Intel® Unnati Industrial Training Program 2024

Intel Unnati Industrial Training Program 2024-2025

Project Report

Problem Statement:

Integrated Common Services to Common People

ConnectGen

Team Name: ConnectGen

Team Members:

AARON ANDE

THEJAS MONIKUTTAN

SHEBIN SAM

JENULIN MAKROS

THIRMALREDDY MARY SHALINI

TABLE OF CONTENT

I. Introduction:

In today's fast-paced world, access to essential services such as healthcare, finance, education, and utilities remains a significant challenge for many individuals, particularly those in underserved and rural communities. These challenges are often exacerbated by the fragmented nature of service delivery systems, where each service operates in isolation, requiring individuals to navigate multiple platforms and interfaces. This fragmentation not only makes it difficult for people to access the services they need but also leads to inefficiencies and increased costs for service providers.

Problem Statement

The core problem addressed by the "Integrated Common Services to Common People" project is the lack of a unified platform that consolidates various essential services into a single, easily accessible interface. The current state of service delivery is characterized by:

1. Fragmentation of Services: Individuals must interact with separate platforms for healthcare, financial services, education, and utilities, leading to a disjointed and time-consuming user experience.

2. Accessibility Issues: Many existing service platforms are not user-friendly, particularly for individuals with limited digital literacy or access to advanced technology.

3. Inefficiencies and Redundancies: Service providers often duplicate efforts in areas such as user verification, data management, and communication, resulting in higher operational costs and reduced service effectiveness.

4. Lack of Personalization: Current platforms fail to offer personalized recommendations and services tailored to the unique needs and preferences of each user.

Project Objectives

The "Integrated Common Services to Common People" project aims to develop a comprehensive web application that integrates multiple essential services into a single platform. This platform will provide:

A Unified Interface: A single point of access for various services, simplifying the user experience.

Enhanced Accessibility: User-friendly design and support for multiple devices to ensure broad accessibility.

By addressing these issues, the project seeks to improve the overall quality of life for individuals by making essential services more accessible, efficient, and user-centric.

Team members & Contributions:

i. **Aaron Ande:**
   - Team lead
   - Front end Development
   - Resource collection
   - RSS

ii. **Thejas Monikuttan:**
   - Front end Development
   - Resource collection
   - Report writing

iii. **Shebin Sam:**
   - Backend Development
   - Resource collection
   - RSS

iv. **Thirmalreddy Mary Shalini:**
   - Front end Development
   - Chatbot Integration
   - Resource collection

v. **Jenulin Makros:**
   - Front end Development
   - Chatbot Integration
   - Resource collection

II. Literature Review

Previous Work and Existing Solutions

The development of web applications to assist common people with day-to-day issues has been an ongoing effort in various domains such as health, education, transportation, finance, and government services. Numerous applications exist that address specific needs within these domains, but few integrate all these services into a single, user-friendly platform.

1.Health: Applications like Practo and Zocdoc provide platforms for booking doctor consultations, finding nearby hospitals, and accessing diagnostic services. These apps often include user reviews and cost details, enhancing the decision-making process for users.

2.Transportation: Ola, Uber, and Google Maps offer extensive services related to transportation, including vehicle booking, cost comparisons, and real-time traffic updates. These platforms have revolutionized the way people navigate urban spaces.

3.Finance: Applications like Mint and YNAB provide tools for personal finance management, while government services like the IRS app facilitate tax-related activities. Banking apps also offer information on loan interest rates and tax-saving plans.

4.Government Services: The Digital India initiative has led to the creation of various apps like UMANG, which integrates several government services including Aadhar, ration card, and passport services.

Analysis of Existing Web Applications Providing Similar Information

Several existing web applications offer similar services, but they typically focus on individual domains rather than providing a comprehensive solution. For instance:

Health Applications: Practo and Zocdoc excel in connecting users with healthcare professionals and services. However, they do not integrate elder care or emergency services, which are critical components of comprehensive healthcare support.

Transportation Services: Ola and Uber provide robust vehicle services but lack integration with public transportation information and vehicle maintenance services.

Finance Applications: Mint and banking apps offer detailed financial information but do not cover the breadth of government services such as Aadhar or land registration.

Government Service Platforms: UMANG is a notable example that integrates multiple government services, yet it does not extend to areas like housing services and community helpers.

The integration of these services into a single platform is a unique approach that aims to streamline access to essential information and services, thereby improving user convenience and efficiency.

Theoretical Framework and Key Concepts

The development of the web application is grounded in several key theoretical frameworks and concepts:

1. User-Centred Design (UCD): This framework emphasizes the importance of designing the application around the needs, preferences, and limitations of end-users.

2. Service Integration: Integrating various services into a single platform is based on the concept of service-oriented architecture (SOA). SOA allows for the modular design of applications, where each service can operate independently yet cohesively, providing a seamless user experience.

3. Information Retrieval and Management: Efficient information retrieval is crucial for the application's success. This involves the use of databases and algorithms to ensure that users can quickly access relevant information. Concepts such as indexing, search algorithms, and data normalization play a vital role in this process.

4. Security and Privacy: Given the sensitive nature of the information being handled (e.g., health records, financial details), the application must adhere to stringent security protocols.

Concepts such as encryption, secure authentication, and data privacy regulations (e.g., GDPR) are fundamental to protecting user data.

5. Quality of Service (QoS): The application must meet certain QoS requirements to ensure optimal performance. This includes factors such as network speed, server response times, and the ability to handle concurrent users without degradation in performance.

By synthesizing these frameworks and concepts, the web application aims to provide a comprehensive, user-friendly solution that addresses the diverse needs of common people in their daily lives. This integrative approach not only fills existing gaps in individual service areas but also enhances the overall user experience by providing a one-stop solution for various day-to-day issues.

III. Requirement Analysis

Functional requirements.

1. User Management

- **User Registration and Login:** Secure registration and authentication for users.
- **User Profile Management:** Users can view and edit their profiles.
- **User Roles:** Different roles (e.g., admin, service provider, general user) with appropriate permissions.

2.Service Integration

- **Transport Services:** Access to official and trustworthy IRCTC website and some of the state bus services' website to surf through and Metro Maps and details of metros of various cities.
- **Healthcare Services:** Access to the location of nearby Healthcare facilities and Pharmacies.
- **House Services:** Booking home maintenance and repair services, including cleaning, plumbing, electrical services and information about packers and movers.
- **Government Services:** Access to government services such as document requests, applications, and public information.

- **Financial Services:** Can locate nearest ATMs and banks and can get to calculate Monthly Budget.

3. Dashboard

- **Unified Dashboard:** A central dashboard displaying relevant information and access points for all integrated services.

4. Search and Navigation

- **Search Functionality:** Users can search for specific services or information across the platform.
- **Navigation:** Easy and intuitive navigation between different services and sections of the platform.

5. Security and Privacy

- **Data Encryption:** Secure storage and transmission of user data.

Non-Functional Requirements

1. Usability

- **User-Friendly Interface:** Intuitive and accessible design for users with varying levels of digital literacy.

2. Security

- **Authentication:** Strong authentication mechanisms to ensure secure access.
- **Authorization:** Role-based access control to protect sensitive data and functionalities.

3. Maintainability

- **Modular Architecture:** A modular design to facilitate easy maintenance and updates.
- **Documentation:** Comprehensive documentation for developers and users.

4. Compliance

- **Legal and Regulatory Compliance:** Adherence to relevant laws and regulations, such as data protection and privacy laws.

System Requirements

1. Hardware

- **Servers:** Reliable servers to host the platform and handle the expected load.
- **Storage:** Adequate storage capacity for user data, service information, and logs.

2. Software

- **Backend:** Django and Nodejs for server-side operations.
- **Frontend:** Bootstrap for responsive and user-friendly interfaces.
- **Database:** A robust database system (e.g. SQLite) for data storage and retrieval.

3. Network

- **Internet Connectivity:** Reliable internet connection for accessing online services and data.

Stakeholder Requirements

1. Users

- Easy access to integrated services.
- Secure and private handling of personal data.

2. Service Providers

- Efficient service management and delivery.
- Secure and compliant platform for service provision.

3. Administrators

- Platform monitoring and maintenance tools.
- User and service management capabilities.
- Comprehensive reporting and analytics for platform performance.

IV. Implementation

Technologies and Tools Used

- Programming Languages: Python, HTML, CSS, RSS
- Frameworks: Django
- Libraries: Bootstrap
- Databases: SQLite
- Version Control: Git
- IDE: VS Code
- Chatbot used for easy access to all the services

Development Environment Setup

Install Python:

Ensure Python 3.x is installed on your system. Verify by running

python --version.

Install Django:

Use pip to install Django:

pip install django.

Install Required Libraries:

Install Django and Bootstrap

pip install django

pip install django-bootstrap4

-

Start a New Django Project:

Initialize the Django project and app:

```
django-admin startproject integratedservices
cd integratedservices
```

```
django-admin startapp integratedserviceforcommonpeople
```

Configure Django Settings:

Update settings.py to include the new app and any required configurations:

```
INSTALLED_APPS = [
    ...
    'integratedserviceforcommonpeople',
]
```

Implementation Details of Major Modules/Components

Authentication Module

Signup and Signin:

Using Django's built-in authentication system for user management.

Models:

Django's User model is used for managing user accounts.

Views:

signup: Handles user registration.

signninw: Handles user login.

Forms:

Utilizing Django's UserCreationForm for signup.

Custom forms can be created if additional fields are required.

Templates:

signup.html and signin.html for rendering the respective forms.

Signup View:

```
def signup(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=password)
            login(request, user)
            return redirect('home')
    else:
        form = SignUpForm()
    return render(request, 'signup.html', {'form': form})
```

Signin View:

```
def signin(request):
    if request.method == 'POST':
        form = SignInForm(data=request.POST)
        if form.is_valid():
```

```
            user = form.get_user()
            login(request, user)
            return redirect('home')
    else:
        form = SignInForm()
    return render(request, 'signin.html', {'form': form})
```

Service Registration and Listing Module

Models:

Define the Service model to store service details.

Code:

```
SERVICE_TYPES = [
    ('hospitals', 'Hospitals'),
    ('cab', 'Cab'),
    ('government', 'Government'),
    ('tourism', 'Tourism'),
]

class Service(models.Model):
    name = models.CharField(max_length=100)
    location = models.CharField(max_length=100)
    service_type = models.CharField(max_length=20, choices=SERVICE_TYPES)
    mobile_number = models.CharField(max_length=15)
    email = models.EmailField()
    whatsapp_number = models.CharField(max_length=15)
    website = models.URLField(blank=True, null=True)
    address = models.CharField(max_length=255, blank=True, null=True)

    def __str__(self):
        return self.name
```

Views:

service_registration: Handles the creation of new services.

service_list: Displays a list of registered services.

Templates:

service_registration.html for the service registration form.

service_list.html for displaying the list of services.

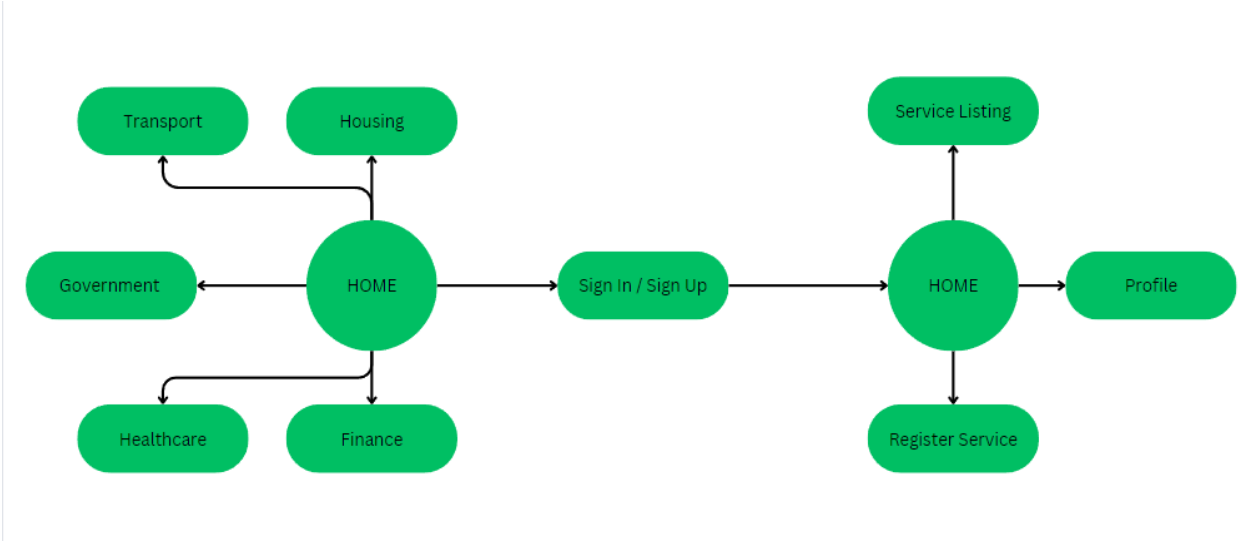
Register Service View:

```
@login_required
def service_registration(request):
    if request.method == 'POST':
        form = ServiceRegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('service_list')
    else:
        form = ServiceRegistrationForm()
    return render(request, 'service_registration.html', {'form': form,'username': request.user.username})
```

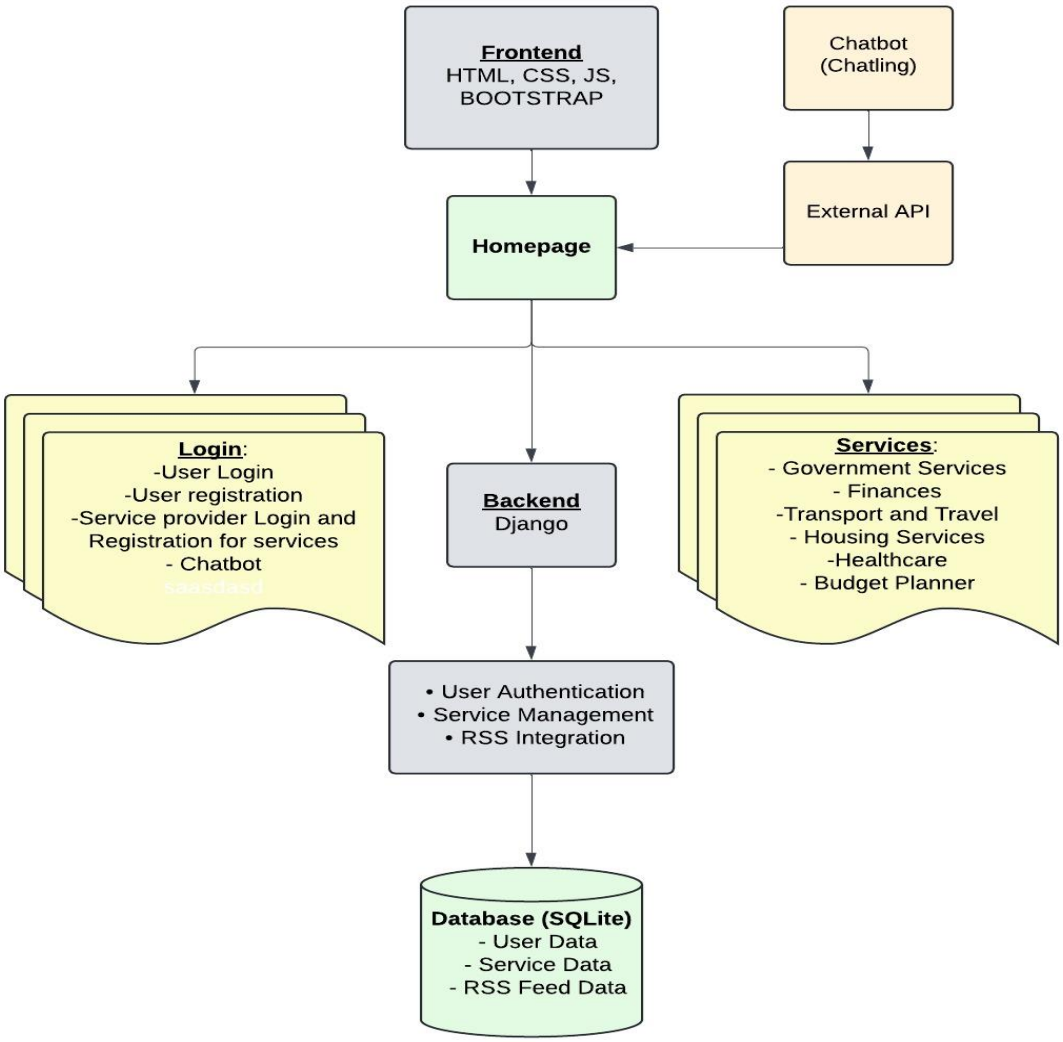
List Services View:

```
@login_required
def service_list(request):
    services = Service.objects.all()
    service_type = request.GET.get('service_type')
    if service_type:
        services = services.filter(service_type=service_type)
    return render(request, 'service_list.html', {'services': services, 'service_types': SERVICE_TYPES,'username': request.user.username})
```

Diagram



Architecture Diagram:



V. Testing

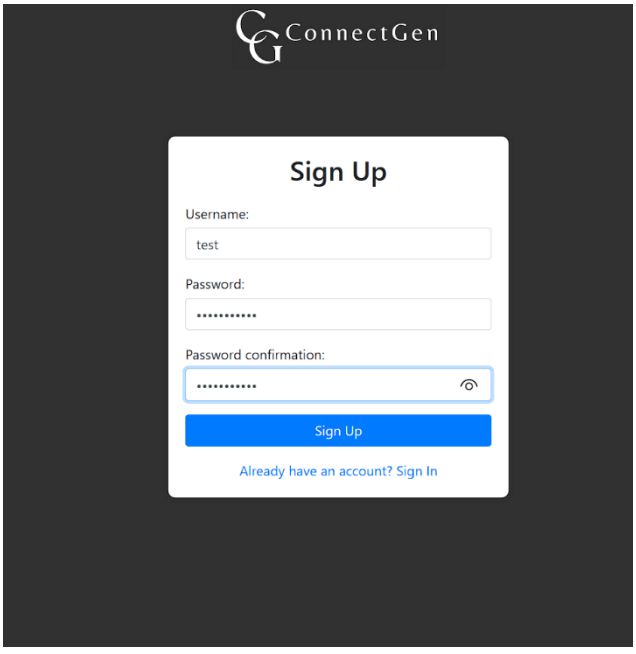
Testing Methodologies Used

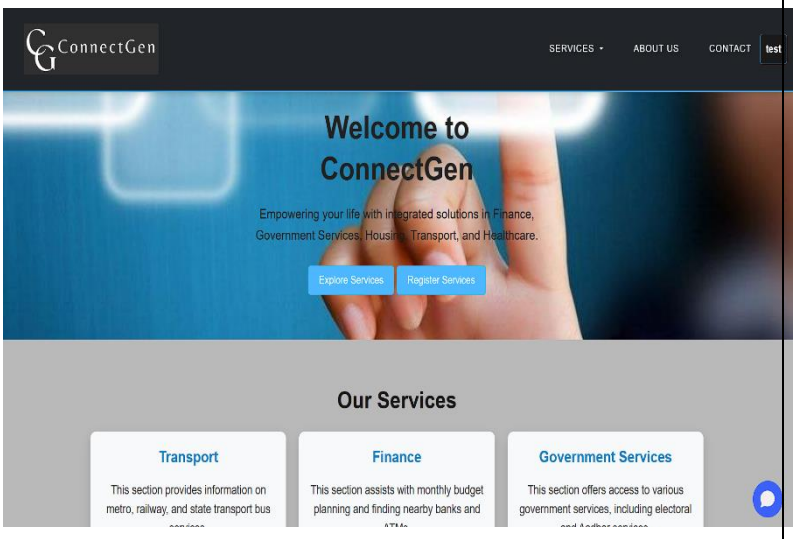
- **Unit Testing**: Testing individual components and functions in isolation.
- **Integration Testing**: Testing the interaction between different modules to ensure they work together correctly.
- **System Testing**: Testing the entire system to verify that it meets the specified requirements.

Test Cases and Test Scenarios

1. **Signup Test**:
    - **Scenario**: A user signs up with valid credentials.
    - **Steps**:
        1. Navigate to the signup page.
        2. Fill in the form with valid data.
        3. Submit the form.
    - **Expected Result**: User is redirected to the signin page.
    - **Actual Result**: Pass
2. **Signin Test**:
    - **Scenario**: A user signs in with valid credentials.
    - **Steps**:
        1. Navigate to the signin page.
        2. Fill in the form with valid credentials.
        3. Submit the form.
    - **Expected Result**: User is redirected to the home page.
    - **Actual Result**: Pass

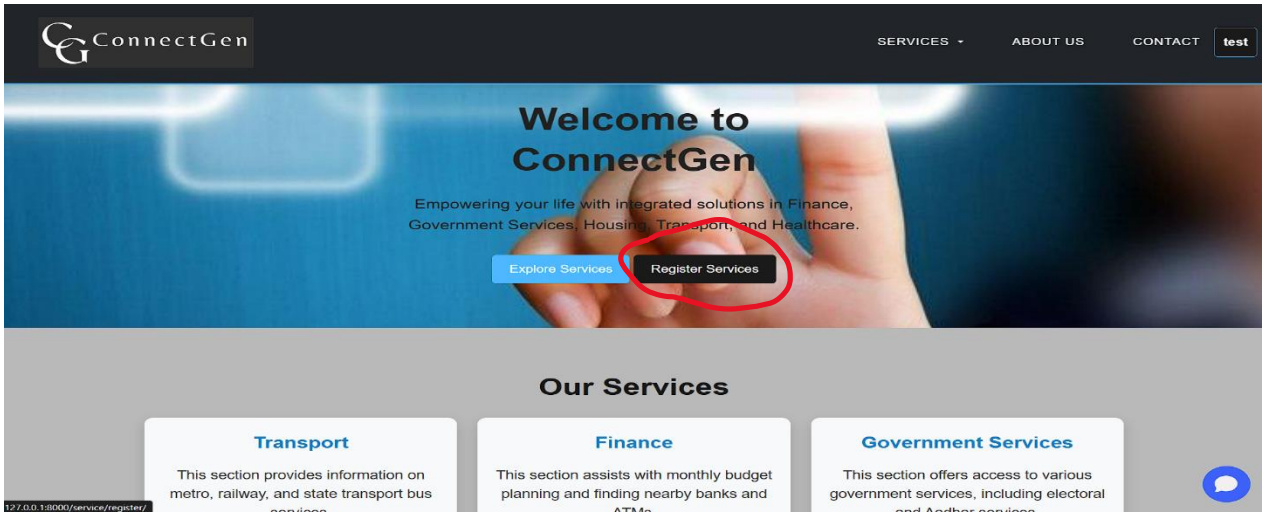
SCREEN SHOTS:

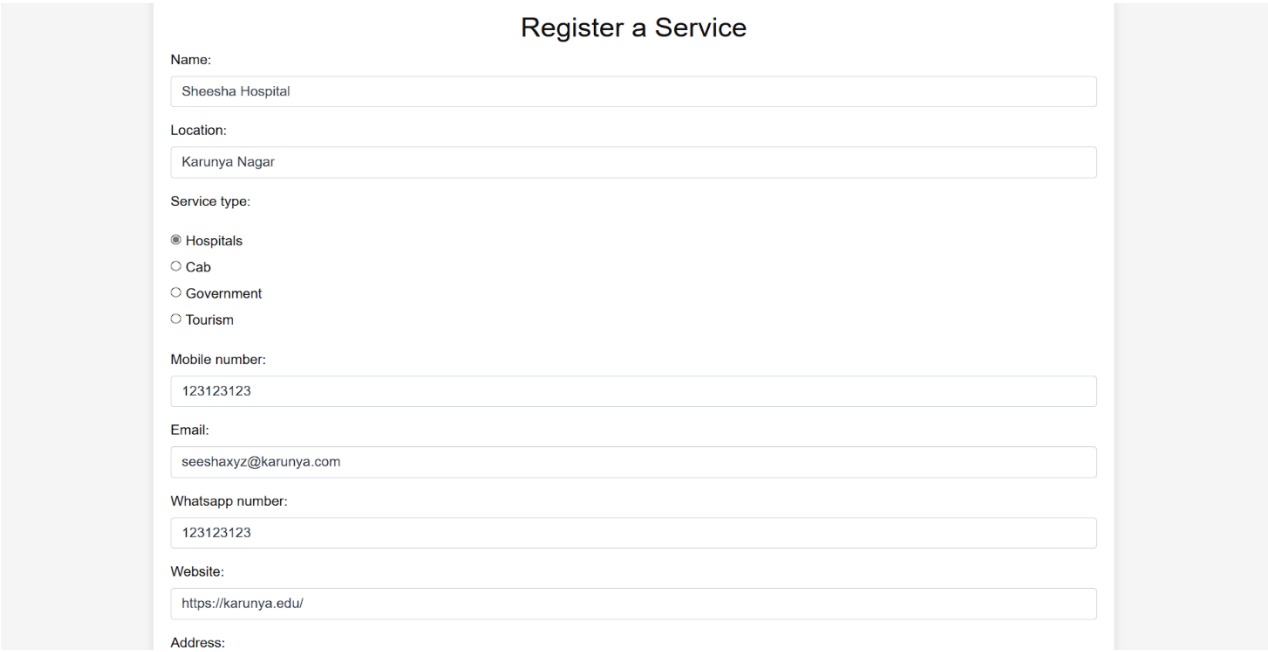




3. **Service Registration Test**:
    - **Scenario**: A user registers a new service with valid details.
    - **Steps**:
        1. Navigate to the service registration page.
        2. Fill in the form with valid service details.
        3. Submit the form.
    - **Expected Result**: Service is listed on the services page.
    - **Actual Result**: Pass

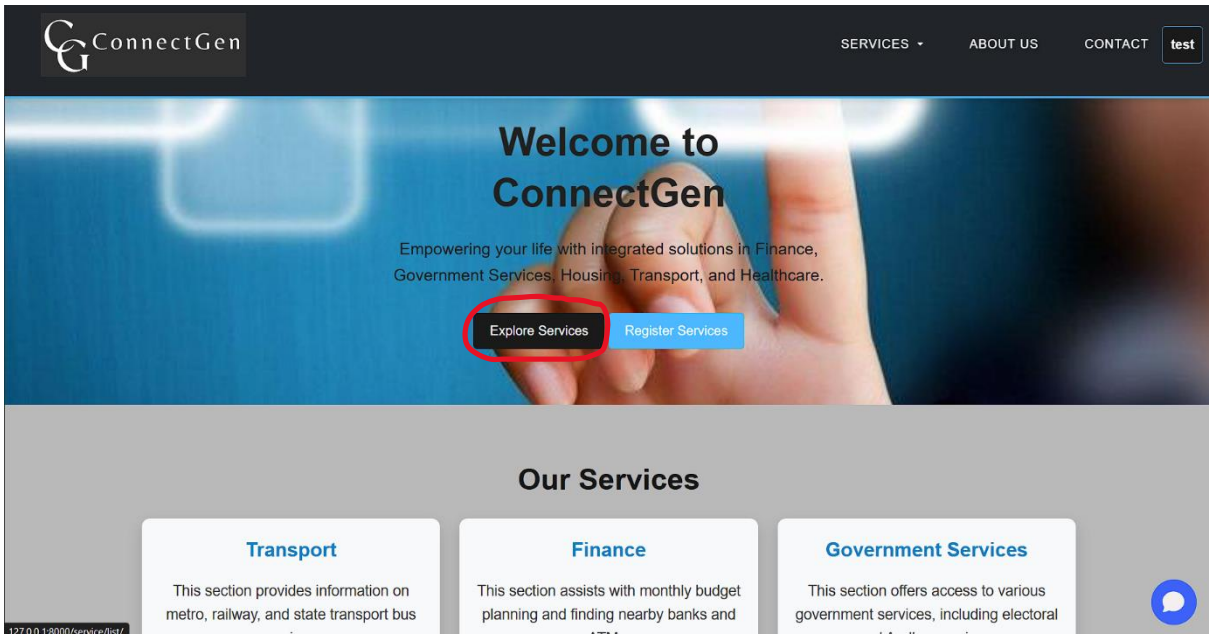
SCREEN SHOTS:

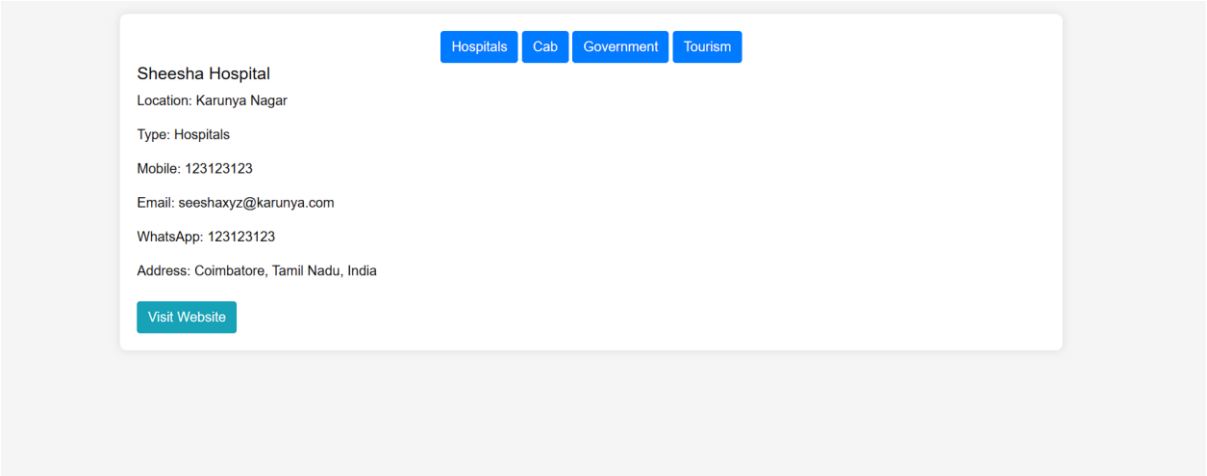




4. **Service Listing Test**:
    - **Scenario**: A user views the list of registered services.
    - **Steps**:
        1. Navigate to the services listing page.
    - **Expected Result**: The page displays a list of all registered services.
    - **Actual Result**: Pass

SCREEN SHOTS





VI. Conclusion and Future Work

Conclusion

The "Integrated Common Services to Common People" project under the Intel Unnati Industrial Training Program 2024-2025 has successfully developed a comprehensive web application that addresses the core issue of fragmented service delivery systems. By consolidating Transport, Healthcare, House, Government, and Financial services into a unified platform, the project provides a streamlined and user-friendly interface that simplifies access to essential services. This integration not only enhances the user experience but also reduces operational inefficiencies and costs for service providers. The project's key achievements include:

- **Unified Interface:** Successfully developed a central platform where users can access multiple services seamlessly.
- **Enhanced Accessibility:** Implemented a user-friendly design that caters to users with varying levels of digital literacy and supports multiple devices.
- **Service Integration:** Integrated a wide range of services, including healthcare, transport, and financial services, into a single platform.
- **Operational Efficiency:** Reduced redundancies and improved service delivery efficiency for providers.
- **Personalization:** Leveraged data analytics to offer personalized recommendations and services tailored to user needs.

The project's success demonstrates its potential to significantly improve the quality of life for individuals, particularly those in underserved and rural communities, by making essential services more accessible, efficient, and user-centric.

Future Work

While the current implementation of the "Integrated Common Services to Common People" project provides a solid foundation, several areas offer opportunities for future enhancement and development:

1. Expansion of Services:

- Integrate additional services such as education, legal aid, elder care, and entertainment to further broaden the platform's utility.

2. Mobile Application Development:

Develop a mobile application to complement the web platform, ensuring that users can access services on-the-go with ease.

3. Advanced AI Integration:

Incorporate more advanced artificial intelligence and machine learning algorithms to enhance personalization, predictive analytics, and automated support.

4. Localization and Multilingual Support:

Implement localization features to cater to diverse geographical regions and support multiple languages to reach a wider audience.

5. Enhanced Security Measures:

Continuously update security protocols to address emerging threats and ensure the highest level of data protection and user privacy.

6. Community Engagement Features:

Introduce features that foster community engagement and peer support, such as forums, user groups, and community events.

7. Partnerships and Collaborations:

Establish partnerships with additional service providers, NGOs, and government agencies to expand the range of services and resources available on the platform.

8. User Feedback and Continuous Improvement:

Implement a robust feedback mechanism to gather user input and continuously refine and enhance the platform based on user needs and preferences.

By focusing on these areas, the project can continue to evolve and adapt, ensuring it remains a valuable and indispensable resource for all users. The ongoing commitment to innovation and user-centric design will drive the success and sustainability of the "Integrated Common Services to Common People" platform.

Project video Link

https://youtu.be/xtRLUWlZknA?si=fM7fxwNnp4x2fIMS