Google Developer Student Clubs

3H-ACKATHON Problems

NOTE: Feel free to use any assets, tools, or libraries available to you. There are no limitations on asset usage - focus on creativity and functionality!
There are no limitations on any technical stacks. Many of the problems are open-ended so try to work around and implement the best features!

1. Parallel Pals: A Dual-Control 2D Platformer (4)

Problem Statement:

Design and develop a 2D platformer game where players control two characters simultaneously using the same set of controls. The characters navigate different terrains with unique obstacles, requiring players to carefully coordinate their movements to overcome challenges. The game should feature asymmetrical level designs, platforming challenges, and a shared win condition where both characters must reach their respective endpoints or solve puzzles together.

Key Challenges:

Implement shared controls that move both characters simultaneously.
Create distinct environments for each character with unique physics and obstacles (e.g., one upside-down, one underwater).
Design levels that test players' timing and coordination skills.
Ensure both characters survive and complete tasks to win.

2. Mini Pokémon Go (6)

Problem Statement:

Develop a game inspired by Pokémon Go, where players’ location is updated live on a GPS Map. The game should also include an AR-based monster catching system - or any other similar use of AR to ensure an engaging platform.

Key Challenges:

Integrate Maps APIs to fetch live locations.
Allow players to interact with models of monsters and catch them.

2. EarthGuess (5)

Problem Statement:

Develop a game inspired by GeoGuessr, where players are placed in random locations on Google Street View and must guess their location on a world map. The game should include navigation mechanics, a scoring system based on guess accuracy, and time-based or attempt-limited gameplay.

Key Challenges:

Integrate Google Maps API (or GoMaps for free access) to generate randomized locations.

Allow players to explore the surroundings via Street View while restricting exact location visibility.

Implement an interactive world map for guess submissions.

Develop a scoring system that rewards proximity to the actual location.

Add time limits or restricted guessing attempts for increased difficulty.

3. Gesture-Controlled Snake (4)

Problem Statement:

Create a voice-controlled version of the classic Snake game where players use gestures to direct the snake's movements instead of traditional controls. The game should incorporate hand-gesture recognition models and maintain the core mechanics of Snake, including continuous movement, food collection, and collision-based game termination.

Key Challenges:

Integrate voice recognition technology (e.g., Web Speech API, Vosk, or Google Speech-to-Text) for real-time command processing. (BONUS -> Implement Real-Time Voice Controls instead of Gestures)

Ensure smooth gameplay with commands like "Up," "Down," "Left," and "Right".

Implement basic Snake mechanics such as food collection for score increments and collision detection for game over conditions.

Optimize voice input responsiveness to avoid delays in movement.

4. DAO: InnovateDAO – The Idea Incubator (4)

Concept:

Build a decentralized autonomous organization (DAO) that serves as an idea incubator. Participants can submit innovative project proposals, vote on them, and receive micro-grants from the DAO treasury—all automated via smart contracts.

Hackathon Scope:

- Submission & Voting: Create a minimal interface where users can post proposals and cast votes.
- Smart Contracts: Implement simple smart contracts to automate proposal registration, voting, and token-based reward distribution.
- Simulation: Use a local blockchain network (like Ganache) to simulate a few rounds of proposals and votes.

Innovation:

This DAO not only decentralizes decision-making but also incentivizes community-driven innovation, ensuring that good ideas are funded transparently.

5. Zero-Knowledge (ZK): ZK-Identity Proof (6)

Concept:

Design a dApp that leverages zero-knowledge proofs to verify user credentials (for instance, proving age or educational status) without revealing any sensitive details. Users can prove they meet a criterion without exposing the underlying data.

Hackathon Scope:

- Proof Generation: Implement a simplified ZK-proof mechanism (or simulate one) where a user "proves" they're over a certain age.
- Smart Contract Verification: A smart contract verifies the proof without learning the actual age.
- User Interface: A basic UI that walks a user through the process and displays the verification status.

Innovation:

This project highlights privacy-preserving authentication, addressing the growing demand for secure identity verification without compromising personal data.

6. BlockchainX (-)

Concept:

Integrate Blockchain with any other domain.

Innovation:

This project is purely a matter of innovation and what you can provide.

7. Physics-Based 3D Obstacle Course Game (-)

Problem Statement:

Develop a single-player 3D obstacle course game inspired by Fall Guys, where players navigate through levels filled with dynamic obstacles, physics-based movement, and time-based challenges. The character should have clumsy or ragdoll-like controls that add fun complexity to the gameplay.

Key Challenges:

Implement physics-based character movement with soft-body or ragdoll-like mechanics.

Design multiple levels with increasing difficulty and diverse obstacles.

Introduce dynamic elements such as moving platforms, rotating barriers, or collapsing structures.

8. Custom Embedding Generation (-)

Problem Statement:

Many organizations plan to use up less memory for storing semantic vector embeddings. A finance company doesn’t have use for words like “king” or “love” leading to a smaller search space and a potential for extracting less features from text for a semantic knowledge base.

Allow custom embeddings to be stored for any textual dataset of your own choice. Preferably - one with smaller size than common generators.

Key Challenge:

Develop an application that converts sentences to vector embeddings without the use of any existing embeddings converters.

9. dsc.chatbot.ai (4)

Create a chatbot with memory, file parsing and image generation capabilities.

Goals:

- Memory must be present.
- Add edge-case handling and multiple files should be parseable.
- User Interface: A UI interface.
- Deployed back-end.

10. DSCMail (5)

Create an application similar to GMail.

Goals:

- Ensure proper database management and proper mailing.
- Add spam detection.
- User Interface: A good user interface.
- Add a prediction for predicting the next word the user is about to type while sending a mail to someone.

11. C++NN (4)

Implement CNN architecture from scratch - mimic common libraries.

Goal:

Test out and improve model accuracy on any small dataset of your choice. Try to include parallelization libraries to ensure faster computations.

11. numC++ (4)

Implement much of the features of the numpy module in Python from scratch - mimic the library in C++. Add ANN-based architecture for model training.

Goal:

Include parallelization libraries to ensure faster computations.

12. CricBot (4)

Create a cricket-based AI Agent to answer any complex queries of the user. Try out web scraping or open-sourced models to answer a query.

Examples:

“Is Virat Kohli going to beat Sachin in a 1v1?”

“MS Dhoni runs against leg spinners and fast bowlers”.

Goal:

Maximise your range of complex queries that can be answered.

13. Gallery Face Scan (4)

Problem Statement:

Develop an application that can automatically classify and group images in a gallery based on unique faces. The system should be able to identify and cluster unlabeled faces, similar to a face recognition clustering model, without any prior labels. The goal is to organize the gallery by grouping images that contain the same person’s face, enabling easy access to photos of specific individuals.

Key Challenges:

Implement a face recognition algorithm that can detect and extract facial features from images.

Use unsupervised learning or other techniques to group images with similar faces.

Ensure the application can scale for large galleries and handle variations in image quality and angle.

14. Face Swap (5)

Problem Statement:

Develop an application that can detect and select a face from a given input image and seamlessly swap it with another face provided by the user. The system should accurately align facial features (eyes, nose, mouth) and ensure a natural blend with the target image, creating a realistic face-swapping effect.

Key Challenges:

Implementing face detection and alignment algorithms to accurately identify facial features.

Using image processing techniques to swap and blend the faces smoothly.

Handling varying lighting, angles, and skin tones to ensure a realistic result.