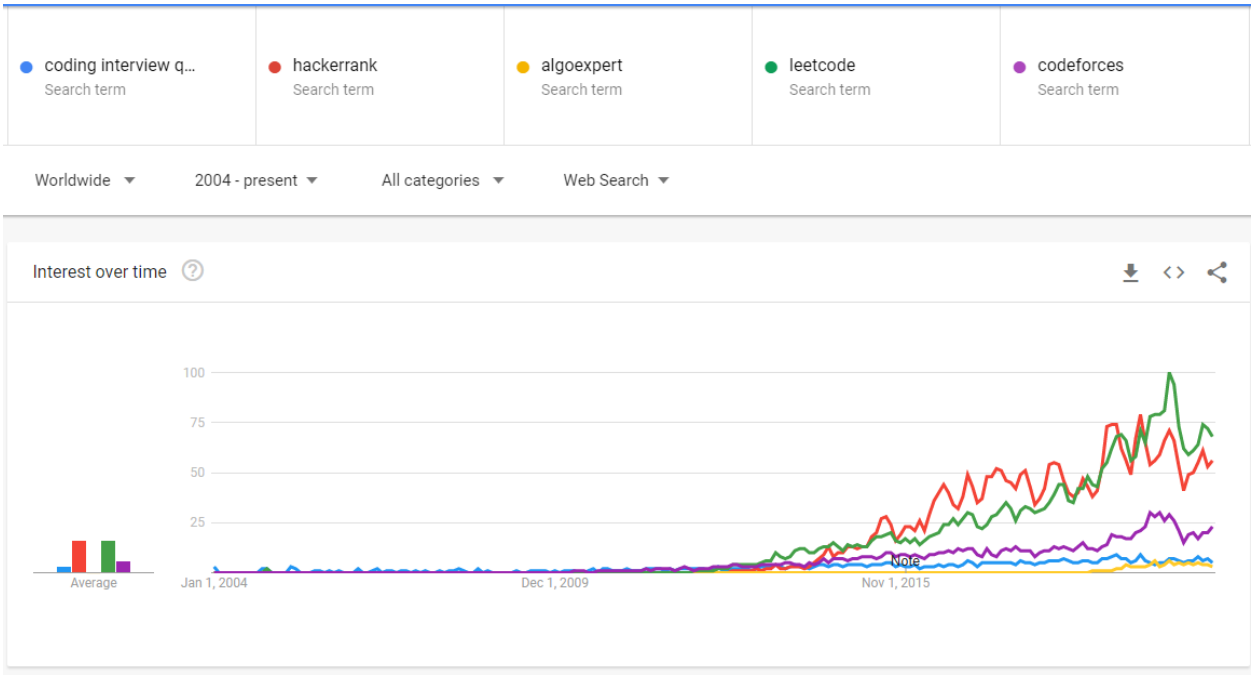
Cloud Computing Homework 5 Report

Irimia Andreea Roxana, Oloieri Alexandru

Case Study



In the last couple of years, online coding platforms have become more and more popular as we can see in the image above.
Each platform comes with its own particular features. For example:

- Hackerank (https://www.hackerrank.com/) offers the option to host remote interviews
- AlgoExpert (https://www.algoexpert.io/) has a gradual difficulty approach and explanatory tutorials created by the creators
- LeetCode (https://leetcode.com/) shows you questions by company and interview type (online assessment, phone interview, online interview)
- Codeforces (https://codeforces.com/) hosts competitions

We can see that the platformes advertised as places where you can prepare for interviews such as Hackerrank and Leetcode are gaining popularity. AlgoExpert is also on the rise, probably thanks to the fact that they have an aggressive advertising campaign going on right now, we are sure we're not the only ones that heard "So you want to become a software engineer at Google?" at least a few times. These platforms are the perfect example of Freemium, they offer a friendly interface, helpful tutorials and guidance but you only get a taste of it if you don't pay them.

On the other hand, platforms such as Codeforces don't rely on advertising; they are promoted and supported by the competitive programming communities in schools and universities. They don't offer you guidance, but they offer an impressive selection of problems and resources that carry a lot of value if you know what to do with it.

What we want to do with our application is to have the best of both worlds. Inspired by Pluralsigth, we want to offer a vast selection of problems, more than enough for someone with a competitive programming background, but also the guidance needed by someone at the start of their programming journey.
How? By allowing our users to contribute with problems that can be solved and rated by everyone, and tutorials and solutions, available only to our premium users. To encourage users to contribute to our platform, they will be rewarded with a share of the profits based on the number of submissions and the rating of their problems.

Existing Solutions

algoexpert.io

https://www.algoexpert.io/product

Architecture

Multiple back-end services:

- Problems service - contains all the problems data
- Payments service - redirects payments request to other services which are specialized in this: Stripe and Paypal
- Remote Code Execution service - runs the user code against all the problem tests and returns errors and execution messages. A docker container is created for each execution
- “Jarvis” service - runs jobs which make sure that after some changes are made to the code, all the problems are still ok (their solutions for the problems still pass the tests)

The front-end is a React app written in javascript and has Stripe and Paypal integration

Technologies

Backend: node.js? (express was used)
Frontend: react.js + javascript + react redux
Hosted on an **NGINX** server

Marketing approaches

- After the successful launch of algoexpert.io, its founders also launched SystemsExpert.io, an app which presents 10 of the most popular system design interview questions.
- algoexpert.io makes all of its money from users paying a yearly subscription. There are currently 3 plans: algoexpert - 99$/year, SystemsExpert - 79$/year, algoexpert + SystemsExpert - 139$/year. In a recent video posted on his channel, Clément Mihailescu said that they currently have more than 50000 paying users

Sources:

How We Built AlgoExpert's Backend (building a web application)
How We Built AlgoExpert's Front End (building a web application)
How We Built AlgoExpert's Remote Code Execution Engine
https://builtwith.com/algoexpert.io

Codeforces

https://codeforces.com/

Technologies

Backend: Java
Frontend: Javascript
Hosted on an **NGINX** server
Hosting provider: Telia Company

Marketing approaches

- They are hosting sponsored contests (some companies pay for a contest and as prizes offer positions in the company)
- They are accepting donations
- They are sponsored by Telegram

An example of a sponsored contest: https://codeforces.com/blog/entry/90185
Huawei offers some prizes: “Prizes for the top **20 Global Challenge performers** are as follows: 1-4: Huawei MateBook X, 5-8: Huawei P40 Pro, 9-12: Huawei Matepad Pro, 13-20: Huawei Watch GT 2 Pro” so they can attract competitors, and in return they can advertise their ICPC Training Camp hosted by Huawei to some of the world’s best competitive programmers.

Sources:
https://builtwith.com/codeforces.com

Leetcode

https://leetcode.com/

Technologies

Perl
Apollo GraphQL
Frontend: Angular
Hosted on an **NGINX** server
OS: Ubuntu
Web hosting providers: Linode, Cloudflare, Digital Ocean
Email hosting providers: Amazon SES, Sendgrid

Marketing approaches

- They offer premium accounts, which have access to more questions and more explanations. There are currently two plans: Monthly - 35$/month and Yearly - 159$/month

Technologies

Programming Languages and Frameworks

Front End

The main UI is an Angular 9 application. We have chosen it over other popular frameworks because it comes with everything you need already included: a router, http support; and because it allows us to easily create highly reusable components since it has a really good dependency injection framework.

Back End

We have implemented multiple services, and we will refer to them in the following way:

- **Service#1: Problems Service**, the service which contains all the problem related data, and offers API for retrieving and creating new entries in the databases. It is written in node.js using typescript (for type checking and faster code errors detection) and the main reasons for this choice are the speed, performance and asynchronous processing.
- **Service#2: Users Microservice**, the service which contains and manages all the user related data (tokens, refresh token, users, users profiles). It is written in node.js using typescript (for type checking and faster code errors detection) and the main reasons for this choice are the speed, performance and asynchronous processing.
- **Service#3: Evaluation Service**, the service which takes an evaluation submission (problem id + source code + programming language), runs all the tests for that problem and writes the verdict and score to a database. It is written in Python using the Flask microframework. The reason for using this programming language is the following one: we need to manipulate files, create OS processes, and these kinds of operations are easily done in python due to its libraries and syntax (you don't need a lot of code to compare two files for example). The reason for using Flask is because we needed a light framework which allows us to create one single route, for evaluating a source.

Cloud Services

Google Cloud Services

1. Cloud Storage: We use it for storing test cases and the submitted source code in a file format, and this service was perfect for this use case: “Object storage for companies of all sizes. Store any amount of data. Retrieve it as often as you’d like.”
2. Cloud Datastore: We use it as a database for the problems microservice because we needed fast access to data and high scalability, and this service was perfect for this use case: “Datastore is a highly scalable NoSQL database for your web and mobile applications.”
3. App Engine: We use it for hosting the problems microservice. We wanted it to be highly scalable and efficient, without worrying about the environment where it runs, so a serverless platform was the perfect choice, and App Service is advertised as this kind of platform: “Build highly scalable applications on a fully managed serverless platform.”
4. Cloud Tasks: We use it for creating evaluation submission tasks, which are represented by a HTTP Post request to the Evaluation Service. The reason we chose this service is because it allows for multiple retries and excellent monitoring.
5. Cloud Functions: We use it for the piece of code which creates and evaluation submission entry in a database and submits a task to the Cloud Tasks service (**Function#1)**, and we decided to use it because it can “run our code with zero server management.” and it is automatically scaled based on the load, so if a lot of users submit solutions to problems at the same time we don’t have to worry that our servers will crash.

Azure Services

1. Azure Cosmos DB: Since we host our users microservice on Azure, it was natural to have the database for that also as a Azure service, and we decided to use Cosmos DB because it guarantees single-digit millisecond response times and it’s really easy to use.
2. Azure Load Balancer: As the number of users will grow, to reduce the waiting time after an evaluation submission, we decided to create multiple instances of the evaluation service (since multiple code sources can be evaluated at the same time without causing any problems), and to put a load balancer in front of them was the logical thing to do, and this service meets our needs.
3. Azure Virtual Machines: Due to the nature of our evaluation service, which needs access to OS support for files and processes (which are not available in serverless platforms), our only choice was to host it on an OS set up from scratch, and this service was the best choice.
4. Static Web App: We use it for hosting the Angular Web App because it’s very easy to use and we don’t have to worry about creating a server which would serve our static files that are created after building the app for production (**ng build --prod**). Also, it comes with an integrated CI/CD pipeline which is easy to set up and allows us to always have the latest version of the app running in the cloud.

5. Azure App Service: We use it for the users microservice. On the service web page the following statement is written: “Quickly build, deploy, and scale web apps and APIs on your terms.”, and that’s exactly what we needed.

Other Technologies

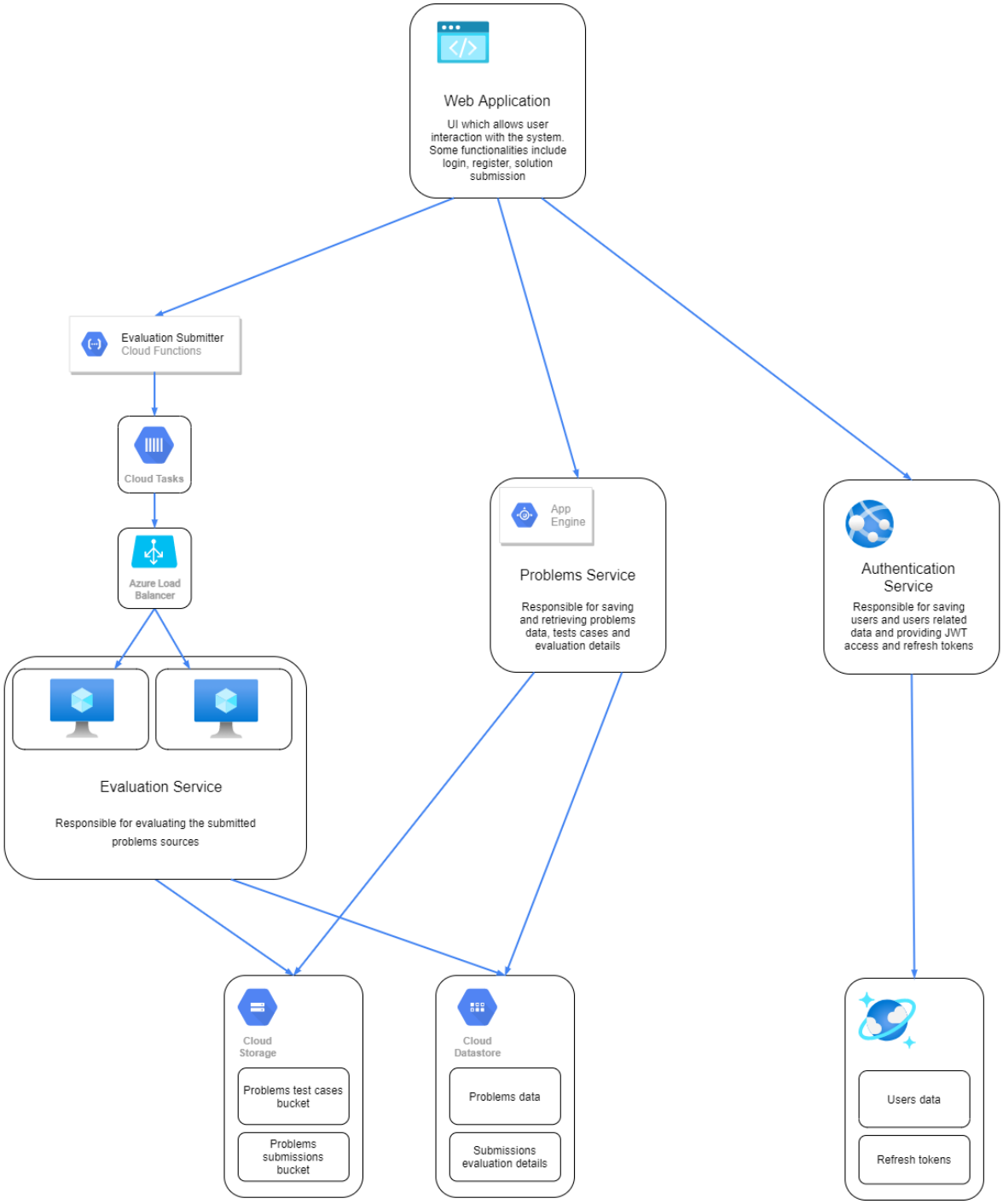
- For the user authentication we have used JSON Web Tokens (JWT), because it’s an easy, secure and fast way to store claims and user auth information. When an users logs in he gets a refresh token, and using it he can get access tokens which must be sent to either of the microservices as a parameter in a HTTP header so he can be authorized.
- For the payment processing an idea would be to use Stripe, if that part would be implemented by the end of the semester. The argument for this is because it doesn’t make sense to implement a payment service from scratch, because we would need to store credit cards and other related information, and Stripe takes care of all this and offers a simple API for transactions.

Business Canvas

1. https://docs.google.com/drawings/d/1Zw3LCAbN8QMqJo7ui7XTiFRwyf9DlvuNhP7L2LRPse0/edit?usp=sharing

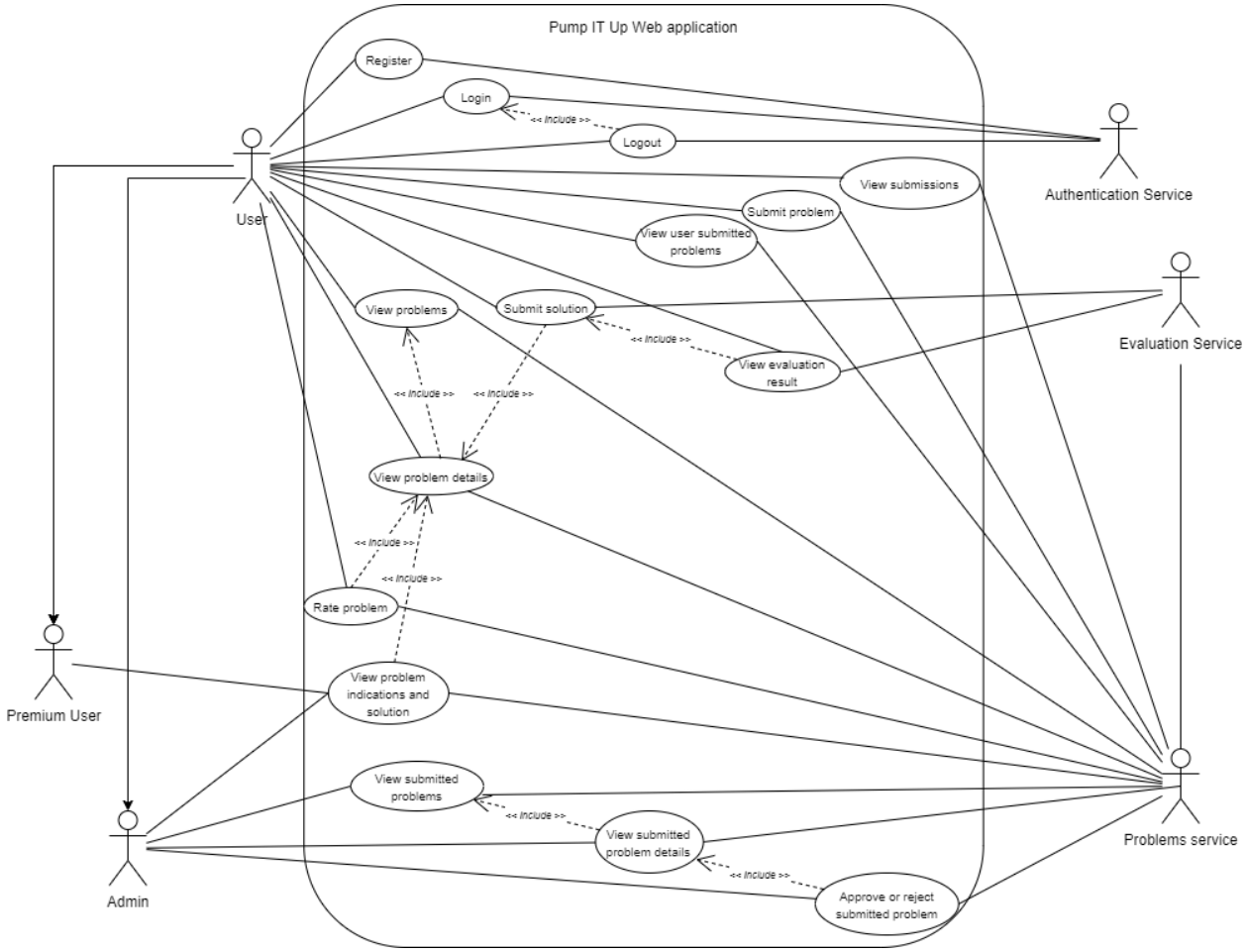
Architectural Diagram

1. **Draw.io diagram:** https://drive.google.com/file/d/1H9YH_80xb0zMnW0cWxnqnw-wcsdajJKr/view?usp=sharing
2. **image url diagram:** browser image url



Use Case Diagram

1. **draw.io diagram:** https://drive.google.com/file/d/1RlKBe0hI1VHIsHVFNlB1eFzGzAY53DtH/view?usp=sharing
2. **image url diagram:** browser image url



APIs

1. **Service#1 API:** https://app.swaggerhub.com/apis/alexoloieri/ProblemsMicroservice/1.0.0
2. **Service#2 API:** https://app.swaggerhub.com/apis/OloieriAlexandru/UsersMicroservice/1.0.0
3. **Service#3 API:** https://app.swaggerhub.com/apis/OloieriAlexandru/Evaluator/1.0.0
4. **Function#1 API:** https://app.swaggerhub.com/apis/OloieriAlexandru/EvaluationSubmission/1.0.0

Functionality Flows Documentation

1. **Pdf on Google Drive:** https://drive.google.com/file/d/13OtM8pgFtZZIZsQigfPA6Yu_msnxjZM6/view?usp=sharing

2. **Pdf on GitHub:**
   https://github.com/Roxanica123/CC-Homeworks/blob/master/Homework5/all.pdf