Onsite tasks

Frontend (150 pts min)

Easy(50 pts)

- Create a mail system implementing basic mail features like compose, send , and all emails. Users should be able to create an email, perform basic formatting, and save it in local storage.
- Create a signup/login page where users signup using a username and password. Check if the password is strong or not. If it is not strong, suggest a strong password. A strong password contains uppercase, and lowercase letters, digits and special characters.
- Create an application to generate posters; users should be able to add pictures(one or more), add some text and download the final poster.

Medium(100 pts)

- Create a website that analyses patterns in data based on csv files .Create graphs or charts from these files (reference - chart.js) . E.g. user provides a file containing names, gender,age , academic level of a colony and displays bar graphs grouping people of same age gender etc.
- Create a game where the user can move forward or backwards. In the path, there are occasional bombing and treasure boxes fall. If a bomb falls on user he dies, if treasure boxes fall on him, his points are increased.
- Create an application which allows users to search Twitter accounts from their user names , visit profiles and see their followers and following.

Hard(150 pts)

- Create a fishing game using HTML5 Canvas.
- Create a fishing game where the player has to catch as many fish as possible within a given time limit.
- The player will control a fishing hook using the mouse or touch screen to catch the fish.
- Each fish caught will add points to the score, and the game will end when the time limit is reached.
- Display the score and remaining time during the game.
- The game can be made more challenging by adding obstacles such as rocks or seaweed that the player needs to avoid catching with the fishing hook.

Backend (150 pts min)

Easy(50 pts)

- Create an application that generates a QR code for a given URL or text input, and allows users to download or print the generated code.

- Create a website where users can create and manage their to-do lists. They can add new tasks, mark them as completed, update details, and delete any task. Implement a proper authentication and authorization system.
- Create an API that generates random quotes or jokes, and returns them in a JSON format when requested. Allow users to specify the category or type of quotes/jokes they want to receive.

Medium(100 pts):

- Develop a 2-Player Game similar to skribbl.io, but instead of drawing, the first player has to use a series of emojis to create a clue related to the word generated by the site. The second player then has to guess the word based on the emoji clue.
- Create a video streaming website specifically for musicians and music enthusiasts. The website should allow users to upload and share their own music videos or covers, as well as watch and rate the videos of other musicians. The site should also include features such as the ability to follow specific musicians, create playlists, and connect with other music lovers for collaborations or networking opportunities. Additionally, the site should have tools and resources available for musicians to learn and improve their craft, such as tutorials, workshops, and feedback from experienced musicians.
- Develop an application to create tables in a relational database and also insert, update, and delete data in those tables.

Hard(150 pts):

- Develop a Discord Bot that can help users learn a new language. The bot should be able to perform the following tasks:

    - ❖ Provide users with daily vocabulary words, with the option to choose a difficulty level.
    - ❖ Offer users grammar tips and explanations of commonly used phrases.
    - ❖ Allow users to practice their language skills with quizzes and interactive exercises.
    - ❖ Provide users with translations of words and phrases they input.
    - ❖ Give users feedback on their language skills and progress over time, with the ability to set goals and track their improvement.

    The bot should also have the ability to customize language learning plans for individual users based on their interests and skill level.

- Create a site monitor that checks a specified URL at set intervals and sends an email notification when the site goes down or comes back up. The user should input the URL, time interval, and email address. The email should only be sent once for each status (i.e., when the site is down after being up or when the site is up after being down) change.

- Implement a Node.js server clustering solution that takes advantage of multi-core systems by splitting a single process into multiple processes or workers. The server should be able to scale up applications such as matrix multiplication and the Fibonacci sequence by distributing the workload across multiple processes. By default, Node.js servers only run in a single thread, which limits their performance on multi-core systems. The clustering solution should enable the server to utilize all available cores for optimal performance.