Lessons Video Script (4-5 min)

Notes on what we want to talk about in our retro video. Put notes below:

- Doing retro reviews every week is very helpful on finding the problems
- Having a regular meeting time that everyone can attend is important for pushing the project forward
- Instead of trying to make everything perfect in the beginning, have a ugly demo running is more important.
- Everything can be iterative, such as testing, CI/CD, even meeting organizations. We can always improve.
- We realized we need to be more directed and focused during our meetings, instead of trying to figure out how to solve problems in the meeting.
- Modularizing/separating into the different webpages made merging/parallel work much easier.
- The massive importance of iterative and incremental development.
- A lack of clarity about who was working on what caused confusion when managing code changes and task assignments. Using GitHub issues helped us track what was assigned and brought more clarity to our workflow.
- We planned regular group coding sessions during the mid sprint to ensure we're aligned.
- We all contributed based on our strengths, which allowed us to maintain steady progress.

Summary of above notes:

Lessons	How did it help us?	Consequences
Weekly retrospectives	Helped us identify and resolve problems early, improving project flow.	Time-consuming if not kept focused and concise.
Regular meeting time	Ensured consistency and accountability for progress.	Scheduling conflicts sometimes made attendance challenging for all team members.
"Ugly demo" over perfection	Allowed us to test ideas quickly and iterate effectively.	Risk of technical debt if early "ugly" implementations aren't addressed later.
Iterative development for all aspects	Enabled continuous improvement, from testing to meeting formats.	Can lead to frequent disruptions if changes aren't well-planned.
Focused meeting goals	Made discussions more productive and actionable.	Required extra preparation time before meetings.

Modular work structure	Simplified merging and enabled parallel development.	Can create dependencies between modules if communication isn't clear.
Using GitHub for task tracking	Provided clarity on responsibilities and task status.	Learning curve for those unfamiliar with GitHub's features.
Group coding sessions mid-sprint	Kept everyone aligned and allowed for real-time problem-solving.	Could be less productive if not well-organized or focused.
Contributing based on strengths	Leveraged individual skills to maintain steady progress.	Risk of knowledge silos if team members only stick to their strong areas.