

Degree Thesis, Åland University of Applied Sciences, Bachelor of Information
Technology

PICTIONARIZER

- “Pictionarize, visualize, and personalize”

Shinichi Takagi



2020:44

Date of approval: 17.12.2020
Academic supervisor: Agneta Eriksson-Granskog

DEGREE THESIS

Åland University of Applied Sciences

Study program:	Informationsteknik
Author:	Shinichi Takagi
Title:	Pictionarizer - “Pictionarize, visualize, and personalize”
Academic Supervisor:	Agneta Eriksson-Granskog
Technical Supervisor:	

Abstract
<p>My project is a web application that I made for language learning. With this application, users are able to visualize and personalize what they are learning when they try to build vocabulary.</p> <p>The application is equipped with basic CRUD (create, update, delete) functionality and some SNS-like functions, such as login, like, comment, follow, searching etc.</p> <p>The data is retrieved from the API that I built on my own by Java Spring backend and a MySQL database. All the requests by the users are handled in the form of HTTP requests, and they are processed according to which request type they are.</p>

Keywords
React, TypeScript, Postman, Java, Spring Boot, MySQL, AWS, Docker, CircleCI

Serial number:	ISSN:	Language:	Number of pages:
2020:44	1458-1531	English	39 pages

Handed in:	Date of presentation:	Approved on:
16.12.2020	17.12.2020	17.12.2020

EXAMENSARBETE

Högskolan på Åland

Utbildningsprogram:	Informationsteknik
Författare:	Shinichi Takagi
Arbetets namn:	Pictionarizer - "Pictionarisera, visualisera, och personalisera"
Handledare:	Agneta Eriksson-Granskog
Uppdragsgivare:	

Abstrakt
<p>Mitt arbete är en webbtjänst jag gjorde för språkinläring. Med hjälp av den här tjänsten kan användare visualisera och personalisera det som de lär sig när de försöker bygga ett ordförråd.</p> <p>Applikationen har grundläggande CRUD-funktionalitet (create, update, delete) och några SNS-liknande funktioner, bland annat att logga in, kommentera, följa, söka osv.</p> <p>Data hämtas genom ett API som jag byggt på egen hand med Java Spring för backend och MySQL för databas. Alla användares förfrågningar tas hand om i formen av HTTP-förfrågningar, och de skickas till olika slutpunkter enligt vilken typ av förfrågningar de är.</p>

Nyckelord (sökord)
React, TypeScript, Postman, Java, Spring Boot, MySQL, AWS, Docker, CircleCI

Högskolans serienummer:	ISSN:	Språk:	Sidantal:
2020:44	1458-1531	Engelska	39 sidor

Inlämningsdatum:	Presentationsdatum:	Datum för godkännande:
16.12.2020	17.12.2020	17.12.2020

TABLE OF CONTENTS

1. Introduction	6
1.1 Purpose	6
1.2 Method	6
1.3 Limitations	7
2. Background	8
2.1 Why web service?	8
3. Technologies	8
3.1 MySQL	8
3.2 Java & Spring Boot	9
3.3 Postman	10
3.4 React	11
3.5 TypeScript	12
3.6 JUnit	13
3.7 Jest & Enzyme	13
3.8 Amazon Web Services	14
3.9 Docker	15
3.10 CircleCI	16
4. Program Structure	17
4.1 User Object	17
4.1.1 Create User	18
4.1.2 Update User Profile	19
4.1.3 Delete User	20
4.2 Word Object	20
4.2.1 Create Word	21
4.2.2 Update Word	21
4.2.3 Delete Word	22
4.3 SNS-like functions	23
4.3.1 Like	23
4.3.2 Comment	24
4.3.3 Follow	25
4.3.4 Recommendation	26
4.4 Other functions	27
4.4.1 Login	27
4.4.2 Searching	28
4.5 Infrastructure	30
4.5.1 Amazon Web Services	30
4.5.1.1 VPC	31
4.5.1.2 EC2	32

4.5.2 Docker	33
4.5.3 CircleCI	33
5. Result	35
6. Conclusion	35
References	36

1. INTRODUCTION

1.1 Purpose

When developing an application, it usually takes the same series of steps. First, it starts by finding an issue that you want to solve. Second, you analyze the cause of the issue. Third, you come up with an idea that addresses the problem. And lastly, you make an application that is based on the idea.

This applies to my project “Pictionarizer” as well. As someone who decided to leave his own country and move to Finland several years ago, I have always felt a strong need for learning new languages, because my native language is Japanese and this language is not commonly used outside of Japan.

One of the biggest challenges in learning a new language is vocabulary building. New words are difficult to remember and easy to forget. Although there are already some “flashcards” apps, I doubt if they really help to better remember new words in the target language, because all it does is essentially the same as “tasteless memorizing”.

This is where “Pictionarizer” comes in handy. Rather than just memorizing, it helps learners visualize and personalize the new words in the target language, so that they can remember what they have learned in context, which makes the process of retrieving memory much more efficient and easier.

This idea of “visualize and personalize” is commonly employed by winners of Memory Championships, and they often mention this fact when they are asked about how they can learn and remember so many new things in such a short time.

In addition to offering a better learning experience, I have one more purpose for working on this project. This is my final year at this university and I am eager to look for a job in the IT industry, and developing an application like Pictionarizer can also demonstrate my skills, so I can use this project as my portfolio in job-hunting.

1.2 Method

This application is delivered in the form of a web service. It mainly consists of CRUD (create, update, delete) functions, and user requests such as button click are handled as HTTP requests. This is not a project given by a company, so there is nothing like a pre-built API, so I needed to build my own API from scratch.

The plan was to make this web service SNS-like, which means that there are various users and they can interact with each other through functions such as “like”, “comment”, “follow” etc. Because of this, it was necessary to keep track of the user’s login state, so that the application has the control over which user has an access to which information, what kind / level of authority the user should be given etc. For example, when you want to update your own user profile, only you should be able to access your “user update” page.

Firstly I made this application in the local environment, which is my own PC, and then I deployed the project in public via Amazon Web Service so that anyone with Pictionary's URL can access the page and use the website.

As I mentioned in the introduction, this application is meant to be used as my portfolio in job-hunting as well. Therefore, it would look more natural to have some active users with various backgrounds already when the project became public, so I had prepared several pieces of user information by INSERT statements in MySQL from the very beginning. Pictionarizer requires login process before it can allow any user to make any CRUD (create, update, delete) related action, but for those people who finds it bothersome to have to create a new account or type in both email address and password to login, I had the app equipped with "easy login" function, with which you can log in as a "Test User" just by clicking the "Easy Login" button.

As for what technologies are used, it is explained in more detail in the section "Technologies" later in this paper.

1.3 Limitations

This project is not a "real social network service" such as Facebook, Twitter, Instagram etc, and I had a limited amount of time to develop the application. Because of this, I needed to make a compromise on certain aspects.

For example, I didn't use a secure way when making the login function. Since the login process requires users to give sensitive information such as passwords, ordinarily the password must be encrypted in a proper way. There are different frameworks or libraries / packages for password encryption depending on which programming language you use, and in my case it was Java, so at a first glance Spring Security, which is offered by Java Spring Frameworks, seemed to be the right choice. However, compared to some other languages' frameworks, Spring Security could be time-consuming, so in order to secure more time for the entire application development I decided not to use it, although I am well aware of the vulnerability.

Another example is about handling multiple simultaneous requests. In real life situations such as Twitter, Instagram etc, what can be expected is that a tremendous number of users access the same page at the same time. This means that the server needs to have the capacity to handle it, and the server will crash if it is not properly taken care of. On an enterprise level this must surely be taken into account, but in my case, I decided that the priority for this matter would not be so high.

There are also a lot of functions that I would like to work on. For example, when searching for a particular user by entering a keyword in the search box, it will be more user-friendly if the search box is equipped with an auto complete function as well. But this is so-called "nice to have" function, not a "must have" function. Nice to have functions are also to be added when there is extra time for development, but when there isn't, it can be omitted.

2. BACKGROUND

2.1 Why web service?

I have two reasons for choosing a web service as my project. One of them is that web development skills are currently in great demand. In this Internet era, websites are ubiquitous. Most companies have their own website regardless of the company size. New applications are downloaded to PCs and mobile devices all the time, and they are distributed online. In this kind of circumstances, working on something in the field of web development means to give a supply.

The other reason is that my final project can also be a compilation of what I have learned here so far. Web development consists of frontend, backend, database, and infrastructure. This university's IT program's curriculum covers programming languages for frontend and backend, and the basics of databases. Although I didn't have an opportunity to study infrastructure during the curriculum, I thought this would be a great chance to work on it by myself.

3. TECHNOLOGIES

3.1 MySQL

Databases are useful in web services, because what happens when you use a web service is that some pieces of data are created, updated, or deleted. Your user information or any other information displayed on the screen is a result of retrieving the data from the database.

In my application, a database is necessary, as without it the changes that have been made on the data will be refreshed and lost every time the project gets restarted.

I chose MySQL for my project's database. It is a relational database management system, in which the data is stored in the form of a table which consists of rows and columns (Oracle, 2020).

One of the reasons I chose MySQL is that it is one of the most popular database management systems in the world. As of November 2020, it is the second most popular database in the world, as shown in Figure 1 (DB-ENGINES, 2020).

360 systems in ranking, November 2020

Rank			DBMS	Database Model	Score		
Nov 2020	Oct 2020	Nov 2019			Nov 2020	Oct 2020	Nov 2019
1.	1.	1.	Oracle +	Relational, Multi-model	1345.00	-23.77	+8.93
2.	2.	2.	MySQL +	Relational, Multi-model	1241.64	-14.74	-24.64
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model	1037.64	-5.48	-44.27
4.	4.	4.	PostgreSQL +	Relational, Multi-model	555.06	+12.66	+63.99
5.	5.	5.	MongoDB +	Document, Multi-model	453.83	+5.81	+40.64
6.	6.	6.	IBM Db2 +	Relational, Multi-model	161.62	-0.28	-10.98
7.	↑8.	↑8.	Redis +	Key-value, Multi-model	155.42	+2.14	+10.18
8.	↓7.	↓7.	Elasticsearch +	Search engine, Multi-model	151.55	-2.29	+3.15
9.	9.	↑11.	SQLite +	Relational	123.31	-2.11	+2.29
10.	10.	10.	Cassandra +	Wide column	118.75	-0.35	-4.47
11.	11.	↓9.	Microsoft Access	Relational	117.23	-1.02	-12.84
12.	12.	↑13.	MariaDB +	Relational, Multi-model	92.29	+0.52	+6.72
13.	13.	↓12.	Splunk	Search engine	89.71	+0.30	+0.64
14.	14.	↑15.	Teradata +	Relational, Multi-model	75.60	-0.19	-4.75
15.	15.	↓14.	Hive	Relational	70.26	+0.71	-13.96
16.	16.	16.	Amazon DynamoDB +	Multi-model	68.89	+0.48	+7.52
17.	17.	↑25.	Microsoft Azure SQL Database	Relational, Multi-model	66.99	+2.59	+39.37

Figure 1. MySQL is ranked as the second (DB-ENGINES, November 2020)

In fact, many large companies like Google, Facebook etc choose MySQL for their databases, which demonstrates its reliability (MySQL, 2020).

Learning MySQL is included as a part of the curriculum I have been taking at the university, so I also feel comfortable using this particular database.

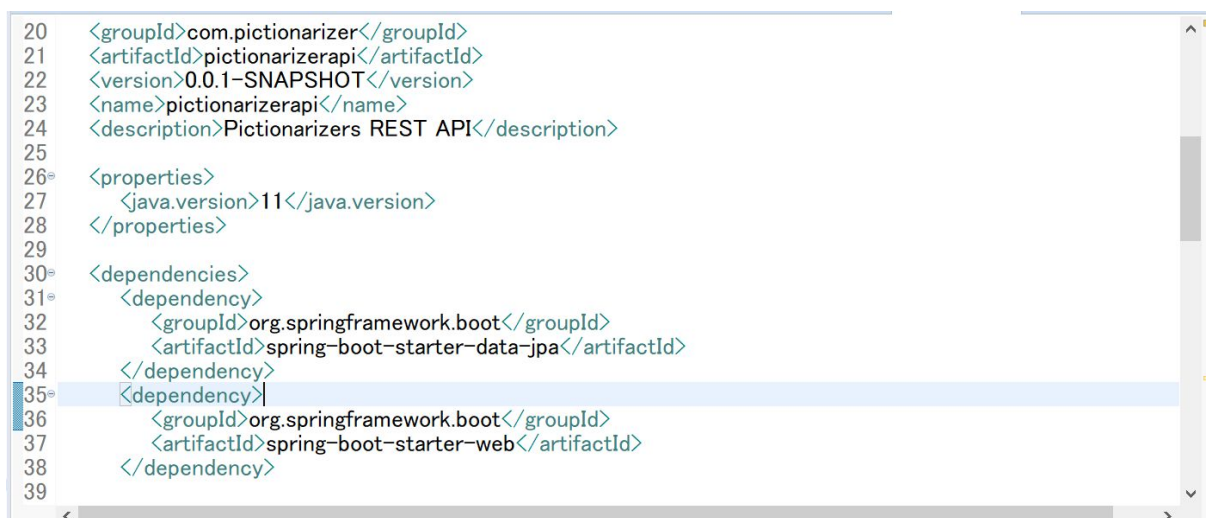
3.2 Java & Spring Boot

This project's backend code is written in Java (GitHub/Shinichi1125/Pictionarizer, 2020). It is one of the most widely used programming languages, and also the main language taught at this university (Deitel, 2017).

Java is an object oriented programming language, and statically typed language as well. In object oriented programming, developers build a program that consists of classes. A class consists of attributes and methods. By breaking the entire program down into smaller chunks like this, it will be easier to organize and manage the project. In statically typed programming, you have to explicitly declare what data type you are using for a variable. This helps the program to be less error-prone, because when there is a data type conflict it is easier to find where the bug occurred

Spring Boot is one of the frameworks for Java, and it is quite helpful especially in the Enterprise context, in other words web application development (TutorialsPoint, 2020). Back in the days when such a framework didn't exist, developers had to do a massive amount of configurations before they could start a project. As a developer everybody wants to focus on developing their project, spending less time for configurations.

Figure 2 is a part of my project's dependencies. It is defined in the form of XML.

A screenshot of a code editor showing XML code for project dependencies. The code is as follows:

```
20 <groupId>com.pictionarizer</groupId>
21 <artifactId>pictionarizerapi</artifactId>
22 <version>0.0.1-SNAPSHOT</version>
23 <name>pictionarizerapi</name>
24 <description>Pictionarizers REST API</description>
25
26< properties>
27   <java.version>11</java.version>
28 </properties>
29
30< dependencies>
31   <dependency>
32     <groupId>org.springframework.boot</groupId>
33     <artifactId>spring-boot-starter-data-jpa</artifactId>
34   </dependency>
35   <dependency>
36     <groupId>org.springframework.boot</groupId>
37     <artifactId>spring-boot-starter-web</artifactId>
38   </dependency>
39
```

Figure 2. Dependencies of my project

Spring frameworks also include others such as Spring Data, Spring Security etc. As mentioned in the “Limitations” section (1.3) I didn't use Spring Security this time, but I wrote a lot of code in which I made the most of Spring Data in order to effectively access the database.

3.3 Postman

When the database is ready and the backend logics including the API parts are written to some extent, the next thing that you want to do before you further develop the project is to test the logics actually work. Of course it is possible to check if they work by writing some frontend code and make a HTTP

request and have the backend process it and see if it retrieves the expected data from the database, but this could make things unnecessarily more complex.

When you test the backend logics, it is commonplace that there is something wrong with the logic and you don't get the value as you expected. But if you test the backend logics by an HTTP request sent from the frontend, it will be hard to figure out whether it was the frontend's fault or backend's fault. It is also a tedious task to make an HTTP request on the frontend side just to test the logic works. Therefore, it makes more sense to begin to work on building the frontend part after making sure that everything is correct with the backend logics (DigitalCrafts, 2020).

And this is where Postman comes into play. Postman enables you to test the backend logics without worrying about the frontend part at all. All you need to do is to choose the request type from GET, POST, PUT, DELETE, specify the HTTP request URL and its parameter(s), and click the send button. You will get a HTTP status and a return value if the method which got called has a return type.

Figure 3 below demonstrates that I successfully confirmed that the HTTP request works fine and it obtains the user name as expected in the local environment.

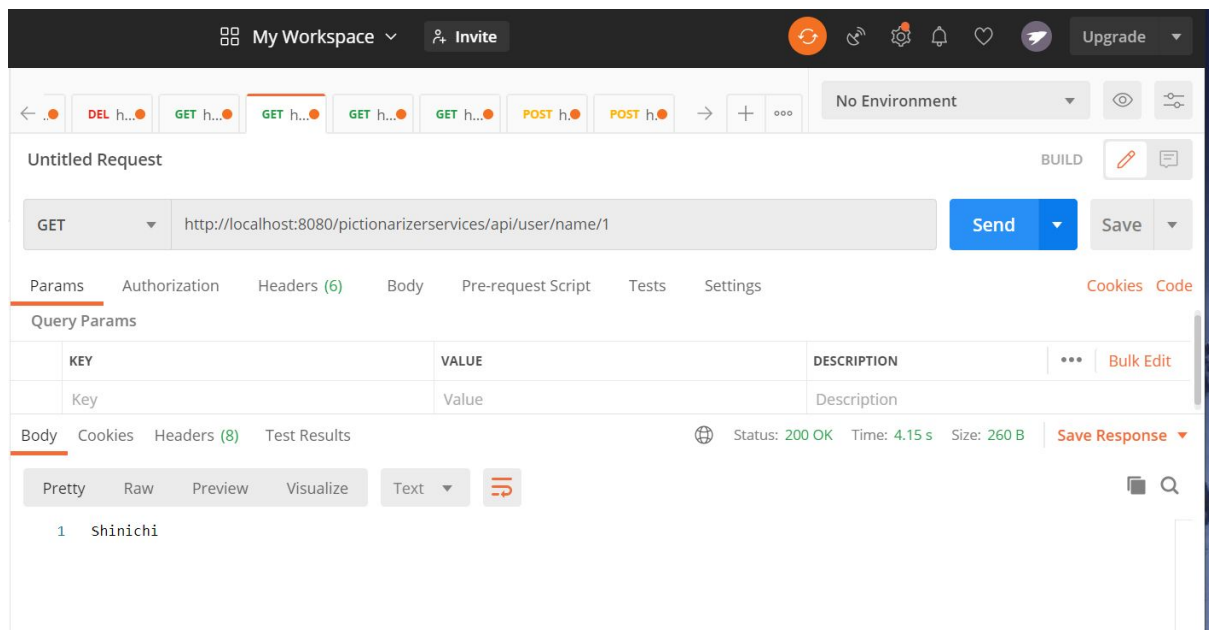


Figure 3. `getUserName` method is working in my local environment

3.4 React

React is one of the JavaScript frameworks which was developed by Facebook 2011 (Skillcrush, 2020). It is currently (as of 2020) the most popular JS framework, according to Stack Overflow Trends (Stack Overflow, 2020).

Figure 4 illustrates the trend of JavaScript libraries / frameworks in the past 10 years. jQuery used to be dominant, but since the emergence of the three other frameworks its trend has been downwards. Although both Vue.js and React have been steadily gaining more attention, React has a steeper curve.

Stack Overflow Insights > Trends

Stack Overflow Trends

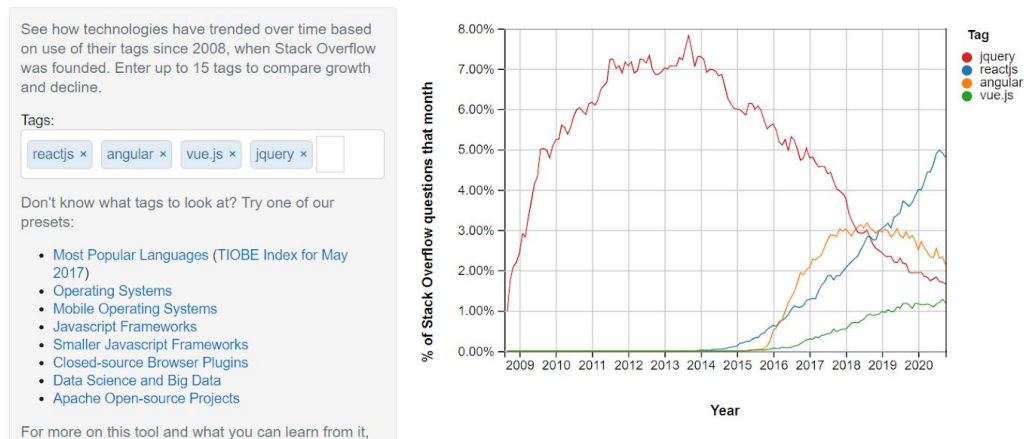


Figure 4. React is currently gaining the biggest attention (Stack Overflow Trends, November 2020)

React is component based. This means that one big project can be divided into many smaller components. Each of the components is reusable and also can be partially customized, so this leads to much less repetitiveness in code, resulting in making the code considerably cleaner and more readable, which also reduces developers’ workload.

One of the important concepts in React is ‘state’. It keeps track of the value of the state, and the page can rerender depending on what the current state is. This makes it very easy to conditionally render a component. For example, suppose you visit a certain user’s page. By checking if your login ID matches the ID displayed on the user page, the component makes sure that the “edit profile” button shows up only if you visit your own user page, and the button won’t be displayed when you visit any other user’s user page.

3.5 TypeScript

TypeScript is a superset of JavaScript, and perfectly compatible with React, which is a JavaScript framework (Dreimanis, 2020). As visualized in Figure 5 below, TypeScript can do anything that JavaScript can do, and on top of that it has some more features.

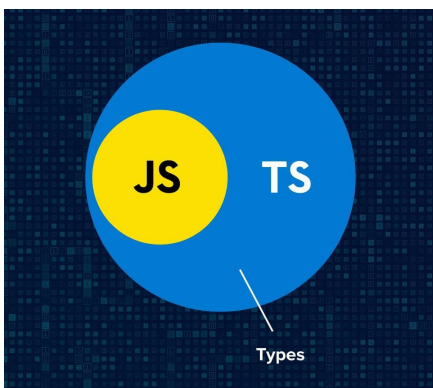


Figure 5. TypeScript is a superset of JavaScript (Dreimainis, November 2020)

As the name TypeScript implies, it has a different feature than JavaScript in handling data types. Whereas JavaScript is a dynamically typed language where you don't have to specify the data type when declaring a variable, you must explicitly declare the data type in TypeScript. In other words, TypeScript is a statically typed language.

One of the big benefits of being stricter about data type declarations is that it can lead to less error-prone code, especially in bigger projects (Dreimainis, 2020). As the project grows bigger, the chances are that developers may sometimes get confused with which variable has which data type and write some code with a wrong assumption about the data type. This will cause a data type conflict, but TypeScript detects the conflict even before actually running the project, which makes it easier to find a bug.

Also, writing code in a statically typed language requires you to be explicit, so this is a good way of gaining a better understanding of your own code, which can eventually lead to a better practice (Dreimainis, 2020).

3.6 JUnit

When a program is written, you want to make sure that all the functions within it work as expected, and this is why testing exists. There are two types of testing. One is manual testing, and the other is automated testing (TutorialsPoint, 2020). The former doesn't necessarily require programming skills. You can just run the program, provide some inputs via the user interface such as clicking a certain button, writing some text in a text field and see how the program responds. The latter has the code test a certain functionality. In an automated test like this, all that is required is to just run the test by a click of a button.

Writing automated test cases is quite important, especially when the size of the project grows (Ascendro, 2013). At some points of the development, there is a necessity to make some changes in code, and fairly often the change can affect the old functionality that you wrote a long time ago. With an automated test, you can check if the change you recently made in the code affects some older parts of the code by simply taking a look at whether the test case for the older part still passes or fails. This is a much more reliable way of protecting the logic in the entire project, as we humans always make mistakes and sometimes even overlook them.

JUnit is a testing tool for Java (TutorialsPoint, 2020). It provides you with a lot of useful functions just by adding an annotation or importing a library, which makes the testing process simpler, as shown in Figure 6.

```

27
28 @ExtendWith(SpringExtension.class)
29 @WebMvcTest
30 public class PictionarizerapiUserControllerTests {
31
32     @MockBean // should be @MockBean instead of @Mock as it's Spring's Bean
33     private UserRepository userRepository;
34
35     @MockBean
36     private WordRepository wordRepository;
37
38     @Autowired // have Spring instantiate the field
39     private MockMvc mockMvc;
40
41     private int alexId = 28;
42
43     @Test
44     @DisplayName("When an update request, the User data gets updated properly, and the updated User data gets")
45     public void testUpdateUser() throws Exception {
46         // User before update
47         User existingUser = new User();

```

Figure 6. Annotations for testing with JUnit

3.7 Jest & Enzyme

Jest is a test-runner made by Facebook, able to test any JavaScript environment (JEST, 2020). And it's most frequently used to test React applications. It also has the ability to mock functions in a very simple way as you can see on line 14 and 27 in Figure 7, so that you can focus on testing the smallest unit of test in an isolated environment, without worrying about dependency problems.

```

10 const axios = require('axios');
11 jest.mock('axios');
12
13 describe('Login', () => {
14     const mockHistoryPush = jest.fn();
15
16     const props: any = {
17         history: {
18             push: mockHistoryPush
19         },
20     };
21     const login = mount(<Login {...props}/>);
22
23     const { location } = window;
24     // delete window.location, and then replace the reload part with a mock object
25     delete window.location;
26     window.location = {
27         reload: jest.fn(),
28     } as any;

```

Figure 7. Mocking functions with Jest

Enzyme is a library developed by airbnb (Airbnb, 2020). It is a testing utility library for React that allows you to work with specific components, including testing the effects of DOM behaviors such as click, typing input etc. It also gives you the ability to find some specific items in the component itself.

With Jest and Enzyme combined, you are fully equipped to test a React application.

3.8 Amazon Web Services

Amazon Web Services or AWS, refers to a collection of more than 100 cloud computing services that Amazon provides (AWS, 2020). Cloud computing refers to various services involving computers that you can use via the Internet, such as server, storage, database, software etc.

In cloud computing, all you need is just one PC and an internet connection, and you are all set to use a server, high volume storage, high speed database etc as much as you need.

Before cloud computing emerged, it was necessary to prepare a physical server, for example if a company wanted to use a server. This way of using a server is called on-premise. In the case of on-premise, not only you have to purchase a physical server and manage it by yourself, you also have to secure enough space to place the server. Besides, purchasing a server alone also costs money.

On the other hand, in cloud computing, you don't have to worry about any of the concerns mentioned above. As long as you connect to the cloud computing services via the Internet, you can use the resources such as the server you need right away. Amazon Web Services made the computing resources far more easily available compared to the traditional way.

Figure 8 is a screenshot of my AWS console's EC2 instance page. I can launch an application thanks to AWS, without having a physical server.

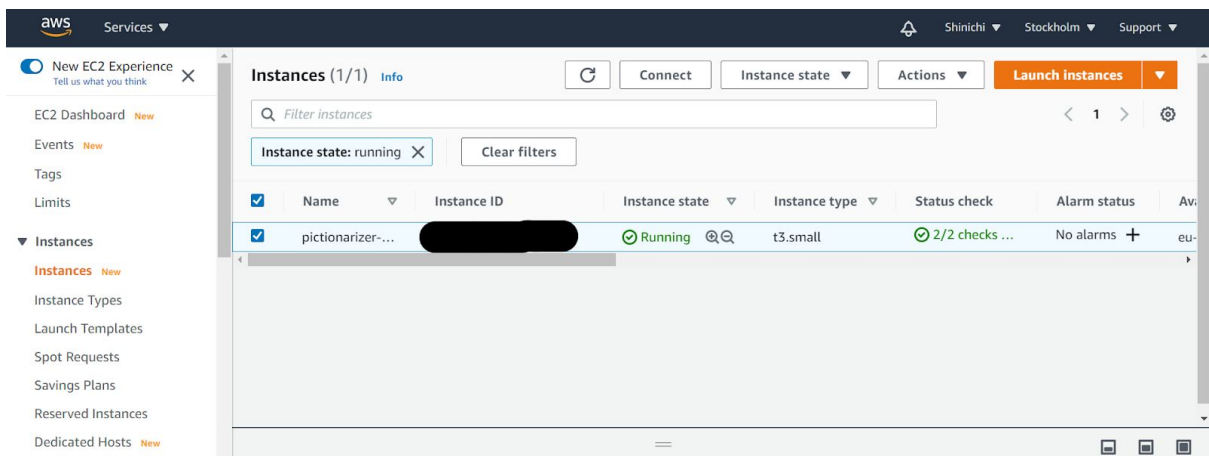


Figure 8. My AWS console displays the information about the EC2 instance that is currently running

3.9 Docker

Docker is a platform where you can create, distribute, and execute a container type virtual environment (opensource.com, 2020). Docker employs the container technology of Linux, and it is often compared with virtual machines.

One of the Docker's big characteristics is that it manages middleware installations and a number of configurations by code. This is called Infrastructure as Code or Iac, and it generates benefits like the ones mentioned below.

- By sharing the configuration files written in code, anybody can create the same environment
- It is easy to distribute the environment that you created

For example, it commonly happens that an application you were building in your development environment was working fine on Windows but it didn't go as expected on the Linux environment because of the different OS, different version of certain softwares etc. This will be easier to prevent by making the most of the benefits that Docker offers, because it considerably reduces the gap between the different PC's environments, which shortens the time for the development environment preparation.

Figure 9 illustrates the difference between a virtual machine and a docker container, and how simpler the latter is.

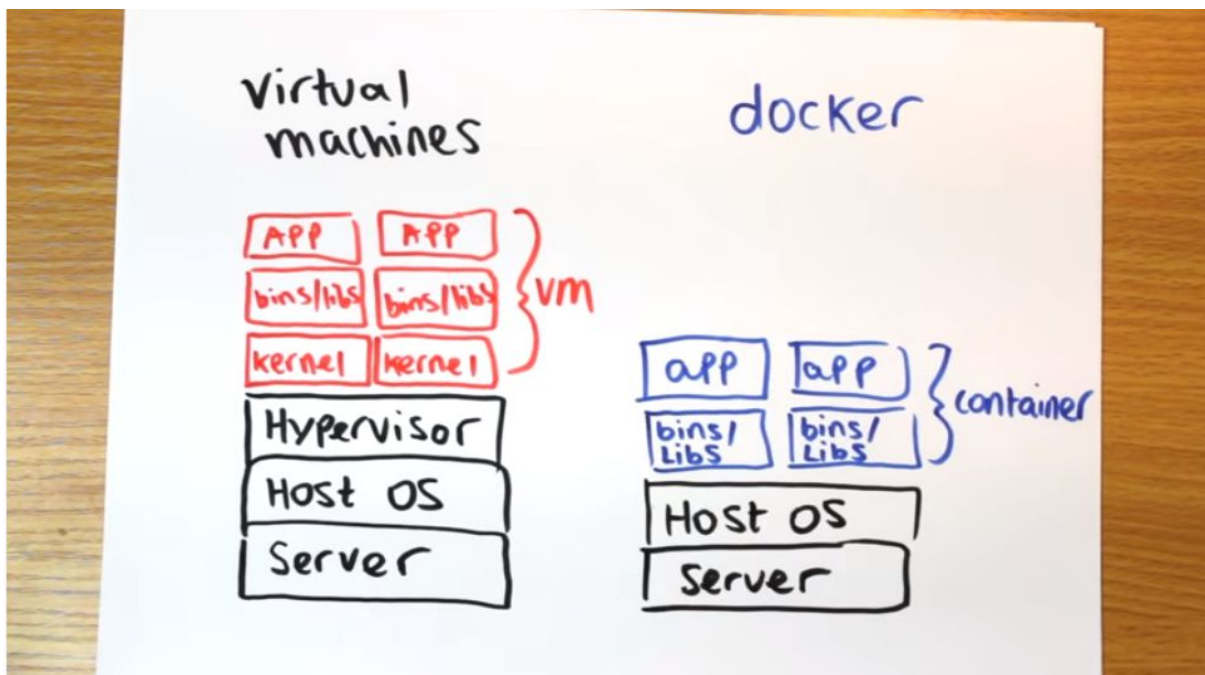


Figure 9. Virtual machines vs Docker containers (Wright, 2016)

Also, Docker makes the process of CI (continuous integration) easy, when it's combined with a CI tool like Jenkins / CircleCI. When the process of testing and building and deployment of the project is automated, the entire process of the development becomes much smoother.

3.10 CircleCI

CircleCI is one of the CI (Continuous Integration) tools (circleci.com, 2020). Continuous Integration means that build and test are frequently and continuously done in a software development in order to find problems on an early stage and make the development more efficient. It can also more specifically refer to the automation of the process by employing the CI tool.

CircleCI's main functionality is the following three things.

1. Build...it builds an executable application from the source code
2. Test...it executes test code to make sure that the application behaves in the expected way.
3. Deploy...once the application passes the build and test phases, it releases the project to the test environment or product environment.

Figure 10 is a screenshot of my own CircleCI dashboard. As soon as I commit and push on Git, it automatically builds, tests, and deploys the project so it will reflect the update.

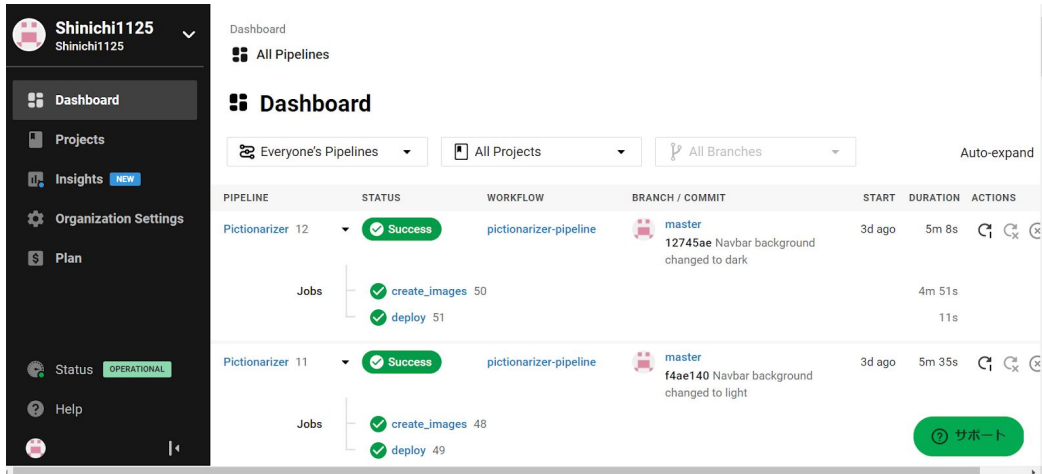


Figure 10. Build, Test, Deploy went successfully

CircleCI is SaaS (Software as a Service), and because of this, it has both pros and cons compared to Jenkins, which is another CI tool.

One of the benefits of CircleCI is its low cost. CircleCI is just a software that sits on top of the platform and infrastructure that someone else takes care of. And this leads to another merit. Unlike Jenkins, you don't have to build your own server, nor do you need to spend time on server maintenance or handle the problem when it breaks down.

However, CircleCI has some restrictions as well. For example, when a technical issue occurs, even if you want to analyze the cause of the trouble and fix the problem, it will be quite difficult because some parts of CircleCI is dependent on a third party's service. Also, it's assumed that the version control is handled by GitHub, so if you are using SVN (Apache Subversion, which is another version control system), CircleCI will not be applicable to that project.

4. PROGRAM STRUCTURE

The core part of the application “Pictionarizer” consists of two objects: User and Word. It is self-evident that an application needs users, so there should be no question about why we need User objects. As for Word objects, it is essential too, as Pictionarizer is an application which helps users build their vocabulary in their language learning.

As the application is meant to work like Social Network Services and it has SNS-like functions such as “like”, “comment”, “follow” etc, and there are also some other features like “recommendation”, “search user / word by keyword” etc. We will dive deeper into each of them.

4.1 User Object

User objects are defined in the following form, as shown in Figure 11.

```
CREATE TABLE user
(
  id int AUTO_INCREMENT,
  name varchar(32) NOT NULL,
  own_language varchar(16) NOT NULL,
  target_language varchar(16) NOT NULL,
  country varchar(16),
  email varchar(32) NOT NULL,
  password varchar(256) NOT NULL,
  image longblob,
  description varchar(255),
  PRIMARY KEY (id),
  UNIQUE KEY (email)
);
```

Figure 11. User Table

Each user has a user ID so that everyone can be uniquely identified. In order to avoid duplication, email is set as a unique key, which means that there shouldn't be multiple users sharing the same email address.

4.1.1 Create User

A user can jump to the “Create User” page by clicking on the “Sign up” link on the right side of the header. This “Sign up” link only appears when the user is in the state of “logged out”. See Figure 12.

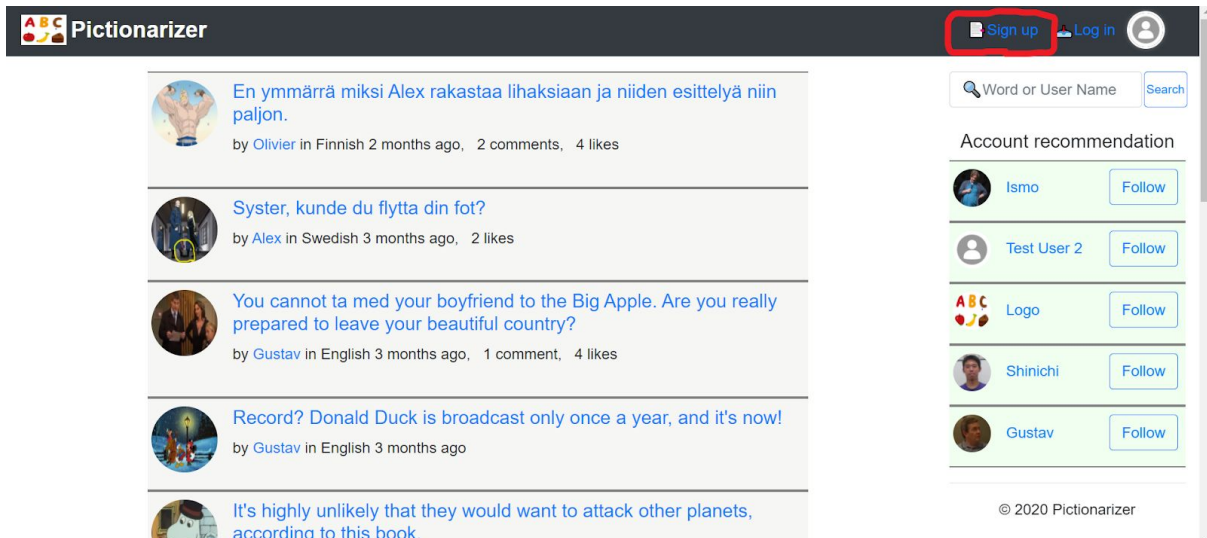


Figure 12. Sign Up link is displayed only when the user is not logged in

The Create Account page provides input fields for the user's name, target language, native language, country, email address, password, description, and file selector to choose a picture for the user profile. Some fields are mandatory, and if the "Sign up" button is clicked without filling in them, the POST request will not be sent to the backend and an error message will be displayed to let the user know that he / she hasn't filled out the form properly.

As mentioned in 4.1 User Object section, each email address must be unique, so if a user fills in an email address that is already used by someone else and send a POST request, the backend will handle the request as an exception and return an error message, which will be displayed on the frontend side.

It is not mandatory to set a profile picture when creating a new account. If all the required text fields are filled and the "Sign up" button is clicked without choosing a picture, the default avatar will be automatically set as the new user's profile picture.

If a new account is successfully created, the user's login state will automatically change to "logged in", and he or she will be redirected to the top page.

4.1.2 Update User Profile

The "Update User Profile" page will show up when the user clicks the "edit" link on the user information page. User information pages are accessible to anyone and it is visible even when the user is logged out, but the "edit" link appears only when the user visits his or her own user information page.

Figure 13 shows that my user profile "Shinichi" displays the option "Edit profile", as I am visiting my own page after logging in, as the picture on the top right corner indicates.

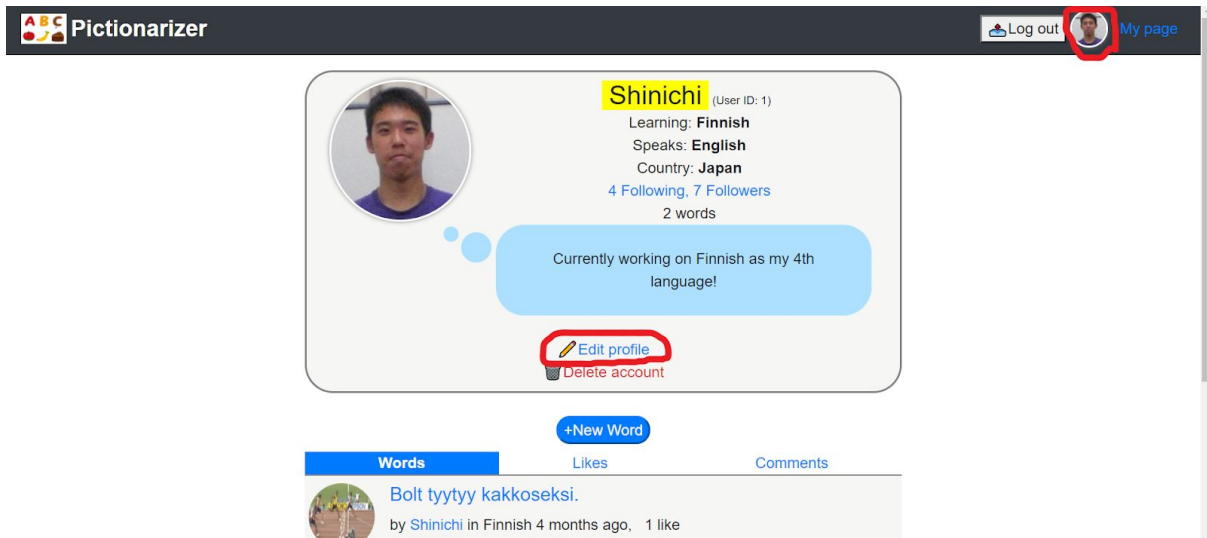


Figure 13. “Edit profile” appears when you visit your own profile page after logging in

When the user is redirected to the “Update User Profile” page, the fields are already filled with the existing data of the user that is fetched by the database, but other than that, how the form works is mostly the same as the “Create User” page.

4.1.3 Delete User

The “Delete User” page will show up when the user clicks the “delete” link on the user information page. User information pages are accessible to anyone and it is visible even when the user is logged out, but the “delete” link appears only when the user visits his or her own user information page.

It takes a further step to actually delete the existing user account. The user has to enter his or her email address and password before he or she can send a delete request. This is to prevent users from accidentally deleting their accounts with an unintended click on the “delete” button. Figure 14 shows how it looks.

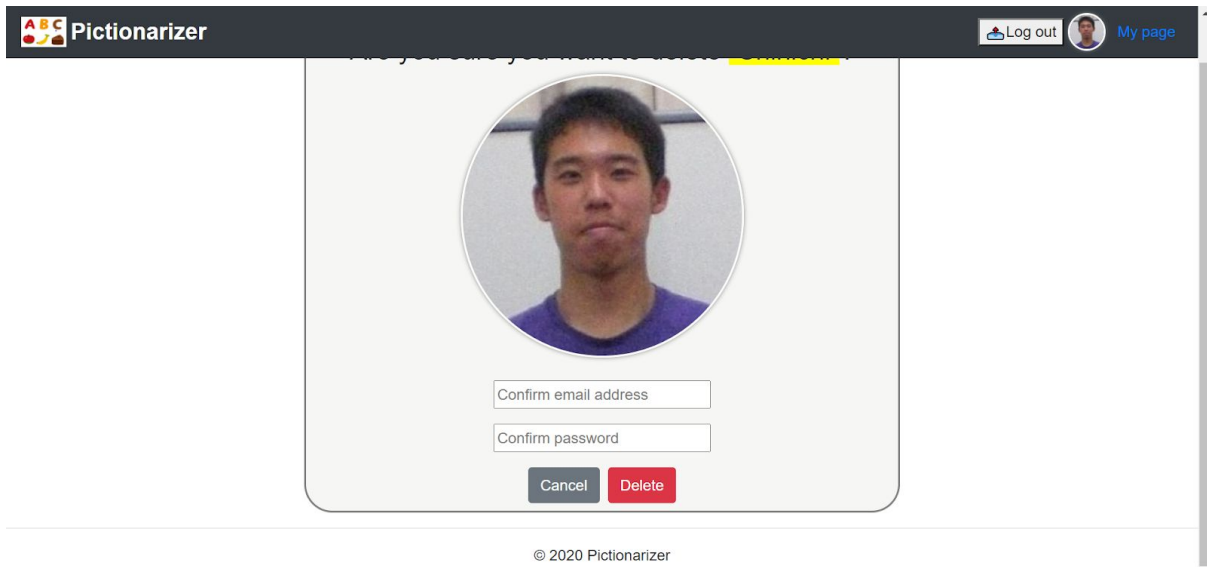


Figure 14. Deleting a user requires the email and password confirmation

If the correct combination of email address and password is provided when clicking on the “delete” button, the account will be deleted, and the user will automatically log out and get redirected to the top page.

4.2 Word Object

Word objects are defined as Figure 15 below.

```
CREATE TABLE word (  
  id int AUTO_INCREMENT,  
  user_id int,  
  own_lang_word_name varchar(64) NOT NULL,  
  target_lang_word_name varchar(64) NOT NULL,  
  own_lang_ex_sentence varchar(255),  
  target_lang_ex_sentence varchar(255),  
  created_date datetime NOT NULL,  
  image longblob,  
  PRIMARY KEY(id),  
  CONSTRAINT fk_user FOREIGN KEY (user_id) REFERENCES user(id)  
);
```

Figure 15. Word Table

Besides the id that identifies which word object it is, there is a user_id as well so that it identifies which user created the word object. As the purpose of the application “Pictionary” is to visualize and personalize words in building vocabulary, word objects contain a number of variables to give learners enough context, including varchar type variable for storing an example sentence in the target language / the learner’s own language, longblob type variable to store an image that well-describes the word.

4.2.1 Create Word

A user can jump to the “Create Word” page by clicking on the “New Word” button on top of the words list on the top page. Alternatively, a user can find the same “New Word” button on his or her own user profile page, as shown in Figure 13.

The Create Word page provides input fields for the word in the target language, the word in the user’s own language, example sentence in the target language, example sentence in the user’s own language, and file selector to choose a picture that describes the situation of the example sentence. The values for user ID and Created Date are automatically set. If the submit button is clicked without filling in the mandatory fields, the POST request will not be sent to the backend and an error message will be displayed to let the user know that he / she hasn’t filled out the form properly.

In creating a word object, providing an image is mandatory, because visualizing is the core feature of this application. If a user tries to create a word object without specifying a picture, an error message will be displayed as in Figure 16.

Figure 16. The error message for not choosing any picture

4.2.2 Update Word

The “Update Word” page will show up when the user clicks the “edit” link on the word details page. Word details pages are accessible to anyone and they are visible even when the user is logged out, but the “edit” link appears only when the user visits the page of the word that he or she created on his or her own.

When the user is redirected to the “Update Word” page, the fields are already filled with the existing data of the word object that is fetched by the database (See Figure 17), but other than that, how the form works is mostly the same as the “Update User” page.

Figure 17. The data fields are already filled with the existing data

4.2.3 Delete Word

The “Delete Word” page will show up when the user clicks the “delete” link on the word details page. Word details pages are accessible to anyone and they are visible even when the user is logged out, but the “delete” link appears only when the user visits the page of the word that he or she created on his or her own.

Just clicking the “delete” alone does not actually delete the word object. When a user clicks the “delete” link, he or she will get redirected to a confirmation page, where he or she will be asked if it is really okay to delete the word object (Figure 18). If the “delete” button is clicked here as well, a delete request will be sent to the backend and the data will be deleted from the database. If the “cancel” button is clicked, the user will be redirected to the word details page.

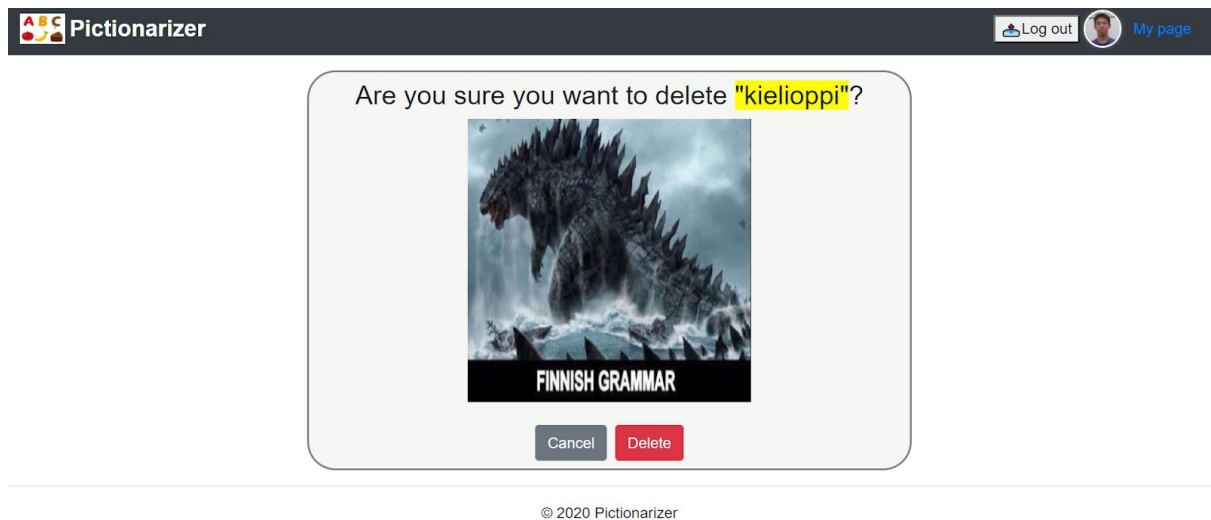


Figure 18. Delete Word Confirmation

When deleting a word object is successful, the user will be redirected to his or her own user information page.

4.3 SNS-like functions

Social Network Services’ core functionality is to let users interact with each other. In order to achieve the goal, it is vital to keep track of which user is communicating with which user / word object. Therefore, on the backend side I made entities that map relationships between users (or user and word).

SNS-like functions assume that the users in question are existing users. This means that you need to be logged in when you want to “like”, “comment” on a word, or “follow” someone. If you click any of these SNS-like buttons without being logged in, the application will not allow the request to be processed and you will be redirected to the login page, which will be further explained in the later part of this paper.

4.3.1 Like

In this project, `like_relation` is defined to keep track of who liked what word object. The table is defined as Figure 19 below.

```
CREATE TABLE like_relation
(
  like_id int NOT NULL AUTO_INCREMENT,
  word_id int NOT NULL,
  user_id int NOT NULL,
  PRIMARY KEY(like_id)
);
```

Figure 19. `like_relation` table

It consists of `like_id` which uniquely identifies the like relationship, `word_id` and `user_id`. More detailed information about the word or user can be fetched either on the backend or frontend side on demand, so the like relation does not have to include any more information.

A user can make a HTTP request by clicking the “like” button. The same “like” button can send either a “like” or “unlike” request, depending on whether the user has already liked the particular word or not.

When the page is loaded, the frontend automatically calls the `isLiked` method, which returns a boolean value to check if the word is already liked by the user. If the value is true, the state `isLiked` is set as true, otherwise false.

The “like” button is displayed and works in a little different ways depending on the state of `isLiked`. If it is true, the button color will be blue and it fires an “unlike” request when it is clicked. If it is false, the color will be still blue but more transparent and it fires a “like” request when it is clicked.

A “like” request is a POST request, and an “unlike” request is a DELETE request from the backend’s perspective. If a “like” request is made, a new `like_relation` will be created on the database, automatically incrementing the `like_id`. If an “unlike” request is made, the `like_id` will be identified by the `word_id` and the `user_id`, and then the “like_relation” gets deleted.

4.3.2 Comment

Figure 20 demonstrates how the comment table is defined.


```

CREATE TABLE comment
(
    comment_id int NOT NULL AUTO_INCREMENT,
    word_id int NOT NULL,
    user_id int NOT NULL,
    text varchar(255),
    date datetime NOT NULL,
    PRIMARY KEY(comment_id)
);

```

Figure 20. comment table

When a comment is posed, the user’s profile picture, name, and the comment text will be displayed. Each comment’s `user_id` is compared to the current user login state, and a delete button appears only if the id matches. In other words, the user can only delete his or her own comment(s).

A comment includes a date. This is to display the information about when the comment was posted right below the comment bubble, something like “10 minutes ago”, “3 days ago” etc. The number changes dynamically, and this is thanks to the data of the date and the JavaScript library “npm moment”, which makes the calculation quite easy and significantly reduces the amount of code. Once the library is installed and it’s imported to the file, all it takes is just a line of code like the one below (Figure 21).

```

<span className="indentation">{moment(date).fromNow()}</span>

```

Figure 21. Dynamically displaying the time by npm moment

4.3.3 Follow

You can see how the `follower_relation` table is defined in Figure 22 below.

```

CREATE TABLE follower_relation
(
    pair_id int NOT NULL AUTO_INCREMENT,
    followee_id int NOT NULL,
    follower_id int NOT NULL,
    PRIMARY KEY(pair_id)
);

```

Figure 22. follower_relation table

It is quite similar to how the `like_relation` is structured. There are states like “isFollowed” and “isFollowing”, and depending on the boolean values of those states, the design of the functionality of

the “Follow” button and whether the pages of certain users display some extra text like “follows you” on their profile.

The `followee_id` and `follower_id` can also be used on the backend side to search for a particular user’s number of followers or how many people he or she is following. It is enabled by some customized SQL queries that I added to the `FollowerRelationRepository` interface. The queries are defined as below (Figure 23).

```
@Query("SELECT fr FROM FollowerRelation fr where fr.followerId = :followerId")
List<FollowerRelation> findAllByFollowerId(@Param("followerId") int followerId);
|
|
@Query("SELECT fr FROM FollowerRelation fr where fr.followerId = :followerId AND fr.followeeId = :followeeId")
FollowerRelation findByFollowerIdAndFolloweeId(@Param("followerId") int followerId, @Param("followeeId") int followeeId);
```

Figure 23. `FollowerRelationRepository` interface

And all you have to do is just call the method in the controller like the example below (Figure 24). This example also shows that you can fetch the number of followers by measuring the length of the array where the fetched result is stored.

```
// fetch the number of followers the user has
@RequestMapping(value = "/no-of-followers/{id}", method = RequestMethod.GET)
public int getNoOfFollowers(@PathVariable("id") int id){
    int noOfFollowers;
    List<FollowerRelation> followerIdList = repository.findAllByFolloweeId(id);
    noOfFollowers = followerIdList.size();
    return noOfFollowers;
}
```

Figure 24. Calling the method via the repository

4.3.4 Recommendation

In Social Network Services, it is commonplace that the system displays a list of users that you might be interested in following. Nowadays this kind of recommendation is optimized by Machine Learning technology, but as I don’t have enough knowledge and experience yet in that field, I used my own algorithm to make it function like recommendations.

In my recommendation algorithm, the backend fetches 4-5 people from the database as a list of people to recommend for the user. The criteria to pick up those people are the following.

- The user has not followed them yet
- They are learning the same target language as the user
- They speak the user’s target language as their own native language

The criteria above are applied when the user is logged in, but if he or she is logged out, the logic mentioned above will not take place, and instead the list of users will consist of people who have been randomly selected. Figure 25 shows a list of recommended accounts when logged in as Shinichi.

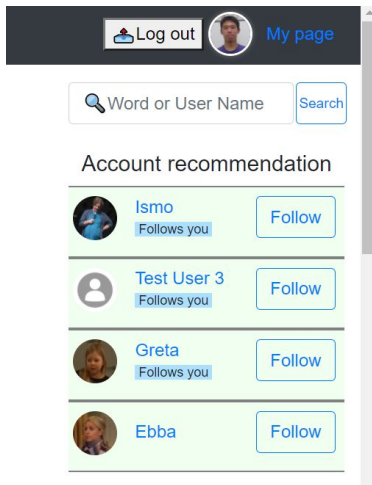


Figure 25. Recommendation for Shinichi

Of course, irrespective of whether the user is logged in or not, the user himself / herself will not be included in the list of users.

4.4 Other functions

4.4.1 Login

In order to be able to tell who the current user is, the system must make sure that the user has logged in when he or she is trying to use any SNS-like function.

There are two ways for a user to reach the login page. One is to click the “login” link on the header, the other one is to click a SNS-like function button such as “like”, “follow” and so on and the user gets redirected to the login page.

On the login page, the user is supposed to enter the email address and password that he or she is registered with. If the combination of the two pieces of information is found on the database, the user will be allowed to log in. If the email address and the password don't match, an error message will be displayed on the screen.

To some people, it is a pain to manually type an email address and password each time to log in. So I added “Easy Login” buttons which allow you to log in as “Test User” just by clicking on it (Figure 26).

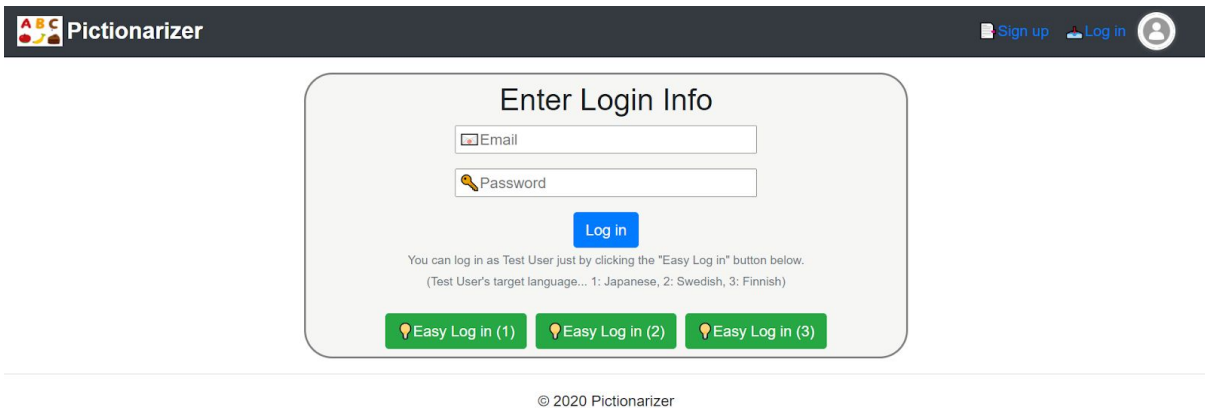


Figure 26. Easy Login available

When the “Easy Login” button is clicked, the `easyLogin` method in the `Login.tsx` gets called. In the `easyLogin` method, the email address and the password that the “Test User” is registered with are assigned to the variable “values” of `LoginInfo` data type, and then the variable gets passed to the `userLogin` method, which makes a GET request (Figure 27).

```
userLogin(loginInfo: LoginInfo){
  return axios.get(`${API_URL}/login`, {
    params: {
      email: loginInfo.email,
      password: loginInfo.password
    }
  });
}
```

Figure 27. Axios get request gets called by calling the `userLogin` method

As mentioned in the **1.3 Limitations** section, a framework like Spring Security is not used here, and this project doesn’t have a logic to safely encrypt passwords before storing them in the database. I’m well-aware of the security risk if this way of handling login were used for the real services. Maybe it is more appropriate to call my login function “something that looks like a login function”, because what I’m doing here is just keeping track of the value of the `localStorage`, which itself isn’t ordinarily used for taking care of login data.

The below is all I’m doing with the `localStorage`. You can see that it just refers to a single number in the local storage as shown in Figure28, and I make it look like a login function by comparing the number stored in the `localStorage` with the user ID returned from the database.

```
const loginId: string = '0';

export const setLoginId = (id: string) => localStorage.setItem(loginId, id);

export const getLoginId = () => localStorage.getItem(loginId);

export const logout = () => localStorage.setItem(loginId, '0');
```

Figure 28. Login ID is stored in the form of localStorage

4.4.2 Searching

In real SNS like Facebook, Twitter, Instagram etc, sometimes you want to search certain users, posts, or tweets. In order to enable users to search something, every SNS has a search box somewhere on the page.

In my case, the search box is located on the right side of the top page. Here a user can enter a keyword, and when the search button is clicked a HTTP request is called.

The value of the search box is kept track of by two way binding. The input field is equipped with `onChange`, which detects any change made in the field and passes the event to the `onHandleChange` method, which is defined as Figure 29 below.

```
onHandleChange(event: { currentTarget: HTMLInputElement; }){
  this.setState({
    searchField: event.currentTarget.value
  });
}
```

Figure 29. Two way binding on the search field

The value of the `searchField` state is always monitored, and the state of the “Search” button also changes according to that. When a user types something in the search box the button becomes blue and clickable, but when the search box is empty (in other words, when the value of the `searchField` is equal to an empty string), the button’s color stays transparent and it remains unclickable (Figure 30).

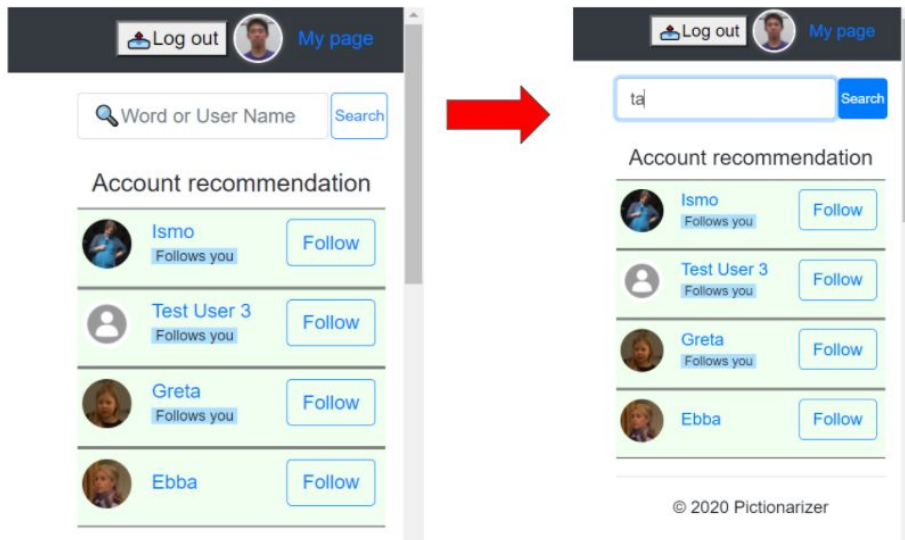


Figure 30. The search button turns blue when the text box is filled

The backend fetches both users and words if the keyword is contained. When the search result is displayed on the frontend, it is filtered by “user” / “word” buttons (Figure 31, 32).



Figure 31. The search result filtered by user name

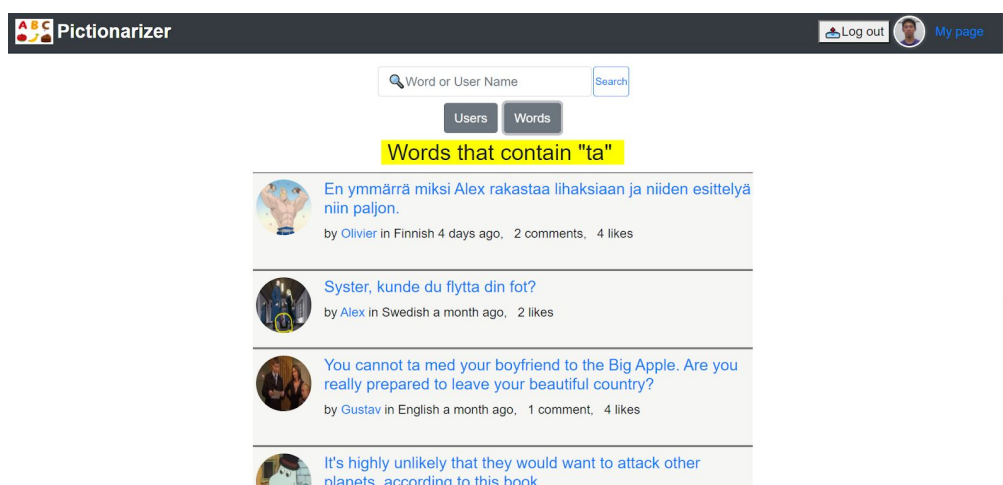


Figure 32. The search result filtered by word

4.5 Infrastructure

If you are working on a web-application development, you probably want to make the app publicly accessible when the development phase is complete. Since the application that has been developed in your local environment can be seen only in your local PC, you will need to establish an infrastructure to make it public.

4.5.1 Amazon Web Services

In the field of cloud computing, there are a few different services provided by different companies. A few of the examples are Amazon Web Services (or AWS) by Amazon, Azure by Microsoft, Google Cloud Platform (or GCP) by Google.

Although each of them has of course both advantages and disadvantages on certain aspects, I decided to choose Amazon Web Services mainly because of its wide usage throughout the world. In terms of worldwide cloud share, Amazon Web Services outweighs the other two, which means that it is the world's most leading service. This also means that it has a lot of resources available, such as question posts on Stack Overflow and various online articles / tutorials, and this helps me learn how the infrastructure works from scratch.

As AWS is highly comprehensive and it takes too many pages to explain everything, I picked up just a few of the important services to explain a bit further, which are VPC and EC2.

4.5.1.1 VPC

AWS provides a number of services, but in order to be able to use them it is not enough with just creating an AWS account. You need to set up a proper environment.

VPC stands for Virtual Private Cloud. This is something like drawing a line between your own territory and the public space, or someone else's territory. For example, taking a look at the physical world, in order to build a house you need some space, and you need to own that space as "your land". The space where your neighbor's house stands is owned by your neighbor, and roads are a good example of "public space".

Something similar happens in the world of the Internet. "From this point to this point, it is my own private space". That is what VPC allows you to do.

Setting up a VPC involves some more detailed information about your own private space, such as subnet, internet gateway etc. Subnet forms different groups depending on what kind of data communication you intend to make with your VPC. Some may only be used for internal communications, some may be allowed to interact with some networks outside of your VPC etc. An Internet gateway is something equivalent to the door of your house. Just like you always enter your house via the same door, your VPC also needs a gate to connect the world outside of your VPC and the world inside of it. Figure 33 below is a brief illustration of this description.

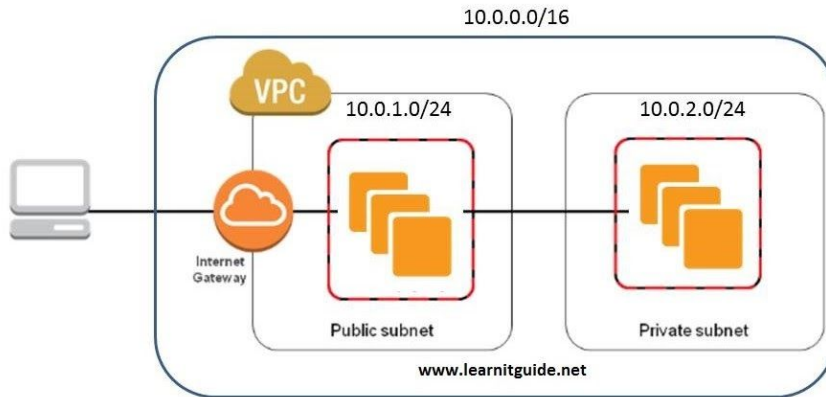


Figure 33. VPC, subnet, Internet Gateway (LEARNITGUIDE.NET, 2019 January)

When using VPC, it is also vital to make Route Tables. They are responsible for defining the rules for which EC2 instance in which subnet communicates with which place. In other words, EC2 instances cannot communicate with anything if it is not defined in the Route Tables.

4.5.1.2 EC2

Elastic Compute Cloud, or EC2 is a service that constructs a virtual server that Amazon provides. EC2 enables you to make a server environment with what is called an instance, which is a virtual server equipped with an Operating System. With EC2, it is also possible to create multiple instances.

Since an EC2 instance is a virtual server, not a physical server, it is quite flexible in scaling. At first you may not need a large capacity because your project is originally small, but when the project grows you can easily adapt to the change.

The main concept for the pricing of EC2 instances is basically “You only pay for what you use”, but there are several different pricing models.

One of them is On-Demand Instance, which charges you based on the hours you are using the server. This is most likely beneficial when you intend to use the server only for a short period of time.

Another model is Reserved Instance. Contrary to On-Demand Instance, you decide how long you plan to run the server and pay for it in advance. This is more reasonable when you want to keep the server active longer than a certain period of time.

There is also something called Spot Instance. That is an instance that someone else has already created but not currently in use. The advantage is that you can use those instances for considerably lower price, but the downside is that the instances can be taken away from you any moment, as they technically don't belong to you and you don't have the right to claim the ownership of those instances, because you are just temporarily borrowing them.

After you have chosen which instance type to go with, you will need to set a role and security group. This is to keep an eye on who has what level of accessibility to the service. For example, when there is a system that manages students' exams' data, you need to make sure that

“Only the teacher who is in charge of the class should be allowed to read and write (update) the students’ exam score data”

“Students can be given an authority to access the online classroom and read the exam result, but should not be given the ‘write’ authority”

“People outside of this educational institution should not be given read right either”

etc.

In most cases, it is the best practice to keep the level of authority to give the users minimum. This is to prevent malicious or unintentional modification of the data, which could possibly lead to a disaster.

Then you obtain a keypair with which you can log in to the EC2 instance via SSH. SSH is a secure way of connecting to the instance, in which the content of the communication is encrypted. Once the login is successful, you are set to do what you want to do with the EC2 instance.

4.5.2 Docker

When you want to deploy your application on AWS, it means that you are running the project on your EC2 instance, not on your local machine. Here comes a question. Will the project work the same way as it does in your local environment?

There are different types of Operating Systems and of course they are not exactly the same. Because of this, what happens sometimes is that what has been working without a problem at all on your computer might not function properly on someone else’s Mac, since the environments are different. This issue is certainly applicable to the application deployment on the EC2 instance, as I am using Amazon Linux as its OS and it differs from Windows OS.

This is where Docker comes in handy. It containerizes all the necessary configurations and dependencies within a single container, and as long as the machine has an environment where Docker can be used, the project can be run in the same way. Before Docker, developers used to have to manually make sure that all the configurations and dependencies are set the right way, which required elaborate work, but it is no longer necessary thanks to Docker.

Fairly often multiple Docker containers are needed to function as a full single project. It is technically possible to run each one of the Docker commands on the terminal, but it is a tedious task. In order to mitigate this problem, there is a tool that makes running multiple containers with all the configurations much easier, and that is Docker compose.

In using Docker, you need to specify which port number to use, what image you are using etc, and with Docker compose you can define them in a very structured way, and once you defined all of them, all it takes to start the container(s) is a single command `docker-compose up`.

4.5.3 CircleCI

If you have finished writing the code and deployed the project, in most cases that is not the end of the application development. Sooner or later, you will need to make some changes in the code, because

later on you might want to add some functions, modify the design of the web page, or refactor the code etc.

Once you have made those changes, you need to build, test and deploy the project again, otherwise the changes will not be reflected. Technically, you could just manually take care of the build-test-deploy process. However, as it usually involves a number of lines of commands, it is clearly a pain in the neck to go through a series of those steps each time you make a change in the code.

This is where CircleCI comes into play. By defining the entire process of the build-test-deploy steps, the integration and deployment can be automated. You need to have a CircleCI account, but if you already have a GitHub account, you don't have to manually create a new account, because if you log in to CircleCI using your existing GitHub account, it is equivalent to obtaining your own CircleCI account.

As for how to use CircleCI, ordinarily you add a `.circleci` directory on your working repository, and make a `config.yml` file there. That is where you write a series of chunks of commands to define how the build-test-deploy process goes.

The structure of the `config.yml` file is not too unfamiliar for those who have written code in almost any programming language. In the field of programming, a program consists of functions, and each of the functions consists of statements. Just like that, CircleCI's `config.yml` file is also written in a way that is relatively straightforward to break the entire flow into smaller chunks. What is called "jobs" are defined, and the jobs consist of "steps", and those steps consist of even smaller elements etc.

Figure 34 below shows how my "create_images" is defined.

```
9 jobs:
10   create_images:
11     machine: true
12     steps:
13       - checkout
14       - run:
15           name: Install docker compose
16           command: |
17             sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)"
18             sudo chmod +x /usr/local/bin/docker-compose
19       - run:
20           name: Build images
21           command: |
22             git clone https://github.com/Shinichi1125/pictionary_v2.1.git
23             cd pictionary_v2.1
24             set -x
25             docker-compose build
26       - run:
27           name: Start containers
28           command: |
29             set -x
30             cd pictionary_v2.1
31             docker-compose up -d
32       - run:
```

Figure 34. `config` file

The virtue of this technology that I would especially like to mention here is the "command: |" parts. In each of them a series of commands is put together, and all of them will be run just by running the "create_images". In other words, you don't have to manually type

```
git clone https://github.com/Shinichi... (and Enter)
```

```
cd pictionary_v2.1 (and Enter)
set -x (and Enter)
.....
```

and so on in the terminal.

And just like you can build a program by writing functions and combining them, you can define the “workflow” by combining the jobs like this (Figure 35).

```
58 workflows:
59   version: 2.1
60   pictionary-pipeline:
61     jobs:
62       - create_images
63       - deploy:
64         requires:
65           - create_images
66
```

Figure 35. CircleCI workflows

It starts with “create_images” and next comes “deploy”. As you can see in the workflow, “deploy” requires “create_images”. This means that the “deploy” will not start until the “create_images” is complete.

5. RESULT

After getting the infrastructure ready, the application is available to anyone with an Internet connection. You can visit the website by clicking the link below.

<http://pictionarizer.work>

However, the link may expire in the future, as it requires a monthly cost just to maintain the EC2 instance and I don't intend to keep it public longer than it is necessary. Still, even when the website is no longer publicly available, it is possible to replicate the same project if you clone it via my GitHub link.

<https://github.com/Shinichi1125/Pictionarizer>

6. CONCLUSION

Working on a web application development truly requires a lot of work. It is highly comprehensive and time-consuming. It took me several months. My project consists of the technologies that have been mentioned so far, but it is obviously much smaller and simpler than the real web services that are actually in use today. So there is no wonder why a number of developers and product managers are needed to make a single project complete on an enterprise level, since Social Network Services like Facebook, Twitter, Instagram etc are considerably bigger and more complex.

This means that the field of web development has so much further to explore. I consider this project to be just a beginning of my journey, and would like to keep improving my skills in this field for my future career as a developer.

References

- Oracle. (2020). *What Is a Relational Database?*. <https://www.oracle.com/database/what-is-a-relational-database/>
- DB-ENGINES. (2020, Nov). *DB-Engines Ranking*. <https://db-engines.com/en/ranking>
- MySQL. (2020). *Why MySQL?*. <https://www.mysql.com/why-mysql/>
- Shinichi Takagi. (30 October 2020). *Pictionarizer*. <https://github.com/Shinichi1125/Pictionarizer/tree/master/pictionarizerapi/src/main/java/com/pictionarizer>
- Deitel, H., & Deitel, P. (2017). *Java 9 for Programmers* (4. ed.). Prentice Hall.
- Tutorialspoint. (2020). *Spring Boot - Introduction*. https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm
- DigitalCrafts. (2020). *What is Postman, and Why Should I Use It?*. <https://www.digitalcrafts.com/blog/student-blog-what-postman-and-why-use-it>
- Skillcrush. (2020). *TECH 101: WHAT IS REACT JS?*. <https://skillcrush.com/blog/what-is-react-js/>
- Stack Overflow. (2020, Nov). *Stack Overflow Trends*. <https://insights.stackoverflow.com/trends?tags=reactjs%2Cangular%2Cvue.js%2Cjquery>
- Bolgurtseva, O., & Dreimanis, G. (18 juni 2020). *Why You Should Choose TypeScript Over JavaScript*. <https://serokell.io/blog/why-typescript>
- Tutorialspoint. (2020). *JUnit - Overview*. https://www.tutorialspoint.com/junit/junit_overview.htm
- Ascendro. (2 August 2013). *Software testing approach by project size*. <http://ascendro.de/blog/article/107/Software+testing+approach+by+project+size?language=en>
- JEST. (2020). *JEST*. <https://jestjs.io/>
- Airbnb. (2020). *Enzyme*. <https://airbnb.io/projects/enzyme/>
- AWS. (2020). *What is AWS*. <https://aws.amazon.com/what-is-aws/>
- opensource.com. (2020). *What is Docker?*. <https://opensource.com/resources/what-docker>
- Wright, J. (6 september 2016). *Learn Docker in 12 Minutes* [Video]. <https://www.youtube.com/watch?v=YF12mCHdv24>
- circleci.com. (2020). *Overview*. <https://circleci.com/docs/enterprise/overview/>

LEARNITGUIDE.NET. (2019). *AWS VPC - Create New VPC, Subnets, Internet Gateway*.
<https://www.learnitguide.net/2019/01/aws-vpc-create-new-vpc-subnets-internet.html>