CODE 4 GovTech
Building Digital Public Goods

C4GT DMP - Proposal Template

Name	Aditya Sharma
Email ID	aditya0202sharma@gmail.com
Phone Number	+91 [redacted]
GitHub ID	https://github.com/adsh16
Discord ID	https://discord.gg/UDmYU2YX
Current occupation *(Working Professionals - add current organization & years of exp)*	Student
Education Details *(College Name - Degree Name and branch of engineering or other course/specialization)*	College Name : Indraprastha Institute Of Information Technology (IIITD) Degree : B.Tech Major : Computer Science
Technical skills with level *(Mention tech skills/languages known/UI-UX and level - Novice/Intermediate/Expert)*	JavaScript/TypeScript: Intermediate Web Development: Intermediate Python/Flask: Expert React/Next.js: Intermediate API Design: Intermediate Cloud Services (Firebase): Intermediate Docker/Containerization: Intermediate SQL: Intermediate CI/CD (GitHub Actions): Intermediate Security/Cryptography: Intermediate OAuth/OpenID Connect: Novice UI/UX Design: Intermediate

Title: Scheme Training Discovery Microservices for Open Networks(ONEST/ONDC)

Summary

My vision is to unlock Haqdarshak's extensive library of over 5,000 vital training modules, available in 12+ languages, making them easily discoverable for everyone, everywhere. Currently, accessing these valuable skilling resources can be challenging. By integrating with open networks like ONEST/ONDC using the Beckn protocol, we can create a seamless, standardized discovery experience, significantly broadening last-mile access to crucial skilling opportunities and empowering individuals across India.

To achieve this, I propose building a robust and scalable microservices architecture. This solution involves two core components: a BPP (Provider) microservice that exposes Haqdarshak's training catalog via multilingual, filterable search APIs compliant with ONEST specifications, and a BAP (Seeker) client SDK with a companion React widget to allow effortless integration by partner platforms. Key deliverables also include essential telemetry logging for usage analytics and comprehensive documentation to ensure smooth partner onboarding. The technical approach prioritizes resilience and interoperability, utilizing Node.js/Express complemented by Elasticsearch for powerful multilingual search, and emphasizes rigorous testing including unit, integration, and crucial contract tests.

Project Detail

1. Project Overview

a. Understanding of the project

Haqdarshak aims to make its extensive catalog of verified training schemes (5000+ modules, 12+ languages) openly discoverable on networks like ONEST (Open Network for Education and Skilling Transactions) / ONDC. This requires building interoperable microservices based on the Beckn protocol:

- **BPP (Provider) Service:** Exposes Haqdarshak's training modules via standardized APIs, supporting filtered (topic, level, language) and multilingual search according to ONEST domain specifications.
- **BAP (Seeker) Service/Client:** Enables third-party platforms to easily integrate and query for these training modules for their users (agents, citizens).

- **Telemetry:** A system to track and visualize usage patterns, search effectiveness, and discoverability metrics.
- **Ease of Integration:** Deliverables must be "plug-and-play" with clear documentation and tools (SDK/Widget) for seamless partner onboarding.
- **Optional Goals:** Voice-enabled search PoC.

The core goal is to enhance last-mile access and foster interoperability within the digital public goods ecosystem for skilling, aligning with the vision of open networks.

📘 Sample JSON Input for a Training Module


```
{
  "id": "module-001",
  "descriptor": {
    "name": {
      "en": "Digital Literacy Basics",
      "hi": "डिजिटल साक्षरता की मूल बातें"
    },
    "short_desc": {
      "en": "An introductory course on digital tools and internet usage.",
      "hi": "डिजिटल उपकरणों और इंटरनेट उपयोग पर एक परिचयात्मक पाठ्यक्रम।"
    },
    "long_desc": {
      "en": "This module covers the fundamentals of using smartphones, accessing the internet, and understanding online safety.",
      "hi": "यह मॉड्यूल स्मार्टफोन का उपयोग करने, इंटरनेट तक पहुंचने और ऑनलाइन सुरक्षा को समझने की मूल बातें शामिल करता है।"
    }
  },
  "tags": [
    { "code": "topic", "value": "digital-literacy" },
    { "code": "level", "value": "beginner" },
    { "code": "language", "value": "hi" }
  ],
  "location": {
    "city": {
      "name": "New Delhi"
    },
    "country": {
      "code": "IN"
    }
  },
  "provider": {
    "id": "haqdarshak.training",
```

```
  "name": "Haqdarshak Training Services"
 },
 "time": {
  "label": "duration",
  "duration": "PT2H"  // ISO 8601 duration format
 },
 "content": {
  "url": "https://training.haqdarshak.org/modules/digital-literacy-basics",
  "format": "video",
  "language": "hi"
 }
}
```

b. Issues that might come up and the support needed from the org (if any)

- **Beckn/ONEST Specification Nuances:** Ensuring precise adherence to potentially evolving or complex protocol specifications for the skilling domain.
    - *Support Needed:* Access to mentors/Haqdarshak team for clarifications, latest ONEST specification documents, potentially connections within the Beckn/ONEST community.
- **Multilingual Search Complexity:** Implementing accurate and performant search across diverse Indic scripts and languages within Elasticsearch requires careful analyzer configuration.
    - *Support Needed:* Access to representative multilingual test data from Haqdarshak.
- **Partner Integration Friction:** Ensuring the BAP SDK/widget is genuinely easy for diverse third parties (with varying technical capabilities) to adopt.
    - *Support Needed:* Feedback from Haqdarshak on potential partner needs/technical capabilities; potentially beta testing with a friendly partner.
- **Data Modelling:** Defining a flexible yet standardized schema for evolving training content that aligns with ONEST standards.
    - *Support Needed:* Collaboration with Haqdarshak on finalizing the TrainingModule data schema.
- **Data Nature:** if the dataset for training modules is a static catalog (like JSON files synced periodically) and all discovery happens via Elasticsearch indexing that catalog, need for a NOSQL based database is removed. We can work with simpler solutions
- **Telemetry Nature needed** : Collaboration with Haqdarshak early on to define the essential telemetry requirements and dashboard views versus desirable future analytics. This will help prioritize implementation and ensure the initial dashboard meets core needs effectively.

c. Solutions

The proposed solution is a robust, scalable, and interoperable system built on a microservices architecture, specifically designed to meet Haqdarshak's goals and integrate seamlessly with open networks like ONEST.

I. Core Architecture & Technology Stack:

- **Microservices Approach:** We propose distinct BPP (Provider) and BAP (Seeker) microservices, potentially separate services for Search Indexing or Telemetry processing.
    - *Justification:* This is superior to a monolith as it allows independent scaling of discovery (BPP) vs. consumption (BAP integration). It enables Haqdarshak to update training content or BPP logic without disrupting integrated partner platforms. Fault isolation ensures high availability, critical for reliable open network participation.
- **MERN Stack:** Utilizing Node.js, Express.js, React.js, and MongoDB.
    - *Suitability:* Node.js/Express provides high-performance, non-blocking I/O ideal for API-heavy interactions. MongoDB is optional and would only be used if dynamic data persistence is required. A NoSQL DB's flexible schema will be able to handle evolving multilingual training data efficiently. React enables a reusable, embeddable BAP widget for easy partner integration.
- **Elasticsearch:** Employed for the core discovery function.
    - *Necessity:* Essential for handling complex, multilingual search across 12+ languages with low latency. Its advanced text analysis (customizable analyzers/tokenizers for Indic languages) and powerful faceting engine directly enable accurate filtering by topic, level, and language, far exceeding basic database search capabilities.
- **API Gateway:** For the initial scope focusing on the BPP/BAP interaction, Express.js routing within the BPP/BAP services themselves might suffice for basic routing and middleware needs (like signature verification).
- **Asynchronous Telemetry:** Possibly we can use the below sequence
    - We can start with structured logging and basic aggregation/dashboarding.
    - If needed a more robust and advanced telemetry sequence can be used.
    - RabbitMQ (or similar message queue) -> Node.js consumer -> Time-series DB (Prometheus/InfluxDB) -> Grafana/Recharts Dashboard.
    - *Justification:* Decouples telemetry generation from processing, ensuring discovery performance isn't impacted. Provides resilient event handling and enables rich analytics on usage patterns, search terms, filter usage, etc., allowing Haqdarshak to optimize content and demonstrate impact.
- **Integration of Voice-enabled discovery:** The implementation would primarily reside on the client-side (BAP), integrating with the existing search flow without requiring changes to the BPP's core Beckn API endpoints.

- Implementation resides entirely on the client-side (BAP), likely within the React widget.
- Leverages the browser's built-in **Web Speech API (SpeechRecognition)** for speech-to-text conversion.
- A UI element (e.g., microphone button) triggers listening for a specific configured language. (example could be 'hi-IN' Hindi-India or 'en-US' English-US).
- The transcribed text is then used to populate the standard search input or directly passed as criteria to the existing BAP SDK search function.
- The subsequent Beckn /search interaction with the BPP remains unchanged, using the transcribed text as input. No modifications are needed on the BPP side.

II. Beckn Protocol Implementation & Data Exchange:

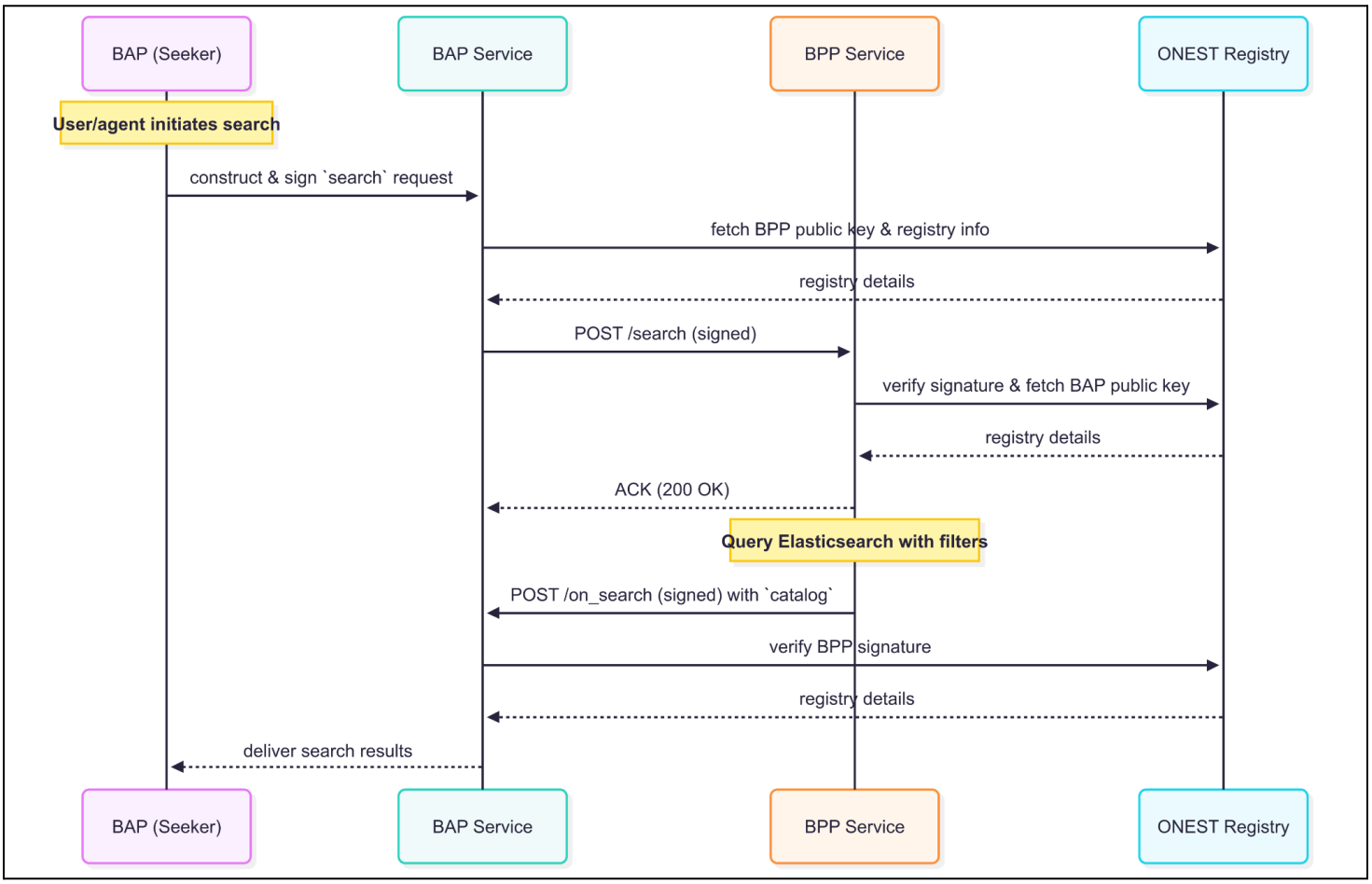

Figure: Demo Proposed Sequence Diagram of Beckn Protocol's Search and On_Search Flow between BAP and BPP, subject to change based on requirements.

- **Protocol Adherence:** We will strictly adhere to the Beckn Protocol specifications as adopted by ONEST for the skilling domain (referencing ONDC-Official/ONDC-ONEST-Specifications or latest sources).
- **Core APIs:** The BPP will primarily implement the Beckn **Discovery APIs**:
    - search: BAP initiates discovery by sending a search request (HTTPS POST) to the BPP's designated endpoint (e.g., /search).
    - on_search: BPP processes the request and asynchronously sends the results (catalog) back to the BAP via a callback URL provided in the search request's context.
- **Data Structure:** Requests and responses will use the standard Beckn JSON structure, including:
    - context: Contains transaction metadata (domain, action, transaction_id, message_id, BAP/BPP IDs, timestamp, callback_url etc.).
    - message: Contains the core payload, primarily the intent for search (specifying criteria like category=Scheme Training, item details like topic/language/level) and the catalog for on_search (containing provider details and discoverable training items/modules).
- **Data Exchange Flow (Discovery):**
    1. Partner Platform (using BAP SDK/Widget) initiates a search.
    2. BAP SDK constructs a Beckn search request JSON (Context + Message with Intent).
    3. BAP SDK sends the search request (HTTPS POST) to the designated Haqdarshak BPP endpoint. The request includes a callback_url.
    4. Haqdarshak BPP receives the request, validates it (including signature verification - see below).
    5. BPP immediately sends back an ACK (Acknowledgement) response to the BAP, confirming receipt.
    6. BPP queries Elasticsearch based on the search intent filters.
    7. BPP constructs the on_search response JSON (Context + Message with Catalog containing matching training modules).
    8. BPP sends the on_search response (HTTPS POST) to the callback_url provided in the initial search request's context.
    9. The BAP (or its gateway) receives the on_search response, validates it, parses the catalog, and the BAP SDK/Widget displays the results on the partner platform.
- **Security:** Adherence to Beckn/ONDC security guidelines is critical. This involves:
    - **Request Signing:** All communication between BAP/BPP (often via Gateways) must be digitally signed using a standard mechanism like HTTP Signatures (using algorithms like XEd25519 as specified in Beckn). Each participant (BAP, BPP) needs a unique subscriber ID and associated cryptographic key pair registered on the network registry.
    - **Signature Verification:** Receiving entities must verify the signature using the sender's public key (looked up via the registry) to ensure authenticity and

message integrity. The BPP will verify signatures on incoming search requests, and the BAP will verify signatures on incoming on_search responses.

III. Documentation and Partner Onboarding Strategy:

Our goal is seamless onboarding for partner platforms wishing to integrate Haqdarshak's training discovery. This will be achieved through:

- **OpenAPI Specification:** The BPP's RESTful API endpoints (including Beckn endpoints like /search) will be meticulously documented using the OpenAPI 3.0 standard.
    - *Deliverable:* A openapi.yaml or openapi.json file in the repository.
    - *Benefit:* Provides a machine-readable contract, enabling automatic generation of interactive documentation (Swagger UI), client SDK stubs, and automated testing. Details endpoints, HTTP methods, parameters, request/response schemas (aligned with Beckn JSON structures), and authentication methods.
- **Interactive API Documentation (Swagger UI):** We will host a Swagger UI instance generated from the OpenAPI spec, allowing partners to easily browse the API, understand its usage, and even make test calls directly from their browser.
- **BAP SDK & React Widget Documentation:** Comprehensive documentation will accompany the TypeScript SDK and React Widget.
    - *Content:* Clear installation steps (npm install @haqdarshak/bap-client-sdk), configuration guides (setting BPP URL, optional parameters), detailed usage examples (embedding widget, calling SDK functions, handling search results/errors), React component props documentation, and a full API reference for the SDK.
    - *Format:* Markdown files (README.md, /docs) within the SDK/Widget repository.
- **Step-by-Step Onboarding Guide:** A dedicated ONBOARDING.md file in the main project repository will guide partners through the integration process:
    - Overview of the architecture and data flow.
    - Prerequisites (e.g., network registration if needed, obtaining credentials).
    - Detailed instructions on integrating the BAP SDK/Widget (recommended approach).
    - Troubleshooting common issues (FAQs).
    - Contact information or support channels for assistance.
- **Code Examples:** Include practical code snippets within the documentation for common integration scenarios in popular frontend frameworks (starting with React).

IV. Deployment & Testing:

- **Deployment:** Containerization using Docker will be prioritized to ensure consistent environments and simplify setup. Dockerfiles for each service and a docker-compose.yml file will be provided for easy local execution of the BPP, BAP (if applicable as a service), and Elasticsearch. We will explore setting up basic CI/CD

pipelines using GitHub Actions for automated linting, testing, and potentially container image building upon code pushes to the main branches.

- **Testing:** We will implement a multi-layered testing strategy focused on ensuring correctness, reliability, and adherence to the Beckn protocol:
    - **Unit Tests:** Using frameworks like Jest (for both Node.js backend and React frontend), we will write comprehensive unit tests for individual functions, modules, and components to verify their logic in isolation. This forms the foundation for catching bugs early.
    - **Integration Tests:** These tests will verify the interactions within each microservice (e.g., API endpoint receiving a request, calling service logic, interacting with Elasticsearch/database, and formatting the response) without mocking external services entirely. For the BAP SDK, this includes testing its internal logic and state management.

This comprehensive solution leverages best practices and appropriate technologies to deliver a robust, scalable, and easily integrable system that meets Haqdarshak's requirements for participating effectively in the ONEST/ONDC open network.

2. Implementation Details with Timelines

Availability

Question	Value
Number of hours available to dedicate per week	Approx 30–40 hours
Do you have any other engagements (projects/internships)?	No
Share any other details about your availability clearly here	I am **fully dedicated** to this project and will prioritize it throughout the entire duration.

Milestones

This project plan is structured into five **2-week sprints**, focusing on delivering incremental value and ensuring core functionality is robustly implemented. The emphasis is on achieving the primary goals of building functional BPP and BAP components aligned with Beckn/ONEST principles, with realistic scope for testing and telemetry within the C4GT timeframe.

- **Sprint 1 (Weeks 1-2): Project Setup, Requirement Refinement & API Design**
    - The initial sprint will focus on establishing a solid foundation. We will set up the complete development environment, including Git repositories, necessary linters, and formatters. A key activity will be a deep dive into the ONEST specifications for the skilling domain and collaborating with Haqdarshak mentors to refine the

precise data schema for training modules, ensuring alignment. Based on this, we will draft the initial OpenAPI (Swagger) specifications for the core BPP /search and /on_search endpoints, defining the expected request/response structures according to Beckn standards. Basic project structure for the BPP microservice (Node.js/Express) will be created, along with initial Docker configuration for development consistency.

- **Sprint 2 (Weeks 3-4): BPP Core Implementation & Elasticsearch Integration**
    - Building on the design from Sprint 1, this sprint concentrates on the BPP service's core. We will implement the basic Express.js server setup, including middleware for request parsing and initial validation. A crucial step will be integrating Elasticsearch, setting up the client connection, defining the index mapping for training modules (considering multilingual fields), and implementing basic functions to index sample data provided by Haqdarshak. We will implement the BPP's /search endpoint to receive incoming Beckn search requests, validate their basic structure, and send the mandatory synchronous ACK response. Initial unit tests for the implemented server setup and Elasticsearch interaction logic will be added.
- **Sprint 3 (Weeks 5-6): BPP Search Logic, on_search Callback & Contract Testing Setup**
    - This sprint focuses on implementing the BPP's primary discovery logic. We will develop the functionality within the BPP to parse the intent from the incoming /search request and translate the filters (topic, level, language) into appropriate Elasticsearch queries, including handling basic multilingual search for a couple of key languages. The asynchronous on_search mechanism will be implemented, where the BPP constructs the Beckn catalog response containing search results and sends it via HTTPS POST to the BAP's callback URL specified in the original request's context. We will also set up the initial framework for contract testing (e.g., using Pact), defining the expected interactions between a hypothetical BAP and the BPP's /search and /on_search endpoints to ensure protocol adherence.
- **Sprint 4 (Weeks 7-8): BAP Client SDK, Basic Widget & Telemetry Logging**
    - With the BPP capable of basic discovery, this sprint shifts focus to the seeker side. We will develop a foundational BAP client SDK in TypeScript, encapsulating the logic for constructing Beckn /search requests (including context generation) and handling the incoming asynchronous /on_search responses. A simple React widget will be created that utilizes this SDK to provide a basic UI for initiating searches and displaying the retrieved training module results. On the BPP side, we will implement structured logging (e.g., JSON format to console output) for key events like receiving search requests (logging filters used) and sending

on_search responses, forming the basis for the telemetry requirement without the complexity of a full pipeline. Contract tests defined in the previous sprint will be refined and executed.

- **Sprint 5 (Weeks 9-10): Testing Refinement, Documentation & Handover Prep**
    - The final sprint is dedicated to consolidation, documentation, and ensuring the project is ready for handover. We will focus on increasing unit and integration test coverage for both BPP and BAP components. The contract tests will be finalized and integrated into a CI check if possible. Comprehensive documentation will be written, including a detailed README covering setup and architecture, finalizing the OpenAPI specification with examples, documenting the BAP SDK/Widget usage, and creating a partner onboarding guide. The Docker setup (docker-compose.yml) will be finalized for easy local deployment and testing of the BPP and Elasticsearch. We will prepare a final demonstration showcasing the end-to-end discovery flow using the BAP widget and BPP service. All code and documentation will be polished and organized in the Git repository.

Personal Information

About Me:

I'm Aditya Sharma, a third-year Computer Science student at IIIT-Delhi. Over the past couple of years I've built solid Computer Science fundamentals. I always try to work across different tech stacks, layers of abstraction and explore different domains. During my time while completing the Entrepreneurial Mindset Course at IIITD incubation center, I got exposed to the idea of social entrepreneurship. Wherein, the idea of creating an impact and monetary benefits goes hand in hand. I feel that Haqdarshak is also creating a good impact on society using technology.

I enjoy automating processes and incorporating new technologies in different products. I've been able to maintain code quality and expedite deployments thanks to my experience with tools like Docker and GitHub Actions. Additionally, I have practical experience with SQL/PostgreSQL databases, REST APIs, and contemporary frameworks like Flask, Node.js, and React.

What really fires me up is open source and public-interest tech. I've pitched in on community projects—fixing bugs, writing docs, polishing features—because there's something powerful about tools that anyone can use and improve. I've watched UPI and Aadhaar change lives, and I'm eager to bring that same spirit to Beckn/ONEST. Helping Haqdarshak turn government

schemes into an easy, multilingual discovery service feels like exactly the kind of project I want to be part of.

What is your motivation to apply for this project?

While browsing through available projects for C4GT, seeing Haqdarshak's name instantly popped up in the shark tank episode in front of me. I truly believe that Haqdarshak's mission to make government welfare schemes accessible to the masses, especially India's marginalized and needy families, is extremely powerful.

On the technical side, I am excited by the opportunity to implement a Beckn/ONEST-compliant microservices architecture, as ONEST's goal of unlocking timely, trusted discovery for 500 million+ young Indians aligns with my interest in scalable, interoperable systems. Designing a BPP Provider service with MongoDB and Elasticsearch for low-latency, language-aware search across 12+ local languages presents a compelling challenge in multilingual information retrieval. Integrating a BAP Seeker module—abstracting Beckn message flows and certificate management via the ONEST registry—offers hands-on experience with cutting-edge open-network protocols.

Please mention if you have solved any issues/tickets for this or other C4GT projects: (Optional)

- **CodeForGovTech Issues Solved [Open Community Projects - Code for GovTech]**

Link to to Issue	Resolution description in short	Link to pull request
Issue Link : Implement trip stitching algorithm to a real mobility workflow in beckn · Issue #1 · beckn/trip-stitching-algorithm	This PR introduces a full-featured Flask-based REST API around the trip-stitching algorithm (~1,196 added lines across five new files). I added: • **Core Algorithm Module** (api/trip_stitcher.py): A NetworkX-backed implementation of the multi-modal path-stitching	**Pull Request Link :** Implement Trip Stitching Algorithm API for Real Mobility Workflows by adsh16 · Pull Request #7 ·

	logic, complete with customizable optimization criteria (distance, duration, cost, transfers) and heuristics for discovering "missing links." • **API Layer** (api/app.py + api/requirements.txt): A modular Flask application exposing endpoints for health checks, trip stitching (POST /stitch-trips), geocoding (GET /find-locations), visualization (GET /visualize-trip/{id}), sample payloads (GET /sample-input), and OpenAPI documentation (GET /api-docs). • **Testing Suite** (api/test_trip_stitcher.py): Unit tests validating core stitching functionality under various optimization modes.	beckn/trip-stitching-algorithm
Issue Link : [C4GT Community]: Add Dynamic Color Scaling to GeoPlot Visualizer · Issue #44 · AgentTorch/visualize	This PR implements dynamic color scaling for the GeoPlot visualizer as requested in issue #44. It adds the ability to automatically detect and use the min/max values from the data to create more	**Pull Request Link :** feat: add dynamic color scaling options to GeoPlot visualizer and

	meaningful visualizations, especially when values vary significantly between episodes.	upd… by adsh16 · Pull Request #53 · AgentTorch/visualize

Previous experience/open source projects (Optional):

In this section you can mention your relevant work experience/projects (not just limited to open-source). You should mention experiences in this section if any with the relevant tech stack of the project (for product usability & design projects, the design software you used; like Figma; can be mentioned)

Project Name	Project Description	Links (if any)
DevOps Intern, HeydoTech	• Engineered and implemented a multi-stage CI/CD pipeline using GitHub Actions for a production application, automating linting, unit testing (pytest), security scanning (OWASP ZAP), and secure deployment via SSH; utilized act for efficient local workflow simulation and debugging. • Explored zero-downtime deployment strategies	HeydoTech : Linkedin **Advisors** • Anuj Grover • Akshmit Dewan

	including Blue-Green; designed and implemented a monitoring solution using Prometheus, black box exporter, and Grafana to visualize key application metrics such as availability, response time, and HTTP status codes.	
Undergraduate Research Intern, Complex Systems Laboratory	Created a RAG based AI Chatbot recommender named, GastroGenie. It is an intelligent recipe recommendation system that leverages Natural Language Processing (NLP) and semantic search to provide users with relevant recipes based on specific criteria such as protein content, preparation time, and regional cuisine. The system processes a dataset of large amount of recipes and employs a two-stage retrieval architecture combining vector search with cross-encoder	• Complex Systems Laboratory • Ganesh Bagler

	re-ranking for improved accuracy.	