Jet Propulsion Laboratory
California Institute of Technology

User Experience with HSML

JPL SIRI Internship Spring 2025 Final Presentation
Interns: Niki Namian, Sydni Yang, Sohee Kim
Section 345C
Mentors: Dr. Thomas Lu and Dr. Edward Chow
June 6th, 2025

Introduction - Niki Namian

- Hometown:
  - Los Angeles, CA
- Education:
  - First Year Computer Science Student at Santa Monica College
- Interests:
  - Baking, running, & playing pickleball
- Role:
  - Build a secure registration and authentication system
  - Test system for latency and optimized user experience

Introduction - Sydni Yang

Hometown: San Gabriel, CA

Education: 2nd Year Data Science Major @ Pasadena City College ⟶ transferring to UC Berkeley in Fall 2025

Interests: Music, reading, and sightseeing

Role:

- Maintained the setup and configuration of AWS EC2 infrastructure and integration of Kafka, MySQL, and HSML_API.
- Tested HSML_API implementation and documented findings.

Introduction - Sohee Kim

Hometown: Fullerton, CA

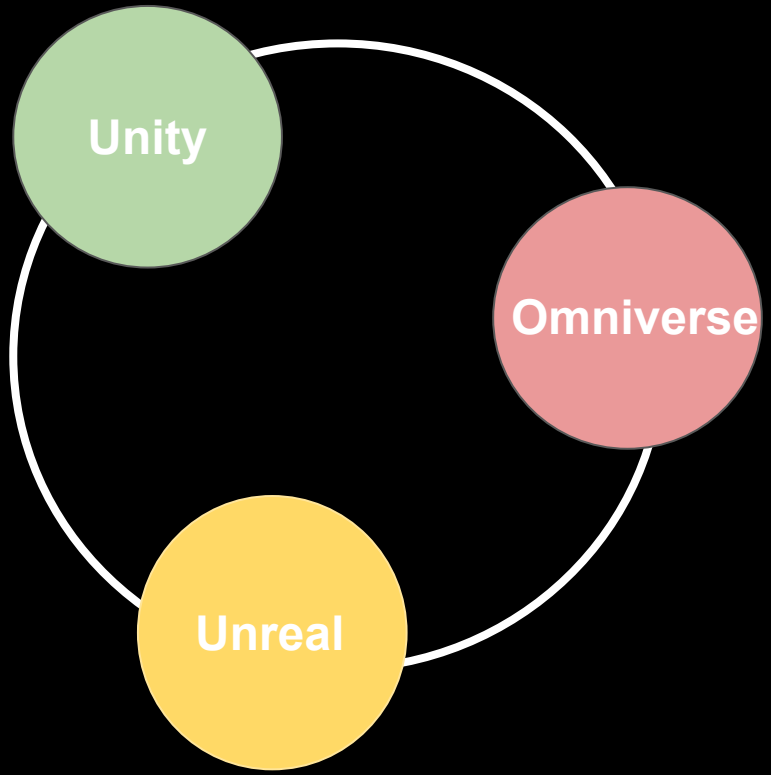
Education: 2nd Year Computer Science Major @ Fullerton College, transferring to UC Berkeley in Fall 2025

Role:

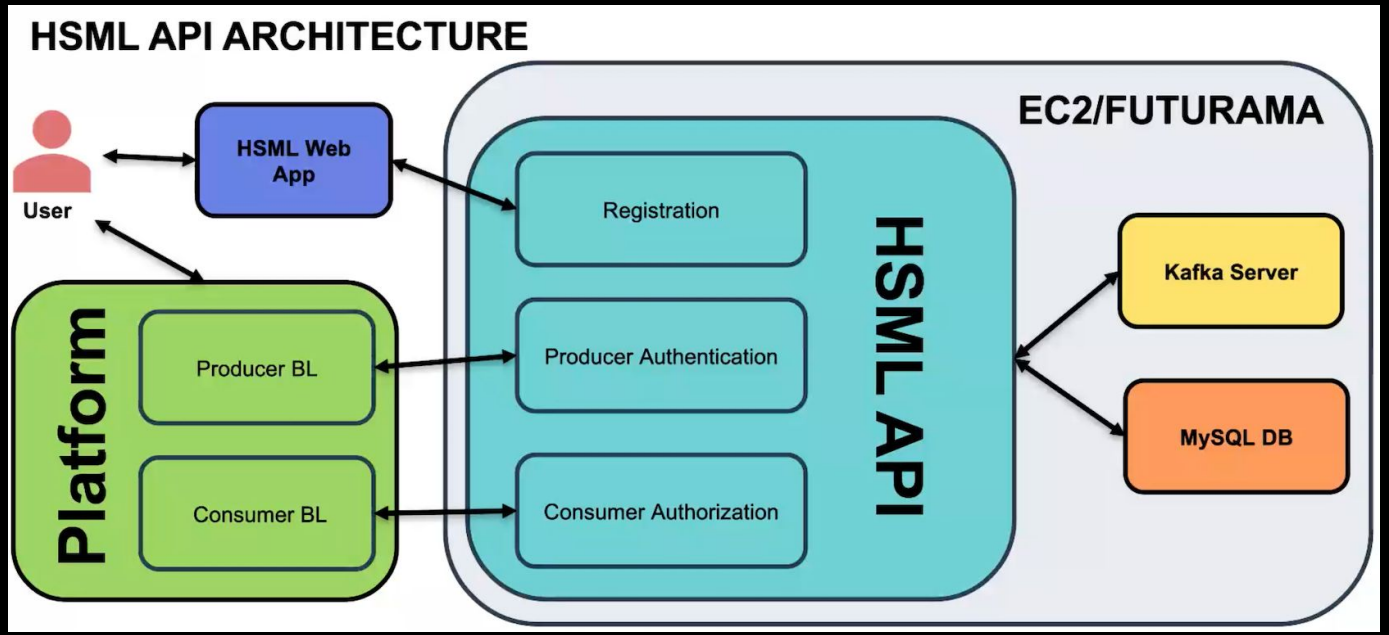
- Determined the business logic between HSML API, Kafka, and simulation platforms
- Performed user testing for HSML implementations
- Cleaned up unit testing documentation and added context to instructions

Importance of Digital Twin Interoperability Project

- ★ Enables real-time communication between digital twins across platforms
- ★ Using HSML (Hyperspatial Modeling Language) to send messages across Unity, Unreal, and Omniverse



Registration and Authentication



Credit to Alicia

This diagram shows the logic behind the registration and authentication components of the project. The user may use the HSML Web App to register themselves into the system. The HSML API is used to allow the platforms to connect. Once the user provides their private key, the system checks the database to validate their credential. If they are authenticated, they may access the system.

Hosting HSML API

- ★ Hosted inside FUTURAMA/EC2
- ★ HSML API is set up and connected to MySQL database and Kafka (EC2/FUTURAMA)
  - HSML Web App is connected to the HSML API

Note: HSML API is a Python package able to be set up in accessible through both FUTURAMA and EC2

Python API - Registration

Checks:

- ★ Validates the inputted JSON file (0.00 seconds)
- ★ Checks if the user is already registered (existing swid) (0.10 seconds)

Registry:

- ★ Generate did:key and connects to the swid property in the JSON
- ★ Stores public key and updated JSON in database
- ★ Provides user with private key (to be used for future logins)
    - Total Registration time: ~0.60 seconds

Python API - Registering Myself

```
It took 0.00 seconds to accept the HSML JSON file.

HSML JSON accepted.

It took 0.00 seconds to generate warning for registration/check required fields.


It took 0.21 seconds to check if swid already exists.

Warning: SWID 'did:key:6Mkw8x42L2aJsJccWb7ABjrWRQhM5SyJ5Q7TcYDv4aGGCWP' in JSON file will be overwritten.

Subprocess output:
Generated DID:key: did:key:6MkqKrmDSN1HrrLkW1ciPu23jtB9v27thzBPKRfbp17Eg4s
Private key saved to private_key.pem

Generated unique SWID: did:key:6MkqKrmDSN1HrrLkW1ciPu23jtB9v27thzBPKRfbp17Eg4s

It took 0.88 seconds to register.

Private key saved to: C:/Users/nnamian/OneDrive - JPL/Desktop/Digital Twin Interoperability/Codes/HSML Examples/registeredExamples\private_key.pem
Updated JSON saved to: C:/Users/nnamian/OneDrive - JPL/Desktop/Digital Twin Interoperability/Codes/HSML Examples/registeredExamples\Niki_Namian.json
```

Python API - Authenticating Myself

```
PS C:\Users\nnamian\Documents\GitHub\hsml_schema\scripts\verification> python .\Registration_API_v4.py
Must be registered in the Spatial Web to register a new Entity. Type 'new' to register or 'login' if already
registered: login
Provide your private_key.pem path: C:\Users\nnamian\OneDrive - JPL\Desktop\Digital Twin Interoperability\Code
s\HSML Examples\registeredExamples\private_key_Niki.pem

It took 0.01 seconds to authenticate.

Welcome Niki Namian, you can now register your new Entity.
Enter the directory to your HSML JSON to be registered:
```

Breakdown of HSML Implementation & Authentication Protocol

- ★ Setup of MySQL database
  - Holds the public key of entity and metadata in HSML JSON files
  - All JSON files use HSML schema
- ★ All users must be registered
  - Entities are registered into the database through the server
- ★ User must be authenticated before accessing topics
- ★ Track relationship between Person, Agent, and Credential
  - Only allow authenticated producers/consumers to access entities

*Note: checks are done through credentials
*Note: user must store private key

DID (Decentralized Identity) Implementation

- ★ Utilizes CLI tool to generate a key pair (private + public)
- ★ User saves the private key (not to be shared), public key is stored in database

Why DID?

- ★ Fast, no storage, no costs
- ★ Likeliness of same did:key being generated is almost zero
- ★ Ensures privacy, security, and interoperability
- ★ Robust way of registering entities in context of Spatial Web

Note: SWID (Spatial Web Identifier) is the identifier for the HSML Schema

User May Now Register/Access Entities

Kafka Producer:

- Provide private_key.pem path
- Once authenticated, user can now publish data in Kafka topic for the Agent

Kafka Consumer:

- Provide private_key.pem path
- Once authenticated, subscribe to the data in the Kafka topic

Note: Can’t have two of the same Kafka Topic Names in MySQL Database

→ Implement check for existing name in database

→ Implement function to randomize Kafka Topic Name

Verification Logic for Entities

- ★ Entity must be registered in the MySQL database

Producers:

- ★ While the Entity is an Agent, the database stores an associated Kafka topic acting as a producer
- ★ All Agents must provide their private key in order to publish data (must match public key in database)
  - Once verified, message is sent through server's topic

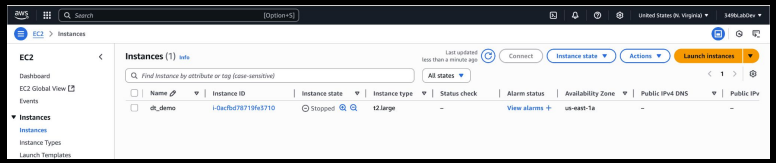
Consumers:

- ★ All consumers must provide private key in order to access Agent's topic
- ★ Once verified through credential in database, they may consume data/access Agent

AWS EC2 Background - Backbone of Integration

Amazon EC2 (Elastic Compute Cloud) → a web service that acts as a cloud-based computer that one can access from anywhere.

- Acts as a remote terminal with full privileges to install tools, run services, and manage files
- Can SSH into the server from any location



Commonly used to host:

- Backend services (ex: Apache Kafka, MySQL, FastAPI)
- Web applications and APIs (HSML_API)
- Real-time simulation environments (Omniverse, Unity, Unreal)

Why EC2?

- Hosts key services like Kafka, MySQL, and HSML_API in one centralized cloud environment, keeping the digital twin ecosystem connected.
- Provides a public-facing IP address, allowing remote access and testing from different terminals (ex: PowerShell).
- Bypasses local machine limitations, enabling cross-platform communication between Unity, Omniverse, and Unreal Engine.

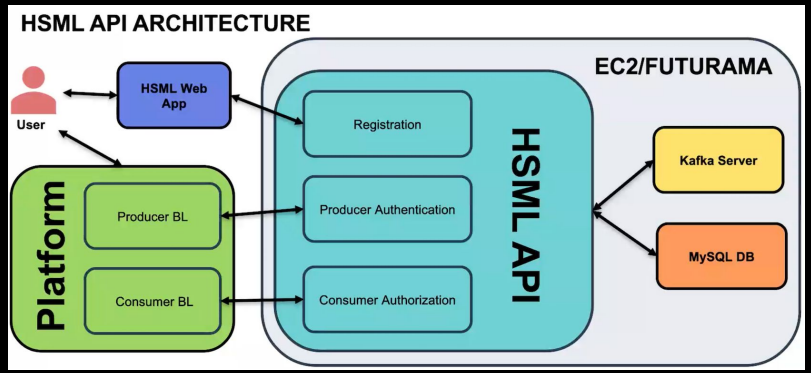


image credit: Alicia Sanjurjo Barrio

Kafka	Handles real-time communication
MySQL	Stores DT identities
HSML_API	Interface for registration and messaging DIDs

MySQL - Digital Twin Identity Registry

- MySQL serves as the identity management system for registering DT users, entities, and agents in the HSML framework.
- Stores information such as DIDs, public keys, metadata, and access credentials

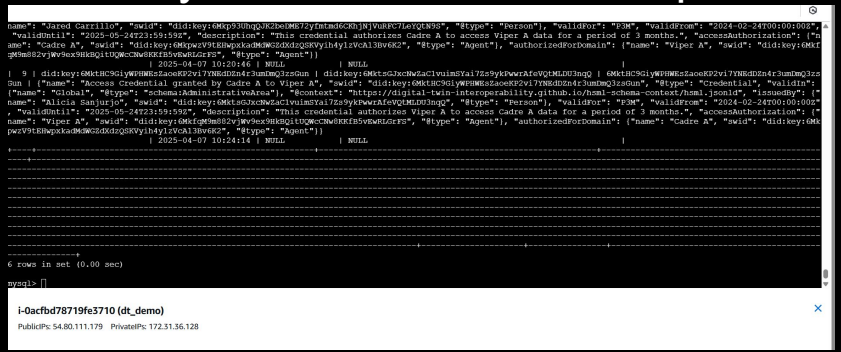
```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_did_registry |
+------------------------+
| did_keys               |
+------------------------+
1 row in set (0.00 sec)

mysql> DESC did_keys;
+---------------+--------------+------+-----+-------------------+-------------------+
| Field         | Type         | Null | Key | Default           | Extra             |
+---------------+--------------+------+-----+-------------------+-------------------+
| id            | int          | NO   | PRI | NULL              | auto_increment    |
| did           | varchar(255) | NO   | UNI | NULL              |                   |
| registered_by | varchar(255) | YES  |     | NULL              |                   |
| public_key    | text         | NO   |     | NULL              |                   |
| metadata      | json         | YES  |     | NULL              |                   |
| created_at    | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| kafka_topic   | varchar(255) | YES  |     | NULL              |                   |
| allowed_did   | text         | YES  |     | NULL              |                   |
+---------------+--------------+------+-----+-------------------+-------------------+
8 rows in set (0.00 sec)
```

MySQL Setup on EC2

- Install MySQL server on the EC2 Ubuntu instance
- Created did_registry database and did_keys table to store registered entities
- Used SQL commands to confirm and inspect data (ex: USE, SELECT *, SHOW TABLES;)

Entity Registration

- Manually registered 6 Digital Twin entities into MySQL to match lab computers (using FUTURAMA)
    - Queried database to confirm accuracy



Kafka Setup on EC2

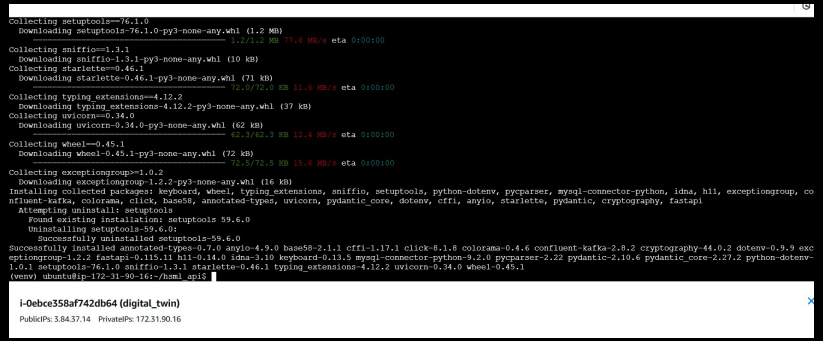
- Used Docker to install Kafka and Zookeeper
- Created three Kafka topics:

```
ubuntu@ip-172-31-90-16:~/kafka_2.13-3.9.0/bin$ ./kafka-topics.sh --create --topic unity-hsml-topic --bootstrap-server localhost:9092
Created topic unity-hsml-topic.
ubuntu@ip-172-31-90-16:~/kafka_2.13-3.9.0/bin$ ./kafka-topics.sh --create --topic omni-hsml-topic --bootstrap-server localhost:9092
Created topic omni-hsml-topic.
ubuntu@ip-172-31-90-16:~/kafka_2.13-3.9.0/bin$ ./kafka-topics.sh --create --topic unreal-hsml-topic --bootstrap-server localhost:9092
Created topic unreal-hsml-topic.
ubuntu@ip-172-31-90-16:~/kafka_2.13-3.9.0/bin$ ./kafka-topics.sh --list --bootstrap-server localhost:9092
omni-hsml-topic
unity-hsml-topic
unreal-hsml-topic
ubuntu@ip-172-31-90-16:~/kafka_2.13-3.9.0/bin$
```

- Modified server.properties for proper public listener configuration
- Ran Kafka services using the custom *start_kafka.sh* script on one instance

HSML_API Configuration

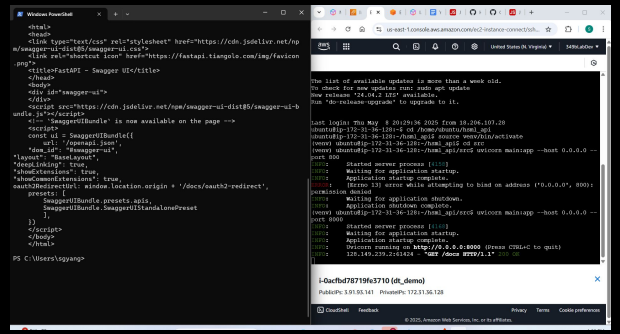
- Cloned the HSML_API GitHub repo onto the EC2 instance
- Set up and activated a Python virtual environment



- Configured FastAPI endpoints to:
  - Register new DIDs and public keys
  - Send HSML-compliant messages to Kafka topics
- Validated that the system supports secure ID registration and real-time communication across simulations

EC2 Testing with HSML_API

- Documented step by step instructions to launch and connect to EC2 instances
- Was able to successfully connect to HSML_API from an external terminal (Windows Powershell)



- Simulated user workflow:
  - Register new users with DIDs and public keys
  - Encountered and began troubleshooting credential registration errors
  - Confirmed that external systems can interact with the EC2-hosted API

https://docs.google.com/document/d/1JgUKaFqL-N2I3eo5GNvA35vcSJyidEqh7YiNICnv6Ic/edit?tab=t.0

EC2 Limitations & Challenges

JPL Firewall Restictions

- EC2 traffic may be blocked by JPL’s internal network policies
- Can prevent API/Kafka access and communication unless on a secure and approved connection

Although EC2 hosts the backend (Kafka, HSML_API, MySQL), users need to download and run the simulation software locally on the machine (Omniverse, Unity, Unreal) since it does not render or display simulations

Business Logic

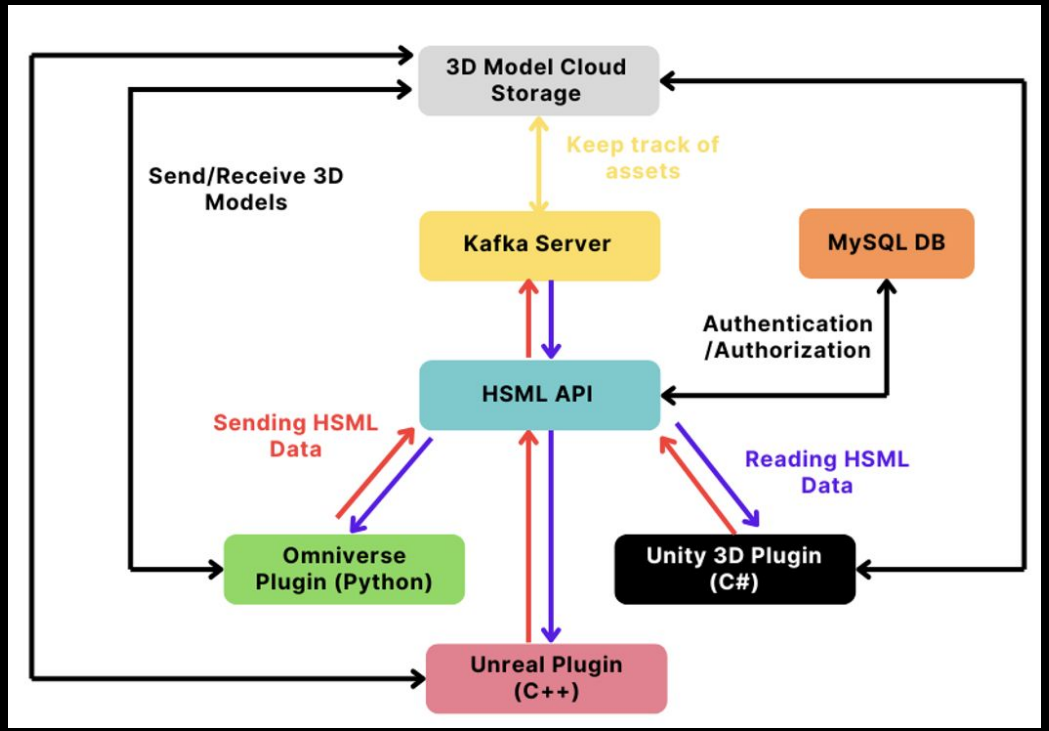


Figure credit: Alicia and Jared

- Previously, platforms directly communicated to Kafka server – meaning anyone could communicate with the server, which raised security concerns
- The HSML API behaves as a middle-man to the simulation platforms – handling registrations, authentications, and authorizations
- The simulation platforms exchange information with the API in the HSML schema language

Authentication and Authorization

A user can behave as a producer or consumer to a Kafka topic, however they must be authenticated, then authorized to access the requested Kafka topic.

A **producer** is a user that publishes data to a topic. A **consumer** is a user that consumes data of a topic.

Authentication

1. User supplies private key (+ Kafka topic user is requesting to publish/subscribe to)
2. System derives public key from private key.
3. System cross-checks with MySQL database that public key:
    - Producer: matches Agent's key for the topic
    - Consumer: is inside "allowed_did" of the Agent's key for the topic

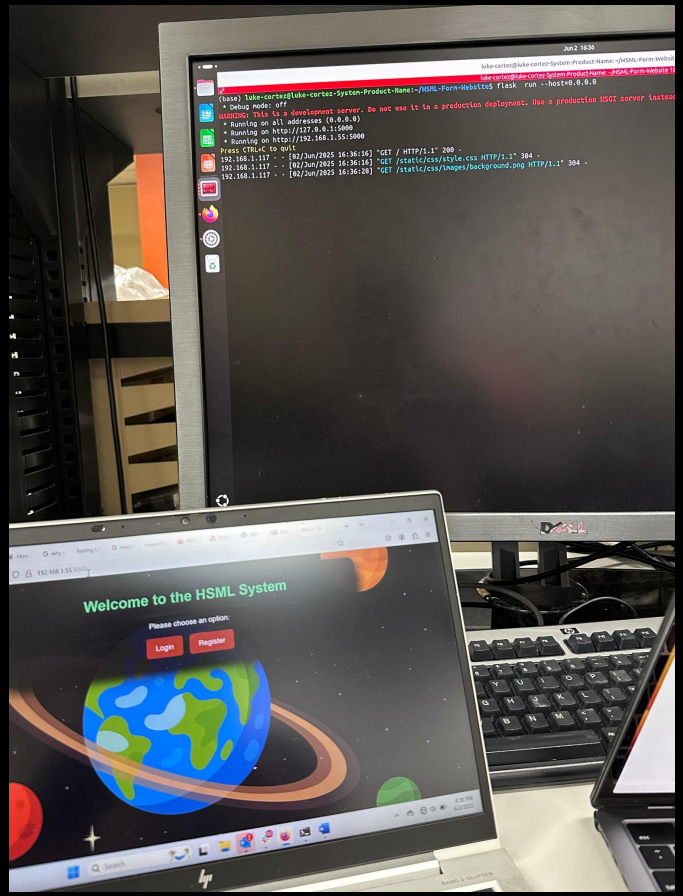
Authorization

1. If authentication is successful, user is authorized as a producer/consumer

Unit Testing - Registration Test

A test to verify if a user can manually register an entity (person or organization) by making a GET request to the /register API endpoint

- PASSED!
- Successfully registered a user through web application
- Queried newly registered user in database
- Initially failed because the server was only being hosted locally, not across the network
    - Fixed by appending –host 0.0.0.0

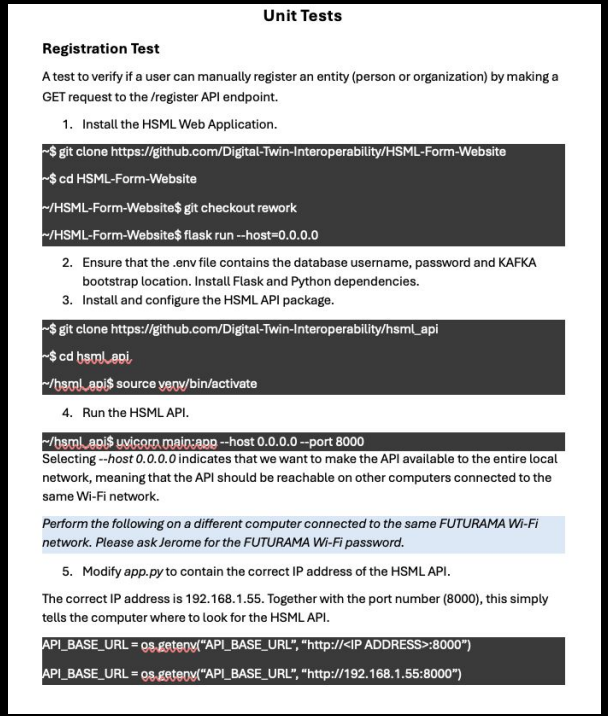


User Painpoints

Painpoint	Why Is This a Problem	Feasible Solution
Constant switching of networks between set-up and usage (FUTARAMA and Eduroam)	The Kafka server will not launch properly if network connectivity is established	In server.properties: - listeners=PLAINTEXT://127.0.0.1:8000 - advertised.listeners=PLAINTEXT://127.0.0.1:8000 Essentially forcing the Kafka server to bind to localhost
HSML API and Web Application installation is not automated	Not friendly to non-Python savvy users	Create a script: - Downloads API + Web App - Creates Python virtual environments - Downloads requirements with Pip - Runs API + Web App on specified ports (to eliminate confusion)
User is not automatically logged in using the newly-created key	User experience	Easy fix for a small problem – automatically authenticate using created private key
Name field is not split into first/last name - lack of input validation	Difficult to sort/segment users, unable to perform form validations, prone to injection attacks	Split name field into first and last, limit text inputs to alphabetical characters
Lack of error message when registration fails	Causes debugging difficulties	Forgot HSML API wasn't running, registration was failing but no reason why – forward error to web page

Completed Tasks

- Updated Python scripts to generate JSON files that align with revised data structure
- Wrote the Unity KAFKA producer and consumer
- Performed unit tests to confirm functional user registration and API authentication/authorization
- Cleaned up unit testing documentation and added context to instructions



Unit Tests

Registration Test

A test to verify if a user can manually register an entity (person or organization) by making a GET request to the /register API endpoint.

1. Install the HSML Web Application.

```
~$ git clone https://github.com/Digital-Twin-Interoperability/HSML-Form-Website
~$ cd HSML-Form-Website
~/HSML-Form-Website$ git checkout rework
~/HSML-Form-Website$ flask run --host=0.0.0.0
```

2. Ensure that the .env file contains the database username, password and KAFKA bootstrap location. Install Flask and Python dependencies.
3. Install and configure the HSML API package.

```
~$ git clone https://github.com/Digital-Twin-Interoperability/hsml_api
~$ cd hsml_api
~/hsml_api$ source venv/bin/activate
```

4. Run the HSML API.

```
~/hsml_api$ uvicorn main:app --host 0.0.0.0 --port 8000
```

Selecting *--host 0.0.0.0* indicates that we want to make the API available to the entire local network, meaning that the API should be reachable on other computers connected to the same Wi-Fi network.

Perform the following on a different computer connected to the same FUTURAMA Wi-Fi network. Please ask Jerome for the FUTURAMA Wi-Fi password.

5. Modify *app.py* to contain the correct IP address of the HSML API.

The correct IP address is 192.168.1.55. Together with the port number (8000), this simply tells the computer where to look for the HSML API.

```
API_BASE_URL = os.getenv("API_BASE_URL", "http://<IP ADDRESS>:8000")
API_BASE_URL = os.getenv("API_BASE_URL", "http://192.168.1.55:8000")
```

Conclusion/Current Outcomes of the Project

- Proven communication between digital twins across platforms
- Successful demo implementations of the HSML schema
- Registration and authentication processes completed in milliseconds
- Business logic designed to maintain the data integrity of users
- Performed user testing to identify room for improvement in user experience
- Created a user-friendly web interface for users to interact with HSML system
- Successfully set up and integrated the EC2 cloud service to support real-time communication

Looking forward/Next Steps

- ★ Using EC2 container to make it the program more accessible
  - Benefits of EC2: direct/simple Kafka (server) integration, easy to scale, full control, allows remote access from anywhere (with stable internet)

- ★ Ensure program is user friendly (ex. Our AI assistant)

- ★ Expanding schemas and web application to support for thermal, physics, & circuit design

Lessons Learned

- Understanding of digital twin architecture & interoperability standards and protocols
  - Gain understanding of and experience with HSML messages used to foster communication
  - Make decisions while keeping functionality, latency, scalability, & security in mind
- Importance of Effective Communication
  - Recording & presenting findings routinely to ensure effective collaboration
  - Explaining frameworks to technical and non-technical audiences
- Personal Growth
  - Tackling technical issues methodically (staying patient and persistent)
  - Adapting quickly to rapidly evolving forms of technology

Acknowledgements

- ★ Our Mentors: Dr. Thomas Lu & Dr. Edward Chow
- ★ Jenny Tieu (SIRI Coordinator), Devyn Payne (SIRI Program Support), Jen Pino, Phillippa Kennedy, Arpine Margaryan
- ★ Group 345C Interns: Alicia Sanjurjo, Gerald Parish, Jared Carrillo, Niki Namian, Joshua Drye, Sohee Kim, Diego Cordova, Aaron Hui, Gabriel Venezia, Subhobrata Chakraborty, and Sydni Yang
- ★ Niki Namian: Jinan Darwiche and Howard Stahl (Faculty Supervisors) at Santa Monica College + My supportive family and friends

Thank you!

HSML Conversion & Autonomous Waypoint Navigation

Presented by:
Diego Cordova

Internship Program:
Jet Propulsion Laboratory (JPL) — SIRI Program

Mentors:
Dr. Thomas Lu
Dr. Edward (Ed) Chow

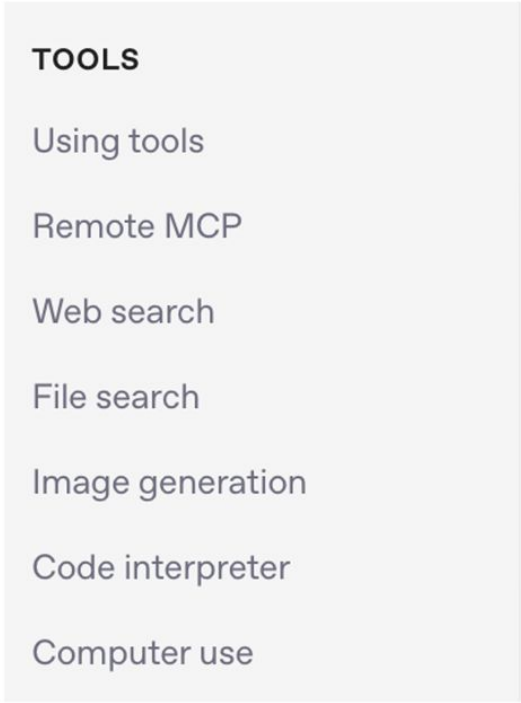
HSML Conversion

- **Purpose:** Convert arbitrary JSON inputs into structured HSML JSON objects using the OpenAI API.
- **Main steps:**
  - Initialize the OpenAI client with credentials.
  - Create and index a vector store for the HSML schema document.
  - Update the assistant to leverage the schema via the file_search tool.
  - Send JSON payloads to the assistant and receive HSML-formatted output.

Install Dependencies

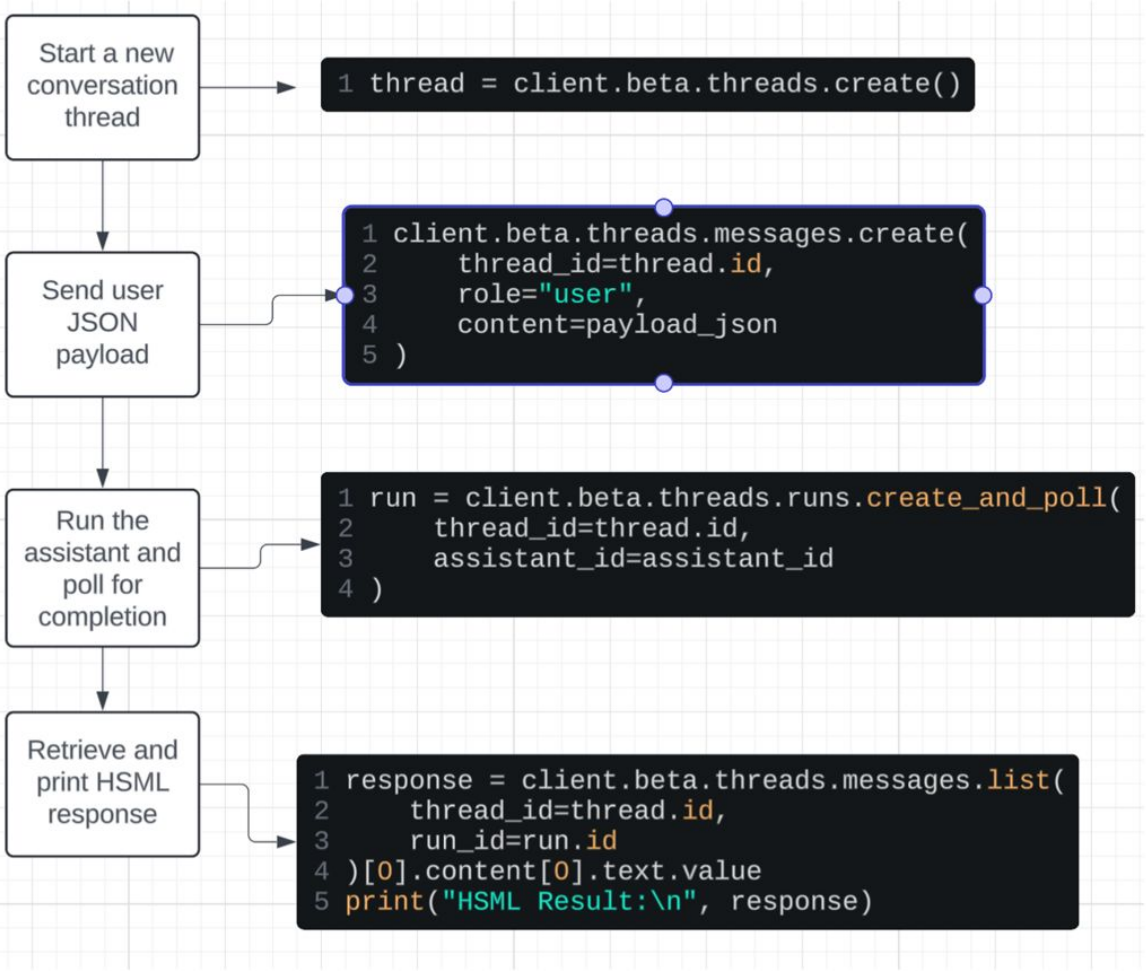
```
pip install openai python-dotenv
```

- **openai library**
  - Provides a Python wrapper around the OpenAI REST API.
  - Simplifies authentication, request formatting, and response handling when interacting with models, vector stores, and other OpenAI services.
- **python-dotenv package**
  - Loads environment variables from a local .env file into os.environ at runtime.
  - Keeps secrets (API keys, assistant IDs) out of your source code, improving security and configurability.

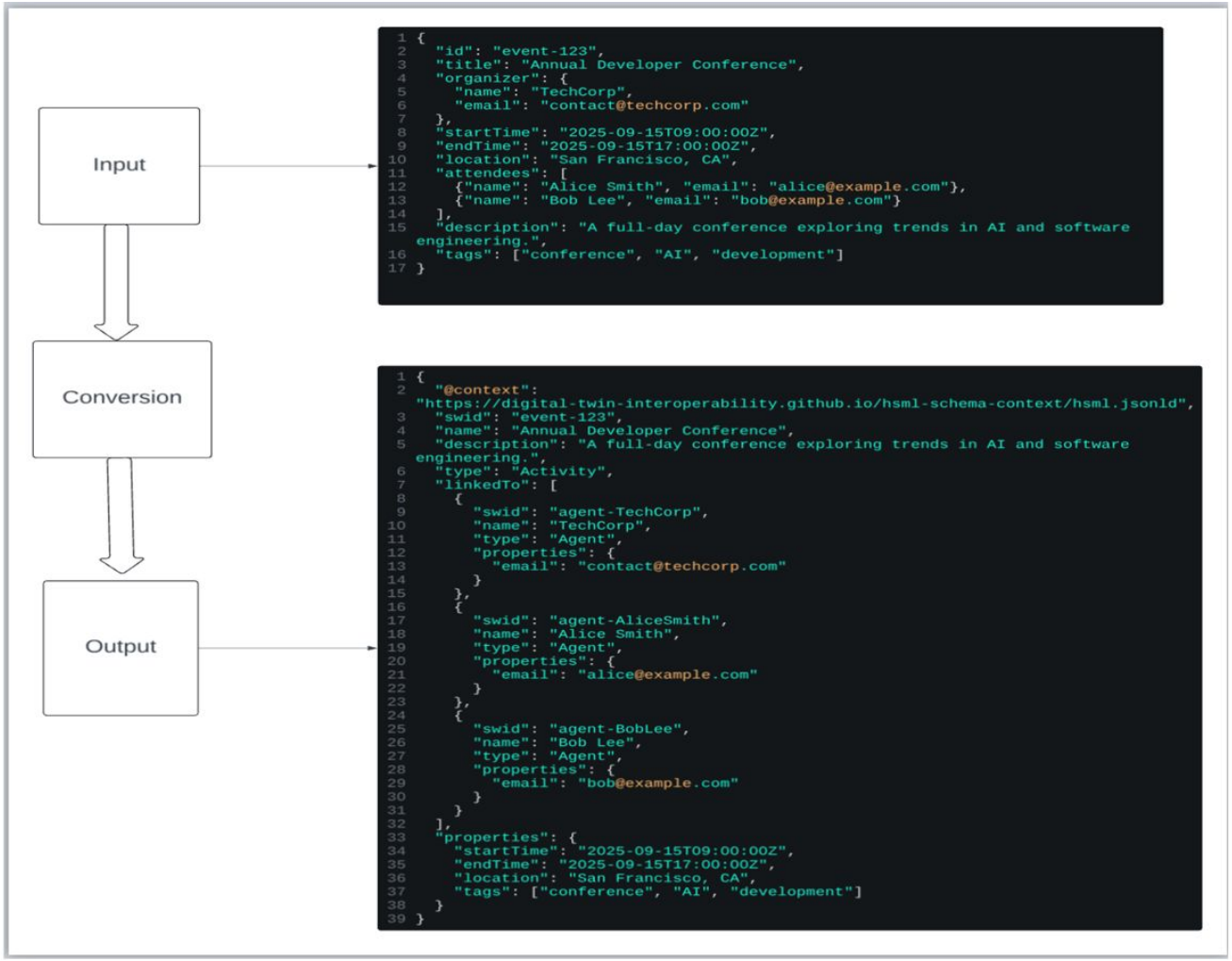


Conversion Pipeline

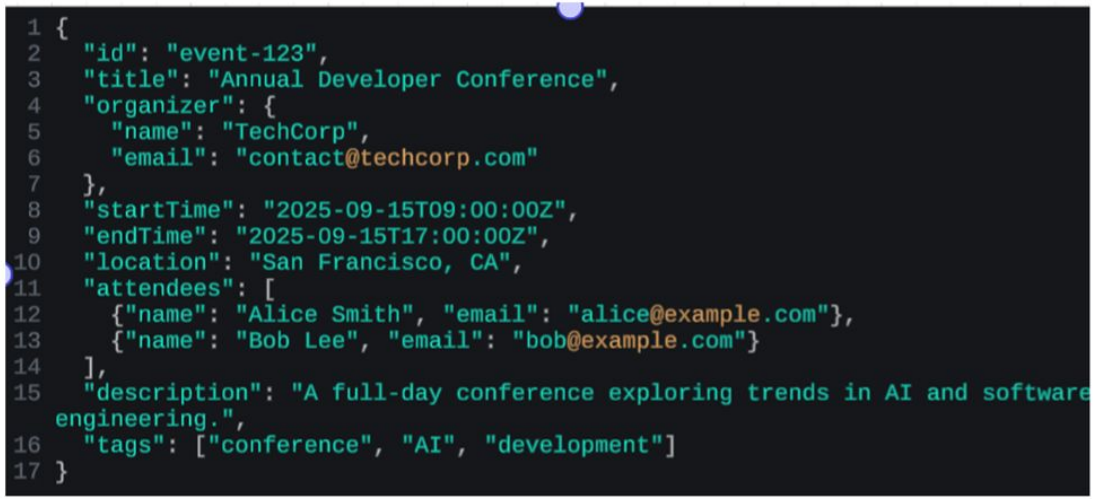
- **Threads:** Maintain context for multi-step conversions or follow-up queries.
- **Polling:** Ensures synchronous behavior in a script environment.



Example Conversion



INPUT

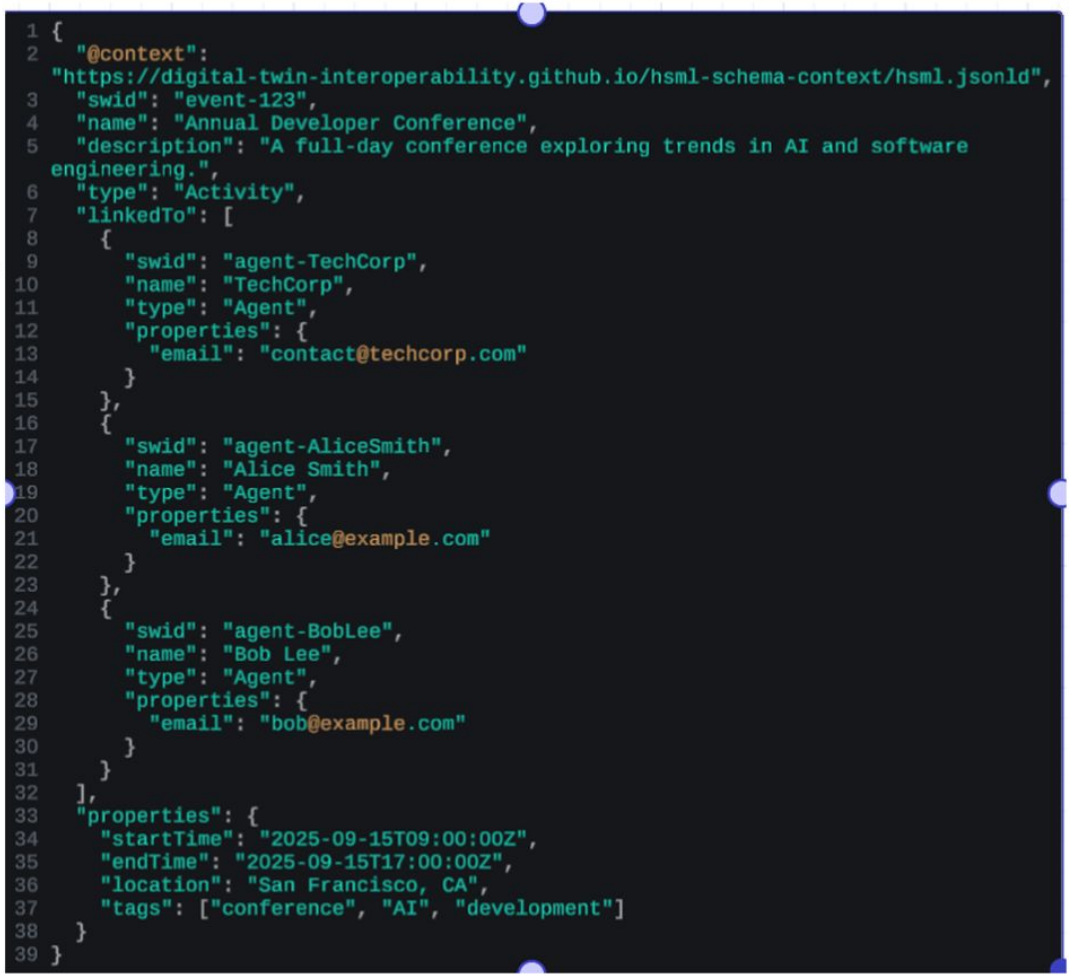


```
{
  "id": "event-123",
  "title": "Annual Developer Conference",
  "organizer": {
    "name": "TechCorp",
    "email": "contact@techcorp.com"
  },
  "startTime": "2025-09-15T09:00:00Z",
  "endTime": "2025-09-15T17:00:00Z",
  "location": "San Francisco, CA",
  "attendees": [
    {"name": "Alice Smith", "email": "alice@example.com"},
    {"name": "Bob Lee", "email": "bob@example.com"}
  ],
  "description": "A full-day conference exploring trends in AI and software engineering.",
  "tags": ["conference", "AI", "development"]
}
```

OUTPUT

Key Mappings & Highlights

- **id → swid:** Ensures unique identifier within HSML.
- **title → name:** Human-readable label.
- **organizer & attendees → linkedTo:** Automatically creates Agent entries for each person/organization.
- **Remaining fields → properties:** Cleanly encapsulates additional data.



```
{
  "@context": "https://digital-twin-interoperability.github.io/hsml-schema-context/hsml.jsonld",
  "swid": "event-123",
  "name": "Annual Developer Conference",
  "description": "A full-day conference exploring trends in AI and software engineering.",
  "type": "Activity",
  "linkedTo": [
    {
      "swid": "agent-TechCorp",
      "name": "TechCorp",
      "type": "Agent",
      "properties": {
        "email": "contact@techcorp.com"
      }
    },
    {
      "swid": "agent-AliceSmith",
      "name": "Alice Smith",
      "type": "Agent",
      "properties": {
        "email": "alice@example.com"
      }
    },
    {
      "swid": "agent-BobLee",
      "name": "Bob Lee",
      "type": "Agent",
      "properties": {
        "email": "bob@example.com"
      }
    }
  ],
  "properties": {
    "startTime": "2025-09-15T09:00:00Z",
    "endTime": "2025-09-15T17:00:00Z",
    "location": "San Francisco, CA",
    "tags": ["conference", "AI", "development"]
  }
}
```

Integration

- **Agents A & B Interaction:** When **Agent A** needs to send data to **Agent B**, raw JSON is first transformed by our HSML conversion script.
- **HSML as Shared Language:** The converted HSML message ensures both agents speak the same schema, regardless of their native formats.

Autonomous Waypoint Navigation System in Unity

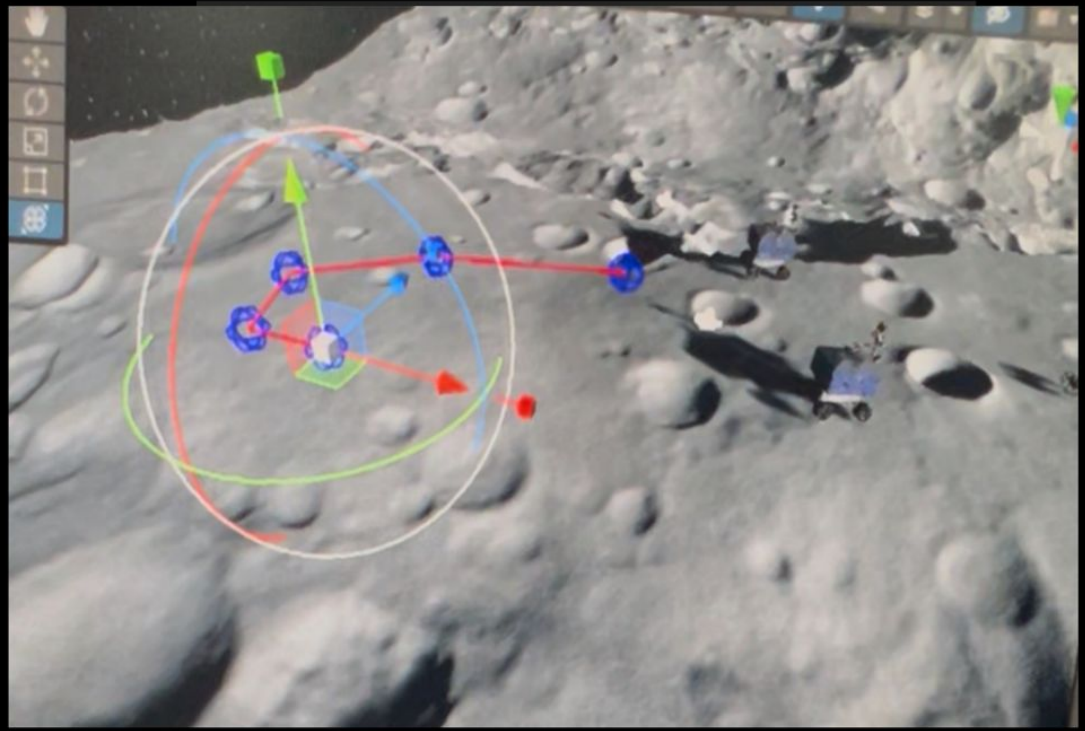
- **Created a modular system enabling rovers to follow user-defined waypoint paths**
- **Built with two main C# scripts: Waypoints.cs (path definition & visualization) and WaypointMover.cs (motion & rotation logic)**
- **Designed for reusable, inspector-configurable setup for any Unity object**
- **Supports smooth, realistic movement and rotation**

Challenges & Solutions

- **Model Import Issues:** Incorrect axis orientation → wrapped rovers inside parent GameObject for transform control
- **Motion Realism:**
  Enhanced rotational smoothness by using Quaternion.Slerp for interpolation, resulting in natural and fluid turns compared to standard linear interpolation methods.
- **Waypoint Visualization:** Used OnDrawGizmos for visual debugging in Editor
- **Inspector Tuning:** Exposed parameters for speed, rotation, and waypoint proximity for easy adjustment

How the Rover Moves & Turns

- Linear movement with Vector3.MoveTowards at constant speed
- Smooth heading adjustments using Quaternion.Slerp towards target waypoint
- Waypoint reached when within user-defined distance threshold, then advances to next waypoint
- Motion parameters adjustable in real-time through Inspector



Testing Strategy and Outcomes

- Tested within crater terrain Unity simulation environment
- System confirmed stable and reliable across all test runs

Waypoint Mover Script Flowchart

Why use [SerializeField] for these fields?

Field	Type & Default	Purpose / Description	Benefit for User
waypoints	Waypoints (Custom class)	Reference to the Waypoints object that holds the list of path points (waypoints) the rover will follow.	Allows the user to assign any waypoint path in the Unity Editor without modifying code.
moveSpeed	float (default: 2f)	Controls how fast the rover moves toward each waypoint.	Users can easily adjust rover speed in Unity's Inspector to fine-tune motion without touching the script.
distance	float (default: 1f)	Minimum distance to consider the current waypoint "reached" and advance to the next waypoint.	Lets users set how close the rover must get before moving on, controlling path precision from the Inspector.
rotateSpeed	float (default: 1f)	Controls how fast the rover rotates to face the next waypoint.	Users can adjust rotation smoothness and responsiveness easily via Inspector parameters.

Why This System Matters

- Modular and reusable — any Unity object can follow waypoints with minimal setup

https://drive.google.com/drive/folders/1GabpHfk9YEdtrqwkoDMZ7gcqlBROW8FM

Future Steps

HSML Conversion Tool:

- Implement real-time data ingestion by integrating this tool with the current HSML web app. This integration will allow users to input JSON files, natural language, or any data format, which the tool will then convert into valid HSML schema format and send through the web app for seamless interoperability.

- **Current Testing Observations:** The tool works well with small input data, maintaining HSML structure and schema compliance. However, with large input data, the `@context` field is not constructed correctly, despite the rest of the structure following HSML guidelines.

- **Next Step:** Explore fine-tuning the model using OpenAI API tools, as prompting alone has not been sufficient to handle larger data inputs accurately. Fine-tuning may improve performance and ensure correct context generation.

Lessons Learned

- Modular code with Inspector settings makes testing and changes fast and easy.
- Keeping coordinate systems and object setups consistent is key for the simulation to work right.
- Using visualization tools like Gizmos helps a lot when finding and fixing bugs.
- Being close with mentors and asking lots of questions helps clarify what’s expected and ensures the project meets high standards.
- Real projects need a balance between technical accuracy and what’s practical to implement.

Acknowledgements

- A huge thank you to **Dr. Edward (Ed) Chow** and **Dr. Thomas Lu** for being so involved and supportive throughout my internship.
- Their guidance made this exciting experience possible, and I learned a lot in a new environment that I really enjoyed.
- Special thanks to the **program coordinators** for providing valuable resources and support during the SIRI meetings—it helped me stay on track and succeed.