SDE Assignment

Deadline: 25th May ’25

🔍Problem Statement

In many online learning platforms, progress is often marked as “complete” when a video finishes playing. However, this isn’t enough to know if the student truly paid attention. A student might skip around or watch the same section more than once, which doesn’t show real progress. Our goal is to build a system that saves and updates the user’s progress based on the unique parts of the video they have watched

🎯Objective

Create a tool that accurately tracks how much of a lecture video a user has really watched. Instead of simply recording whether the video was played to the end, your solution must only count progress when new parts of the video are seen..

🌟 What We Want to Achieve

- **Track Real Progress:**
    - **Measure Unique Viewing:**
      Only add to the progress when the user watches parts of the video they haven’t seen before. For example, if a student has already watched 0–20 seconds and 50–60 seconds, watching 10–20 seconds again should not increase the overall progress.
    - **Prevent Skipping:**
      If a user jumps ahead (fast forwards) to the end without watching the middle parts, that skipped time should not be counted as progress.
- **Save and Resume:**
    - **Persistent Progress:**
      Save the user’s watching details, such as the unique intervals they have viewed,

so that when they return, the system knows exactly where they left off.

- **Seamless Experience:**
  When the user logs back in, the video player resumes at the correct position, and the recorded progress (as a percentage) reflects only the new content that has been seen.

.

🛠️What You Need to Build

User Interface

- **Lecture Video Page:**
  - The user can play the lecture video.
  - The progress is shown as a percentage based on the unique parts watched.
  - The player automatically resumes from the last saved position when the user returns
- **Page Indicator**

  Display a visual progress bar (or a simple percentage readout) that updates only when new seconds of the video are watched.

Functionality (Logic for Tracking Progress)

- **Track Watching Intervals:**
  - Record the start and end times of every segment the user watches.
  - Ensure that overlapping segments or parts that have already been seen are not counted multiple times.
- **Calculate Unique Progress:**
  - Merge the recorded intervals to figure out the total number of unique seconds watched.
  - Convert this total into a percentage based on the overall length of the video
- **Handle Edge Cases:**
  - If the user jumps ahead (fast forwards) and then watches parts that were not seen before, only count the new segments.

- If the user re-watches a section, ensure that the progress does not increase further for that section.

Data Persistence

- **Save User Progress:**
  - Storing details like which intervals have been viewed.
  - Keeping a record of the overall percentage progress.
- **Resume Correctly:**

  When the user returns to the lecture, the video should start from the last position they watched, and the current progress should still be visible.

💡 What We're Looking For

- **Your Thinking and Approach:**

  We want to see how you break down the problem. Explain your approach for:

  - Capturing and merging the unique watched intervals.
  - Calculating the progress without counting repeated views.
  - Managing data persistence to resume progress correctly.
- **Clean and Simple Code:**

  Ensure that your solution is easy to understand, well-documented, and modular so that each part (tracking, calculating, and saving progress) can be reviewed on its own.

- **User Experience:**

  Your interface should be simple and intuitive for the user, with a clear display of their progress.

🎁 Deliverables

- **Source Code Repository:**

  Provide a repository (GitHub, GitLab, etc.) that includes:

    - The code for the frontend and backend.
    - Clear documentation (a README file) on how to set up and run your application.
    - Explanations of your design decisions and how your code works.
- **Design Documentation:**

  A short write-up (can be part of the README) explaining:

    - How you tracked the watched intervals.
    - How you merged intervals to calculate unique progress.
    - Any challenges you encountered and how you solved them.
- **Demo (Optional):**

  A brief video or a set of screenshots showing your application in action, demonstrating how progress is tracked and resumed.

- **Live Link:**

  A working URL of your hosted application so we can access the live demo.

Submission Instructions

- **Repository**: Provide access to your Git repository (e.g., GitHub). **Make sure to do proper commits . Multiple Small Commits are Always better than single large commit.**
- **Branching**: Use meaningful commit messages and consider using branches for different features.

Submission Link : https://forms.gle/6z9titKchv4TFM2i7

Deadline : 25th May, 11pm.