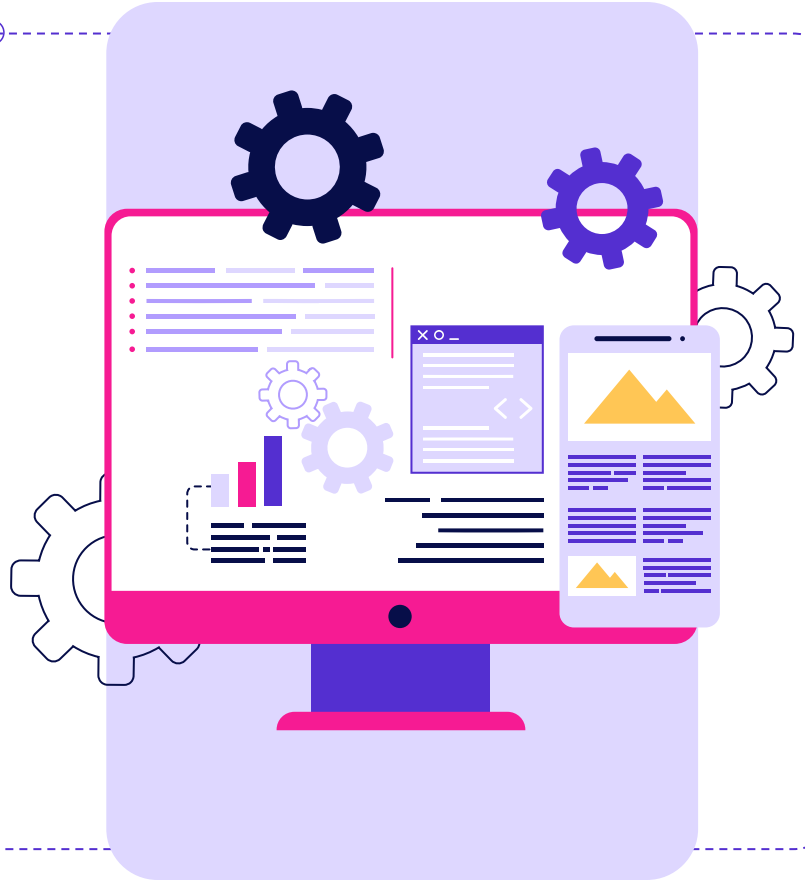
Agile Creative Labs technical work

In this presentation, I will talk about my internship work in Agile Creative Labs.

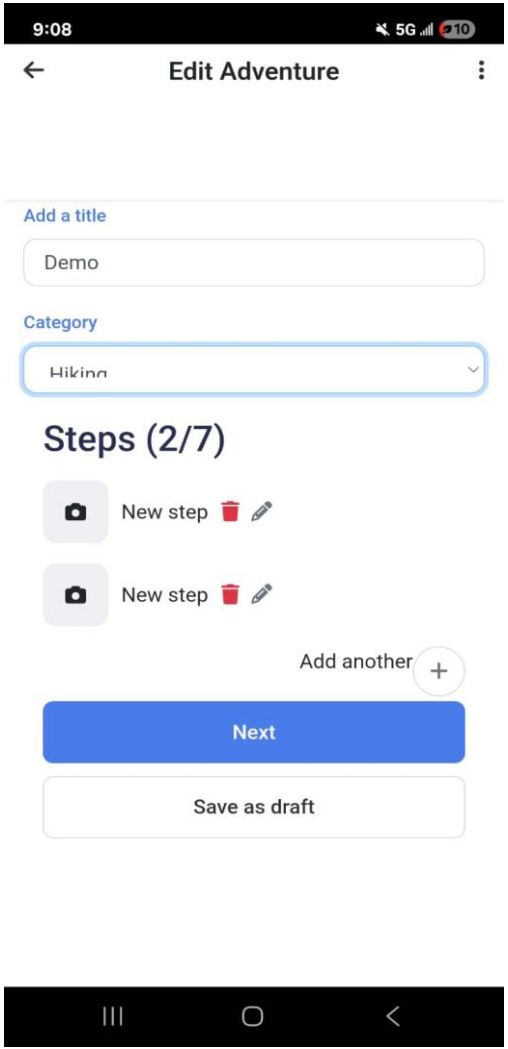


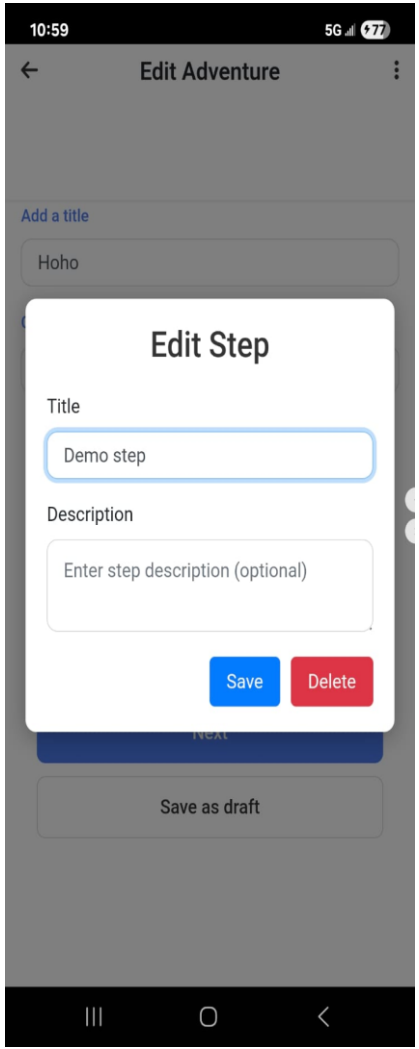
CONTENTS OF PRESENTATION

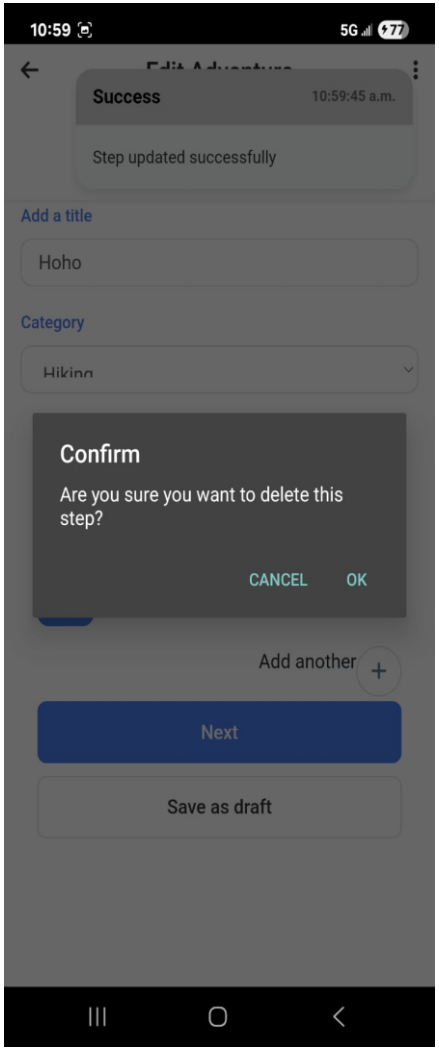
- **Content:** My work during my internship had 2 aspects. Planning and implementation. I drew diagrams and then applied the solution.
- **Tech stack:** HTML/CSS/JavaScript that are wrapped in a Cordova container.
- **Summary:** During my internship, I contributed to the development of a hybrid mobile application for a social media platform designed for children with autism.

01

STEPS ADDITION FOR ADVENTURES







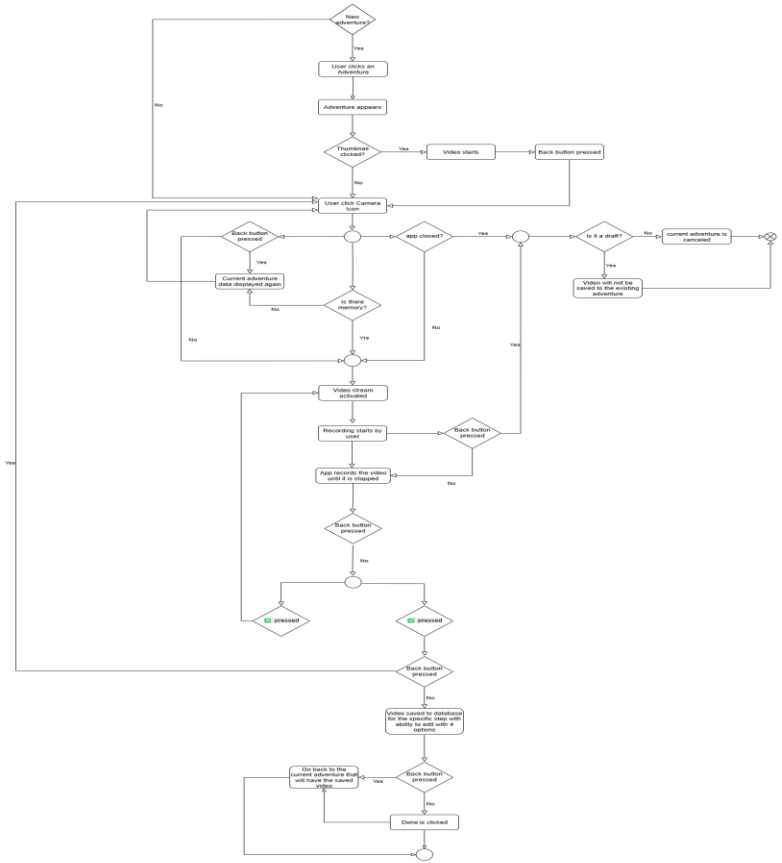
STEPS FOR ADVENTURES

I created the steps addition in this screen with the ability to edit and delete each step.

02

Video Recording Feature

DIAGRAM OF VIDEO FUNCTIONALITY

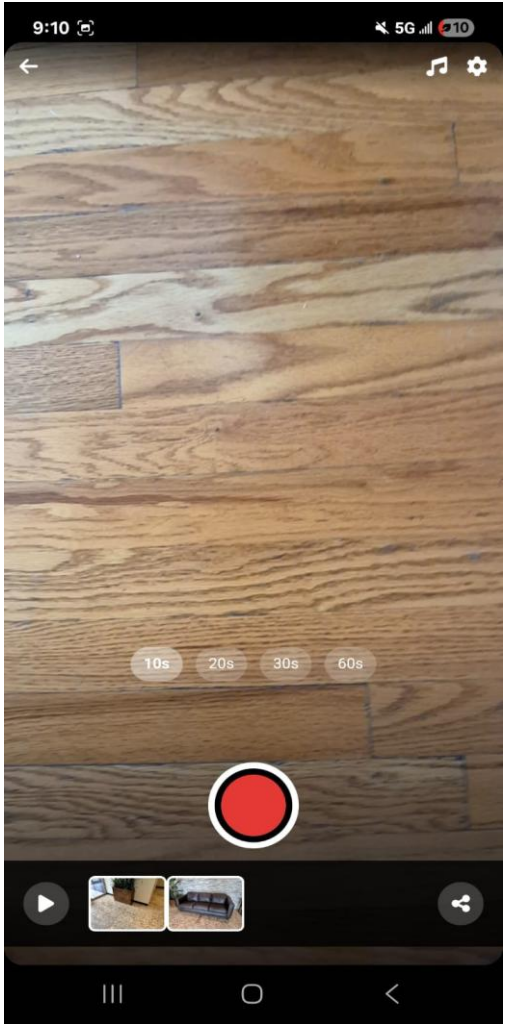


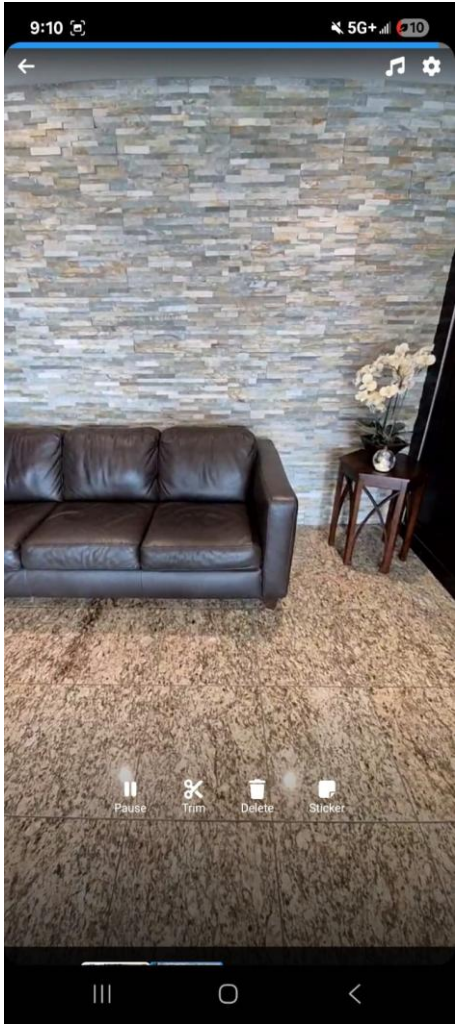
1. Database Changes
Add a video_id field to the steps table (nullable)
Each step can have 0 or 1 video
When recording a new video, it replaces the old one

2. Step Display Logic
No video: Show camera icon (record new video)
Has video: Show play button (view existing video)
Click play button: Show video player with "re-record" option

3. User Flow
Step without video: Camera icon → Record → Save to step
Step with video: Play button → View video → "Re-record" button → Replace old video
Always one video per step

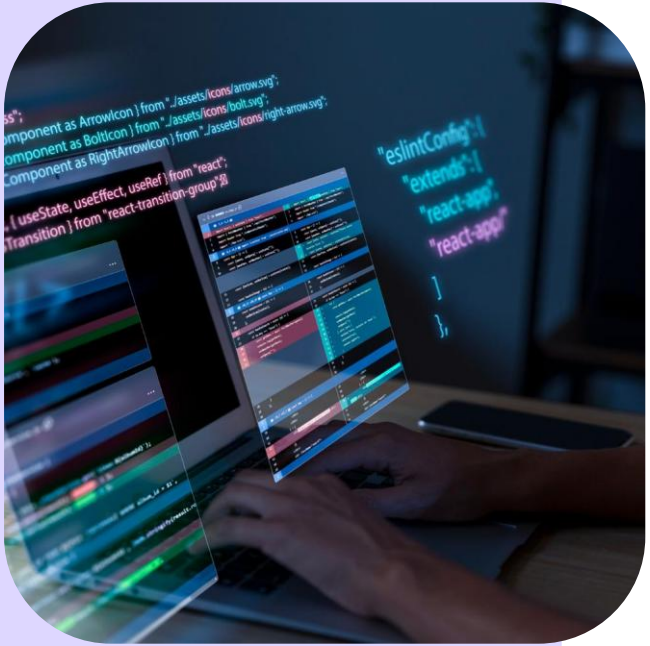
4. Implementation Steps
Update database schema (add video_id to steps table)
Add step_video table.
Modify StepManager.js to check video status
Update AdvView.js to show play button





VIDEO WITH CLIPS

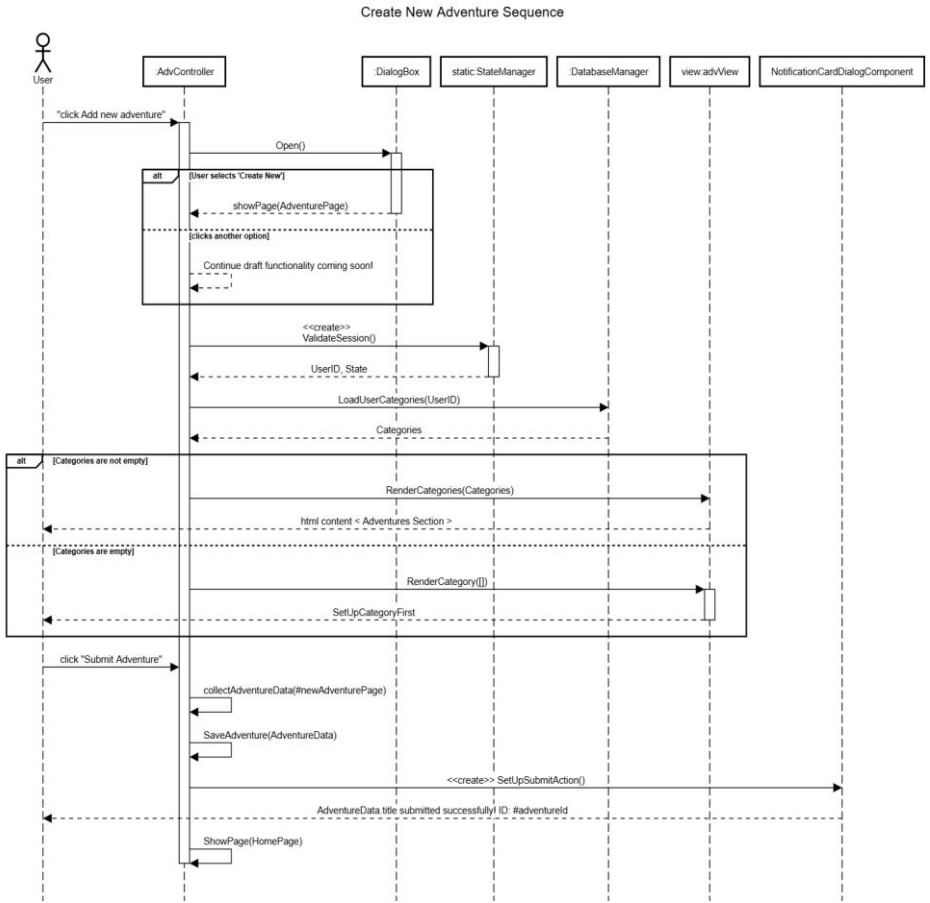
I expanded the video recording feature by having clips that work dynamically with each video. Refrences of the clips are saved in the database and the videos are in the phone’s gallery. The videos could be retreived for each step uniquelly. The user can record and edit old and new clips at any point.



03

Create new adventure

DIAGRAM OF NEW ADVENTURE



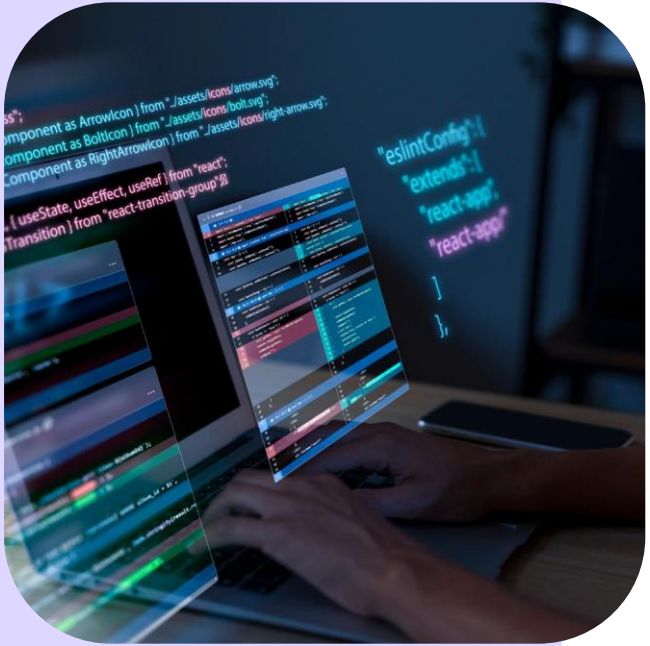
Task: I looked through the code to get a better idea of how everything works under the hood.

What I did:

- Reviewed the component, controller, and service code tied to this page.

- Traced the key logic flows: how inputs render, how form state is handled, and how the draft mechanism works.

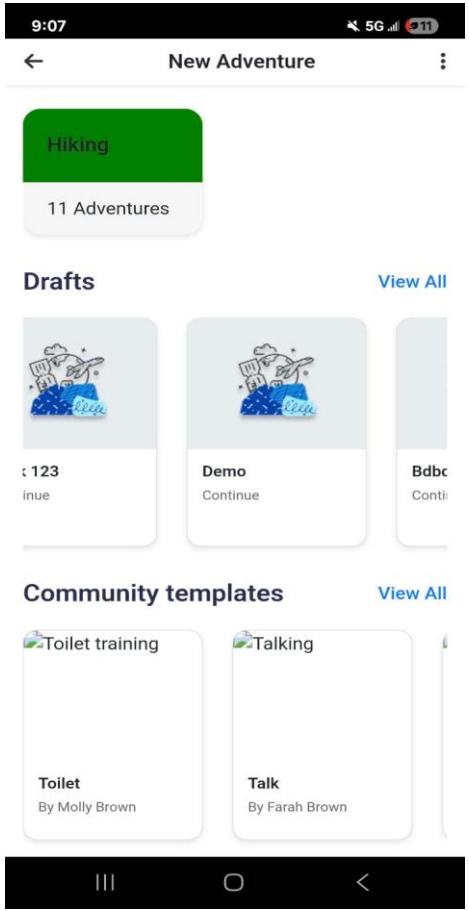
- Drew sequence diagram from my findings

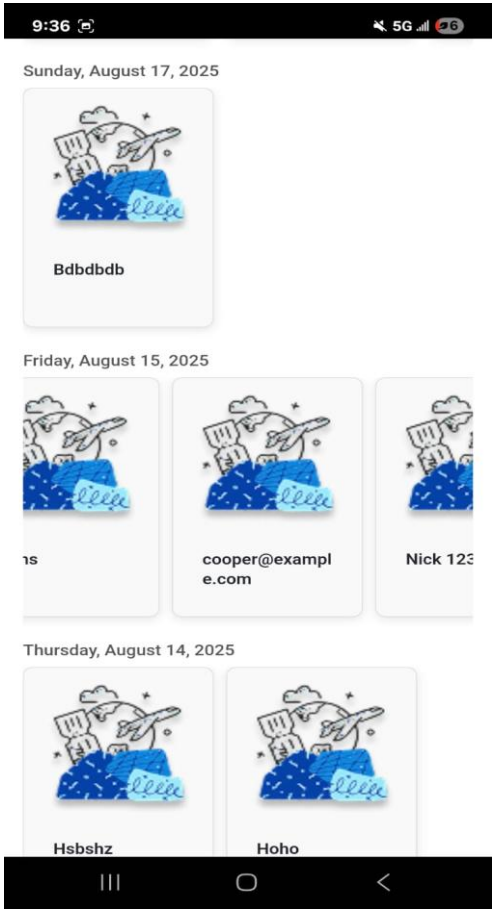

Component as ArrowIcon } from "../assets/icons/arrow.svg";
Component as BoltIcon } from "../assets/icons/bolt.svg";
Component as RightArrowIcon } from "../assets/icons/right-arrow.svg";
{ useState, useEffect, useRef } from "react";
Transition } from "react-transition-group";
"eslintConfig": {
"extends": [
"react-app",
"react-app/jest"
]
},

04

DRAFTS

ADVENTURES AS DRAFTS





🧩 Task 1: Create New Draft

Add a “New Draft” button.
Initialize and store a new empty draft.
Open it in the editor for the user.

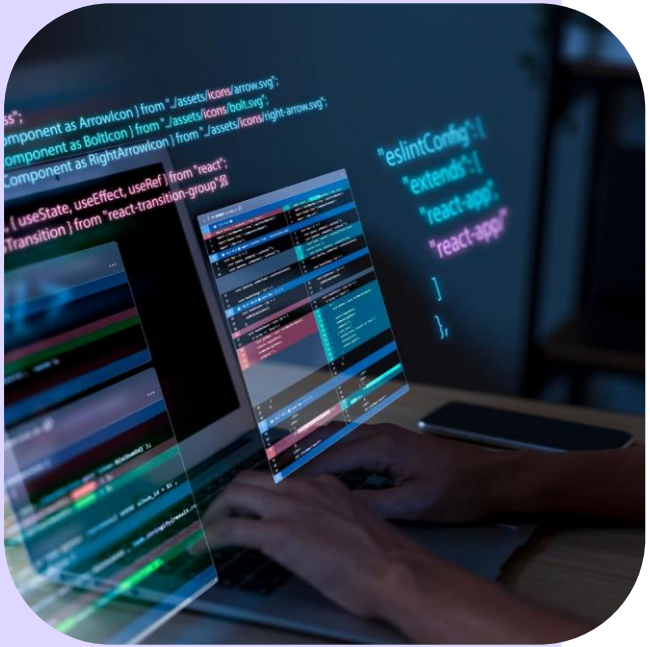
🧩 Task 2: Save Draft

Add a “Save Draft” button.
Capture current draft data and store it.
Include metadata (ID, timestamp) and show feedback.

🧩 Task 3: Group Drafts by Date

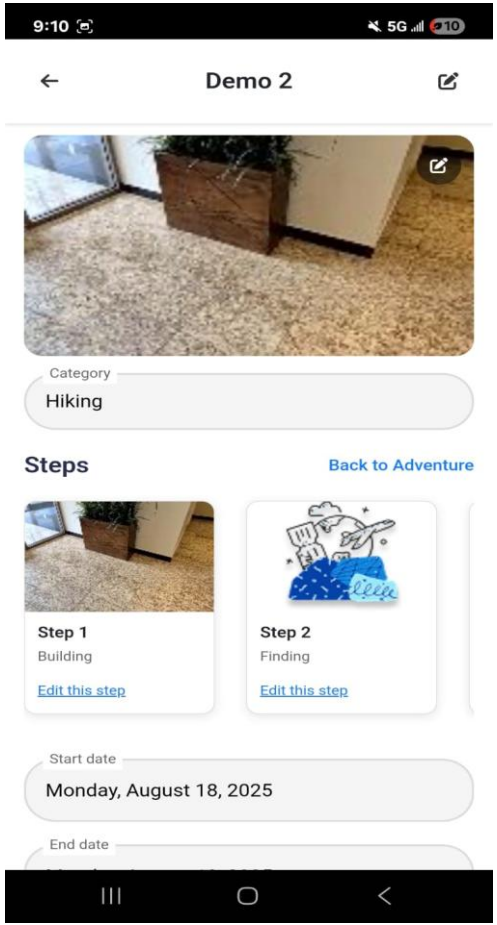
Fetch saved drafts.
Sort and group by date (Today, This Week, Earlier).
Display under date headers in the UI.

05



RESUME AND EDIT STEPS FROM NEXT

STEPS AFTER NEXT BUTTON CLICKED



- Added the steps slides in after the user clicks next from the draft page.

- The slide should have the same functionality as described in the Figma design.

- Edit functionality should work for old and new clips.

- Thumbnails are shown as the main image of each card.

06

CODING PRACTICES EXAMPLE

```
class StepManager
{
    constructor(controller)
    {
        this.controller = controller;
        this.logger = logger;
        this.databaseManager = window.databaseManager;
        this.maxSteps = window.maxSteps; // Maximum number of steps allowed per adventure
        this.logger.log('StepManager: Initialized');
    }
```

StepManager Class

- **Purpose**: Centralizes all logic for creating, managing, and interacting with adventure steps in the application.
- **Key Responsibilities:** Adding/removing step UI elements for adventures
- CRUD operations for steps in the database
- Enforcing maximum step limits per adventure
- Rendering and refreshing step lists
- Handling step editing/deletion interactions

StepManager Class

The Problem: Adventure app needs to manage multiple steps

- Each step has complex operations (CRUD, UI, validation)
- Controller was getting overwhelmed with step logic

The Solution: StepManager class centralizes ALL step-related functionality

- Single responsibility: Manage adventure steps
- Clean separation from AdvController

Architecture Benefits:

Before (Monolithic Controller):

AdvController
- Handle user input
- Manage DOM
- Database operations
- Business validation
- UI updates

After (Separation of Concerns):

AdvController
- High-level flow
- User feedback
- Page navigation

StepManager
- Step CRUD
- DOM manipulation
- Event handling
- Business rules

THANKS!

Do you have any questions?
nicola98b@gmail.com