Firebase Genkit

2025/03/27

Firebase

- Firebase is a **Backend-as-a-Service (BaaS)** platform by Google.
- Provides tools for building web & mobile applications **without managing servers**.
- Core services: **firestore (NoSQL DB), authentication, hosting, cloud storage and cloud functions**.

Firebase Genkit

- A **toolkit** designed to develop **complex AI services** for web and mobile apps.
- **Seamlessly integrates** with various AI APIs, including **OpenAI, Gemini, and more**.
- **Simplifies development** of advanced AI workflows, including **multi-agent collaboration, tool calling, and complex AI flows**.
- Use Typescript

Typescript

- **Superset of JavaScript:** All JavaScript features + additional TypeScript-specific tools.
- For this lab:
  - Type aliases: you can specify a name for a any type, such as string, number, and object.
  - Asynchronous functions
    - async tells TypeScript that the function contains asynchronous code. await will pause execution until it get the return.

Genkit Code

- **Configure plugin**
- **Prompts**
- Chat session
- Tool
- **Retrieval-augmented generation**
- **Flow**
- Multi-agent system

Genkit Code –

Generating content

```
import { genkit } from 'genkit';
import { vertexAI } from '@genkit-ai/vertexai';
```

```
export const ai = genkit({
  plugins: [
    vertexAI({
      projectId: getProjectId(),
      location: 'us-central1',
    }),
  ],
});
```

```
async function main() {
    const { text } = await ai.generate(‘prompt here’);

    console.log(text);
}
```

Genkit Code –

definePrompt

- Use it to config model, define input and output schema, and write your prompt.

```
const hello = ai.definePrompt({
   name: 'hello',
   system: 'talk like a pirate.',
   prompt: 'hello {{ name }}',
   input: { schema: z.object({
         name: z.string()
      })
   }
});
const { text } = await hello({name: 'Genkit'});
```

Genkit Code –

dotPrompt

- **What is a .prompt file?**
  - A file dedicated to **model configuration and prompt design**.
  - Separates the **function logic, model configuration, and prompt content** for better organization.
- **Where to place it?**
  - Store .prompt files in the **prompts/ folder** inside your Genkit project.
- **How to structure a .prompt file?**
  - See the code

Genkit Code –

run dotPrompt: ai.prompt()

```
const recipeGenerator = ai.prompt('customRecipe');
const response = await recipeGenerator({
    suggestRecipe: recipes[0],
    ingredients: input
});
const customRecipe: Recipe = response.output;
```

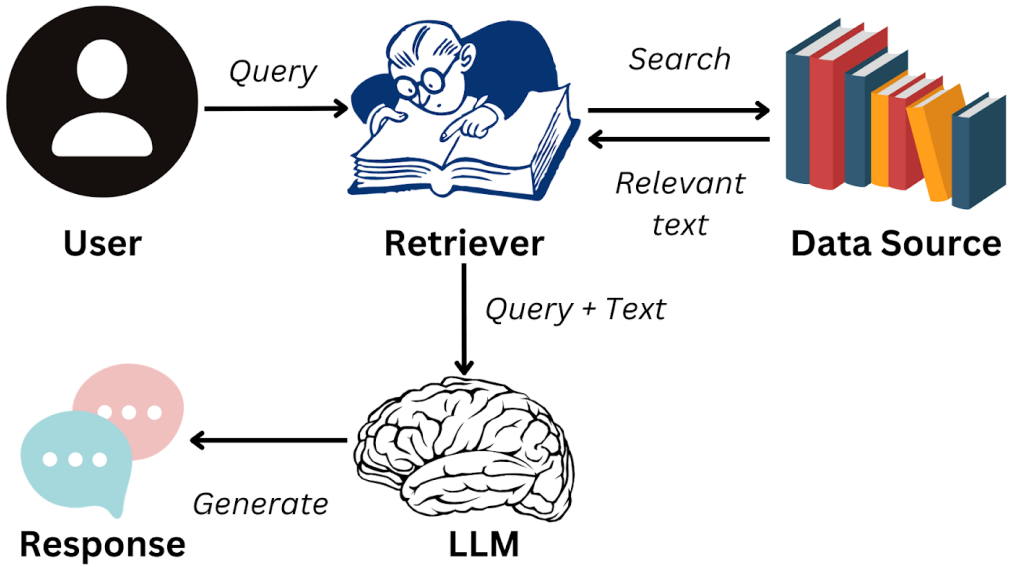
Retrieval Augmented Generation (RAG)

Retrieve relevant documents for external knowledge

Augment the retrieved information for the AI model

Generate the response based on retrieved data and its own knowledge

Purpose -- reduce hallucination and keeps AI up to date



Genkit Code –

RAG

Set up retriever

```
export const recipieRetriever =
defineFirestoreRetriever(ai, {
  name: 'recipeRetriever',
  firestore,
  collection: 'recipe_with_vector',
  contentField: 'ingredients',
  vectorField: 'ingredients_embedding',
  embedder: textEmbedding005,
  distanceMeasure: 'COSINE',
});
```

Call retriever

```
const docs = await ai.retrieve({
   retriever: recipieRetriever,
   query: input,
   options: {
      limit: 1,
    },
});
```

Flow

- Pack pre- and post-processing steps that must accompany the model call into a function, together with the model call. For example:
  - RAG
  - Retrieve history session
  - Type safety check
  - Combining several models

Genkit Code –

Flow

```
export const customRecipeFlow = ai.defineFlow({
   name: 'customRecipeFlow',
   inputSchema: z.string()
},
   async (input) => {
      ........
   }
}
```

Genkit Workflow –

Development Tool

- cd functions
- npx genkit start -- npx tsx --watch src/index.ts
- Go to your browser and search: http://localhost:4000
- You can test and debug your code on it. (Make sure index.ts import all the flows you want to test.)

Genkit Workflow –

Deploy the flow

- After you finish coding, you should deploy the flow to the firestore so that the frontend can call the flow as a function.
- First, expose this flow as a callable function using onCallGenkit:

```
import { onCallGenkit } from 'firebase-functions/https’;
export generatePoem = onCallGenkit(generatePoemFlow)
```

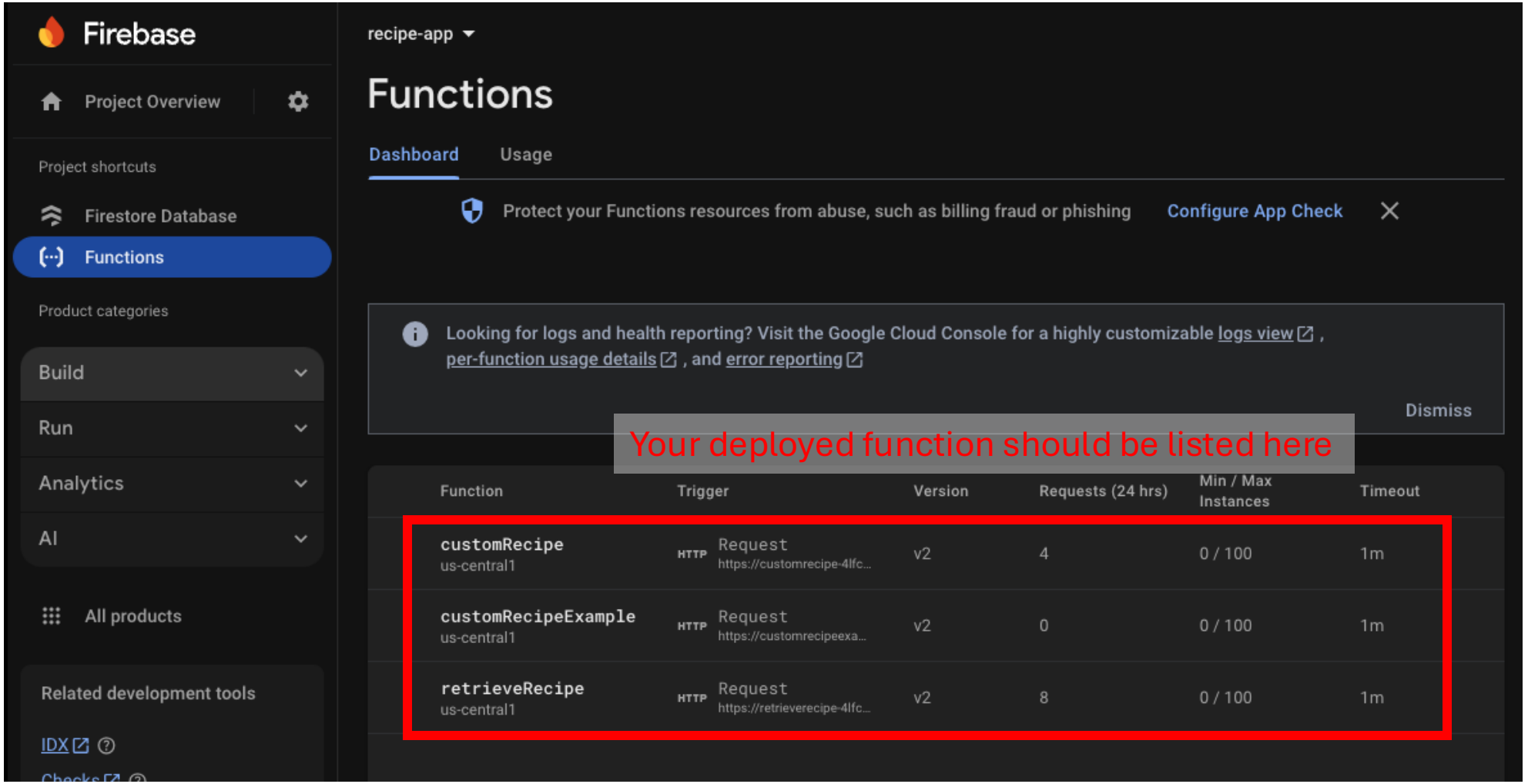
- Then, deploy it

```
cd $PROJECT_ROOT
firebase deploy --only functions
```

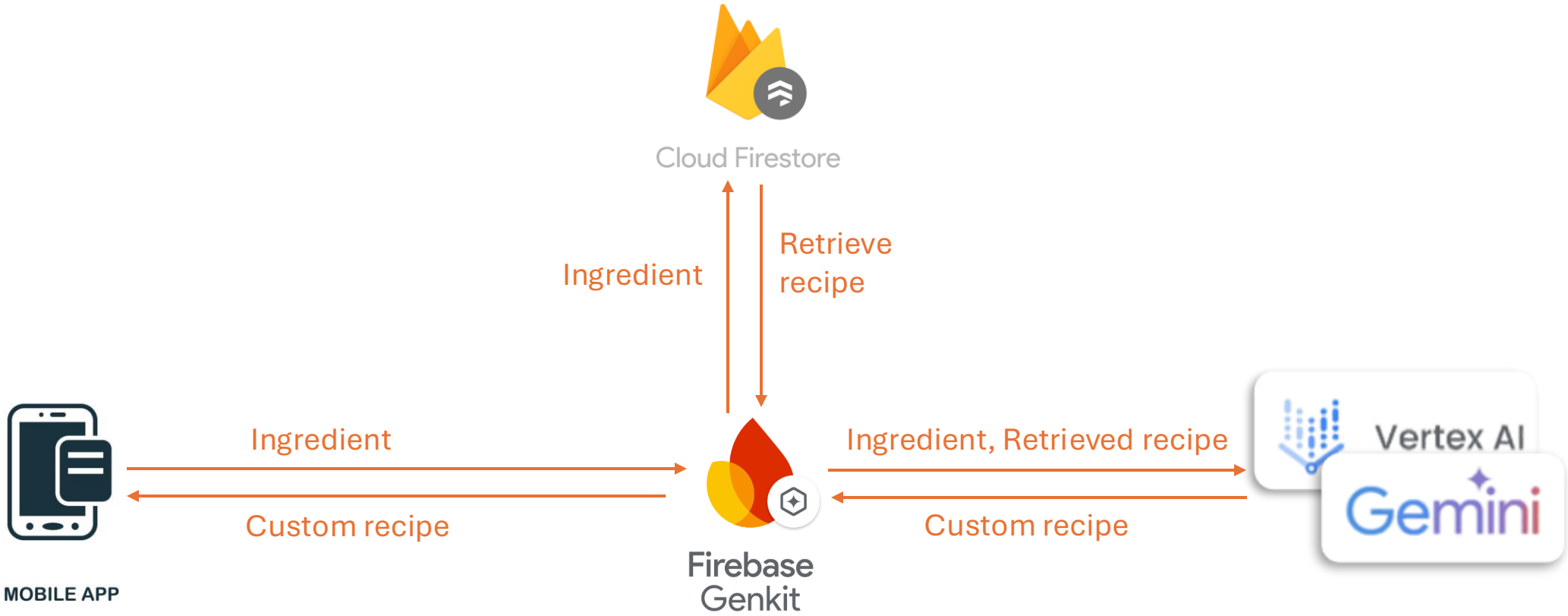
- Wait for the deployment complete, you should see the functions on firebase functions

Genkit

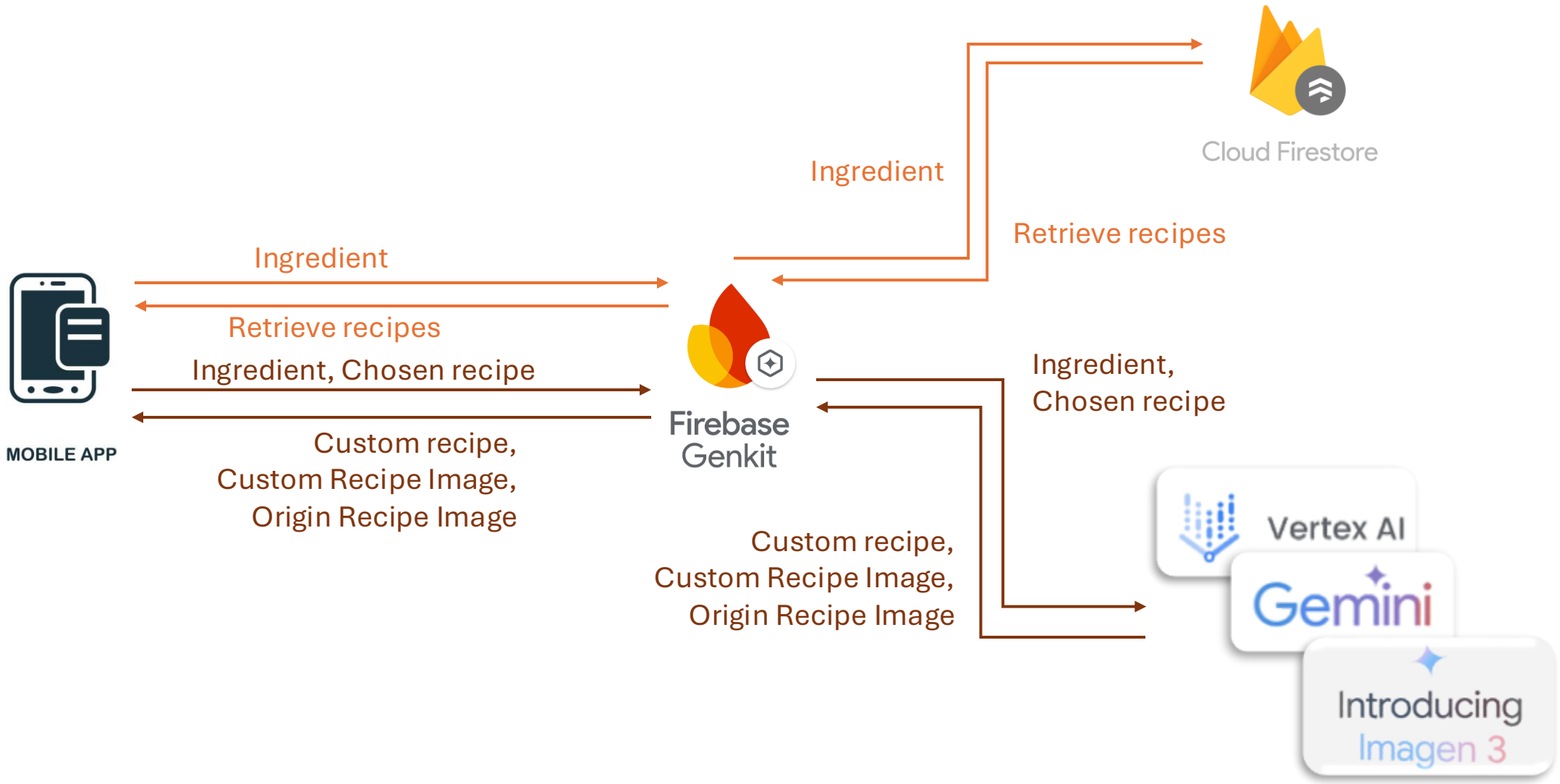
Deploy the flow



Example code -- Flow



Lab code -- Flow



Example APP for image generation