Google Developer Student Clubs
Vellore Institute of Technology

GDSC Inner Core Enrollments ‘24

Technical Task Round

General Instructions

- **These instructions are applicable to all Tech subdomains.**
- In case a domain has multiple tasks, you may attempt **either/both** task(s). Your attempt will be graded according to the features you have implemented, and their corresponding complexity and/or value.
- Create a public GitHub repo with a branch for each task you are attempting, and try to make **frequent, atomic commits** while working instead of one big commit containing everything at the end.

- Do **as much as possible**. Even if you’re not able to complete a task, make sure to have a README detailing your process and the steps you would take next given more time.
- Have fun :D
- **Deadline:** Your interview slot 🙂

Tech: Frontend

Instructions

- Try to **deploy** your finished task on Netlify/Vercel/etc.
- Feel free to use vanilla JS, TS or any framework/library of your choice like React, Vue, Svelte, etc.

Task One: Interactive Data Visualisation

Create a website that allows users to visualise and interact with data related to a specific topic, such as climate change, finance, or sports statistics. The choice of data is up to you — find a topic you have prior knowledge or interest in, or which has the nicest public APIs/data dumps available.

The website **should include** the following features:

- Data Visualization: Implement at least three different types of data visualisations (e.g., line charts, bar charts, scatter plots, maps) to display the data.
- Filtering and Sorting of data based on various criteria (as per the data chosen)
- Responsive Design on mobile and all screen sizes.

Implement the following for **brownie points** 🧁:

- Implement a sharing feature that allows users to share specific visualisations or views of the data via social media or other platforms.

Hint: use various open data APIs, or data from the stock market or orgs such as NASA.

Task Two: Yet Another Social Media Clone

Create a basic clone of your social media website of choice — Twitter, Reddit, LinkedIn, etc. Use Firebase, Supabase or a similar service for auth, DB, etc. The following features should be implemented:

- User Login via email/password or social auth.
- Creation of a new post.
- User feed/homepage with all posts.
- Basic searching/filtering.

Implement the following for **brownie points** 🧁: Follows, Likes/Upvotes, and make a PWA.

*Hint: If you aren’t too sure about Firebase, **start off with building the UI** at least, and populating it using a mock/test API.*

Tech: Fullstack

Instructions

- Feel free to attempt this task if you’ve been selected for Frontend and/or Backend, or even otherwise:)
- **Dockerization** is a must. Try to deploy the frontend on a platform like Vercel/Netlify, and if possible the backend on a platform where you have credits (a VM, or platforms like Fly).
- Frontend: Feel free to use vanilla JS, TS or any framework/library of your choice like React, Vue, Svelte, etc. Backend: use the language of your choice:)

Task: Build a clone of apps like Gartic

Reference: https://gartic.io/

The website **must allow users to**:

- Login/Signup via at least one of the following: email/password or social auth.
- Create game rooms and share the invite code/room uuid to others.
- Join game rooms via the room uuid/invite code.
- Website must also obviously implement the core gameplay of Gartic: turn-based play in which one user is shown a word, and they must draw its representation on a canvas. This canvas should be visible to other players in the room in real time, and they should be able to submit their guesses as to what the word is. Points should be allotted in a FCFS manner.
- Users should be able to see a history of games they have participated in, and the corresponding results/final scorecards.

Implement the following for **brownie points** 🧁:

- Anonymous users, who can participate in games without an account, but don't have access to their game history.
- A global leaderboard system.

Tech: Backend

Instructions:

- Use a suitable backend framework (e.g., Django, Flask, Express.js) and database (e.g., SQLite, PostgreSQL) for the development.
- Ensure secure API endpoints with appropriate authentication and authorization mechanisms.
- Implement error handling and logging for better debugging and user experience.
- Write clear documentation for API endpoints, including request/response formats and usage examples.

Task: News API Integration and User Management Backend

You are tasked with developing the backend system for a News client application The primary focus is to retrieve and manage stories, comments, and user profiles. Additionally, you are encouraged to implement optional features to enhance user experience.

- Implement endpoints to fetch and display stories from API, including categories such as top, new, and best stories.
- Develop an endpoint to fetch and display comments for a selected story.
- Create endpoints to fetch and display user profiles from your API, showcasing relevant information such as user's submissions, comments, creation date, and karma points.

Implement the following for **brownie points** 🧁:

- Implement a search functionality allowing users to search for stories or comments by keywords using appropriate API endpoints.
- Develop endpoints and functionality to allow users to bookmark their favorite stories or comments for easy access later.

Submission:

- Provide a GitHub repository with the backend codebase, including clear instructions on setting up and running the application.
- Include API documentation detailing endpoints, request/response formats, and any additional features implemented.
- Optionally, provide a demo or walkthrough video showcasing the implemented features and functionalities.

Tech: App

Instructions

- Unless specified otherwise, feel free to use whichever language or framework you feel most comfortable with to build the mobile application.

FLUTTER/ANDROID: Build a Hacker News client app.

Create a mobile application that serves as a client for Hacker News, utilising the official Hacker News API. This application should provide a seamless and efficient way for users to interact with the content on Hacker News, including reading stories, comments, and viewing user profiles. The following features are must-haves:

- API: https://github.com/HackerNews/API
- Story Listing: Display a list of stories from Hacker News, including categories such as top, new, and best stories. Each story item should show the title, author, points, and number of comments.
- Story Details: Allow users to tap on a story to view its details. Display the story's URL in an in-app web view or browser. Show comments in a threaded view, with proper indentation or other visual cues to depict the hierarchy.
- User Profiles: Implement functionality to view Hacker News user profiles by tapping on the user's name. Show relevant information such as user's submissions, comments, creation date, and karma points.

Implement the following for **brownie points** 🧁:

- Search Functionality
- Bookmarking/Favourites
- Offline Reading

IOS APP DEVELOPMENT: Be a Swifty!

Create an iOS app to browse all Taylor Swift Songs using iTunes API. The user should be able to like songs which should be shown separately.

- App should be built using Swift and SwiftUI.
- It should use the Observation Framework (https://developer.apple.com/documentation/observation).
- It should use Swift Concurrency a.k.a. async/await.(https://developer.apple.com/videos/play/wwdc2021/10132/).

- Should store local data using SwiftData (https://developer.apple.com/xcode/swiftdata/).

Implement the following for **brownie points** 🧁:

- Caching of song data would be a plus point.
- Ability to search for more artists would also be a good feature to have.
- Liked songs could be stored locally but having a Backend/Firebase/Supabase/Appwrite to store it would be a plus point.

Tech: Game

Instructions

- This is a free-for-all task. Go for it 😀

Task: Build a game with the theme: *You only get one.*

- The game can be of any genre, 2D or 3D and can use any engine or no engine at all.
- You can use prebuilt assets but cannot submit a previously made game
- The game should have atleast a windows or web version.
- The game **must** incorporate the theme in a direct or indirect way.
- The game should ideally be made available via itch.io or some other similar platform.

Tech: ML

Task: Custom text query system with RAG (Retrieval-Augmented Generation)

Implement RAG with an open source LLM with Llamaindex. Given a directory of various text documents, create a chat application for the context.

- Pick 3-5 of your favorite movie scripts to work with as your data. They can be in either text or pdf form. You can easily find movie script data available. If you don’t want to use movie scripts, you can use some short stories.
- Start with a cool open-source language model (LLM) and get it tuned up.
- Use LlamaIndex to organize your scripts – this makes finding the right bits super fast. Implement a re ranker if you can.
- Make sure your model understands even tricky questions about the scripts.
- The system should be able to ask basic questions about the context (in this case the movie scripts)
- If you are unable to find an open source LLM, feel free to use the Gemini API or the OpenAI API. But using an open source model for re-ranking and embeddings is a must.
- Make sure your jupyter notebook (or colab notebook, or kaggle notebook, doesn’t really matter) is well documented.

Brownie points:

- Can your system give you short summaries of the script parts it finds? Either use the map-reduce or the refine technique to generate summaries of text provided.

Task: Credit Card Fraud Detection with Deep Learning

Dataset Link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

- Process Data: Prepare and clean data for analysis.
- Enhance Data (Optional): Create new features and improve existing ones.
- Build Deep Learning Model: Experiment with different architectures and fine-tune parameters.
- Train and Validate: Split data strategically and address class imbalance.
- Evaluate Performance: Focus on AUC-ROC, with additional metrics for better understanding.

Tech: Python

Task: CLI Kanban Board

A kanban board is used to track the status and priorities of various subtasks to be done in a software project. Implement the following features:

- Implement a basic kanban board to be used from a command line interface with 3 statuses (Todo, In progress, and Done)
- Addition and removal of tasks to each, and moving them between statuses
- Assignees and Reporters
- Add more status fields to the Kanban Board
- Support for multiple boards for different projects
- Allow priorities and sorting by priority (High, Medium, Low)

Brownie points:

- Usage of a mouse within the CLI to have drag and drop functionality.

Please make use of an appropriate storage solution to persist tasks, boards, and statuses across runs.

Task: Solitaire!

Build your own implementation of the card game "solitaire" in python, without using external libraries.

Brownie points:

- Hook it up to Gemini to allow a computer to play an automated game, or to distribute the cards to ensure a winning strategy.