Development of an application used in testing algorithmic problems.

Introduction

Our application is a platform designed for testing algorithmic problems, leveraging Google Cloud services for its architecture. The frontend is built using Next.js, offering an intuitive interface for users to solve or add algorithmic problems. For the backend, Firebase is utilized, with collections including users, problems, solutions, results, and tests.

Architecture

Frontend: Developed using Next.js, providing a user-friendly interface for interacting with the application. Enables users to solve algorithmic problems or contribute new ones.

Backend: Utilizes Firebase for managing data and user authentication.

Consists of several collections:

- Users: Stores user information and authentication data.
- Problems: Contains details about algorithmic problems.
- Solutions: Stores solutions submitted by users.
- Results: Holds the results of test executions.
- Tests: Contains test cases for the problems.

Google Cloud Function (Solution Publisher):

Triggered when a solution is added to the database.
Submits a processing request to a PUB/SUB topic named "compiler".

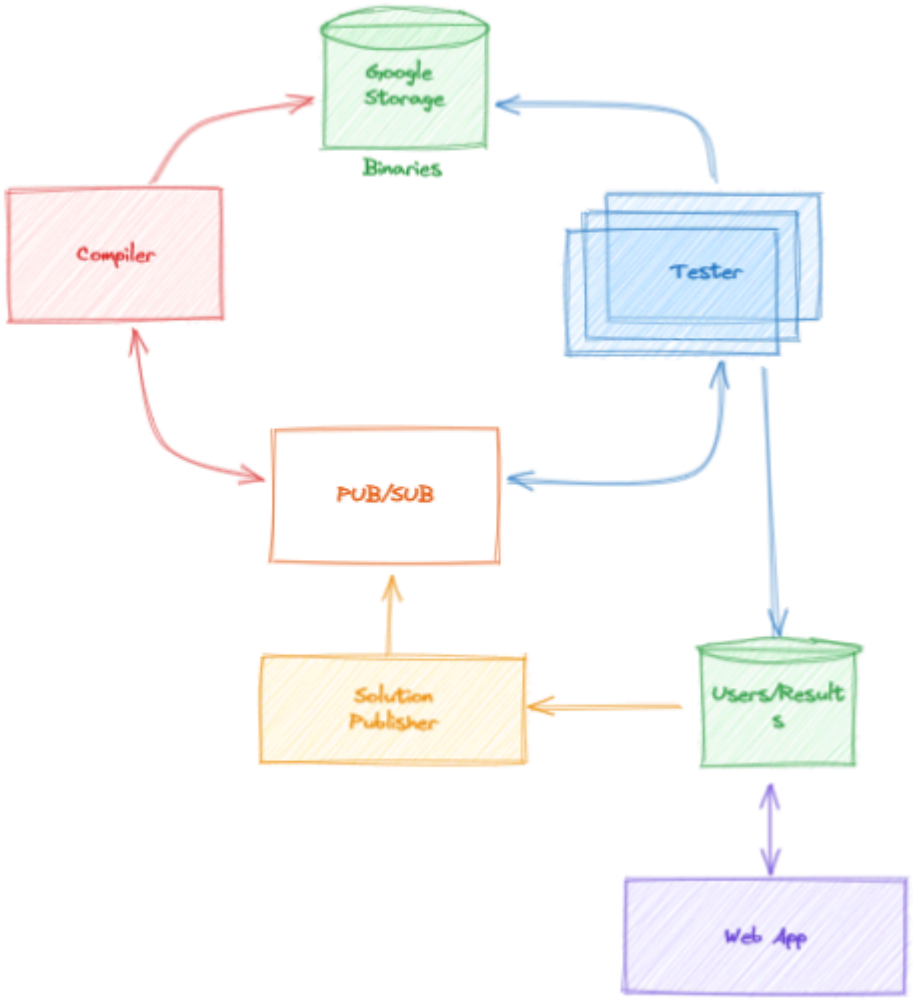
Compiler EC2 Instance:

Listening to the "compiler" topic.
Upon receiving a request, processes the received program.
Uploads resulting binaries to Google Cloud Storage.
Publishes a message to the "executor" PUB/SUB topic for execution.
Google Cloud Storage:
Stores the compiled binaries.
Google Cloud Function (Executor):
Triggered by messages published to the "executor" topic.

Executor EC2 Instance:

Listening to the "executor" topic.
Retrieves binaries from Google Cloud Storage.
Executes the provided tests.
Saves the results in Firestore.

This architecture ensures scalability, as Google Cloud services handle various tasks such as storage, message queuing, and computation, while Firebase manages data storage and authentication. The separation of concerns between frontend, backend, and cloud services enables efficient development, deployment, and maintenance of the application.



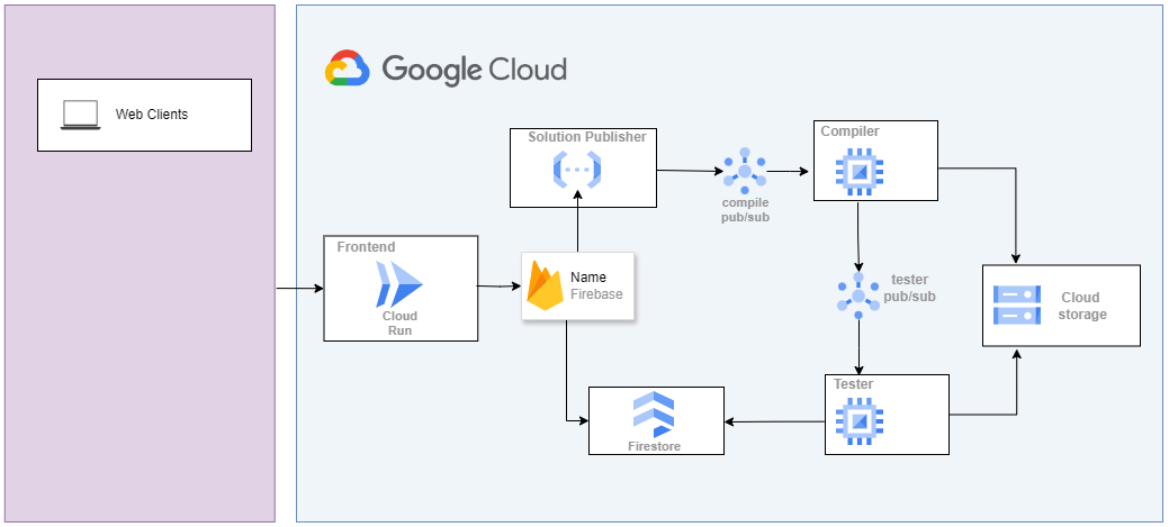
Google Cloud Components

Google Cloud Function: Used for serverless execution of code in response to events, such as adding a solution to the database. Triggers processing tasks and message passing between components.

Google Cloud Pub/Sub: A messaging service for exchanging asynchronous messages between independent applications. Facilitates communication between the compiler and executor components through topics.

Google Cloud Storage: Provides scalable object storage for storing and retrieving binary files, such as compiled programs. Used to store binaries compiled by the compiler component.

Google Compute Engine (EC2 instances): Virtual machines offered by Google Cloud for running applications and backend services. Instances are utilized for both compiling code (compiler) and executing tests (executor) in this architecture.



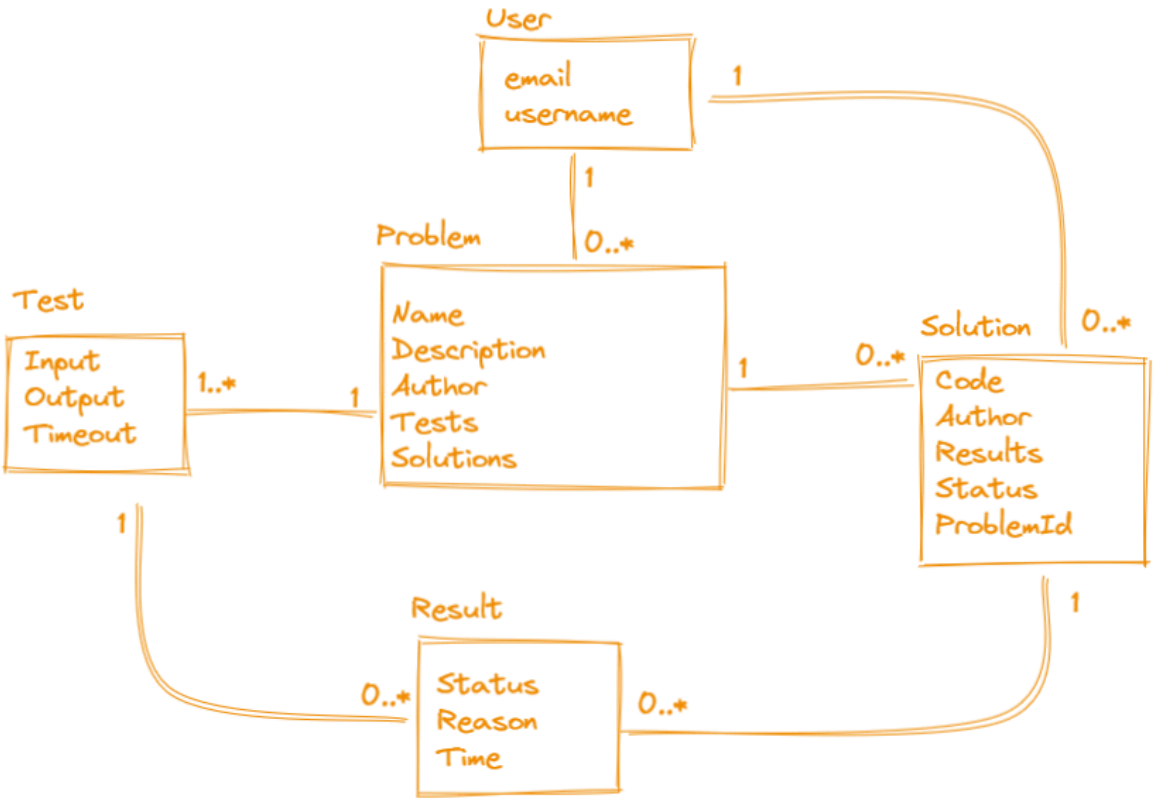
Relationships within the collections

Users Collection: A user can have one or more solutions or problems associated with them.

Solutions Collection: Each solution belongs to one user. A solution can have multiple associated results.

Problems Collection: A problem can have zero or more solutions. Each problem has one or more tests associated with it.

Results Collection: Each result belongs to one solution.


User
email
username
1
1
0..*
Problem
Name
Description
Author
Tests
Solutions
Test
Input
Output
Timeout
1..*
1
1
0..*
Solution
0..*
Code
Author
Results
Status
ProblemId
1
1
Result
Status
Reason
Time
0..*
0..*