go/ghacks25

gHacks ‘25 Project Submission

Instructions:

- Make a copy of this doc (go/ghacks25-submission-template)
- Give (silent) Edit or Comment access to ghacks25-judges@
- Make sure ghacks25-judges@ has access to artifacts included below
- Link to the doc in the hackathon submission tool by **5pm PT on May 29, 2025**
    - Select “Add new documents”
    - Add the link to your submission document

Project name: Mini-Apps Generator for Android

Project summary:

The Mini Apps Generator empowers users to create their own, personalized, small and functional apps on Android. The generator comes in the form of an Android app that lets the user enter their own prompt to create an app which is powered by the device capabilities and APIs. The user can use it to create an app for repetitive tasks (like calculating coffee-water brewing ratios), a travel guide app for an upcoming trip, or just something fun (like a space invaders based apps launcher).

Technical description:

The app is built on top of the existing GenUX infrastructure that was developed by the AIR gen ux team (go/genux-status),

The simplified app flow, given a user prompt, consists of several LLM calls to produce an HTML based app that is then presented to the user using a WebView component. The WebView has access to several Android APIs that are visible to the

app using the JavaScriptinterface API.. The app is persisted on device using a local SqLite database so the user can access the apps they have already built.

In more details:

Given a prompt by the user, the app would ask Gemini to generate a few different follow up prompts and queries:

1. A prompt for generating a widget/icon for the app as an entry point.
2. A detailed prompt for generating an HTML app that will be later presented on the device using a WebView. (The initial prompt also contains a list of existing APIs and capabilities that are accessible from the webview).
3. A list of search queries that will be used to fetch images that are relevant for the app.

After the initial LLM call returns, the GenUX framework generates the widget/icon for the app using (1), and the app fetches images using the search queries generated in (3).
The LLM (Gemini 2.5) is then called again with a larger prompt that is built with the prompt generated in (2) and also a list of available images.
The result of this call is the HTML app that is persisted locally on the device (together with the prompts and the widget).

When the user accesses the app using the generator home screen, the app code is loaded from the DB and shown to the user using a WebView component.

In the app screen, the user has also the ability to update the app and request changes. This is implemented by calling the LLM again, but this time the app code is also provided in the prompt, so the result is a modified app. This can be helpful to adjust the design of the app, add features or ask the LLM to fix bugs.

3-minute video demonstration: [insert link]
Prepare a concise video demo for your project that showcases the following:

- **Core functionality:** Provide a demo of the project from the perspective of a user!
- **How you implemented it:** Give us an idea of how the project works under the hood. Get specific! Technical details are part of the judging criteria.

- **User Experience and Accessibility Considerations:** Part of a great project is thinking about the end user. What was your process for making the experience as accessible and seamless as possible? Walk us through that.
- **Project Background and Potential Impact:** Why did you build it? Tell us the story.

TODO:

Source code:
https://source.corp.google.com/piper///depot/google3/java/com/google/research/air/frameworks/genux/demo/miniapps/

Running demo (if applicable): go/get-mini-apps (Needs to install an APK on an Android device)
See this deck for more information on running the APK.