Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

PROJECT AND TEAM INFORMATION

Project Title

(Try to choose a catchy title. Max 20 words).

NeuroShell: Blending Unix Power with Machine Learning Intelligence

Student/Team Information

Team Name:	Kernel Mind
Team member 1 (Team Lead) (Name, Student ID, Email, Picture):	 *Anshika Saklani-2318420* *Student ID : 23012076* *Email :* *anshikasaklani894@gmail.com*

Team member 2 (Name, Student ID, Email, Picture):	 Rakhi Bist-2319380 Student Id : 23012177 Email : rakhibisht24122004@gmail.com
Team member 3 (Name, Student ID, Email, Picture):	 Akriti Rawat – 2318263 Student Id: 230221541 Email : akritirawat12345@gmail.com
Team member 4 (Name, Student ID, Email, Picture):	 Ayush Chand – 2318582 Student Id: 230112536

	Email : ayushchand862@gmail.com

PROJECT PROGRESS DESCRIPTION

Project Abstract

(Brief restatement of your project's main goal. Max 300 words).

Traditional command-line shells offer limited, rule-driven assistance, leading to manual overhead and reduced efficiency when dealing with complex or repetitive tasks. This project addresses the gap by developing an AI-driven, Unix-like shell that seamlessly merges a robust C-based POSIX core with an integrated, context-aware machine learning (ML) engine written in Python. The ML engine provides intelligent support via typo correction, next-command prediction, and flag/template recommendations that dynamically adapt to the user's Git repository and working environment. Crucially, the system supports self-learning from local history for continuous improvement and guarantees offline operation. By providing explainable suggestions alongside its predictions, this novel shell enhances user trust, usability, and overall developer productivity in the command-line interface.

Updated Project Approach and Architecture

(Describe your current approach, including system design, communication protocols, libraries used, etc. Max 300 words).

Our project utilizes a robust, modular **two-component architecture** to integrate low-level systems functionality with intelligent guidance, ensuring stability, portability, and continuous, context-aware user assistance.

Component 1: Shell Core (C, POSIX)

The **Shell Core (C, POSIX)** forms the stable foundation, handling all direct OS interaction and the core execution loop. It implements the **REPL**, command parsing, and core **process management** using standard Unix calls like **fork()** and **execvp()**. Comprehensive **job control** features are included, alongside robust signal handling (SIGINT, SIGTSTP).

For data capture, the Core integrates a **SQLite history store** to persistently log enriched context—including command text, working directory, and exit status—providing high-quality training data. Communication with the intelligence component is established using high-performance **Unix domain sockets** that exchange low-latency, newline-delimited JSON messages.

Component 2: Suggestion Engine (Python)

The **Suggestion Engine (Python)** operates entirely offline, guaranteeing privacy and reliability while delivering adaptive and **explainable recommendations**. It combines outputs from three specialized suggestors:

1. **Typo Fixer:** Uses the **Damerau-Levenshtein distance** metric to swiftly correct mistyped commands against known history.
2. **Next Command Predictor:** Employs **N-gram and Markov models** trained on sequential history data to anticipate the user's next action in a workflow.
3. **Flag/Template Recommender:** Leverages **TF-IDF** combined with **cosine similarity** to recommend highly relevant arguments and flags based on the user's environment and Git context.

A lightweight **Ranker** merges these suggestions, scores them, and crucially, generates a **rationale** for each prediction, ensuring users understand *why* a recommendation was made. This architecture supports a **self-learning mechanism** where models periodically retrain on the local SQLite data, ensuring dynamic adaptation to evolving user patterns.

Tasks Completed

(Describe the main tasks that have been assigned and already completed. Max 250 words).

Task Completed	Team Member
Core System Developer **ML Suggestion & Model Engineer**	**Anshika Saklani** **Akriti Rawat**
Database & Preprocesssing	**Ayush Chand**
IPC & Communication Engineer	**Rakhi**

Challenges/Roadblocks

(Describe the challenges that you have faced or are facing so far and how you plan to solve them. Max 300 words).

GCC not recognized in non-Unix environments (Windows compatibility issue).
- Debugging socket timeout responses.
- Managing concurrent background processes efficiently

Area Of Improvement

(Describe the main tasks that you still need to complete. Max 250 words).

Area Of Improvement	
Explainable AI: Shows why each suggestion was made Privacy-First: Operates entirely offline with no cloud dependency	

Project Outcome/Deliverables

(Describe what are the key outcomes / deliverables of the project. Max 200 words).

The project delivers a fully integrated, two-component intelligent shell that unites POSIX stability with adaptive machine learning. The outcome includes a stable C/POSIX Core with robust job control and persistent, enriched history logging via SQLite. This powers a real-time, offline Python Suggestion Engine that provides adaptive command assistance. Key features include Damerau-Levenshtein typo correction, N-gram/Markov next command prediction, and TF-IDF flag recommendations. We validate low-latency IPC using Unix domain sockets and demonstrate a self-learning mechanism that continuously improves models on local history. Crucially, every suggestion is delivered with an explainable rationale, resulting in the first context-aware shell designed to minimize user cognitive load and maximize command-line productivity.

Progress Overview

(Summarize how much of the project is done, what's behind schedule, what's ahead of schedule. Max 200 words.)

The project is in late-stage development, with the two-component architecture fully implemented and transitioning to final testing. The **C/POSIX Shell Core** is 100% complete, featuring robust job control, command parsing, and stable process management. Crucially, the **SQLite history module** is fully integrated, providing rich, contextual data for the ML pipeline. The **Python Suggestion Engine** is fully operational, incorporating the Damerau-Levenshtein Typo Fixer, N-gram/Markov Predictor, and TF-IDF Flag Recommender. The core innovation—**low-latency IPC via Unix domain sockets**—is validated for real-time performance. Furthermore, the essential **self-learning mechanism** for offline retraining and the **explainable rationales** feature are fully functional, achieving all primary project objectives.

Codebase Information

(Repository link, branch, and information about important commits.)

Repo :- “https://github.com/Anshika-111105/Intelligent-Unix-Shell”