ReadySet Go Developer Manual

For developer operation of the site

Version History

Version	Date	Author	Description
1.0	4/23/2021	Kien Nguyen	Document created
1.0	4/28/2021	Kien Nguyen	Revision

Texas Christian University

Contents

CHAPTER 1

Introduction

This is a developer manual provided along the Ready-Set-Go project for TCU Senior Design 2020-2021. The manual will briefly talks about the servers and the re-deployment process if there are new updates to the project in the future. Further, it describes the system architecture, which is rather complicated for a maintaining developer. It will also includes the API documentation hosted on Swagger Hub.

CHAPTER 2

Servers

2.1 Description

The environment consists of 3 separate servers:

- Ubuntu 20.04 as the main server that hosts the Spring Boot application
- Ubuntu 18.04 as the play server that hosts the Node application
- Ubuntu 18.04 as the database server that hosts MongoDB.

2.2 Main (Spring Boot) Server

The main repository is `~/Ready-Set-GO` on `go.cs.tcu.edu`. There is a `deploy.sh` file inside the repository that takes care of the deployment from compiling `webpack` for front-end, building a `jar` file for the Spring Boot application to starting the server. Therefore, the developer needs to do 2 steps to re-deploy the main server while inside the main repo.

```
1    $ git pull
2    $ ./deploy.sh
```

.

2.3 Play (Node) Server

The main repository is `~/Desktop/RSGPlayServer` on `rl.cs.tcu.edu`. We use `pm2` to manage Node processes on this server. Refer to this tutorial. To re-deploy,

```
1    $ systemctl restart pm2-ai-lab nginx
```

.

2.4 Database (MongoDB) Server

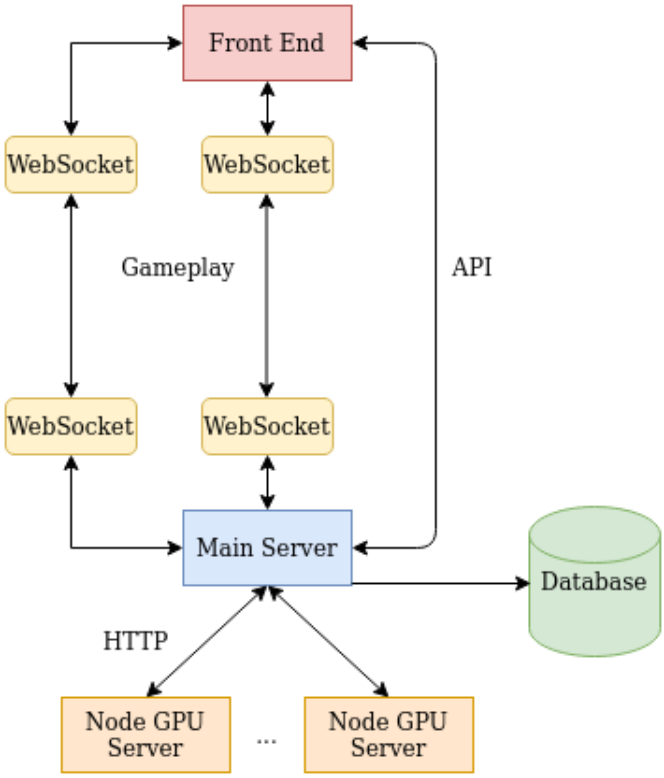
It's best not to touch the database server often. However, when needed, we can access the server through a `mongo` shell command while inside the main server shell.

```
1    $ mongo "mongodb://<user>:<password>@eurekad.cs.tcu.edu/readysetgo"
```

.

CHAPTER 3

System Architecture



3.1 User Management

For this feature, the client only talks to the main server, which then talks to the database server for CRUD operations.

3.2 Research Feature

The research feature requires the play server to come into play. The client talks to the main server, which then talks to the play server through APIs.

3.3 Gameplay Feature

This is the most complicated feature yet. The client creates a web socket instance with the main server, which then creates another parallel web socket instance with the play server. The parallel sockets help gameplay to be feasible in real time.

CHAPTER 4

Technology Stack

There are quite a lot of technologies involved in this project. We will list them below and explain the responsibility of each one.

- **Spring Boot** is a Java web application framework that works as the main server in the project. It takes requests from the clients and processes them by applying CRUD operations on the database, retrieving data from the play server via RESTful APIs, and instantiating web sockets between itself and the clients and between itself and the play server.
- **NodeJS** is a JavaScript web application framework that works as the play server in the project. Its primary roles are to start training sessions for the AIs, to start gameplay sessions via a Go engine called `leelazero`, and to give out data to the main server when requested. An important package in the play server is `SabakiGTP` that helps the Node application talk to the Go engine in real time.
- **MongoDB** is a NoSQL database in the project. We use a NoSQL database because it is easy to modify on the fly, especially during testing.
- **React** is a front-end JavaScript framework in the project. The framework makes it easier for web pages to communicate with one another during transitions and is modular.

CHAPTER 5

API Documentation

The API documentation can be found here.