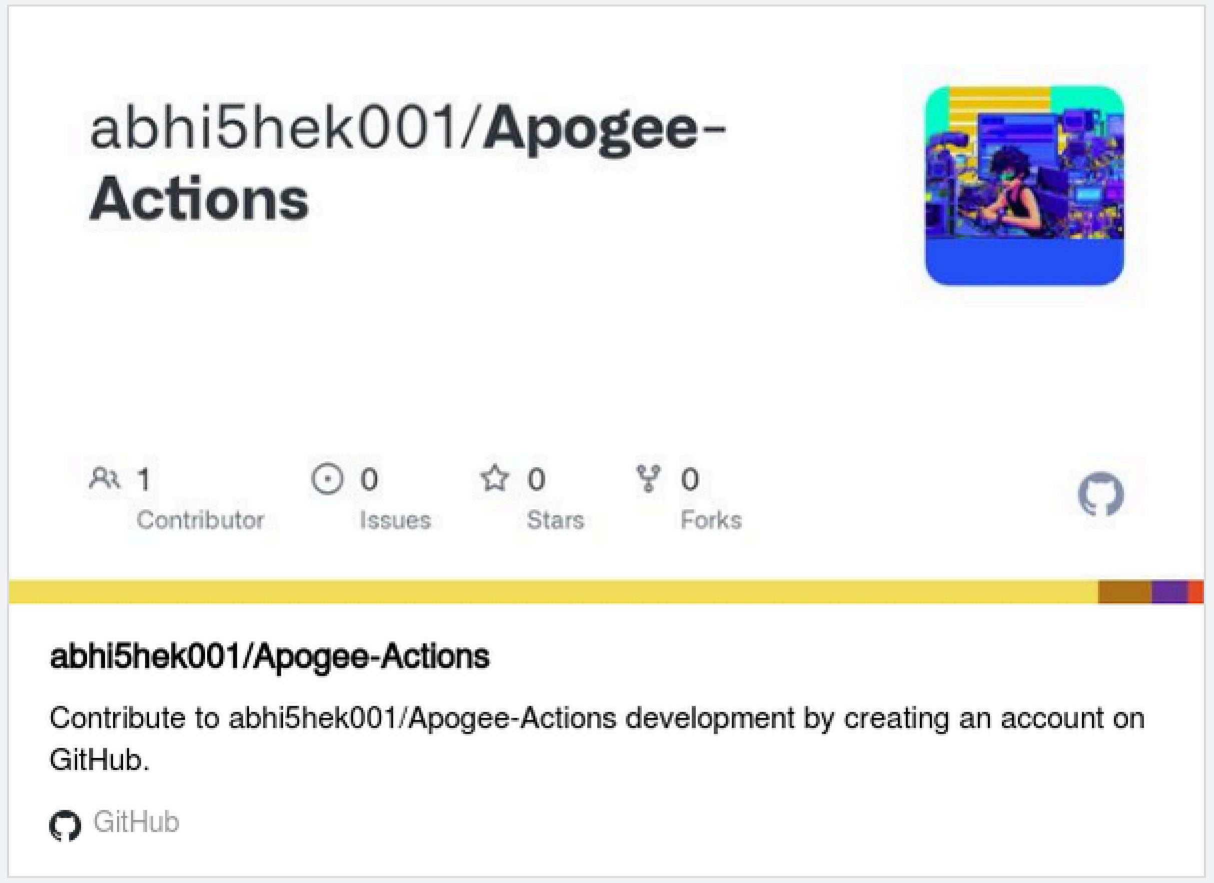
APOGEE ACTIONS MANAGER


abhi5hek001/**Apogee-Actions**

1 Contributor · 0 Issues · 0 Stars · 0 Forks

abhi5hek001/Apogee-Actions

Contribute to abhi5hek001/Apogee-Actions development by creating an account on GitHub.

GitHub

By Abhishek Sahay

S20220010003

LinkedIn Github Portfolio

QUESTION 1

Tell us something that is not already in your resume. This could be your hobbies, current internship search experience, your passion for sports, philosophical, mythology or any topic that you are interested in. Don’t forget to share why this interests you.

SOLUTION:

Outside of my resume and tech stack, I love **sports and fitness**. I play a lot of **badminton**, because it is a high-pressure game. You have to think of the opponent’s next move to prepare your next move for the win.

I’m also regular at the **gym and running**. For me, it’s not just about being fit; it’s about discipline. There are days when I’m exhausted, but I still show up. That 'don't quit' mindset and endless questions on mind while running motivates me. Also, I’m into movies, especially those with a strong plot. I like analyzing how a character handles conflict to reach their goal, which helps me understand different perspectives by putting myself on their shoes.

QUESTION 2

You are building a simple mobile app that shows a list of items from an API.
I. Write the steps you would take to build this app. What considerations are you making?
2. List assumptions you are making.
3. List at least 3 things that could go wrong and how you'd debug them..

SOLUTION:

1. Project Steps & Implementation Logic:

The management of this project follows a Hybrid Lifecycle: building a high-quality web app first, then "syncing" it into a native mobile container using **Capacitor**.

- Foundation (Web Layer):
  - Initialize the project with Vite and Tailwind CSS for a responsive, mobile-ready UI.
  - Build a dedicated API Service using fetch or Axios to handle data retrieval.

QUESTION 2

- **Mobile Bridging (Capacitor Layer):**
  - Run `***npm run build***` to generate optimized web assets in the ***dist/*** folder.
  - Initialize Capacitor `***npx cap init***` and add the Android platform `***npx cap add android***`.
  - Followed a Native-Bridge workflow. I kept my Android folder in Git to track changes and used `***npx cap sync***` to ensure my React updates were always reflected in the mobile build.

- **Performance Considerations:**
  - Virtualization: If the API list is long, use react-window to only render items visible on the screen, keeping scroll performance smooth at 60 FPS.
  - Native Feel: Implement a "Splash Screen" and "App Icon" using @capacitor/assets so the app doesn't look like a generic website upon launching.

QUESTION 2

2. Strategic Assumptions

- **Database persistence:** I chose MongoDB Atlas for cloud persistence, ensuring the app works out-of-the-box.
- **Single-User Focus:** I assumed the primary goal of this assignment is to demonstrate core CRUD logic, UI/UX skills, and mobile integration. Consequently, I prioritized these features over implementing a complex Authentication/Authorization system (JWT), keeping the barrier to entry low.
- **Code Reusability (Capacitor):** I assumed efficient resource utilization was key. Instead of maintaining two separate codebases (Web + Native), I utilized Capacitor to bridge the React application to Android, demonstrating how a single modern codebase can deliver a native-like experience across platforms.
- **Modern Environment:** I assumed the deployment environment would support modern Node.js (ES6+) and web standards, allowing me to use the latest simplified syntax and tools (Vite, Tailwind) without heavy legacy polyfills.

QUESTION 2

3. Potential Technical Issues & Debugging Strategies

- **Mobile Network & Connectivity Isolation:**
  - **Issue:** The app fails to connect to the backend if localhost is used, as the Android system views it as the device itself rather than the PC server.
  - **Debug:** I would use Remote Debugging via Chrome DevTools (chrome://inspect) to monitor the Network tab and update the API URL to the PC's actual local IP address.
- **Data Integrity & Conflict Management:**
  - **Issue:** Rapid offline updates followed by reconnection can lead to race conditions, causing data corruption or out-of-order writes.
  - **Debug:** I would inspect localStorage payloads and server-side timestamps to enforce a "Last Write Wins" strategy, ensuring the most recent user intent is synchronized.

QUESTION 2

- **Native Plugin & Environment Synchronization:**
  - **Issue:** Capacitor plugins (like Haptics or Camera) might crash on specific Android API levels due to version mismatches or missing permissions.
  - **Debug:** I would utilize Android Studio's Logcat to trace native stack traces and run npx cap sync to realign native project files with the web assets.
- **Memory Efficiency & Scroll Performance:**
  - **Issue:** Rendering massive lists of action items can deplete device memory, causing UI "jank" or app termination during scrolling.
  - **Debug:** I would use the React DevTools Profiler to find bottleneck components and implement List Virtualization (e.g., react-window) to maintain a smooth 60 FPS.

QUESTION 3

You have to build an app that manages all your action items. An action item could be to remind, to send an email, to set up a calendar invite, to prioritise.

SOLUTION:

1. Explain how you would go about doing this.

I would build this as a Hybrid App using a React-based frontend and a Node.js/Express backend. To manage different tasks efficiently, I'll use a "Unified Task Schema." This means every item (Remind, Email, Invite) shares a common data structure in MongoDB but is distinguished by a type field. This allows the UI to dynamically show specific action buttons—like "Open Mail" for email tasks or "Add to Calendar" for invites—based on that type.

2. Implementation Plan

- **Phase 1:** Backend & Schema (Data Integrity): I'll build the Express API first. It's crucial to define the "contract" between the server and the app early so the frontend knows exactly what data to expect.
- **Phase 2**: Responsive Frontend (Core Logic): Using React and Tailwind CSS, I'll build the main dashboard. My focus here is on the "Add Task" flow and "Priority Sorting" to ensure the app is actually useful from day one.

QUESTION 3

- **Phase 3:** Capacitor Integration (Mobile Bridge): Once the web version is stable, I'll wrap it using Capacitor. This turns the web code into a native Android app without me having to learn a completely new language like Kotlin.

3. Web vs. Mobile Interface Considerations

- **Ergonomics:** On the web, users have a mouse and keyboard, so I'll use a sidebar for navigation and shortcuts like Ctrl+N. On mobile, everything needs to be "thumb-friendly," so I'll use a Bottom Navigation Bar and a Floating Action Button (FAB).
- **Interaction:** Mobile users expect tactile feedback. I would use Capacitor plugins to add Haptic Feedback (vibration) when a high-priority task is completed, whereas on the web, a simple color change or animation is enough.

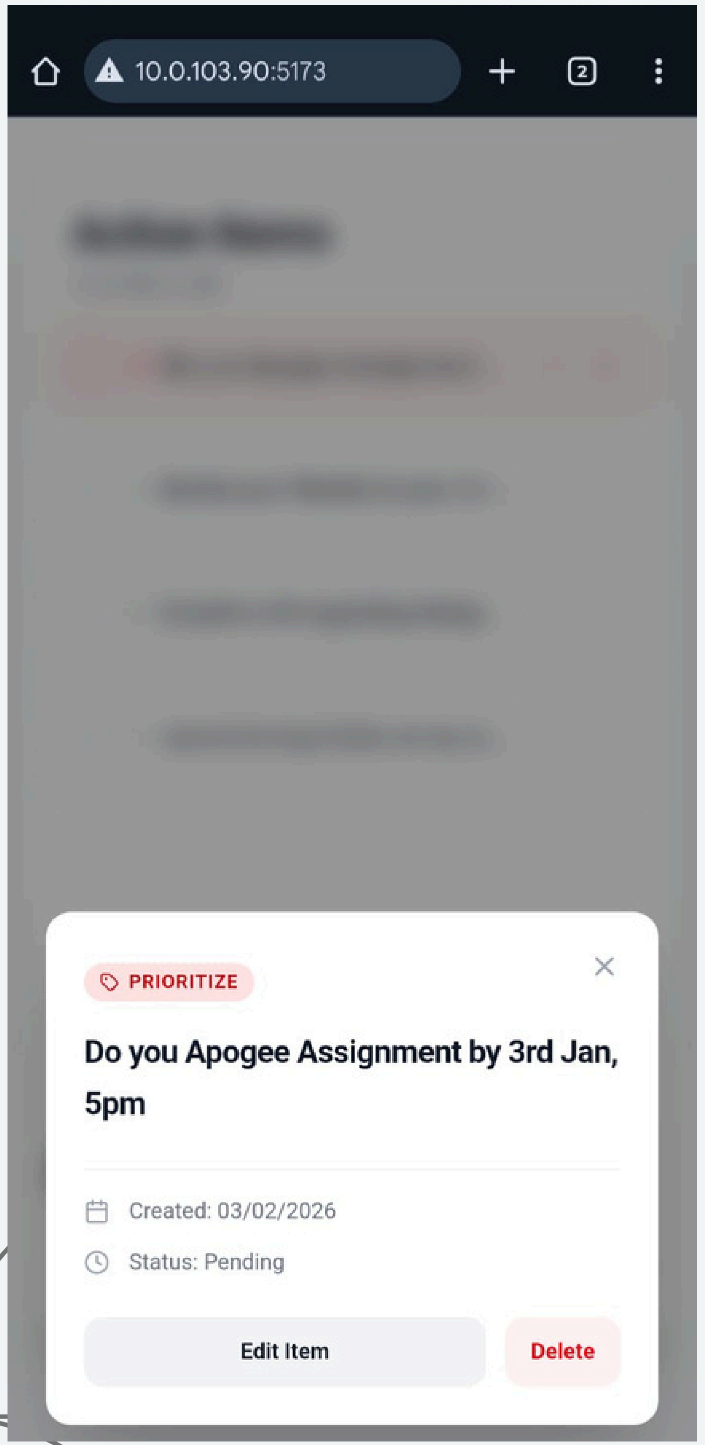
4. Offline Mode & Caching Strategy

- **Caching Strategy:** I would use a Stale-While-Revalidate (SWR) approach.
- **What to Cache:** I'll store the list of action items in the browser's localStorage. This way, the app opens instantly even with no internet.
- **Invalidation Logic:** If the server has newer data, the cache is updated (invalidated) and the UI refreshes automatically. This keeps the app fast and accurate.

QUESTION 3

5. Repository & Proof of Concept:

- I have already started implementing this architecture. You can see my folder structure (Client/Server), the mobile setup, and my automated build pipeline in my repository:
- GitHub Link: https://github.com/abhi5hek001/Apogee-Actions


10.0.103.90:5173

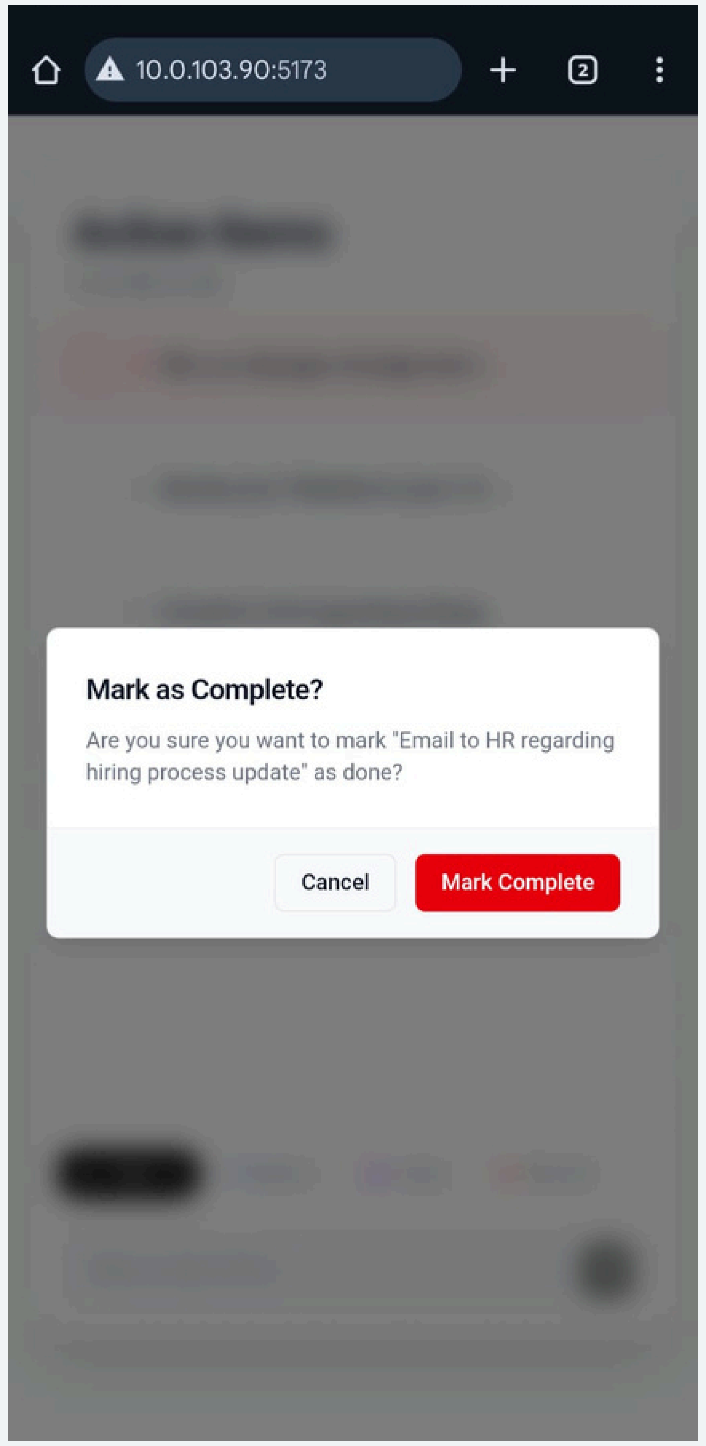
PRIORITIZE

Do you Apogee Assignment by 3rd Jan, 5pm

Created: 03/02/2026

Status: Pending

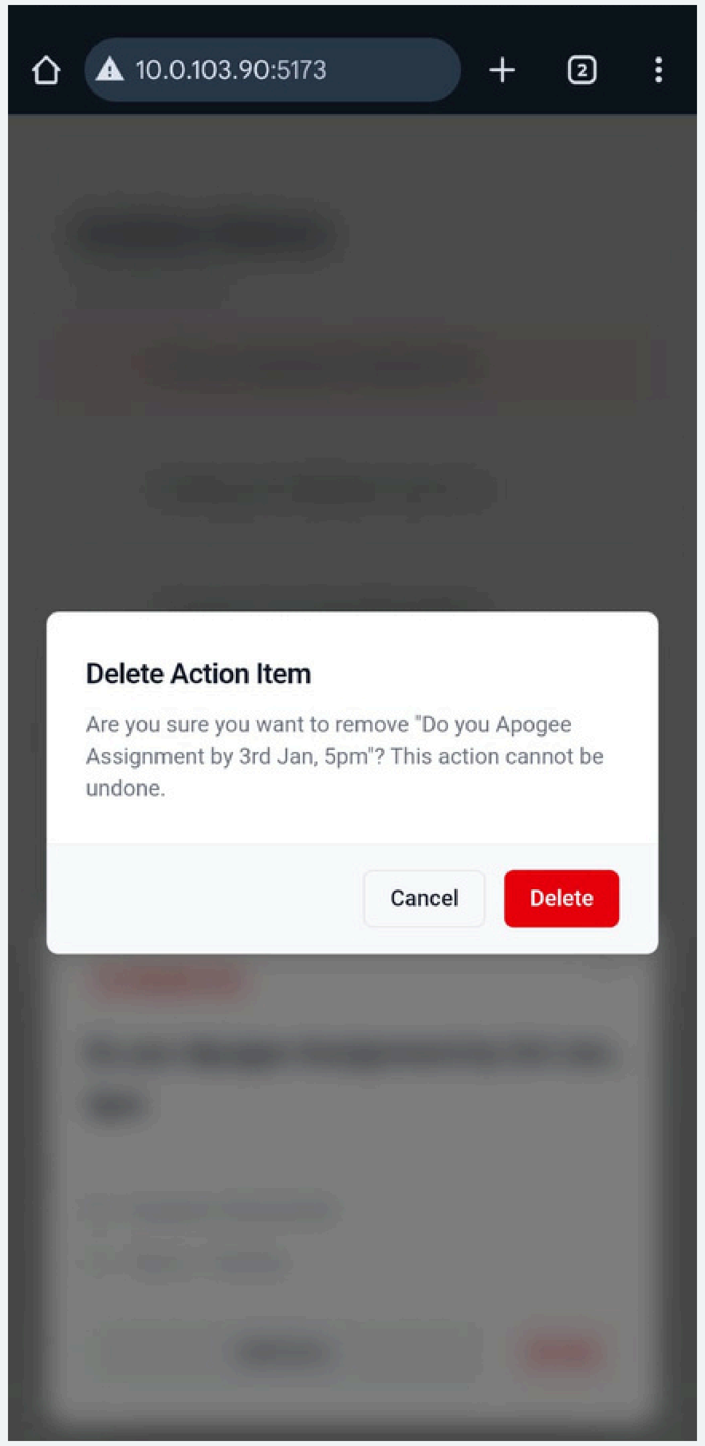
Edit Item | Delete


10.0.103.90:5173

Mark as Complete?

Are you sure you want to mark "Email to HR regarding hiring process update" as done?

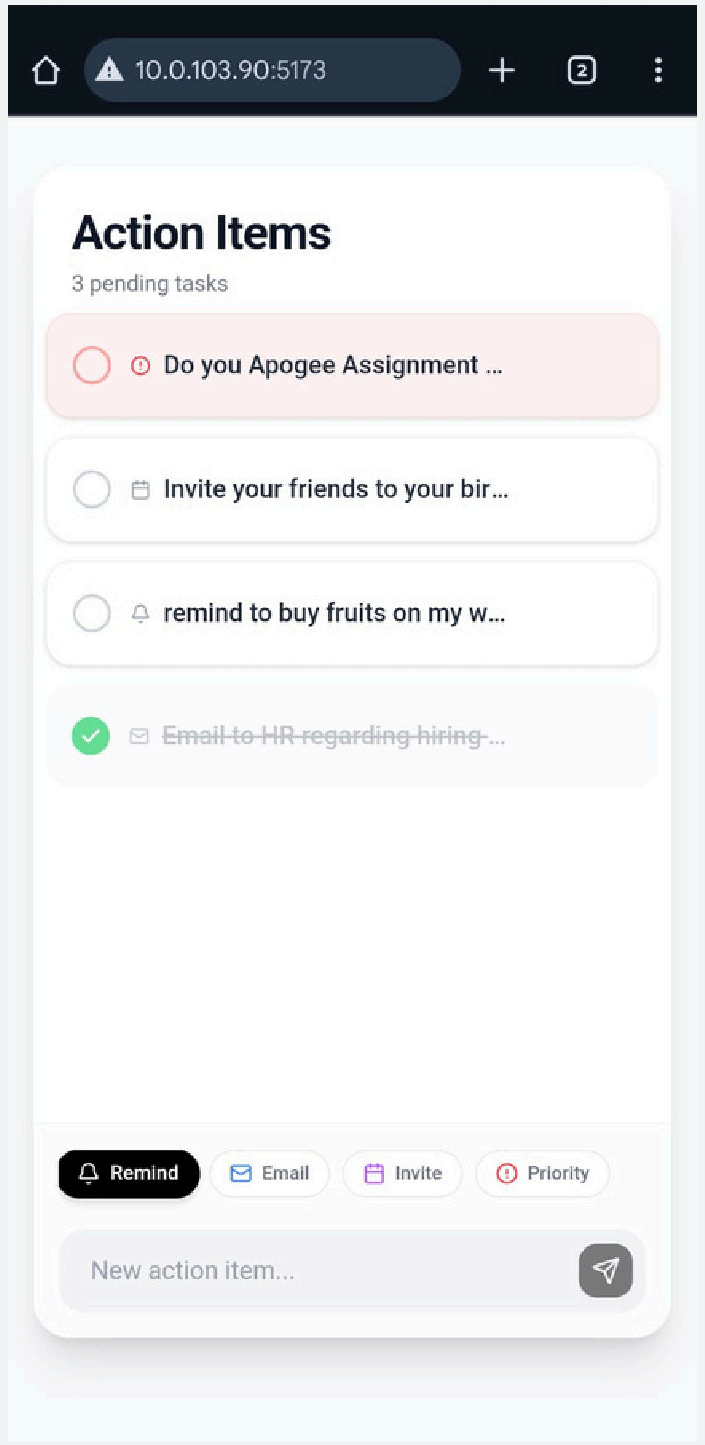
Cancel | Mark Complete


10.0.103.90:5173

Delete Action Item

Are you sure you want to remove "Do you Apogee Assignment by 3rd Jan, 5pm"? This action cannot be undone.

Cancel | Delete


10.0.103.90:5173

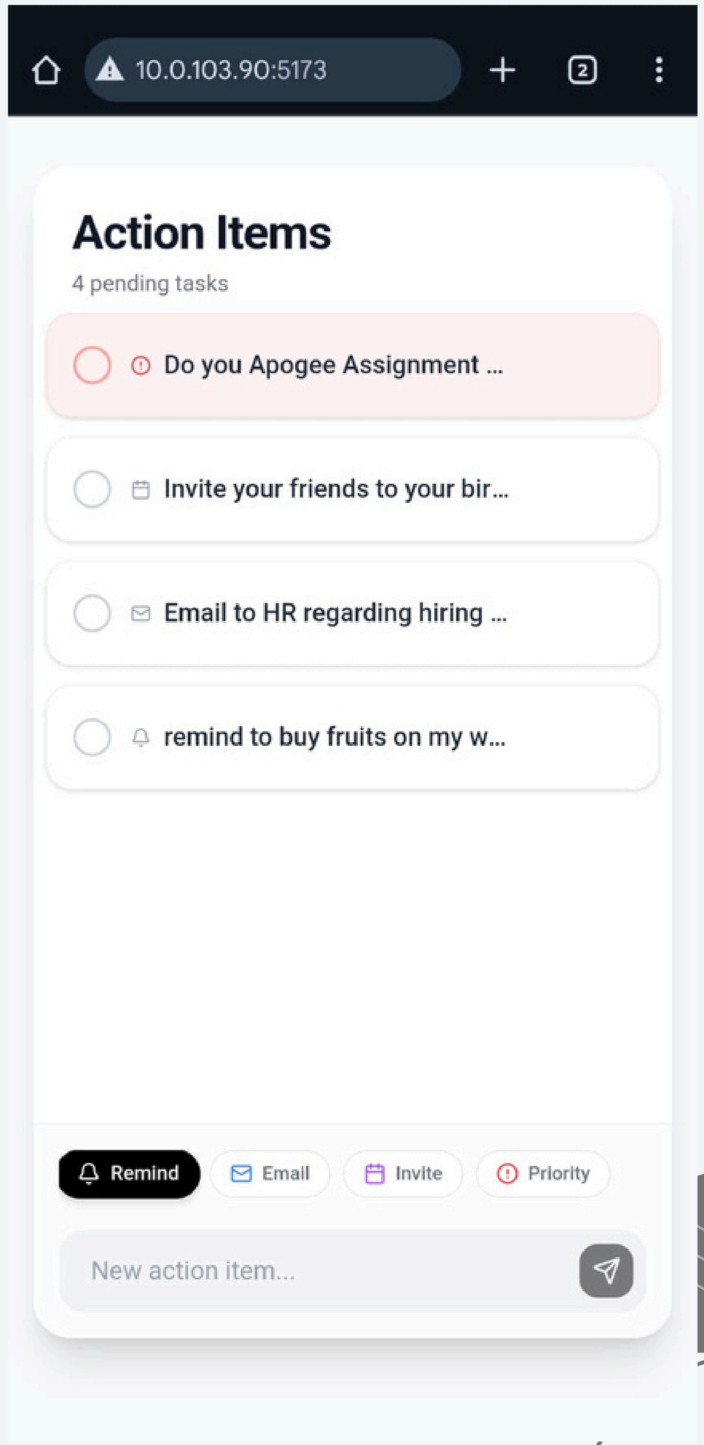
Action Items

3 pending tasks

- Do you Apogee Assignment ...
- Invite your friends to your bir...
- remind to buy fruits on my w...
- ✓ ~~Email to HR regarding hiring ...~~

Remind | Email | Invite | Priority

New action item...


10.0.103.90:5173

Action Items

4 pending tasks

- Do you Apogee Assignment ...
- Invite your friends to your bir...
- Email to HR regarding hiring ...
- remind to buy fruits on my w...

Remind | Email | Invite | Priority

New action item...

THANK YOU