Hello There! My Name is

Your Name
Hammad Javed

Title
Data Analytics & AI Engineer

WELCOME TO MY

PORTFOLIO 2022

ABOUT

Know More About Me

Your Name
Hammad Javed

Title
Data Analytics & AI Engineer

Website
https://portfolio-hammad.vercel.app/index.html

Email
hammadjaved265@gmail.com

Phone Number
(323) 455-9214

Skills

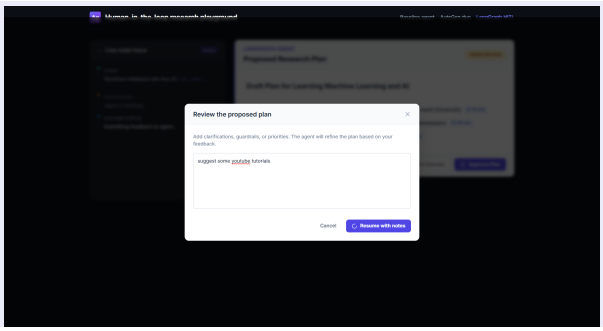
Skill
Agentic AI
Data Engineering
Software Development
Machine Learning

Work Experience

Company	Position/Title	Year Ended
Stylo Private Limited	Associate Machine Learning Engineer	Present
S.A Hamid & Co.	Sr. Software Developer	2025

PORTFOLIO

Check Out Some of My Works.

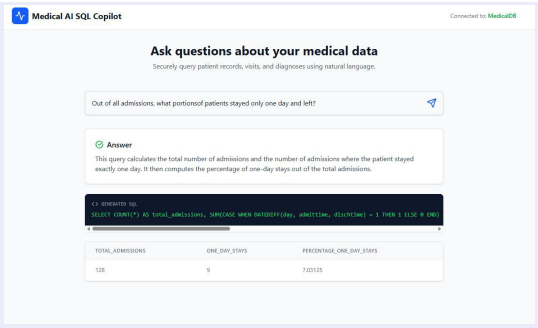


Project Name 1

Human-in-the-loop Slides Maker Agent

Project Type 1

Agentic AI

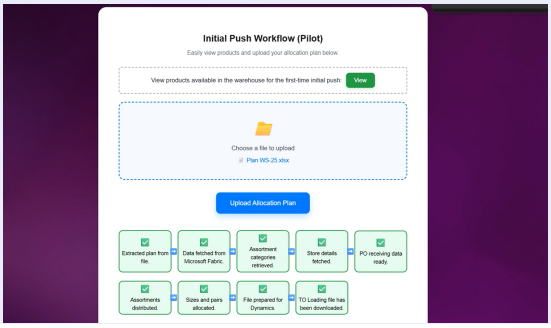


Project Name 2

Data Agent

Project Type 2

Agentic AI

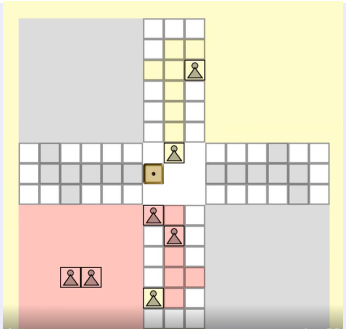


Project Name 3

Supply Chain Automation

Project Type 3

Machine Learning & Agentic AI



Project Name 4

Self-play Ludo

Project Type 4

Reinforcement Learning

Project 1: Human-in-the-loop research playground:

The Human-in-the-Loop Research Playground is an AI-powered system designed to automatically generate structured research or learning plans along with accompanying presentation slides, while keeping humans actively involved in the loop. The goal of this project is to combine the reasoning capabilities of AI with human intuition and feedback, ensuring that the final output whether it's a detailed research roadmap or a polished presentation truly aligns with the user's goals, focus areas, and preferences.

For the frontend, I built the user interface using Next.js with React and TypeScript. This provided a smooth and interactive single-page experience where users can easily enter their research topics, track progress, and view AI-generated outputs in real time. The interface includes a dynamic progress tracker showing each stage of the AI pipeline from plan generation to resource enrichment and slide creation. I styled the UI with Tailwind CSS, keeping it modern, responsive, and minimal. A built-in feedback editor allows users to view and modify the AI's draft before finalization, playing the central role in the human-in-the-loop workflow.

The backend was developed in Python, using FastAPI as the main framework for API endpoints and communication with the frontend. This backend orchestrates the full pipeline it receives the user's input from the frontend, triggers AI tasks using LangGraph, and returns partial or final results back to the interface. LangGraph was the core of the AI logic, as it allowed me to design a graph-based agent workflow where each node represents a modular AI step. For example, one node generates the initial plan structure, another enriches it with references and learning resources, a third converts it into slides, and a checkpoint node pauses execution to wait for human feedback.

At the heart of this system are the OpenAI APIs, primarily using GPT-4-turbo for reasoning and content generation. The PlanGenerator node calls GPT-4-turbo to draft the initial research or learning roadmap, structured with objectives, milestones, and timelines. The ResourceEnricher node uses GPT's function-calling feature to search and attach relevant materials such as papers, articles, tutorials, or datasets to each section of the plan. The SlideCreator node transforms this information into presentation-friendly content, generating slide headings, bullet points, summaries, and even optional visual cues or image prompts. For visuals, I integrated DALL·E for AI-generated illustrations and Plotly for concept diagrams where applicable.

A defining feature of this system is the Human-in-the-Loop checkpoint, which is implemented as a distinct node inside the LangGraph workflow. When the AI completes a draft plan, the graph execution pauses, and the plan is sent to the frontend for user approval. The user can review the generated sections, add focus areas, remove irrelevant parts, or modify the depth of specific topics. These edits are then submitted back to the backend, and the graph resumes execution from that checkpoint. This mechanism ensures that the final AI output isn't purely automated instead, it reflects an intentional collaboration between the model and the user's feedback.

Once the plan is finalized, the backend triggers the final step: presentation generation. Here, the system uses GPT-4 again to convert the approved plan into a slide deck outline. I used python-pptx and Marp to generate slides automatically from structured markdown text. The slides include research goals, learning paths, and visual representations of progress or concepts. The finalized plan and slides are stored in a database either MongoDB or PostgreSQL depending on deployment configuration. The system also supports exporting outputs as PDF, Markdown, or PowerPoint (PPTX) files. For file storage, I integrated AWS S3 or Cloudinary, which handle plan exports and slide assets securely.

The backend communicates with the frontend over REST APIs and WebSocket connections, allowing real-time plan updates and progress streaming. NextAuth and JWT tokens are used for authentication, ensuring that users can securely manage their plans, view version history,

and revisit previous iterations. For deployment, the frontend is hosted on Vercel, and the backend runs on Render or AWS EC2, enabling scalability and smooth integration between the two layers.
In terms of architecture, the workflow can be summarized as:
User Input → Plan Generation → Resource Enrichment → Human Feedback → Slide Creation → Finalization & Export.
Each phase is modular, making it easy to extend the platform to other applications such as course design, project planning, or proposal writing.
Overall, this project demonstrates a practical and human-centered use of AI agents. By integrating LangGraph for multi-step reasoning, OpenAI APIs for intelligent content generation, and a human approval checkpoint for iterative refinement, the Human-in-the-Loop Research Playground bridges the gap between automation and human creativity. It not only accelerates the process of generating structured research materials but also keeps users meaningfully in control of their learning or project outcomes.

Project 2: Build Data Agent Chatbot Data Agent Chatbot:
The Medical AI SQL Copilot is an intelligent analytics assistant designed to translate natural language questions into safe, executable SQL queries for hospital data. It bridges the gap between non-technical users (like hospital administrators or doctors) and complex database systems.
Architecture & Stack
The project is built as a modern full-stack application with a containerized database.
1. Database (Infrastructure)
System: Microsoft SQL Server 2022
Deployment: Docker Container (orchestrated via docker-compose)
Role: Stores patient, admission, and medical data.
2. Backend (API Layer)
Language: Python
Framework: FastAPI (High-performance web framework for building APIs)
Server: Uvicorn (Lightning-fast ASGI server)
AI Integration: OpenAI API (gpt-4o) used to interpret natural language and generate SQL.
Database Driver: pyodbc (Connects Python to SQL Server)
Data Handling: Pandas (Data manipulation) & Pydantic (Data validation & settings management)
Utilities: python-dotenv (Configuration management)
3. Frontend (User Interface)
Framework: React 19 (Latest version of the library)
Language: TypeScript (Type-safe JavaScript)
Build Tool: Vite (Next-generation frontend tooling, extremely fast)
Styling: Tailwind CSS 4 (Utility-first CSS framework for rapid UI development)
HTTP Client: Axios (For making requests to the backend)
Icons: Lucide React (Beautiful, consistent icons)
Key Features
Natural Language Processing: Converts questions like "How many patients were admitted last week?" into SQL.
Context-Aware: The system is aware of the database schema (tables and columns).
Few-Shot Learning: We recently integrated "Golden Key" examples (from QUERIES.csv) to teach the AI how to write accurate queries for this specific dataset.
Safety First: Designed with read-only permissions and safety prompts to prevent data modification or unauthorized access.

Project 3: Improving Supply Chain Operations in Fashion Retail Using Advanced Analytics: Exploring Agentic AI:
Project Foundation and Data Collection This phase involves identifying and collecting relevant datasets at Stylo such as sales, inventory, website traffic, and product attributes. It also includes process research and stakeholder interviews to understand current supply chain operations and pain points. Feature engineering will be performed to prepare the data for analysis. Testing, Operationalization, and Impact Measurement The solution will be tested using Stylo historical data and compared against existing baselines. Collaboration with Stylo IT department will support integration or automation of the system. A live pilot test will be conducted to assess operational performance, and impact will be measured through defined KPIs, capturing improvements in efficiency, decision quality, and profitability. This project's uniqueness lies in its strong industry collaboration, access to internal data, and focus on applied, measurable outcomes. My existing domain expertise as a Data Analytics Engineer at Stylo further enhances the practical value and execution potential of the work.

Project 4: Self-Play Ludo:
1. Technology Stack
Language: Python 3.12
Core Libraries:numpy: Used for all vector math (dot products for Q-value calculation).
pickle: Used for saving/loading the learned weights (the "brain" of your agent).
random: Used for dice simulation and exploration (epsilon-greedy).
matplotlib (implied): Used in create_training_plots.py for visualization.
Environment: Custom-built Ludo class (ludo.py) that simulates the complete game rules (movement, cutting, safe zones, home runs).

2. The Core Algorithm
Method: Linear Q-Learning (Gradient Descent).
Why Not Deep RL? Deep Neural Networks were too slow and unstable for this specific logic-heavy game.
Why Not Tabular? The game has ~11 Trillion states, making a table impossible.
State Representation: You condensed the 11 trillion states into 30 Linear Features:Base (15): Kill, Safety, Target Progress, Danger, Escape, Blocking, etc.
Strategic (15): Polynomials (Progress squared), Interactions (Kill × Safety), and Game State context (Aggression when leading).

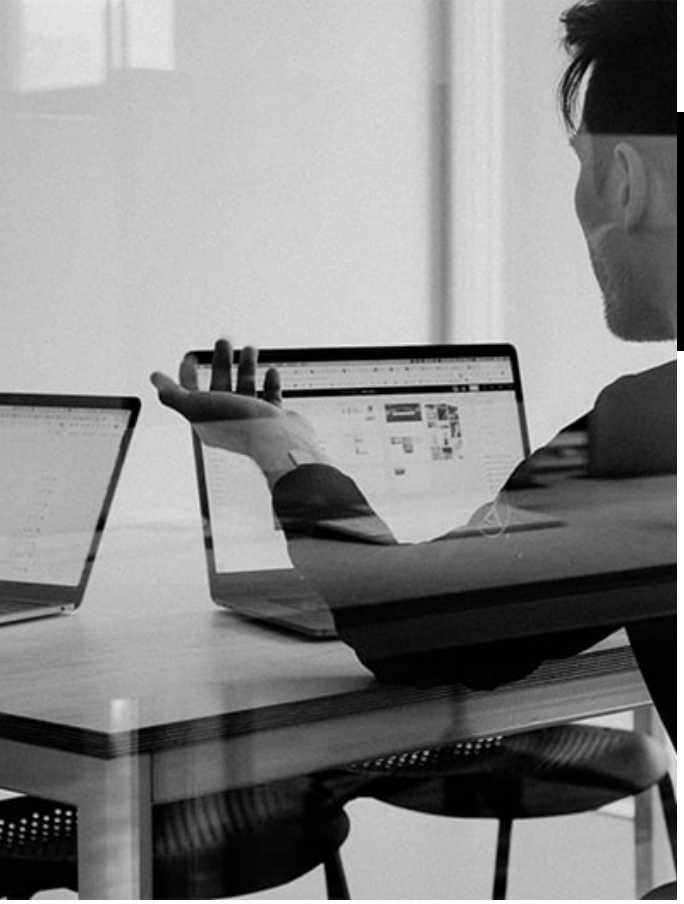
3. Policies Implemented
You created a "League of Opponents" to train against:
Policy_Random: Moves completely randomly. Used as the first "training dummy."
policy_strangers: Your custom Heuristic policy. It uses hard-coded rules (scores) to play strategically.
Policy_Milestone2 (.pyd): The "Final Boss." A compiled, highly optimized heuristic bot provided as the benchmark.
PolicyPureRLLinear (The Agent): The AI you built. It starts knowing nothing and learns by adjusting its 30 weights.

4. Training Strategy: Curriculum Learning
Instead of playing the hard boss immediately, you used a 3-Stage Curriculum:
Stage 1 (Bootcamp): Train vs. Random.Goal: Learn basic rules (don't sit at start, go home).
Result: 90%+ Win Rate.
Stage 2 (Junior League): Train vs. Stranger (Your Heuristic).Goal: Learn defense and basic

strategy.
Result: ~75% Win Rate.
Stage 3 (Pro League): Train vs. Milestone2 (The Boss).Goal: Exploit specific weaknesses in the pro bot.
Result: ~54% Win Rate (Consistently beating the "Master" bot more than half the time).

5. Key Results
Efficiency: Your agent makes decisions in microseconds because it only calculates a dot product of 30 numbers.
Performance:vs Random: ~100% Win Rate (Massacre).
vs Milestone 2: ~54% Win Rate (Verified over 10,000 games).
Stability: The agent uses "Reward Shaping" (extra points for safety, killing) to learn 100x faster than standard win/loss signals.

6. Files Overview
ludo.py: The Game Engine.
policy_pure_rl_linear.py: The AI Code (features & extensive logic).
train_curriculum_rl.py: The Training Loop (Manages the 3 stages).
test_pure_rl_10k.py: The Verification Script (Proves the 54% win rate).
weights_pure_rl.pkl: The saved "Brain" (List of 30 floating-point numbers).



SERVICES

What Can I Do For You

Services Offered

Services Offered
Machine Learning
Agentic AI
Data Engineering
Software Development

CONTACT

Love To Hear From You

Email
hammadjaved265@gmail.com

Phone Number
(323) 455-9214

Website
https://portfolio-hammad.vercel.app/index.html