SANGANITRA

Projects 2025-26

ACM Sanganitra and Team

Velocity Clash

Mentors

Mohnish Hemanth Kumar - 8431186036
Saksham Parmar - 6267733526
Manish Agarwal - 9671565344

Introduction

Abstract :

- Velocity Clash is a real-time, peer-to-peer Air Hockey game for Android that enables smooth offline multiplayer over Wi-Fi LAN or Wi-Fi Direct without relying on servers. Built with Flutter for UI and a native C++ engine for physics, collisions, and networking, integrated via Dart FFI to ensure low-latency updates. The project aims to demonstrate an efficient hybrid architecture for real-time games, combining Flutter's flexibility with native performance.

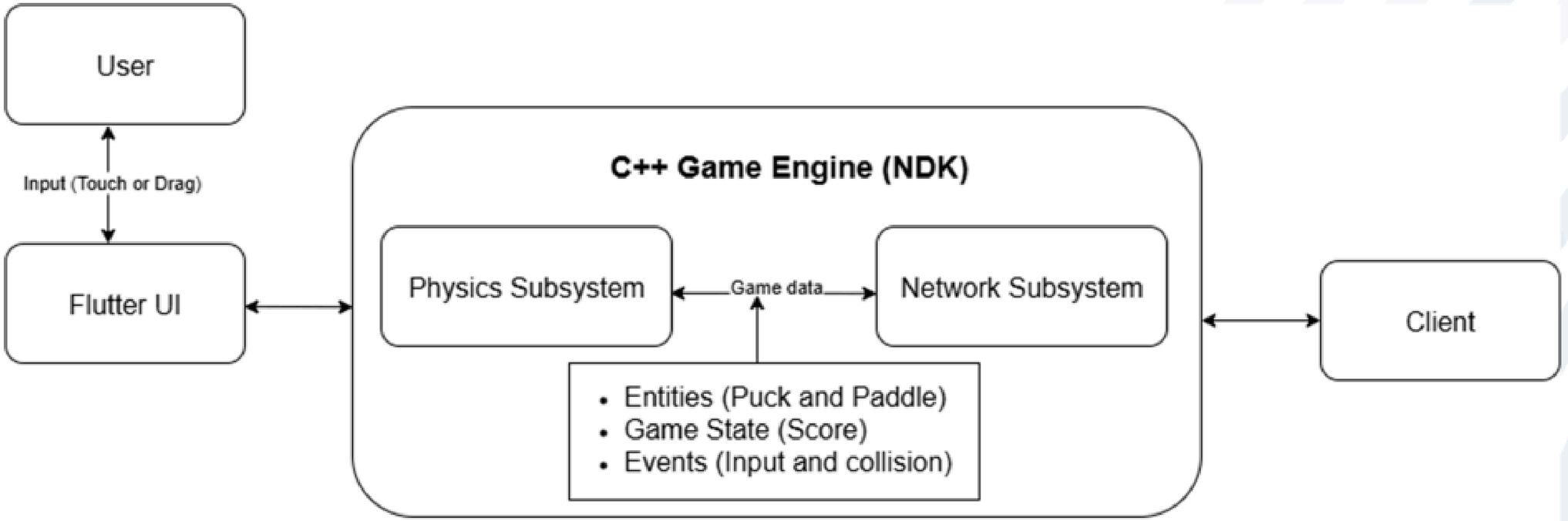
Technologies used :

- Flutter - UI
- Dart FFI
- C++ - Game Engine and Networking
- Android NDK

How is low latency achieved?

- **Efficient hybrid architecture (Flutter + native C++ engine).**
- **UDP-based lightweight communication.**
- **Optimized packet processing with compact binary formats.**
- **Multithreaded engine (physics, networking, rendering).**

Context Diagram



Timeline (3-4 months)

[Link to Proposal]

Phases	*Tasks*	*Start & End dates*
Learning Phase - I	*Week 1: Basics of Flutter + Android NDK setup.* *Week 2: FFI Integration demo.*	*2 weeks*
Learning Phase - II	*Week 3: Physics simulation basics.* *Week 4: P2P networking concepts.*	*2 weeks*
Implementation Phase - Core Game Development	*Project setup, Game engine and physics prototype in C++.*	*2 weeks*
Implementation Phase - Integration	*Flutter UI + C++ integration (single-device mode).*	*3 weeks*
Implementation Phase - Networking	*Implement P2P networking (Wi-Fi Direct).*	*4 weeks*
Implementation Phase - Testing & Publishing	*·Add interpolation, lag handling, Testing, bug fixes, performance tuning.* *·Documentation, UI polish, final demo.*	*3 weeks*

Learnings and Deliverables

Learnings :

- Flutter integration with native C++ using Dart FFI
- Implementation of a modular 2D game engine using C++
- Understand real-time P2P networking on Android
- Develop deterministic game physics with an authoritative host model
- Gain skills in mobile performance optimization for low-latency gameplay
- Handle collision detection, networking, and system-level optimizations in real-time games

Expo Deliverables :

- Fully Functional Android Application:
  - Live 1v1 offline Air Hockey match on two Android devices
  - Flutter UI with smooth 60 FPS puck & paddle animations
  - Demonstration of low-latency P2P gameplay over Wi-Fi Direct/LAN with 30ms ping.
  - Real-time physics and collision handling by C++ engine

ImposterCode

Team Mentor
Dhanush - 9353241312

Team:
Jessica Mariam - 8891227041
Preethi Mondal - 9475081162
Ganesh Madhav - 7907914942

Introduction

Abstract :

- ImposterCode is a multiplayer social deduction game that combines Among Us-style gameplay with collaborative coding challenges.
- Players work together to debug and complete code in a shared VS Code-like environment, while hidden saboteurs must introduce bugs and suspicious code patterns.
- The game features real-time voice chat, AST-based code analysis for detecting sabotage, and voting mechanics to identify imposters, creating an engaging educational experience that teaches code review, debugging, and collaborative programming skills.

Technologies used :

- Frontend: React.js, Monaco editor, Agora Web SDK
- Backend: Node.js, REST API, Socket.io
- Database:PostgreSQL

Timeline

Link To Proposal

Phases	*Tasks*	*Start & End dates*
Learning Phase - I	*Tech stack introduction, Git/GitHub workflows, development environment setup, React.js fundamentals*	*2 weeks*
Learning Phase - II	*Node.js/Express backend development and REST API design, Socket.IO real-time communication and WebSocket concepts*	*2 weeks*
Implementation Phase - Foundations	*Project Setup & Architecture, Basic Game Infrastructure, Collaborative Code Editor*	*4 weeks*
Implementation Phase - Core game features	*Game State Management,AST Analysis Service, Voice Communication*	*3 weeks*
Implementation Phase - Testing & Refinement	*Bug-free, polished game experience Comprehensive testing, UI/UX improvements and responsive design fixes, Performance optimization and load testing, Add game statistics and leaderboard*	*3 weeks*

PHASE 1: GAME SETUP

Player Login & Lobby Creation

↓

Lobby Fill & Role Assignment

PHASE 2: CODING ROUND

Challenge Starts → Voice Chat During Coding

↓

Saboteur Inserts Bug

PHASE 3: MEETING/VOTING ROUND

Timer Ends – Meeting Triggered → Discussion Phase

NORMAL CODERS WIN :

- All saboteurs eliminated
- Code passes all test cases

SABOTEURS WIN if:

- Sabateurs = Normal coders remaining
- Code fails test cases due to bugs

Learnings and Deliverables

Learnings :

- Full-Stack Web Development
- API Integration
- Real-Time Systems & WebSockets
- Advanced Code Analysis (AST)
- WebRTC & Voice Communication
- Authentication & Security
- DevOps & Deployment
- Game Design & User Experience

Expo Deliverables :

- Fully Functional Web Application:
    - Production-ready multiplayer game
    - Responsive UI working across desktop
    - Support for 6-10 concurrent players per game session

YieldVoyager (DeFi Yield Exploration Agent

Team Mentor
Kailash- 91643 93666

Team:
Sarth Shah - 8849079895
Mihir Adhikari - 8827244739
Sourabh Kapure - 7021991449

Introduction

Abstract :

- Yield Analyzer Agent is an AI-driven system that collects and analyzes on-chain and off-chain DeFi data, normalizes and stores it, computes risk/return metrics, ranks yield opportunities, and generates human-friendly recommendations.

- Think of it like a smart investment assistant: it scans “crypto pools” (similar to bank accounts or stock options), compares returns, and suggests options based on your preference — e.g., low risk or high reward. In this environment, depositing funds is like putting money in a bank savings account, but interest rates are dynamically calculated by code and come from people borrowing or swapping crypto in the pool.

- The agent provides simulation-backed trade previews and supports non-custodial user-signed transactions with strict security safeguards.

Technologies Used

Frontend:
- React.js

Backend:
- FastAPI (Python)
- REST APIs

Database:
- Postgres + TimescaleDB
- Pinecone/Weaviate or FAISS

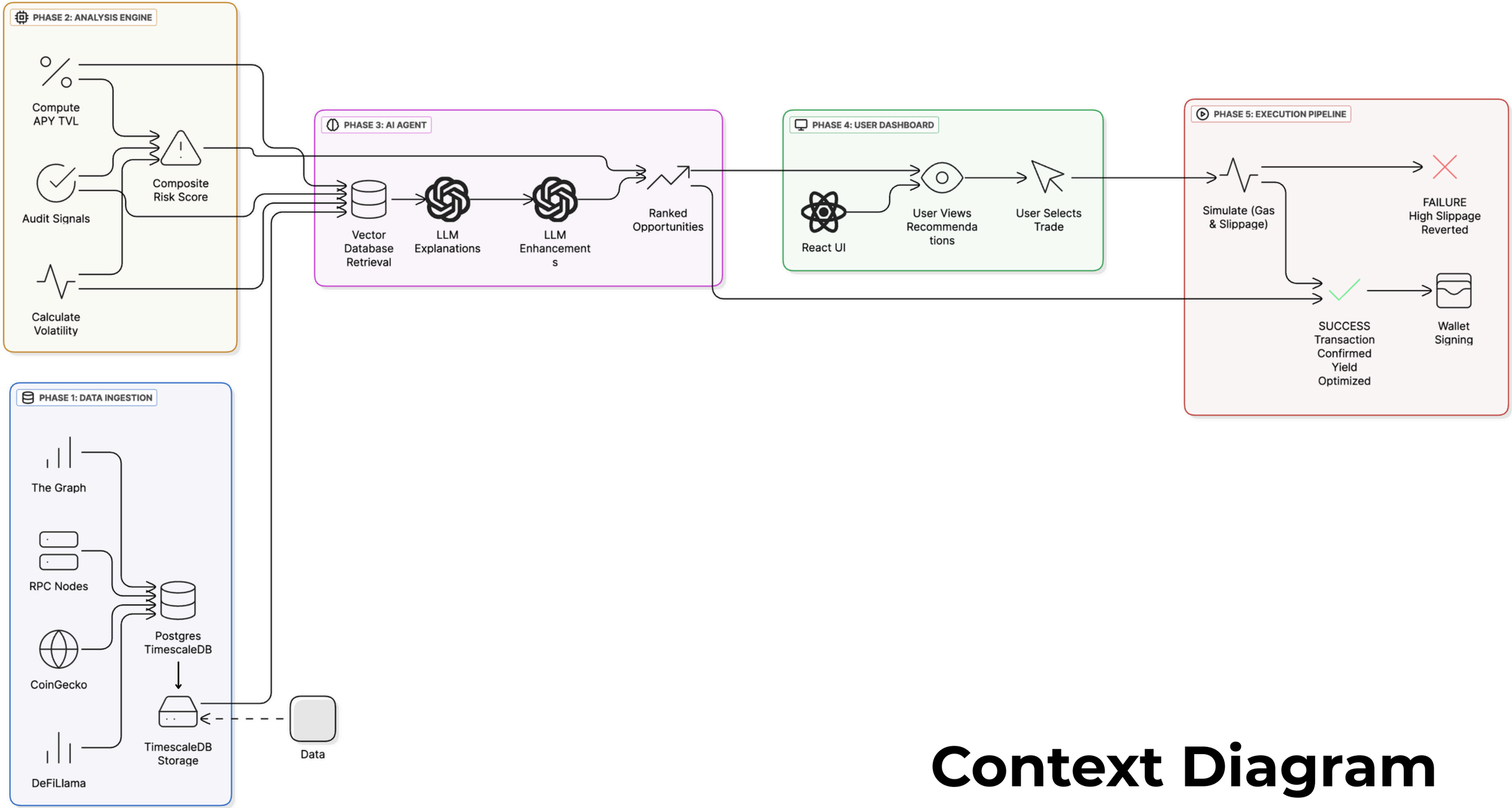
AI/ML:
- LLMs(Gemini, OpenAI models)
- LangChain
- SBERT Embeddings

Blockchain:
- Solidity
- Development environments(Hardhat/Foundry)
- Oracles (Chainlink/Pyth)
- DeFI Protocols (Uniswap, Aave etc.)

Timeline

Link to Proposal

Phases	*Tasks*	*Start & End dates*
Learning Phase - I	*Web stack (React, Databases, Backend) as well as ML(Deep Learning) introduction*	*2 weeks*
Learning Phase - II	*Web3/Blockchain introduction :- Solidity, Smart Contract Design, introduction to Decentralized Finance (DeFi)*	*2 weeks*
Implementation Phase - Foundations	*User Onboarding, API architecture and Database setup*	*3 weeks*
Implementation Phase - Core project features	*Implement yield scoring engine, integrate agent, and explanation system linking AI with live data.*	*4 weeks*
Integration and Testing	*Join all aspects of the project, and try to get them to work together.*	*3 weeks*



Context Diagram

Learnings and Deliverables

Learnings :

- **DeFi Protocol Integration & On-chain Data Analysis**
- **LLM Integration & Prompt Engineering**
- **Vector Database & Embeddings**
- **Transaction Simulation & Gas Optimization**
- **Wallet Integration & Web3 Development**
- **Time-Series Data Processing**
- **Risk-Adjusted Return Calculation**
- **Smart Contract Security Auditing**

Expo Deliverables :

- **Fully Functional Web Application**
- **Production-ready DeFi yield analyzer**
- **Responsive UI working across desktop**
- **Support for Aave, Compound, Uniswap V3, Curve protocols**
- **Real-time risk scoring with LLM-generated explanations**
- **MetaMask integration with transaction simulation**
- **Live dashboard with opportunity listings and risk badges**

PeerReach - Decentralized Mesh Communication

POC : Garv Mandloi (+91 9343939016)
Anshul Dadhich (+91 9001097420).
Charan Gowda (+91 9945219524)

Introduction

Abstract:
PeerReach is a decentralized mesh communication platform that enables message transmission and internet access in low-connectivity or disaster scenarios. The system creates peer-to-peer mesh networks using Bluetooth LE and Wi-Fi Direct, allowing offline message queuing and opportunistic internet relay when gateway nodes are available. It implements Delay-Tolerant Networking (DTN) principles to ensure reliable communication even in challenging network conditions.
The project aims to provide a cost-effective, infrastructure-less communication solution that can operate in remote areas, during disasters, or where traditional networks fail.

Technologies Used:

- **Frontend: React Native, JavaScript**
- **Networking: libp2p, Bluetooth LE APIs, Wi-Fi Direct**
- **Backend: Node.js, Express.js, Socket.io**
- **Database: SQLite (local), MongoDB (cloud gateway)**
- **Encryption: AES-GCM**

PeerReach: Communication Without Internet

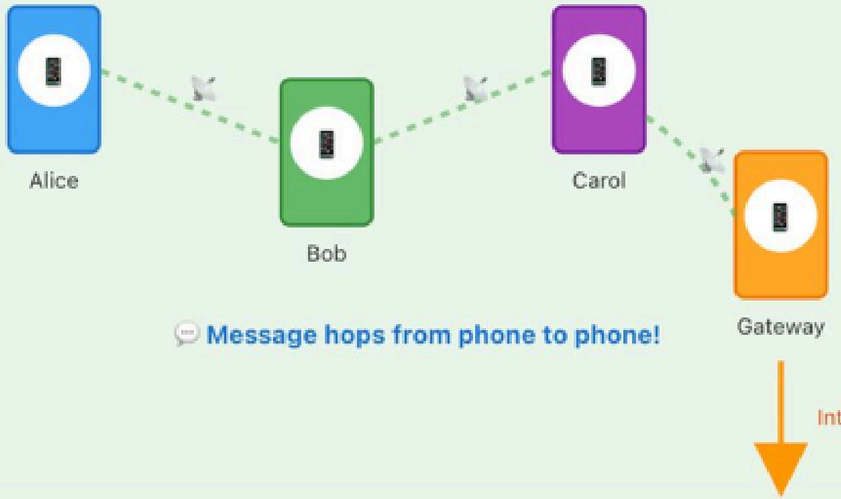
❌ Problem: Network Down

😟 Can't send messages!

😰 No internet!

Disaster, Remote Area, or Network Outage

✓ Solution: PeerReach Mesh Network



💬 Message hops from phone to phone!

How PeerReach Works

1 Send Message
Your phone connects to nearby phones using Bluetooth & WiFi Direct

2 Message Hops
Message jumps from phone to phone until it reaches the destination

3 Gateway Relay
If a phone has internet, it sends your message to the cloud!

✨ Key Benefits:

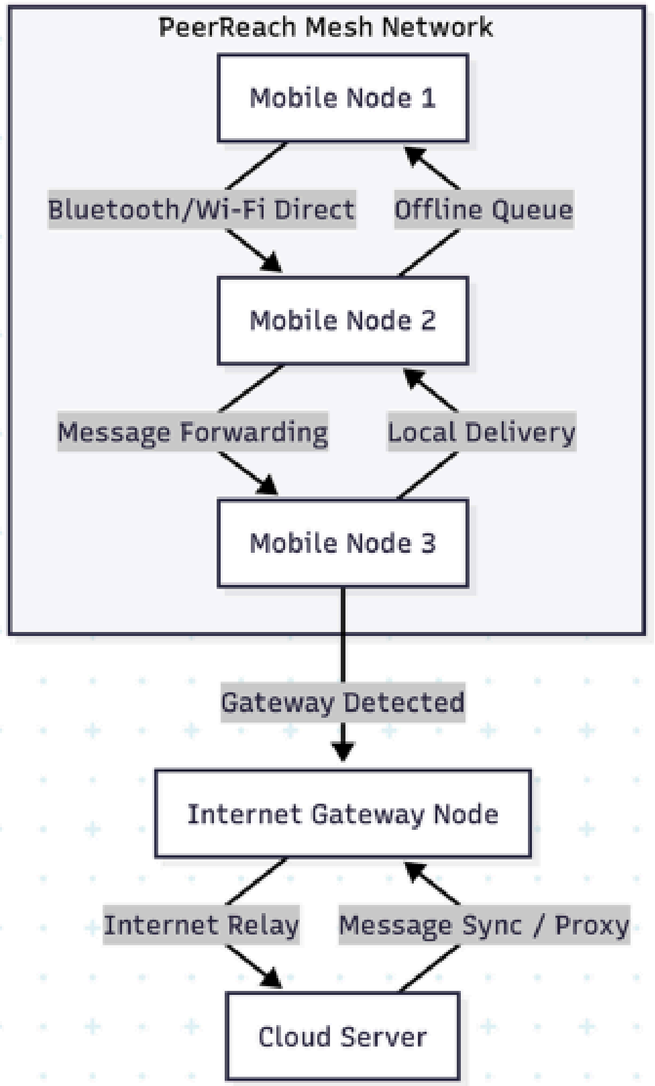
- 🆘 Works during disasters & emergencies
- 🏔️ Perfect for remote areas & hiking
- 🔒 Encrypted & secure communication
- 📵 No internet or cell towers needed
- 💰 Free - no data charges
- 🌐 Messages reach internet when possible

Timeline

Link to Proposal

Phases	Tasks	Start & End dates
Learning Phase - I	P2P networking concepts, libp2p fundamentals	Week 1–2
Learning Phase - II	React Native development, Bluetooth LE integration	Week 3–4
Phase 1 – Foundation	• Project setup, UI design, local storage setup (SQLite) • Bluetooth LE peer discovery and connection	Weeks 5–7
Phase 2 – Core Mesh Development	• Implement message forwarding & epidemic routing • Add encryption & multi-hop message delivery	Weeks 8–11
Phase 3 – Gateway & Internet Integration	• Implement gateway node detection • Enable opportunistic internet relay & caching	Weeks 12–14

Context Diagram



Learnings and Deliverables

Learnings:

- **Understanding of mesh networking concepts and delay-tolerant networks (DTN).**
- **Hands-on experience with React Native app development and state management.**
- **Experience with libp2p and peer-to-peer routing protocols like epidemic routing.**

Expo Deliverables:

- **A fully functional mobile app demonstrating peer-to-peer communication.**
- **Working demo showing offline message delivery across multiple devices.**
- **Integration of gateway nodes enabling opportunistic internet access.**
- **Complete source code and documentation .**

LEDBAT++ and TC CLI implemenation in ns-3

Sambhav Singh - 9354413246
Vedh Adla - 9177992152
Unnati Kumari - 9770714351

Introduction

Abstract :

- LEDBAT++ is a delay-based congestion control algorithm that minimizes latency while maintaining high throughput. Linux Traffic Control manages queuing and directly influences congestion behavior. Since ns-3 lacks both LEDBAT++ and a Traffic Control CLI, our project focuses on implementing LEDBAT++ in ns-3's TCP stack along with a Traffic Control command-line interface.

Technologies used :

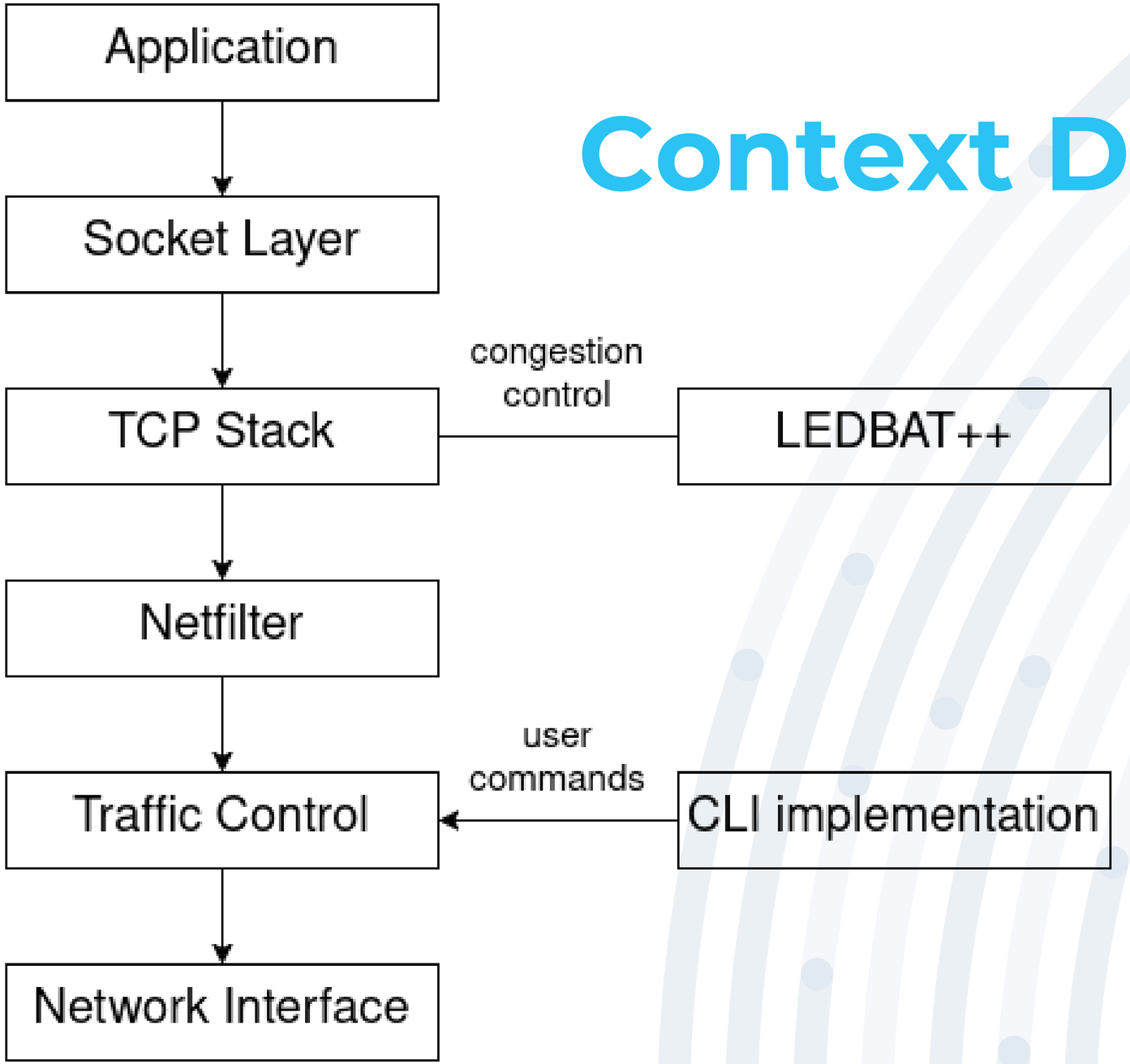
- ns-3 simulator (C++ framework, TCP/IP stack)
- Linux Networking(Traffic Control)
- C++ (for module implementation)
- Python / ns-3 helpers (for scripts, running experiments, and plotting results)
- Git and GitLab(open source contribution)

Timeline

[Link to Proposal]

Phases	*Tasks*	*Start & End dates*
Learning Phase - I	*Computer Networks*	*2 weeks*
Learning Phase - II	*Linux Networking, NS3*	*2 weeks*
Implementation Phase - Foundation and setup	• *Set up ns-3, study LEDBAT* • *Analyze Linux Traffic Control features*	*3 weeks*
Implementation Phase - Implementation and Integration	• *Write the LEDBAT++ C++ module* • *Develop the Traffic Control CLI.* • *Integrating both into ns-3's TCP stack and queuing modules with proper parameter handling.*	*4 weeks*
Implementation Phase - Testing and Open Source Release	• *Conduct extensive simulations and CLI tests under varied network scenarios.* • *finalize documentation for open-source release.*	*3 weeks*

Context Diagram



Learnings and Deliverables

Learnings :

- TCP/IP Networking and NS-3
- Linux Network Stack and Linux Networking Commands
- Git and GitLab(version control for team collaboration and open source)
- C++ modules
- CLI development

Expo Deliverables :

- LEDBAT++ ns-3 Module: Fully implemented C++ congestion control module integrated into ns-3.
- Traffic Control CLI: Interactive command-line interface to modify queueing disciplines and network parameters during simulations.
- Real-time Visualization: Graphs showing RTT, throughput, queue delays, and fairness metrics dynamically as simulations run.
- Simulation Scenarios & Scripts: Predefined experiments with multiple flows and topologies, automated via Python/ns-3 helpers.
- Open-Source Repository: Complete code, scripts, and usage guides for community access.

TrapShield–An Intrusion Detection System + Custom Interactive Honeypot

Mentors:

Ananya A. K	**231CS208**
Ankita A	**231IT007**
Aryan Deshmukh	**231AI006**
Utsav Bhamra	**231CS161**

Introduction

Abstract:

Our project aims to design a **rule-based Intrusion Detection System (IDS)** that continuously monitors live network traffic, detects suspicious activities like port scans or failed logins, and redirects attackers to a custom-built honeypot. The honeypot safely simulates real services such as SSH, HTTP, or FTP, engages with attackers, and logs their activities for analysis.

Through this, we aim to study intrusion behavior, refine detection rules, and create a practical learning platform for network security.

Technologies used :

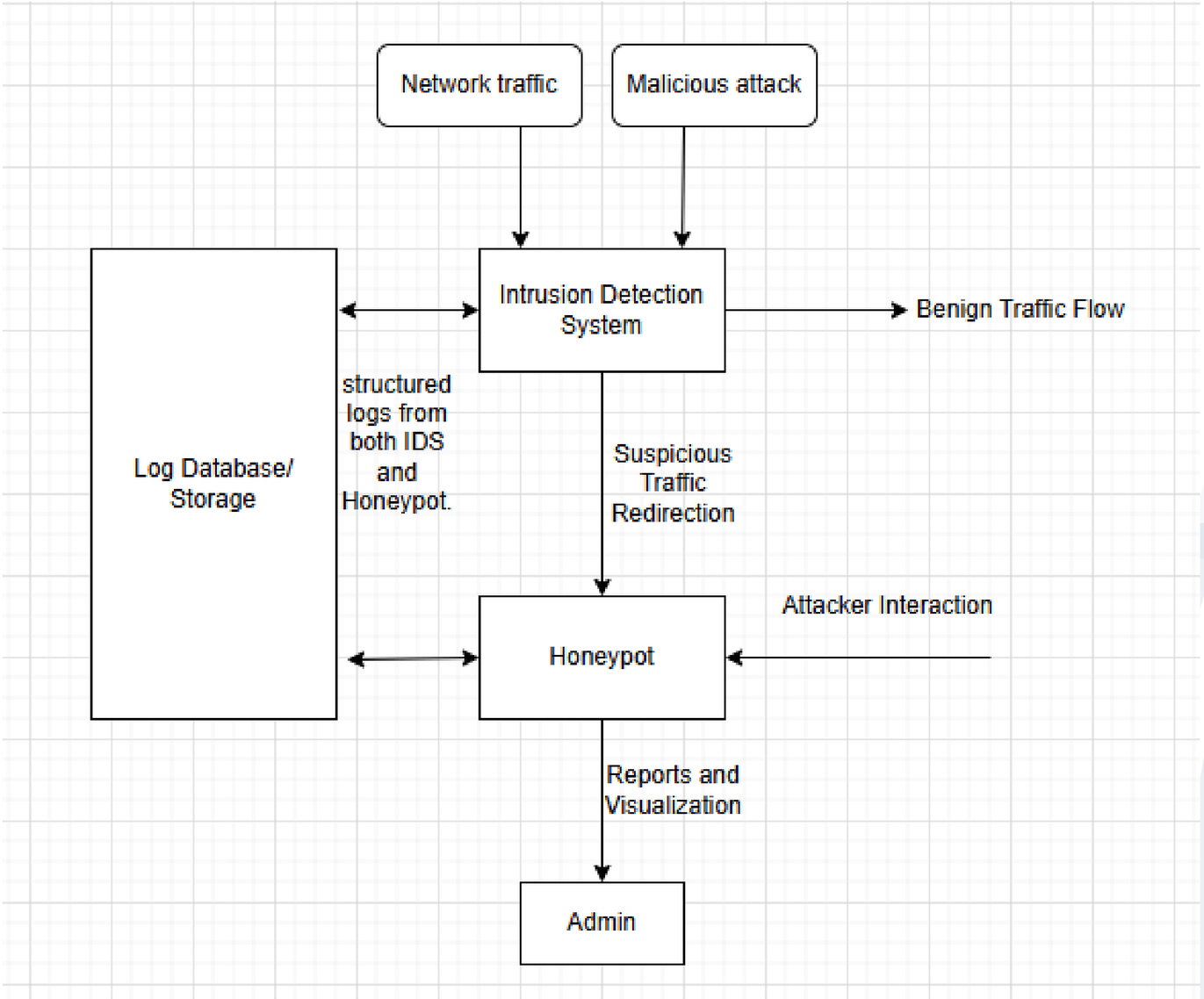
- Traffic Capture & Analysis: Scapy, PyShark, Wireshark
- IDS & Redirection: iptables / nftables
- Honeypot Development: Custom Python scripts (SSH/HTTP/FTP emulation), Cowrie (for reference)
- Logging & Visualization: SQLite / JSON, Pandas, Matplotlib / Seaborn
- Testing: Ubuntu/Linux, Docker / VirtualBox.

Timeline

Link to Proposal

Phases	*Tasks*	*Start & End dates*
Learning Phase	*Basics of networking, packets, and IDS concepts. Hands-on with Scapy, PyShark, iptables. Intro to honeypots .*	*1-2 weeks*
Module 1 – Traffic Capture	*Capture live packets, extract IPs, ports, and protocols. Log traffic to JSON/SQLite.*	*1-2 weeks*
Module 2 – IDS Implementation+ML(optional)	*Build rule-based IDS for suspicious activity detection; optionally test simple ML models (e.g., decision tree) for pattern classification. Validate using Nmap/Metasploit.*	*4 weeks*
Module 3 – Honeypot Setup	*Build custom honeypot (SSH/HTTP/FTP), redirect traffic via iptables, record interactions.*	*3 weeks*
Module 4 – Analysis & Visualization	*Analyze logs, visualize attack trends, refine IDS rules. Prepare reports & demo.*	*2 weeks*

Context Diagram



Learnings and Deliverables

Learnings

- Basics of network packets, IPs, ports, and protocols.
- Hands-on with Python network programming using sockets.
- Understanding how an IDS detects threats and how honeypots trap attackers.
- Working with iptables for redirection and basic data logging and visualization.
- Experience with team-based cybersecurity development and testing.

Expo Deliverables

- Live Demo: IDS detecting suspicious activity and redirecting traffic to honeypot.
- Custom Honeypot Dashboard: Showing attacker logs, commands, and trends.
- Visualization: Attack frequency graphs and behavior analytics.
- Report & Documentation: Architecture, rules used, and insights from captured data.

FinGuard AI: Secure, Explainable, and Intelligent UPI Fraud Detection

Team Mentor
Aaryan Patil - 8855930471

Team:
Varshini Reddy Pateel - 8317326594
Siddhi Dhawale - 8669662251
Sarayu Narayanan - 9741972217

Introduction

Abstract :

•**FinGuard AI** is a secure, explainable, and intelligent UPI fraud detection framework that identifies fraudulent transactions in real time using sequential ML models (LSTM/GRU) integrated with GPT-based contextual assistance.
•It combines **explainable AI (SHAP/Captum)** and **generative AI** to provide transparent reasoning and personalized, actionable user guidance — going beyond traditional fraud detection to enhance trust and usability.

Technologies Used :

Frontend:
Streamlit / Flask
Plotly, HTML, CSS

Backend:
Python (Flask / FastAPI)
OpenAI GPT API

Machine Learning:
PyTorch, Scikit-learn
NumPy, Pandas

Explainability:
SHAP, Captum

Data Simulation:
Faker(Synthetic UPI Data)

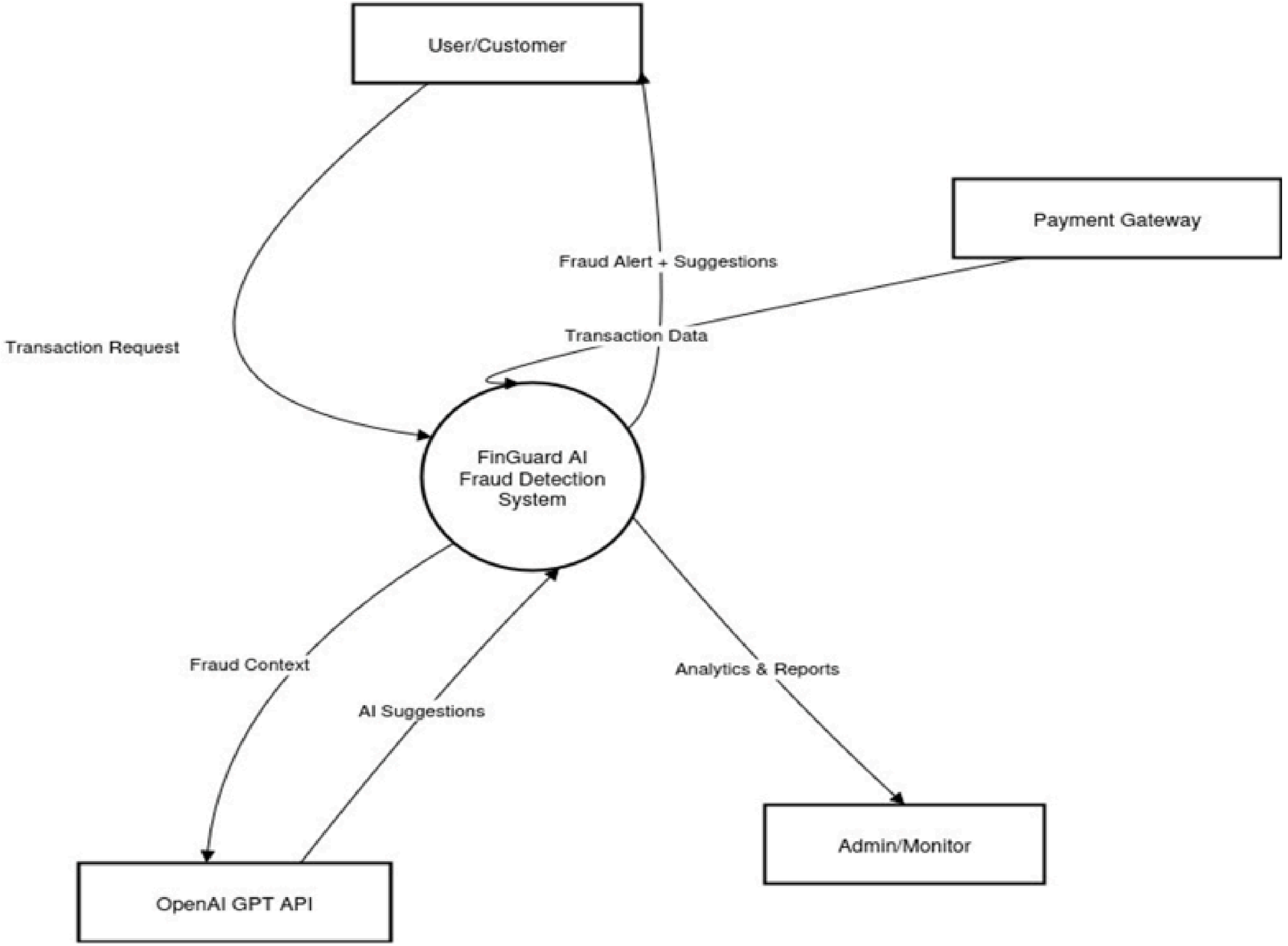
Deployment:
Streamlit Cloud/ Render / GitHub

Timeline

[Link to Proposal]

Phases	*Tasks*	*Start & End dates*
Learning Phase - I	*Machine Learning fundamentals, Time-series and anomaly detection basics, Feature extraction and preprocessing*	*2 weeks*
Learning Phase - II	*Explainable AI (SHAP, Captum), Data simulation using Faker, Model evaluation and performance metrics*	*2 weeks*
Implementation Phase - Foundations	*Fraud detection model development (LSTM/GRU), Dataset integration and training pipeline setup, Model tuning and validation*	*4 weeks*
Implementation Phase – Core features	*Explainability module integration, GPT-based contextual guidance, Streamlit interface design*	*3 weeks*
Implementation Phase - Testing & Refinement	*End-to-end system integration, UI/UX refinement, Testing and performance optimization, Final documentation and demo*	*3 weeks*

Context Diagram



Learnings and Deliverables

Learnings:

- Full-Stack Web Development (React + Flask/Streamlit)
- Machine Learning & Time-Series Modeling (LSTM/GRU)
- Explainable AI (SHAP, Captum)
- GPT Integration for Context-Aware Assistance
- Real-Time Fraud Detection Simulation
- UI/UX Design for Dashboard & User Guidance

Expo Deliverables:

- Fully Functional Web Dashboard:
  - Simulated UPI plugin environment
  - Real-time fraud detection using sequential ML models
  - Transparent explanations of flagged transactions
  - GPT-generated, context-aware user guidance
  - Interactive visualization of transaction flow and fraud reasoning

FortiFy

Real-Time Device Access Monitor

4th year mentor :
Khush Patel - 82001 60232

3rd year Mentors :
Keya - 9606767576
Prathyanga - 8123401103
Shriya - 7760544126

FortiFy

Abstract :

- The Real-Time Device Access Monitor (RTDAM) project focuses on securing digital devices in educational environments by monitoring access in real time.
- The goal is to prevent unauthorised usage and potential theft, ensuring that students’ and faculty’s devices remain safe at all times.
- This system uses artificial intelligence for facial recognition and instant alerts, providing a reliable solution for campus security.

Technologies used :

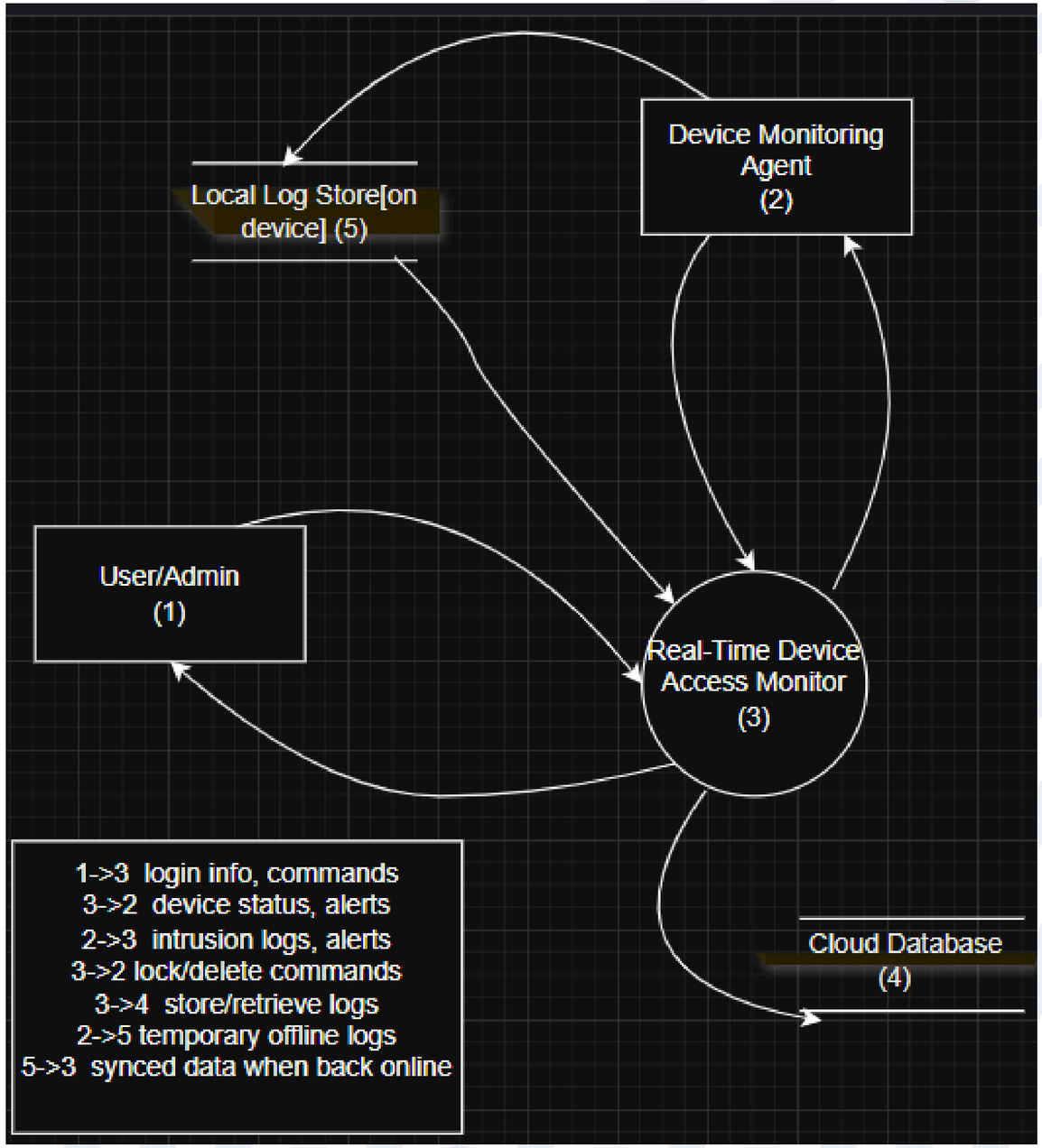
- AI-based Facial Recognition
- Real-time Notification System
- Remote Locking and Data Protection
- Implement a GPS tracking system
- Cross-platform Compatibility: Windows, Linux, Mac OS
- Dashboard for monitoring and control

Timeline

[Link to Proposal]

Phases	*Tasks*	*Start & End dates*
Learning Phase - I	*Implement a simple face detection program using Python & OpenCV.*	*2 weeks*
Learning Phase - II	*learn how build a basic Android app to receive alerts , Learn Docker fundamentals and containerize a small app.*	*2 weeks*
Implementation Phase - Cross-Platform Monitoring Agent Development	*– Develop main agent in Python supporting Windows (via pywin32), Linux (systemd), and macOS (launchd).* *– Implement boot-time startup and local offline log storage using SQLite, or in encrypted local files.* *– Integrate OpenCV for motion detection and photo capture during unauthorized access.*	*4 weeks*
Implementation Phase - Dashboard & Remote Control Integration	*– Develop cloud backend using Django REST / Express.js and connect with Amazon S3.* *– Build web dashboard (React + Tailwind CSS) for viewing logs, issuing lock/wipe/encrypt commands.* *– Use cryptography library for secure file operations and implement WebSocket alerts.*	*3 weeks*
Implementation Phase - AI Recognition, Alerts & Final Testing	*– Integrate AI-based facial recognition using pretrained models from face_recognition / DeepFace.* *– Implement push/email notifications via Firebase Cloud Messaging (FCM) or SMTP.* *– Conduct cross-OS testing, optimize sync and API latency, and finalize documentation + deployment using Docker.*	*3 weeks*

Context Diagram



Learnings and Deliverables

Learnings :

- You will gain practical experience building cross-platform security software for real-life situations.
- Learn to integrate AI-powered facial recognition into device security using Python and OpenCV.
- Get hands-on experience with Flask, Node.js, MongoDB Atlas, and React for backend and dashboard development.
- Understand how to use REST APIs and cloud data sync for secure communication.
- Develop project planning skills and learn best practices for privacy and data security.

Expo Deliverables :

- You will showcase a live demo of the Real-Time Device Access Monitor, detecting unauthorised access and sending instant alerts.
- Present a functional web dashboard with live monitoring, evidence capture, and remote control features.
- Share posters and clear documentation explaining the project’s design and your contributions.
- Demonstrate the product’s reliability and user-friendliness with real evaluation metrics.

DrawGuess AI

Team Mentor:

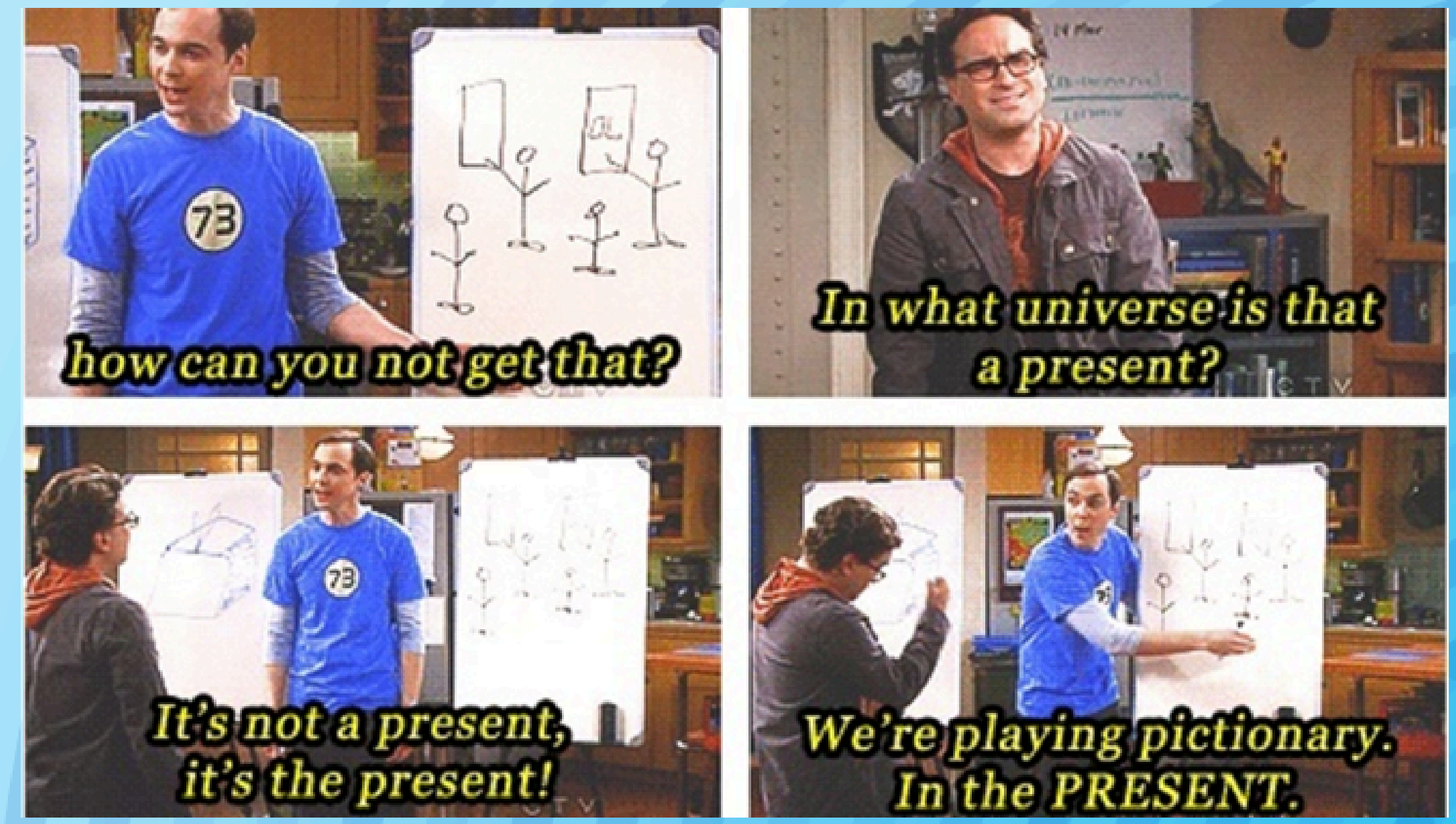
Deepak Nayak - 9353241312

Team:

Sambhav - 9503277804

Dhanya D- 8660142554

Akshaj- 89046 80789



Abstract

DrawGuess-AI is a real-time multiplayer drawing game where multiple players sketch simultaneously while an AI model predicts their drawings in real time. The system uses convolutional and recurrent neural networks (CNN and RNN) trained on both image-based and stroke-based datasets to recognize drawings dynamically as they are made. Built with FastAPI and WebSockets for LAN-based multiplayer support, it provides instant AI feedback, synchronized game states, and automatic round completion when a drawing is correctly identified.

Technologies Used:

Frontend: React.js, HTML5 Canvas, JavaScript (ES6+), Socket.IO Client
Backend: Node.js, Express.js, Socket.IO Server, FastAPI/Flask
Machine Learning: PyTorch (CNNs, RNNs, Transfer Learning)
Multiplayer: WebSocket/Socket.IO for real-time LAN (4–6 players)

Timeline

Phases	Tasks	Start & End dates
Learning Phase - I	• Learn ML basics: datasets, training loops, losses, metrics. • Study CNN concepts (Conv, Pool, FC) and build a small PyTorch CNN (MNIST/CIFAR-10). • Learn RNN fundamentals (sequences, LSTM/GRU). • Implement a simple RNN task in PyTorch (text or sequence prediction).	Oct 15-Nov 21
Pretrained Model Experimentation	Work with pretrained CNN (e.g., ResNet/MobileNet) and pretrained RNN (e.g., Sketch-RNN or text model). Run inferences, understand transfer learning, modify small layers.	Dec 1-Dec 15
Custom model designing and testing	Design and train custom CNN and RNN models in parallel — CNN for doodle image recognition and RNN for stroke-based prediction. Integrate both modules, test real-time stroke recognition, and evaluate inference speed and overall performance.	Dec 15-Jan 31
Multiplayer Game Prototype (LAN)	Build minimal frontend (HTML + Canvas + JS). Setup FastAPI backend with WebSocket connections for multiplayer game logic. Integrate model predictions into the game loop.	Feb 1-Feb 21
Final Integration & Evaluation	Tune model parameters, optimize inference time, finalize LAN testing with 4 players. Documentation and demo preparation.	Feb 22-March 15

Learnings and Deliverables

Learnings:

- Deep Understanding of Networks: Hands-on experience building CNNs and RNNs from scratch for doodle recognition and stroke prediction.
- Transfer Learning: Applying and modifying pretrained CNN and RNN models to understand real-world AI workflows.
- Integration & Real-Time AI: Combining CNN + RNN modules, optimizing inference, and testing predictions in real-time.
- Full-Stack Development: Building frontend (HTML/Canvas) and backend (FastAPI) with WebSocket-based real-time communication.
- Game Design & UX: Designing multiplayer mechanics, responsive UI, and LAN-based testing.

Expo Deliverables :

- AI-Driven Predictions: Real-time doodle and stroke recognition using custom and pretrained CNN & RNN models.
- Interactive Web Interface: Responsive Canvas-based drawing and user-friendly multiplayer experience.
- LAN-Based Multiplayer Game: Real-time sessions supporting 4–6 concurrent players with smooth gameplay.

Drawing 5/6

Draw

pineapple

in under 20 seconds

Got It!

Quantiv (Quant Interactive)

4th year mentor :
Parikshith Bismillah Rahman

3rd year mentors :
Ronin Mukherjee, 7338458641
Purav Nayak, 63624 49045
Rushil Mandavia, 9372212053

Introduction

Abstract

Quantiv is an interactive, web-based dashboard designed to price and visualize financial options in real-time.

- What it is: A professional-grade tool that makes complex financial theory tangible and easy to understand by visualizing the impact of market variables on option prices.
- Core Feature: It allows users to manipulate key financial parameters (like stock price, time, and volatility) through an intuitive interface and see the results of complex calculations instantly.
- Primary Goal: To create a powerful educational and analytical tool that transforms abstract financial models into a tangible, hands-on experience, bridging the gap between theory and real-world application.

Technologies Used

The project uses a dual-language architecture, combining the high performance of C++ with the rapid development capabilities of Python.

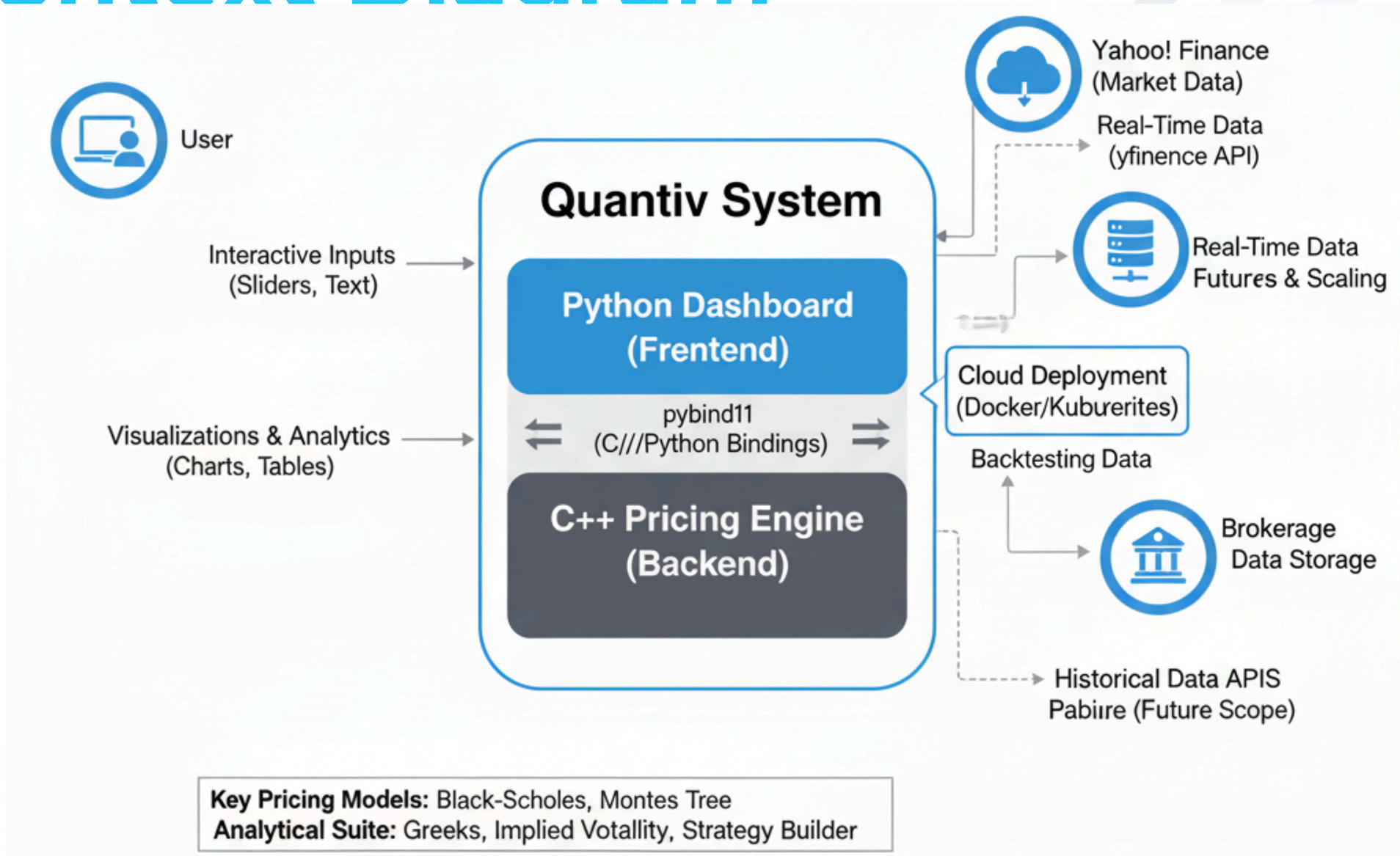
- Backend (C++):
  - Language: C++17 for fast, efficient computation of financial models.
  - Libraries: Standard Template Library (STL) for core data structures.
  - Build System: CMake to manage the compilation process.
- Frontend (Python):
  - Web Framework: Plotly Dash to build the interactive, callback-driven user interface.
  - Data Handling: NumPy and pandas for numerical operations.
  - Visualization: Plotly Express and Plotly Graph Objects to create dynamic charts, payoff diagrams, and 3D volatility surfaces.
  - Market Data: yfinance library to fetch real-time financial data.
- Integration Layer:
  - C++/Python Bindings: pybind11 to create a seamless and high-performance link between the C++ backend engine and the Python frontend

Timeline

[Link to Proposal]

Phases	Tasks	Duration
Month 1: Mentorship Phase	Complete all mentorship assignments to demonstrate foundational skills in C++ (OOP), Python (visualization), and Monte Carlo methods.	4 Weeks
Month 2: Core Engine & Foundational Dashboard	- Develop the C++ engine with BOPM and BSM models. - Establish the Python-C++ bridge using pybind11. - Build the initial static UI with Plotly Dash.	4 Weeks
Month 3: Interactivity & Core Visualizations	- Implement interactive sliders and real-time Dash callbacks. - Develop the dynamic Binomial Tree visualization. - Integrate real-time calculation and display of the Greeks.	4 Weeks
Month 4: Advanced Analytics & Model Expansion	- Expand the C++ engine with Trinomial Tree and Monte Carlo models. - Develop the Implied Volatility (IV) Calculator. - Build the Option Strategy Builder for multi-leg strategies.	4 Weeks
Month 5: Sophisticated Visualizations & Polish	- Create advanced charts: Payoff Diagrams, Greeks Charts, and the 3D Volatility Surface. - Integrate real-time market data fetching. - Refine UI/UX and complete documentation.	4 Weeks
Month 6: Deployment & Future-Proofing	- Buffer time for bug fixing and feature completion. - Containerize with Docker and deploy to a cloud service. - Research and begin proof-of-concept for future scope.	4 Weeks

Context Diagram



Learnings and Deliverables

Learnings for 2nd Years

This project offers a practical immersion into advanced software development and quantitative finance.

- Hybrid C++/Python Development: Master building high-performance systems by integrating a C++ computational backend with a Python interactive frontend (using pybind11).
- Quantitative Finance Foundations: Gain hands-on understanding of option pricing models (Binomial, Black-Scholes, Monte Carlo) and risk metrics (The Greeks).
- Interactive Web Dashboards: Develop dynamic, real-time data visualization and analytics using Plotly Dash.
- Full Software Lifecycle: Experience designing, implementing, and potentially deploying a professional-grade application.

Expo Deliverables

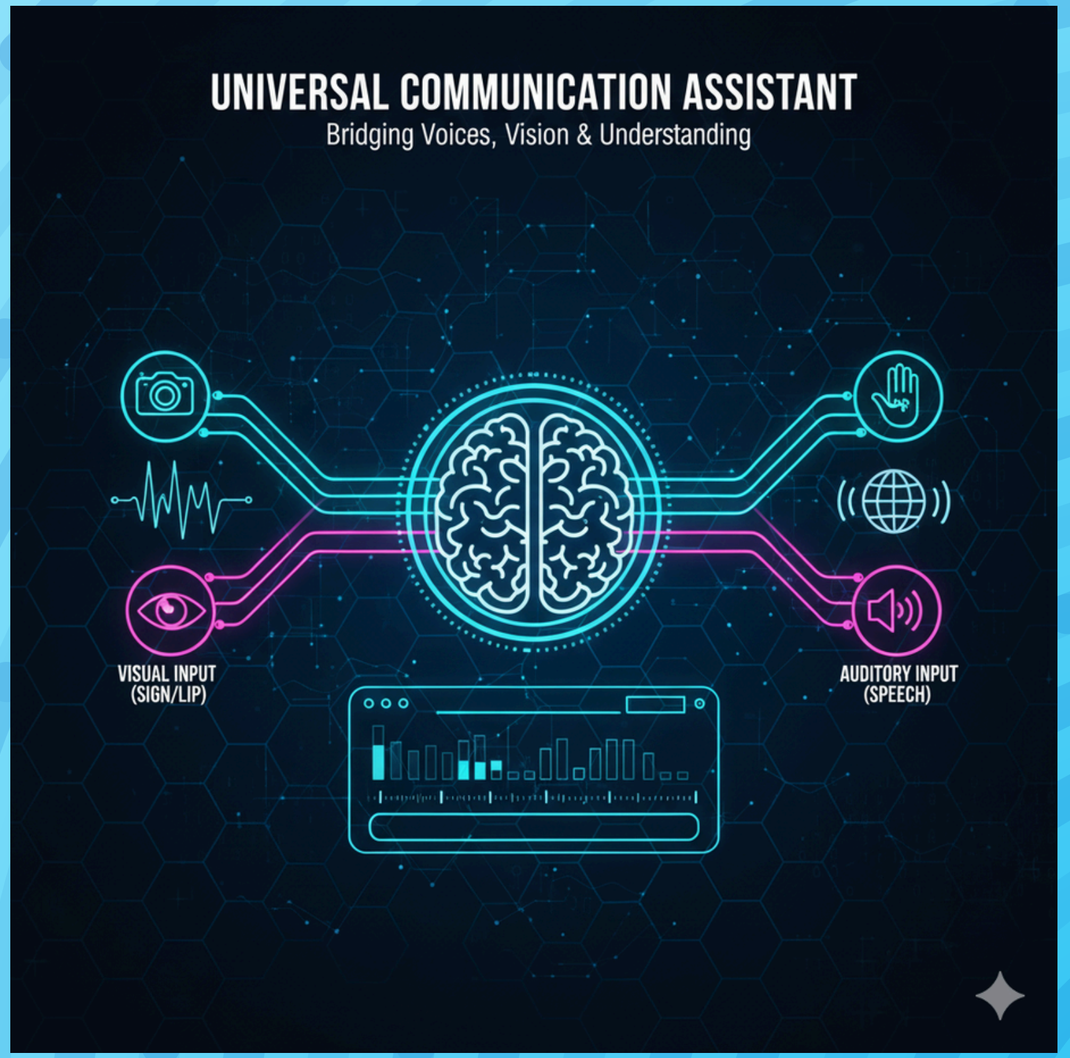
At the expo, we will showcase Quantiv, a live, interactive financial analytics dashboard.

- Live Interactive Dashboard: A functional web application demonstrating real-time option pricing with dynamic input controls.
- Dynamic Model Visualizations: Interactive graphical representations, such as the Binomial Tree, updating instantly with user changes.
- Real-Time Risk Analysis: Display of "Greeks" and payoff diagrams for immediate insight into option sensitivities and strategies.
- Multi-Model Comparison: Ability to compare pricing results from different financial models side-by-side.

UNICOMM

Team Mentor
Kailash-9164393666

Team:
Akshit Vinu - 7259534424
Amogh Kulkarni-8105967138
Samruddhi Deshpande-8605218476



Introduction

Abstract :

- UniComm is an AI-powered communication bridge that unifies speech, sign, and lip movements into one intelligent system.
- It enables real-time understanding and translation across languages and communication modes.
- Whether a person speaks, signs, or lip-syncs, the system interprets and converts the input into meaningful text or translated speech.
- For spoken input, UniComm Nexus performs automatic language detection, ASR (Automatic Speech Recognition), machine translation, and voice-cloned TTS, reproducing the output in the original speaker's voice.
- By integrating vision, speech, and language intelligence, it breaks linguistic and sensory barriers, making communication truly universal, inclusive, and human-like.

Technologies used :

Artificial Intelligence & Deep Learning

- CNNs, RNNs & Transformers for vision and speech tasks.
- Transfer Learning & Fine-Tuning on multilingual datasets.

Sign Language Recognition

- Pose Estimation: MediaPipe / OpenPose.
- Classification: CNN + LSTM / Vision Transformers.
- Dataset: Indian Sign Language / Sign MNIST.

Lip Reading

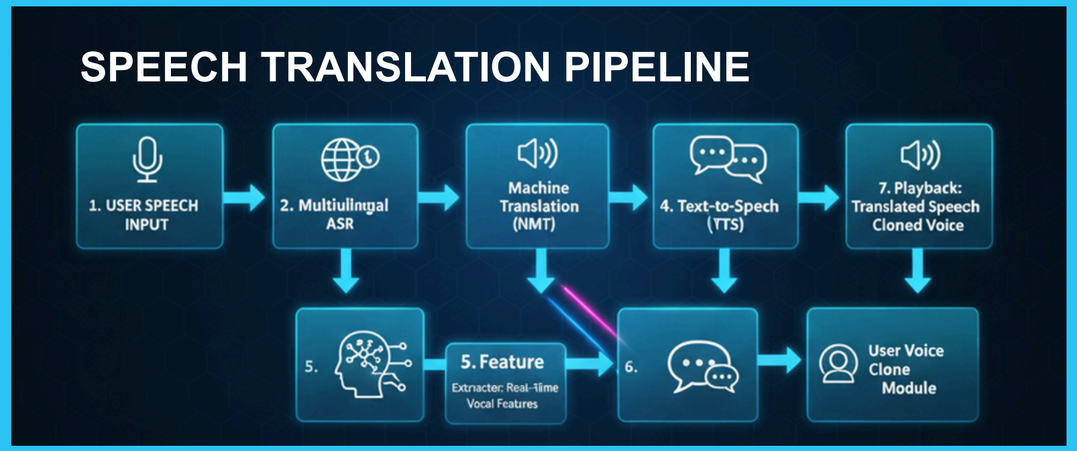
- Model: LipNet / 3D CNN + GRU.
- Tools: OpenCV for mouth ROI extraction.
- Dataset: GRID / LRS2.

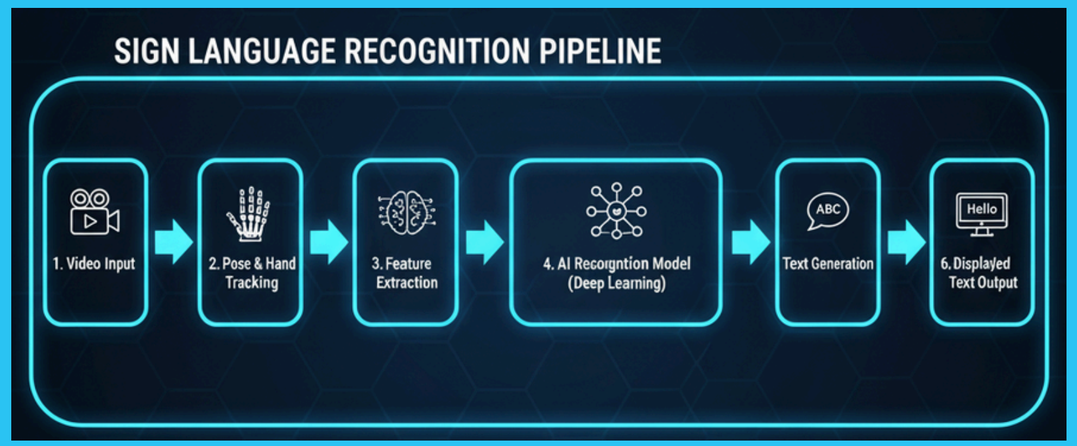
Speech Translation & Voice Cloning

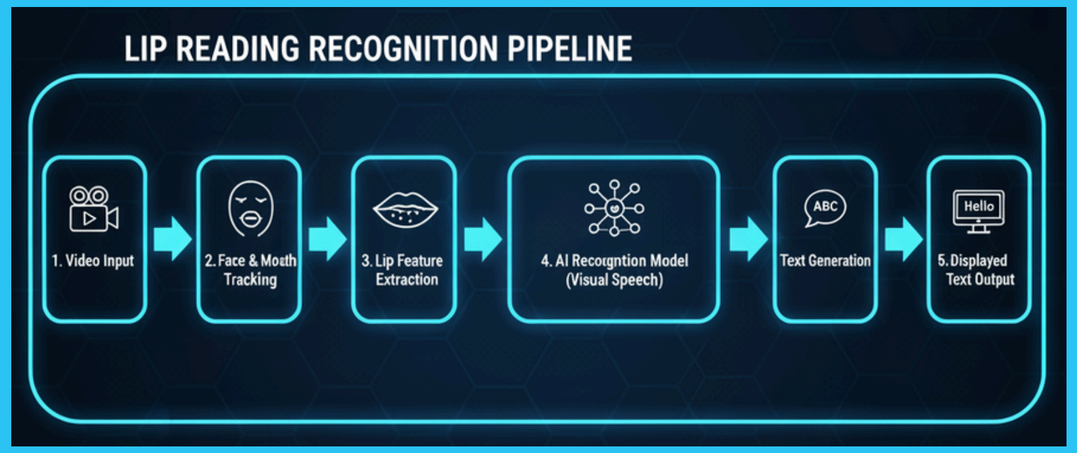
- ASR: Whisper / wav2vec2-xlsr.
- Translation: mBART-50 / M2M100.
- Voice Cloning: Resemblyzer + YourTTS / Tacotron2 + HiFi-GAN.
- Datasets: Common Voice / CSS10 / VCTK.

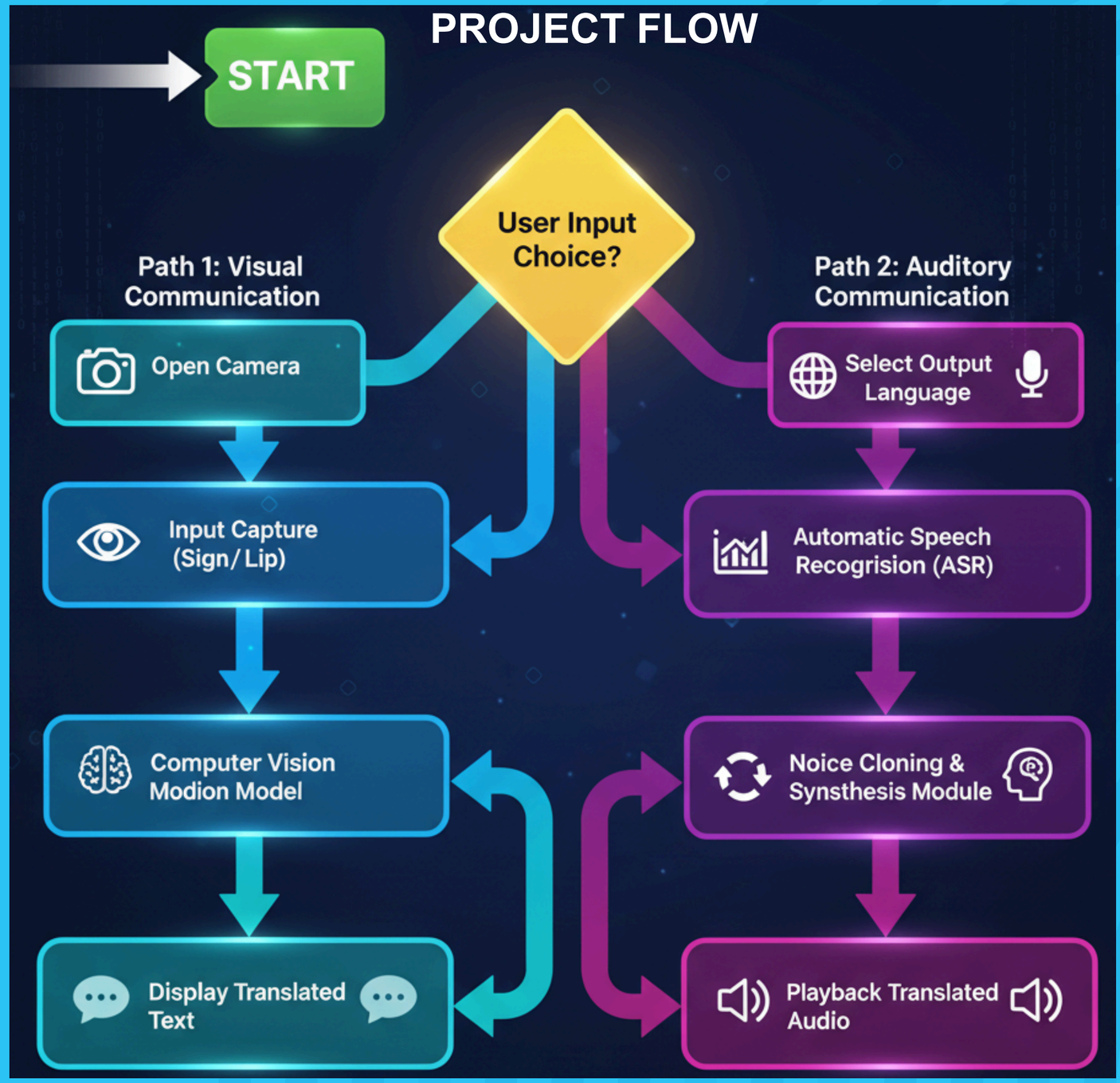
Integration & Interface

- Tech Stack: Python, PyTorch, TensorFlow.
- Frontend: React + FastAPI backend.
- Tools: OpenCV, MediaPipe, torchaudio, pyaudio for real-time input/output.


SPEECH TRANSLATION PIPELINE
1. USER SPEECH INPUT
2. Multiulingal ASR
Machine Translation (NMT)
4. Text-to-Spech (TTS)
7. Playback: Translated Speech Cloned Voice
5.
5. Feature
Extractor: Real-time Vocal Features
6.
User Voice Clone Module


SIGN LANGUAGE RECOGNITION PIPELINE
1. Video Input
2. Pose & Hand Tracking
3. Feature Extraction
4. AI Recognition Model (Deep Learning)
Text Generation
ABC
Hello
6. Displayed Text Output


LIP READING RECOGNITION PIPELINE
1. Video Input
2. Face & Mouth Tracking
3. Lip Feature Extraction
4. AI Recognition Model (Visual Speech)
Text Generation
ABC
Hello
5. Displayed Text Output


PROJECT FLOW
START
User Input Choice?
Path 1: Visual Communication
Open Camera
Input Capture (Sign/ Lip)
Computer Vision Modion Model
Display Translated Text
Path 2: Auditory Communication
Select Output Language
Automatic Speech Recogrision (ASR)
Noice Cloning & Synsthesis Module
Playback Translated Audio

Timeline

Link to Proposal

Phases	*Tasks*	*Start & End dates*
Learning Phase - I	*learn deep learning basics for vision and speech, including CNNs, RNNs, Transformers, object/pose detection, and audio preprocessing.*	*2 weeks*
Learning Phase - II	*study multilingual ASR, machine translation, speaker embeddings, TTS, and multimodal AI integration.*	*2 weeks*
Implementation Phase - Foundations	*fully implement the sign language module while preparing datasets and training models for lip reading and speech pipelines.*	*4 weeks*
Implementation Phase - Create a fully functional pipeline	*Evaluate the performance of the models and integrate lip reading, speech translation, and voice cloning modules to create a fully functional end-to-end multimodal system.*	*3 weeks*
Implementation Phase - Testing & Refinement	*Fix bugs, optimize performance, and ensure accurate, real-time multimodal communication.Build a frontend for deployment*	*3 weeks*

Learnings and Deliverables

Learnings :

- Understanding multimodal AI pipelines integrating vision, speech, and language.
- Hands-on experience with advanced computer vision, Automatic speech recognition, machine translation and Text2Speech technology.
- Working with pretrained models, fine-tuning, and transfer learning for specific languages/datasets.
- Extracting and using speaker embeddings for voice cloning.
- Building real-time systems with React frontend and FastAPI backend.
- Handling data preprocessing, video/audio streams, and multilingual datasets.

Expo Deliverables :

- Frontend which allows the user to choose the medium of input.
- If Sign Language/LipReading is chosen, web camera to capture the input, output text interpreting the message will be seen on the screen.
- If speech is chosen, user gets to choose the target language, take input through a microphone, output audio in target language in user’s voice.

PacDevil

Team Mentor
Kailash - 9164393666

Team:
Rijul Raman - 6360898496
Tejas Bajaj - 9099616168
Sathwik Mavilla - 9036858403

Introduction

Abstract :

PAC-DEVIL is a web-based Pac-Man game enhanced with dynamic, Level Devil–inspired mechanics such as shifting walls, hidden traps, and unpredictable obstacles. Ghost agents in the game are controlled by a reinforcement learning model using Proximal Policy Optimization (PPO), which adapts their behaviour in real time based on gameplay. The project combines engaging gameplay with AI experimentation, allowing players to experience increasingly challenging scenarios while demonstrating adaptive multi-agent behaviours and interactive RL concepts.

Technologies used :

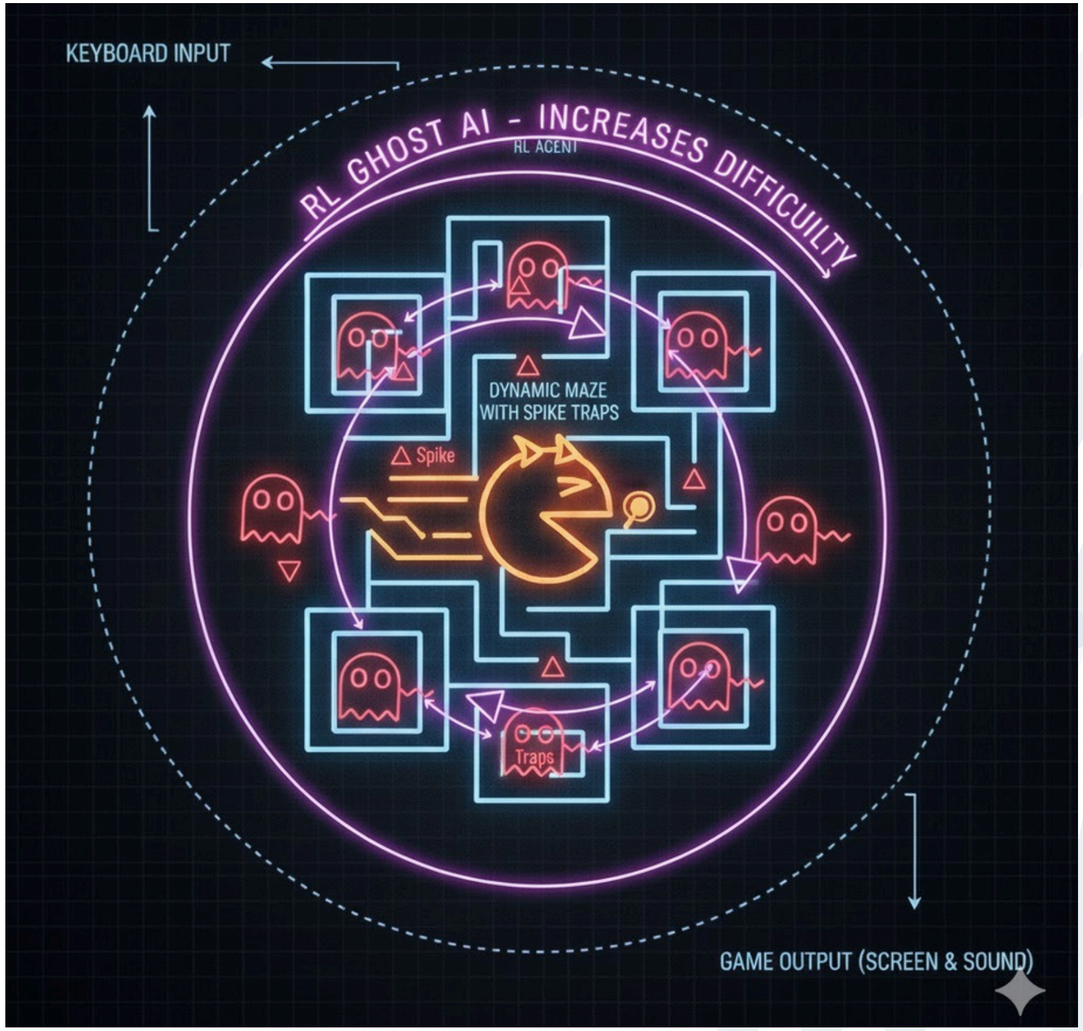
- **Frontend: *PhaserJS***
- **Backend:**
  - ***PyTorch / TensorFlow (PPO Model)***
  - ***Flask / FastAPI***
- **Training Environment: *Custom gym-like environment for RL training episodes using OpenAI gym***

Timeline

Link to Project

Phases	*Tasks*	*Start & End dates*
Learning Phase - I	*Learn important game dev concepts and get familiar with PhaserJS and familiarize yourself with git*	*2 weeks*
Learning Phase - II	*Go through articles and research papers about reinforcement learning specifically PPO*	*2 weeks*
Implementation Phase - I	*Dev team makes a basic version of pacman and the ML team creates an RL training environment for pacman*	*3-4 weeks*
Implementation Phase - II	*Dev team adds level devil to pacman and also makes a server to connect with ML model. ML team implements the PPO RL model and trains the agents (ghosts) on a variety of game states*	*5-6 weeks*
Implementation Phase - Integration and Testing	*Integrate web app with ML model and test*	*2-3 weeks*

Context Diagram



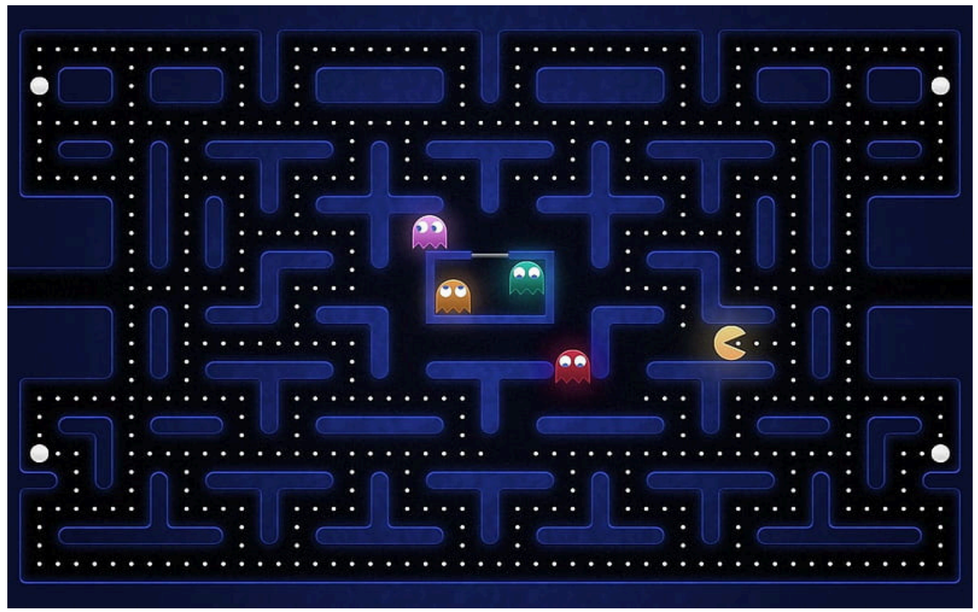
Learnings and Deliverables

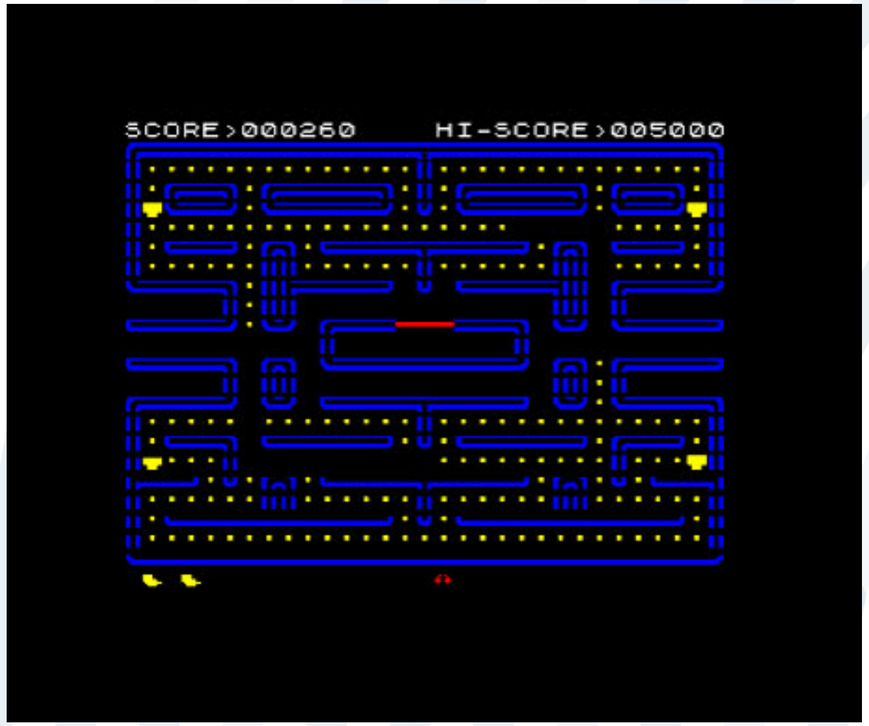
Learnings :

- Develop a great foundation in game development through use of PhaserJS.
- Develop a strong understanding of OOPs concepts through use of OOP principles.
- Application of reinforcement learning for adaptive AI behaviour.
- In hand experience with collaboration tools such as git.

Expo Deliverables :

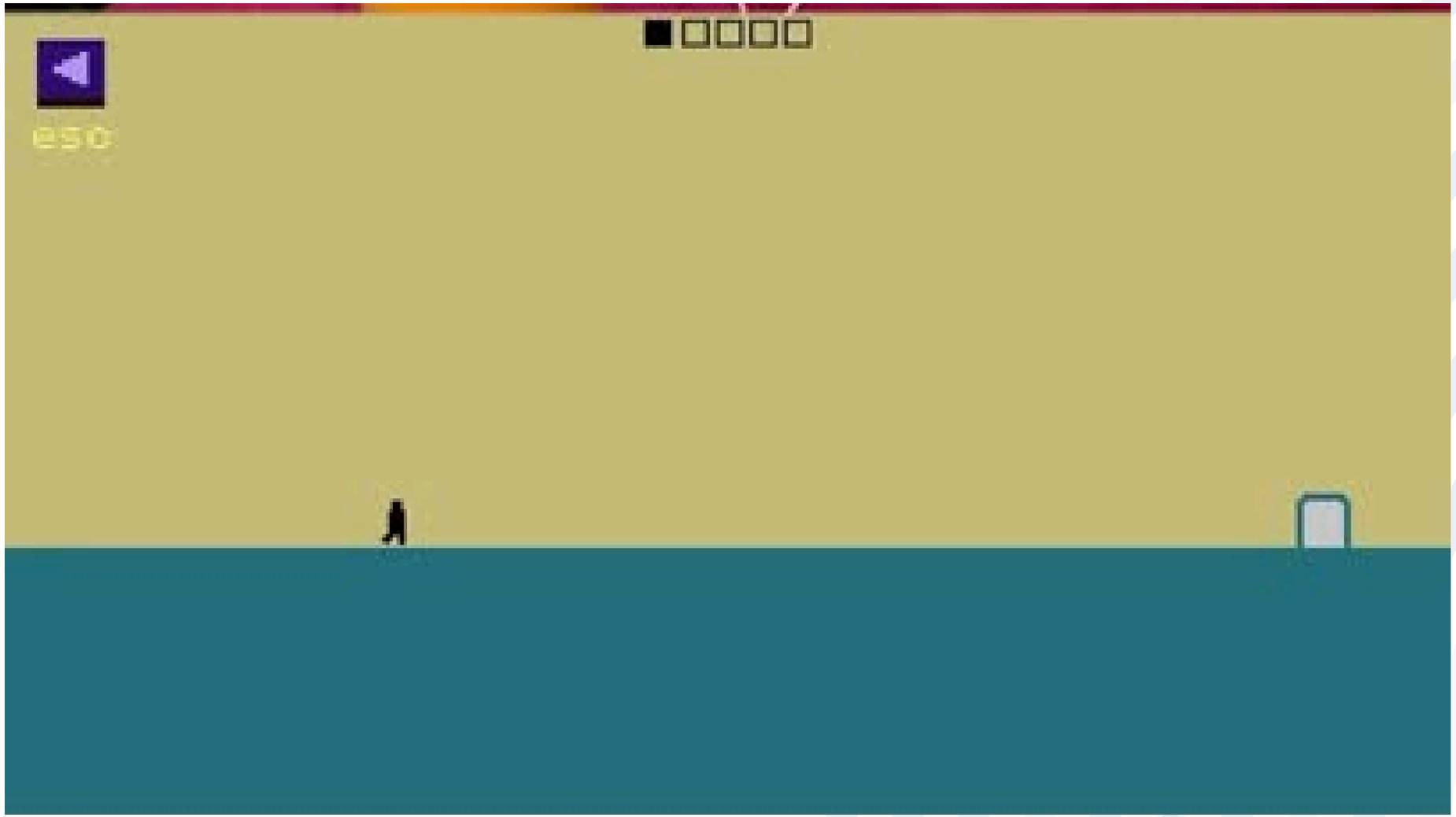
- What you will be showcasing at the expo :





Learnings and Deliverables

And now what the ragebait would look like:



Thank You