**Project Vision and Background (Introduction)**

* **Purpose**: Clearly articulate the problem you're solving with your SPA project.
* **Key Points**:
  + **What is the project?** Describe what your single-page web application does.
  + **Why is it important?** Explain why your project matters, especially in terms of its relevance to your field of study. Discuss the problem or need it addresses. For example, if your SPA is a task management tool, explain how it can improve productivity or efficiency.
  + **Target audience**: Who is the end-user? Why would they find your solution valuable?
* **Tip**: Keep this section concise but informative. It’s the context for your entire project, so aim to explain it in a way that engages the viewer.

**2. Your Project Plan and Sprints Carried Out to Date**

* **Purpose**: Show how you’ve planned your project development so far.
* **Key Points**:
  + **Project Plan**: Outline the phases of your project development. How did you plan your sprints?
  + **Sprints**: Briefly summarize the work you’ve completed in the initial sprints. What tasks were planned, and what have you accomplished? Be specific about deliverables you’ve completed in each sprint (e.g., project design, homepage, first feature implementation).
* **Tip**: Mention any adjustments made to your plan based on the feedback you received or challenges you encountered.

**3. UML Models to Date**

* **Purpose**: Show your technical understanding of the project and how you've structured it.
* **Key Points**:
  + **UML Diagrams**: Share your **Use Case Diagram**, **Class Diagram**, and any **Interaction Diagrams** you have worked on so far.
  + **Explanation**: Briefly explain the key elements of the diagrams. For example, for the Use Case Diagram, show how the user interacts with the system.
* **Tip**: Make sure your UML diagrams are clear and easy to follow. Don’t just show the diagrams – describe their purpose and significance.

**4. An Initial Prototype Running (Demonstration)**

* **Purpose**: Show the prototype of your SPA, if possible, to give viewers a sense of what the project looks like in action.
* **Key Points**:
  + **Prototype demonstration**: Showcase your homepage or any other functional aspect of your web application that you’ve implemented so far.
  + **Tech stack**: Briefly mention the technologies you're using (HTML, CSS, JavaScript, etc.).
* **Tip**: If your prototype is still in progress, show what you have completed (e.g., the layout, homepage structure, some basic functionality). If there are bugs, acknowledge them and mention your plans to fix them.

**5. A Succinct Description of Issues/Challenges Faced**

* **Purpose**: Show you can identify problems and think about solutions.
* **Key Points**:
  + **Challenges faced**: Discuss any technical or non-technical problems you’ve encountered so far (e.g., debugging issues, difficulty integrating certain features, designing the UI).
  + **Solutions**: How did you address or plan to address those challenges? Show that you are actively problem-solving.
* **Tip**: Be honest but concise. Don’t dwell on minor challenges; focus on significant ones that might impact the project’s progress.

**Overall Tips for Creating a Successful Interim Video:**

* **Time Management**: Your video should last no more than 4 minutes. Stay within the time limit by being concise, focusing only on the key points for each section.
* **Visuals**: Use slides, diagrams, or screen recordings to illustrate your points. Make sure everything is clear and easy to read.
* **Narration**: Speak clearly and confidently. Use a script if it helps you stay focused, but don’t read directly from it – sound natural.
* **Clarity**: Avoid technical jargon unless you explain it, and make sure the video is understandable even for someone unfamiliar with your specific project.

**Video Structure (Example Timeline):**

1. **0:00 – 0:30**: Project Vision and Background.
2. **0:30 – 1:30**: Project Plan and Sprint Progress.
3. **1:30 – 2:30**: UML Models.
4. **2:30 – 3:30**: Prototype Demonstration.
5. **3:30 – 4:00**: Challenges and Solutions

**Phase 1: Planning and Research**

1. Research the main features of a task management web application.
2. Identify the target audience (students, professionals, etc.).
3. Define core functionality (task creation, due dates, reminders, task prioritization, etc.).
4. Research similar applications (e.g., Todoist, Trello, Asana) for inspiration.
5. Sketch initial wireframes and layout ideas.

**Phase 2: Design & Prototyping**

1. Create wireframes for the homepage, task list page, and task creation modal.
2. Design the user interface (UI) for adding tasks, editing tasks, and marking them as completed.
3. Choose a color scheme and typography for the website.
4. Create a prototype of the website in Figma or Adobe XD.
5. Get feedback on the prototype from peers or instructors.

**Phase 3: Backend and Frontend Setup**

1. Set up a GitHub repository for version control.
2. Initialize the project with HTML, CSS, and JavaScript files.
3. Create the homepage (header, footer, and basic layout).
4. Set up basic routes for viewing tasks and adding tasks (e.g., /tasks, /add-task).
5. Implement functionality to add tasks (using JavaScript and storing them in memory or local storage).
6. Implement functionality to mark tasks as complete or delete them.

**Phase 4: Task Management Features**

1. Add due dates to tasks.
2. Implement task prioritization (e.g., High, Medium, Low).
3. Set up notifications or reminders for due tasks (using JavaScript).
4. Allow users to filter tasks based on priority or due date.
5. Implement a search bar to filter tasks.

**Phase 5: Testing and Debugging**

1. Test the task creation, editing, and deletion process.
2. Test user interactions (clicking buttons, marking tasks as complete).
3. Test for mobile responsiveness (using CSS media queries).
4. Debug any issues with task storage or layout problems.

**Phase 6: Final Touches & Deployment**

1. Add error handling for tasks (e.g., empty input fields when adding tasks).
2. Polish the UI (animations, hover effects, etc.).
3. Optimize the website for performance (e.g., minifying CSS, JavaScript).
4. Deploy the website to GitHub Pages or Netlify.
5. Write a README.md file for the GitHub repository.
6. Gather feedback and make final adjustments.

--------------------------------------------------------------------------------------------------------------Resources

Sprints: Trello

UML: lucidchart