressing real-world issues but also contributes to broader digital transformation.

3.2 Literature Review

With the rapid growth of urban development and digital transformation, the demand for skilled blue-caller workers has significantly increased. According to the National Electrical Contractors Association (2024), the electrical contracting industry is evolving with the integration of new technologies, which demands a highly skilled workforce capable of adapting to modern tools and safety protocols.

Wells (2024) predicts that the blue-caller workforce will experience a rise of 1.7 million new job openings by 2032 in the US. This rise is similarly reflected in Nepal due to increased infrastructure activities. Burinskiene (2024) further supports this by emphasizing that technological advancements-such as mobile platforms, automation, and data driven are crucial for the success and scalability of sharing economy platforms.

Digital platforms such as Bhetayo, Blujobs, and Sriyog are emerging in Nepal to address this need, offering job-matching services that connect skilled laborers with clients in need of reliable service (Bhetayo, 2024; BluJobs, 2024; Sriyog, 2024). These platforms help build trust through verifications, user reviews, and accessible interfaces, which are vital for sustaining gig economy ecosystems.

Additionally, Anderson (2024) notes how technologies such as Artificial Intelligence (AI), automation, and wearable tech are revolutionizing construction and labor-based industries, enhancing worker safety and operational efficiency. Laudon and Traver (2021) also highlight how e-commerce and digital platforms are transforming traditional industries by offering scalable digital solutions.

These studies collectively emphasize the importance of combining technological innovation with human skills to create efficient and inclusive digital labor marketplaces like the “Blue Caller” system.

Chapter 4: Internship Activities

4.1 Roles and Responsibilities

During my internship at the organization, I was assigned the role of a Python with Django Developer within the backend development team, where I contributed to the development of Blue Caller, a gig economy platform that connects skilled workers with customers seeking services.

My main responsibilities revolved around the backend architecture, secure data handling, role-based access control, and database management. I worked closely with frontend team to ensure that server-side logic functioned properly, securely, and efficiently. Some of my key roles and responsibilities included:

- Developing and maintain backend logic for user authentication, job posting, worker-customer matching, and rating systems.
- Collaborating on the database schema design.
- Implementing secure authentication methods, password hashing, and role-based access control to restrict sensitive areas.
- Improving system performance through query optimization.
- Participating in code reviews, maintaining clean, readable, and well-documented code throughout the development cycle.
- Working in an Agile environment, attending daily stands-up, and actively engaging in bug fixing and feature improvements.
- Supporting also handling media files.

This internship provided me with hands-on experience in backend development, allowing me to improve my Django skills, and problems solving abilities while contributing to a real-world project.

4.2 Weekly log

Table 4.1 Weekly log Sheet

TIME PERIOD	Activity	Activity Details	Challenges
1st week	Orientation and project setup	Attended onboarding sessions, explored the blue caller project architecture. Setup the development environment, installed dependencies and configured Django settings.	Understanding project requirements and adapting to the organizational workflow.
2nd week	User Module Exploration	Studied the accounts app, including user registration, login, and role-based user types(worker/customer). Tested and debugged forms and views.	Handling user role logic and customizing built in authentication features.
3rd week	Database Design and Model Updated	Analyzed and optimized models in accounts and jobs. Modified fields for extensibility and implemented relationships between jobs and users.	Maintaining data consistency and syncing migrations across the team.
4th week	Job-Posting	Implemented job creation, update and deletion features in the jobs app. Created views and templates for job-related forms.	Ensuring correct job status updated and filtering based on tagline.
5th week	Application and Approval System	Developed worker-side application functionality and customer-side approval views. Implemented logic to link jobs with applicants.	Managing token expiration issues and ensuring secure session handling.

6th week	Admin Panel Enhancements	Customized Django admin for managing users, jobs, and applications more efficiently.	Making admin user friendly without breaking permission logic.
7th week	User Profile Management	Enabled users to edit their profile data and view job history(posted/applied). Integrated with Django forms and templates.	Dynamically loading user specific data and ensuring privacy.
8th week	Template Integration	Integrated backend logic with frontend templates using Django's templating system. Rendered job lists	Passing context data correctly and minimizing template repetition.
9th week	Security and Access Control	Implemented role-based access (worker/customer), protected views using decorators, and restricted unauthorized actions.	Managing different permission levels across views.
10th week	Testing & Debugging	Conducted manual testing of all core features (job posting, applying approval).	Debugging multiple actions and ensuring data integrity.
11th week	Documentation and Final Review	Documented codebase, explained app structure, and model relationship.	Writing clear documentation for future developers.
12th week	Deployment Prep and Wrap Up	Finalized requirements.txt, tested deployment readliness.	Ensuring deployment stability and resolving path related issues.

4.3 Description of the Project

During my internship, I was actively in the development of the project titled “Blue Caller”. This project aims to connect skilled blue caller workers with customers looking for their services, providing an efficient and user-friendly job matching experiences. The project focused in building a scalable and secure ecosystem where job providers and service workers could interact seamlessly. The primary objective of Blue Caller was to create a seamless job search and hiring process for both workers and customers. The platform allowed users to register, browse job listings and connect with workers based on their skills and availability.

Technology Stack & Key Features:

Backend-Django with MySQL database.

User Registration and Role Management: Implemented secure user signup and login with role-based access control to differentiate between admi, worker and customer functionalities.

Job Posting: Job listing and worker profile management.

Security: Enhancement including CSRF protection and user data validation.

Django Template Language (DTL): Used to dynamically generate HTML in Django web application.

Database Optimization: Build a relational schema using Django ORM.

Deployment Preparation: Prepared application for production by setting up environment configurations, media handling and migration process.

Additionally, I collaborated with the team to debug issues, optimize performance, and ensure a smooth user experience.

The project also includes a profile management system where workers could highlight their expertise, making it easier for customers to make informed hiring decisions.

The overall goal of Blue Caller was to create an accessible, secure, and user-friendly job matching platform that empowers blue-caller workers with better job opportunities while simplifying the hiring process for customers.

4.4. Tasks / Activities Performed

Throughout my internship, I performed several tasks related to backend development for the Blue Caller. The key tasks I was involved in include:

Research, Environment Setup and Database Design and Management:

- Setup Virtual Environment and MySQL database and understand the authentication logic for secure module development.
- Designed and modified the database schema to accommodate new features like job applications and user roles.
- Worked on model relationships such as Foreign Key and Many to Many Fields for linking users with jobs.

User Authentication, Role-Based and Job posting, worker and customer:

- Implemented secure login, signup and logout functionalities using Django's built-in auth system, integrated role-based permissions, allowing different features for admin, worker and customers and ensured password security through hashing and validation mechanism.
- Developed backend logic for job posting, managing and deleting based on user roles.
- Implemented constraints to allow only authorized users to manage jobs and handled application logic for worker applying to jobs, created logical views to support data display for worker and customer dashboards (e.g. posted jobs, applied jobs), ensured secure data access and role-based filtering for dashboard data (like searching tagline, distance).

Security and Data Protection:

- Applied additional security measures including form validation.
- Prevented unauthorized access through custom permissions and access checks
- Secured critical user data using encryption for sensitive fields.

Testing and Debugging:

- Performed unit-testing using Django's built in test framework to validate models and views.
- Debugged backend errors related to database integrity, login failures.
- Performed load testing to assess system performance under high traffic conditions.

Performance Optimization:

- Optimization database queries for faster data retrieval using select_related and prefetch_related.
- Implement searching functionality.
- Reduced redundant logic and improved server-side data processing.

Deployment Preparation:

- Assists in preparing the project for deployment.
- Configured environment variables, static/media files, and database migration scripts.
- Helped identify and fix deployment-specific issues (e.g. static file handling).

Through these tasks, I gained hands-on experience in backend development while contributing to the successful implementation of Blue Caller as a secure, efficient and scalable gig economy platform.

Chapter 5: Conclusion and Learning Outcomes

5.1 Conclusion

The internship experience at Sipalaya Info Tech Pvt.Ltd. was an invaluable opportunity that provided me with hands-on exposure to backend development and real-world application deployment. Working on Blue Caller was a highly rewarding experience. This System me to develop key components of a gig economy platform, gaining expertise in Python, Django, MySQL, ORM, authentication mechanisms, and security protocols.

Throughout the project, I contributed to building and optimizing backend infrastructure, ensuring secure authentication, efficient database management, and seamless both communication between the frontend and backend. This internship has significantly contributed to my professional development, equipping me with real-world industry experience and reinforcing my passion for backend development. The skills and knowledge gained during this period will serve as a strong foundation for my future career in software development.

5.2 Learning and Outcomes

Technical Skills:

- Gained expertise in backend development using Python with Django, MySQL.
- Developed and optimized role-based features like job posting, dashboards.
- Implemented secure authentication, password hashing, and role-based access control.
- Improved skills in database design, model relationships (Foreign Key, Many to Many Field).
- Learned how to handle exceptions, Django's test framework.

Teamwork and Collaboration:

- Worked closely with frontend developers and database administrators, ensuring smooth integration between different system components.
- Actively participated in team discussion, sprint planning, and code reviews, improving my ability to communicate technical concepts effectively.

Problem-Solving Abilities:

- Solved real-world backend issues like access control bugs, login failure, and data integrity problems.
- Quickly to a production-level codebase and explored new Django features as required.

Security Awareness:

- Implemented input validation, custom permissions, and data encryption for sensitive user data.
- Worked with user authentication mechanisms, reinforcing best practices in web security.

The internship bridged the gap between academic learning and real-world application, providing me with essential hands-on experience in backend development, security, and database management. The challenges faced and skills acquired have prepared me for future roles in backend development and software engineering.

REFERENCES

National Electrical Contractors Association. (2024). *NECA*. Retrieved July 20, 2024, from https://www.necanet.org/

Wells, R. (2024). Blue collar workers wanted: 1.7 million new jobs projected by 2032. Forbes.

Burinskiene, A. (2024, February 19). Sharing economy development: Empirical analysis of technological factors.

Bhetayo. (2024). *Bhetayo*. Retrieved July 20, 2024, from https://bhetayo.com/

BlueJobs. (2024). *BluJobs*. Retrieved July 21, 2024, from https://blujobs.in/

Traver, C.G. Laudon. (2021). *E-commerce: Business, technology, society.* Pearson Education.

Sriyog Consulting Pvt. Ltd. (2024). *SRIYOG*. Retrieved July 23, 2024, from https://sriyog.com/

Anderson, T. (2024). Technological trends in the construction industry: A look into the future. *Construction Today*.

APPENDICES

```python
def get_queryset(self):
    query = self.request.GET.get('q')  # Search query
    filter_param = self.request.GET.get('filter')  # Filter parameter

    queryset = Worker.objects.all()

    # Apply search functionality (by tagline) first
    if query:
        queryset = queryset.filter(tagline__icontains=query)

    # Filter by Rating
    if filter_param == 'rating':
        # Filter workers by rating (only those already searched)
        queryset = queryset.annotate(avg_rating=Avg('workerrating__rating')).order_by('-avg_rating')


    # Filter by Distance
    elif filter_param == 'distance':
        customer = getattr(self.request.user, 'customer', None)
        if customer and customer.latitude and customer.longitude:
            customer_lat = float(customer.latitude)
            customer_lon = float(customer.longitude)

            # Annotate workers with calculated distance
            queryset = queryset.annotate(
                distance=ExpressionWrapper(
                    # Distance formula using latitudes and longitudes (Haversine formula)
                    (F('latitude') - customer_lat) ** 2 + (F('longitude') - customer_lon) ** 2,
                    output_field=FloatField()
                )
            ).order_by('distance')  # Order by the calculated distance

    return queryset
```

Worker List View with search and filter(rating/distance)

```python
average_rating = WorkerRating.objects.filter(appointment__worker=worker).aggregate(Avg('average_rating'))['average_rating__avg']
context['average_rating'] = round(average_rating, 1) if average_rating else 0  # Default to 0 if no ratings exist

total_ratings = WorkerRating.objects.filter(appointment__worker=worker).count()
context['total_ratings'] = total_ratings
full_stars = int(context['average_rating'])
half_star = 1 if context['average_rating'] % 1 >= 0.5 else 0
empty_stars = 5 - (full_stars + half_star)

context['full_stars'] = [1] * full_stars  # List of full stars
context['half_star'] = [1] * half_star  # List for half star (1 or 0)
context['empty_stars'] = [1] * empty_stars  # List of empty stars
```

Worker Detail View with star ratings

```python
def rate_worker(request, appointment_id):
    # Fetch the appointment and ensure the user is the customer
    appointment = get_object_or_404(Appointment, id=appointment_id, customer=request.user.customer)

    # Ensure the appointment is completed before allowing a rating
    if appointment.status != 'completed':
        return HttpResponseForbidden("You can only rate a worker after the appointment is completed.")

    if request.method == 'POST':
        rating_value = request.POST.get('rating')

        # Validate the rating value
        if not rating_value or not rating_value.isdigit() or int(rating_value) < 1 or int(rating_value) > 5:
            return HttpResponseForbidden("Invalid rating value. It should be between 1 and 5.")

        # Check if the user has already rated this appointment
        existing_rating = WorkerRating.objects.filter(appointment=appointment, appointment__customer=request.user.customer).exists()
        if existing_rating:
            return HttpResponseForbidden("You have already rated this worker for this appointment.")

        # Create the new rating
        if WorkerRating.objects.filter(worker = appointment.worker).exists():
            past_rating = WorkerRating.objects.get(worker = appointment.worker)
            total_rating = int(rating_value) + past_rating.rating
            appointment_count = past_rating.appointment.count()  + 1
            new_average = total_rating / appointment_count
            past_rating.rating = total_rating
            past_rating.average_rating = new_average
            past_rating.appointment.add(appointment)
            past_rating.save()
        else:
            new_object = WorkerRating.objects.create(worker=appointment.worker,rating=rating_value,average_rating=float(rating_value))
            new_object.appointment.add(appointment)
```

Rate Worker Form after completion

```python
def accept_appointment(request, appointment_id):
    appointment = get_object_or_404(Appointment, id=appointment_id)
    if request.method == 'POST':
        appointment.status = 'accepted'
        appointment.save()
    return redirect('worker_list')

def reject_appointment(request, appointment_id):
    appointment = get_object_or_404(Appointment, id=appointment_id)
    if request.method == 'POST':
        appointment.status = 'rejected'
        appointment.save()
    return redirect('worker-list')
```

Accept and Reject Appointment

```python
class WorkerCreateView(LoginRequiredMixin, CreateView):
    model = Worker
    fields=['name','profile_pic','tagline','phone_number','bio','citizenship_image', 'certificate_file']
    success_url=reverse_lazy('worker-list')

    def form_valid(self, form):
        form.instance.owner=self.request.user
        form.instance.latitude = self.request.POST.get('latitude')
        form.instance.longitude = self.request.POST.get('longitude')
        return super(WorkerCreateView, self).form_valid(form)

    def add_worker(request):
        if request.method == 'POST':
            form = WorkerCreateView(request.POST, request.FILES)  # Include request.FILES to handle file uploads
            if form.is_valid():
                form.save()  # Save the worker instance
                return redirect('success')  # Redirect after a successful submission
        else:
            form = WorkerCreateView()

        return render(request, 'add_worker.html', {'form': form})

# naming convention of the templates for these classes ..CreateView expect is model_form.html

class CustomerCreateView(LoginRequiredMixin, CreateView):
    model = Customer
    fields=['name','profile_pic','phone_number']
    success_url=reverse_lazy('worker-list')

    def form_valid(self, form):
        form.instance.latitude = self.request.POST.get('latitude')
        form.instance.longitude = self.request.POST.get('longitude')
        form.instance.owner=self.request.user
        return super(CustomerCreateView, self).form_valid(form)
```

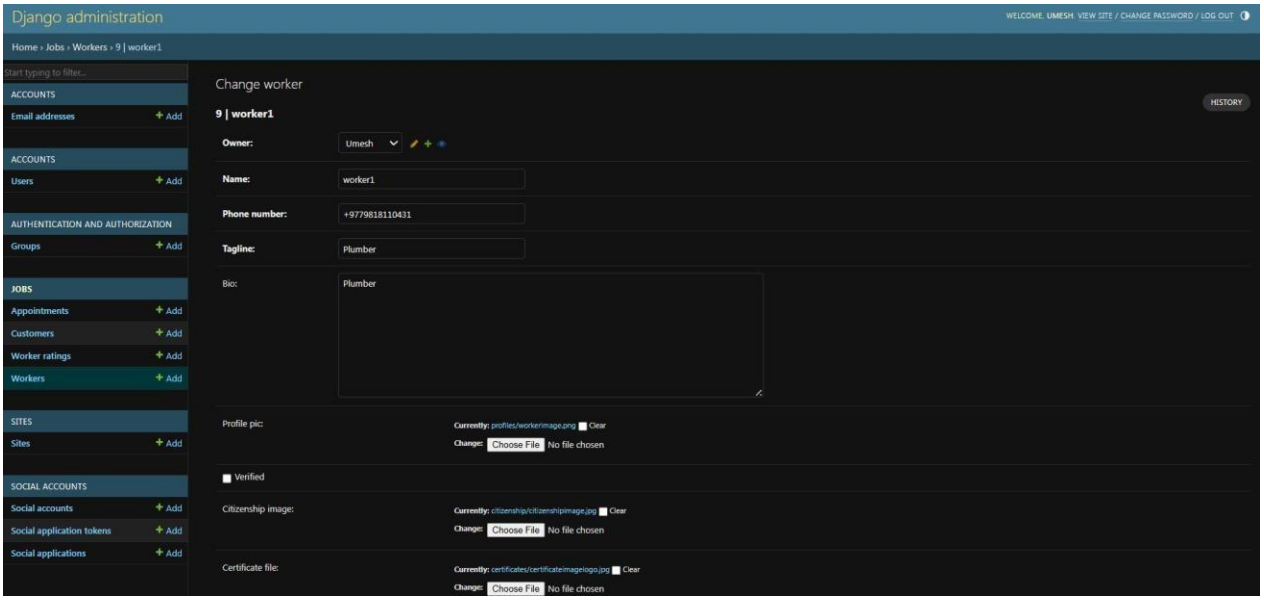
Create worker and customer form

```python
worker = get_object_or_404(Worker, owner=request.user)
appointments = Appointment.objects.filter(worker=worker)
return render(request, 'jobs/worker_appointments.html', {'appointments': appointments})
```

Worker Appointment View

```
customer = get_object_or_404(Customer, owner=request.user)
appointments = Appointment.objects.filter(customer=customer)
return render(request, 'jobs/customer_appointments.html', {'appointments': appointments})
```

Customer Appointment View



Admin Dashboard