CodeGram

Problem Statement

Different platforms for technical interview preparation and competitive programming, such as [leetcode.com](#) or [vjudge.net](#) offer internal statistics on their users, but there is no unified way to view one's progress and achievements across all platforms. Further, many of those platforms lack social or competitive aspects to improving one's skills. Studies show that gamification boosts retention, meaning CodeGram will help programmers improve their skills.

Solution

CodeGram is a competitive social networking app similar to apps such as [Strava](#) but with Algorithms and Data Structures problems as the activity. Users can add friends, create groups and follow people to receive updates when others successfully complete a new problem. Other common social media components like news feeds, likes/comments and profile pages will be implemented to show scores, stats, and updates.

User workflow

A user will create an account and link external platforms they are using to CodeGram. They will be able to add friends, create Groups or follow other users. The user's CodeGram newsfeed will show the most recent updates, including their friends' activities (e.g. "🔔 janedoe123 has completed a new LeetCode Hard problem!"). They can click on their friends' usernames to view their profile which will show stats like how many Easy, Medium, and Hard problems on LeetCode the user has completed. We will explore and implement various gamification and social networking components such as scoreboards, challenges, and rewards.

Internal, External Service(s) and Database(s)
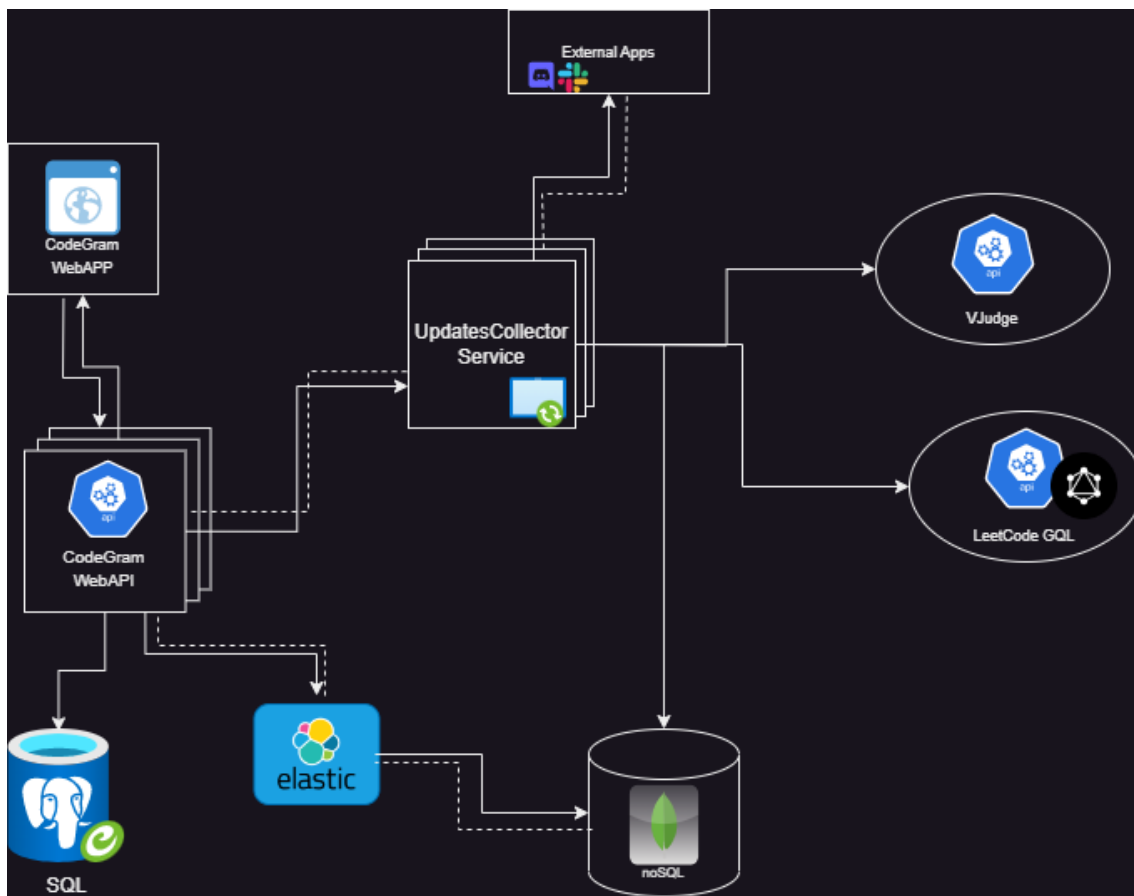
Technology Stack

Frontend framework

React will be used. We are considering using Chart.js for data visualizations such as user stats, etc.

Backend

We will use Node.js + Express on GCP as a PaaS for both services (see Internal Service).

Internal Service

Unfortunately, most platforms like LeetCode do not provide webhooks for user updates, so we wish to explore the option of having a microservice alongside the "main" back-end, which would be responsible for continuously (scheduled job) pulling the user data from other platforms (see External Services) and sending update messages when a person has solved a new problem. As update workflows are event-driven, we are considering using a Pub/Sub system or message queues, which also gives us an easy way to expose our own webhooks. Both our servers will access their databases (more on that in the DB section).

External Services

We will use the LeetCode GraphQL API and VJudge APIs to get users' stats and explore other external services to help bootstrap our application, including but not limited to services for authentication and integration with other social media apps (Instagram, Facebook, etc.).

Database

The current architecture plan involves having a NoSQL MongoDB for data coming from external API. This grants a flexible data model, which is crucial as the external data format is prone to change, and this allows for easy expansion to more algorithm platforms, like CodeForces or Kattis. We are also expecting a high read/write load as we will be continuously getting all user data from external services and NoSQL is designed for horizontal scalability. We will also use an RDB for Events, Groups and Friends (internal data) for efficient joins. We are debating using a search engine like ElasticSearch or an in-memory db like Redis for faster data retrieval; MongoDB also offers in-memory/hybrid options.

Authentication

Users will need to create an account and log in to access the CodeGram functionality. As there is no way for us to verify if the person is not linking their CodeGram account to someone else's external profile, we are adding admin/moderator users with the ability to unlink a CodeGram user from an external platform.