**BITS UNPLUGGED 2.0**

* Overview
  + Develop an innovative platform where users can enhance problem-solving skills without coding.
  + Offer a seamless and user-friendly interface for engaging with data structure and algorithmic challenges.
  + Facilitate conceptual understanding by focusing on problem-solving approaches rather than coding syntax.
* Motivation
  + Our motivation stems from the need to provide a unique learning experience that focuses on conceptual understanding rather than coding syntax.
  + Traditional coding platforms often emphasize syntax over core concepts, hindering true comprehension for many learners.
  + **Real Life Applications**
    - Traditionally Google form is used to take online tests at universities. Where MCQ and Short questions can be set.
    - Teachers can arrange online tests on Data structures and Algorithm theory courses on our site. In this way, students don’t need to write down on paper and upload images in Google form :)
    - One big problem of online tests is plagiarism. Because generally the question is the same for all students. And it is hard for the teachers to create different sets of questions. Our website can generate different canvases for different participants. Like you just set the problem statement “Find the minimum spanning tree” and our website will generate random graphs for each participant.
* Core Functional Features and Scope of the project (Features are ordered based on priority. Scope may decrease based on our progress)
  + Problem Library:
    - Curated collection of diverse data structure and algorithm problems.
    - Categorized by difficulty levels and concepts.
    - An interactive interface will be provided to the problem setters.
  + Interactive Problem Solving
    - Users can solve problems through an intuitive, interactive interface.
    - Again, No coding is needed :)
  + Realtime Contest
  + Problem analytics
    - A detailed analytics, including the percentage of users who successfully solve each problem. This feature offers valuable insights into problem difficulty and allows users to benchmark their performance against the community
  + Learning Paths:
    - Guided learning journeys based on user proficiency and interest.
    - Progress tracking and personalized recommendations.
    - A robust progress tracking system, allowing users to monitor their advancements, identify strengths and weaknesses, and tailor their learning journey accordingly.
  + Gamification:
    - Reward system for accomplishments, encouraging consistent engagement.
    - Rating system, Leaderboards and badges for recognition.
  + Community Engagement:
    - Discussion forums for each problem to encourage collaborative learning.
    - User profiles showcasing achievements and contributions.
* Unique Features and Comparison
  + Bits Unplugged is quite similar to different competitive programming platforms like Leet~~Code~~, ~~Code~~forces, ~~Code~~chef, Top~~Coder~~ etc. I guess you got the difference :)
  + We are just unplugging the coding(bits) part from problem solving.
* **Summary**
  + First, we need problems. So, we will create a problem setting interface.
  + Development of an interactive and intuitive interface for users to solve problems without coding.
  + Now we need to measure the difficulty of each problem. For that we will gather analytics for each problem.
  + Then we need to guide the users to a smooth learning journey. That’s why we need a recommendation system.
  + To ensure consistent engagement of the users, we will provide a reward system for accomplishments. Like a rating of each user and a leaderboard to to benchmark their performance against the community
  + Lastly we need to build a supportive community. For that we will introduce discussion forums for each problem.
* Formal Summary
  + **Problem Setting Interface:** Done
    - Creation of a user-friendly interface for setting up data structure and algorithmic problems.
  + **Problem Solving Interface:** Done
    - Development of an interactive and intuitive interface for users to solve problems without coding.
  + **Difficulty Measurement:** Souvik
    - Implementation of an analytics system to measure the difficulty of each problem.
  + **Recommendation System:** Sayem
    - Introduction of a guidance system to recommend a smooth learning journey for users based on their proficiency and interests.
    - Also Peer recommendation
  + **Reward System:** Dihan
    - Implementation of a reward system to ensure consistent user engagement.
    - Inclusion of user ratings and a leaderboard for benchmarking performance within the community.
    - Enhance user profiles with detailed statistics, achievements, and a portfolio of problems solved
  + **~~Supportive Community:~~** ~~Optional~~
    - ~~Creation of discussion forums for each problem to encourage collaboration and knowledge sharing among users.~~
* It forms a well-rounded platform that not only focuses on the technical aspects of problem-solving but also considers the user experience, motivation, and community engagement.
* Some additional suggested features by ChatGPT
  + **User Feedback Mechanism:**
    - Implement a system for users to provide feedback on problems, solutions, and the overall platform.
  + **Collaborative Problem Setting:**
    - Introduce features that allow users to contribute and create problems collaboratively.
  + **Integration with Educational Institutions:**
    - Explore possibilities for integration with educational institutions, such as partnerships or tailored solutions for classrooms.
  + **Offline Mode:**
    - Consider developing an offline mode for users who may not always have internet access.
  + **User Profiles:**
    - Enhance user profiles with detailed statistics, achievements, and a portfolio of problems solved.
* Problem Library
  + Sorting Algorithms:
    - Drag the steps of a sorting algorithm (e.g., Bubble Sort) into the correct order.
  + Search Algorithms:
    - Arrange the steps of a search algorithm (e.g., Binary Search) in the correct sequence.
  + Graph Algorithms:
    - Drag nodes and edges to construct a graph and visualize the output of a graph traversal algorithm.
  + Dynamic Programming:
    - Arrange blocks representing subproblems and solutions to visualize the process of dynamic programming.
  + Recursion:
    - Arrange blocks of recursive function calls to solve a problem, demonstrating the call stack visually.
  + Tree Structures:
    - Drag nodes to construct a binary tree and perform tree traversal operations (e.g., inorder, preorder).
  + Hashing:
    - Arrange elements into buckets to demonstrate the hash mapping process and handle collisions visually.
  + Greedy Algorithms:
    - Drag items with associated weights and values to optimize a solution using a greedy algorithm (e.g., Knapsack problem).
  + String Manipulation:
    - Arrange characters or substrings to perform string operations like palindrome checking or pattern matching.
  + Backtracking:
    - Drag elements to construct a solution space and visually explore backtracking to find a solution.
  + Heap Operations:
    - Arrange elements in a visual heap structure and perform heap operations like insertion and deletion.
  + Bit Manipulation:
    - Drag bits to demonstrate bitwise operations like AND, OR, XOR, and left/right shifts.
  + Mathematical Algorithms:
    - Arrange mathematical expressions or numbers to solve problems like computing factorials or finding prime numbers.
  + Concurrency and Parallelism:
    - Visualize parallel processes by arranging blocks representing concurrent tasks and demonstrating their execution.