{TypeMania}

CSCI 441 VA – Technical Documentation

Group 2:

Shaka Elmore

Nargis Sultani

Heather Vroman

Alexandra Stevens

Nov 5, 2022

Table of Contents

1. Run the code

1.1 Prerequisites

Before running the code, please ensure the following are installed on your device:
Git - https://git-scm.com/downloads
NodeJS - https://nodejs.org/en/download/
Any modern web browser that can load javascript scripts for web pages.

1.2 First time setup

Once installed, follow these steps to run the code:

1. Run git bash & use its terminal to...
2. Type “cd file/path/to/install/project/to”, navigating to an empty folder of your choosing.
3. Type “git init”, turning the folder into a local repository.
4. Type “git clone -b master https://github.com/TypeMania/TypeMania.github.io.git“, downloading the project to our folder.
5. Type "cd Typemania.github.io/backend", navigating into our backend codebase.
6. Type “npm config set legacy-peer-deps true”, ensuring smooth installation of dependencies.
7. Type "npm i", installing the required dependencies at this location.
8. Add this file as “.env” to the backend folder.
9. Type "npm run dev", starting a developer server from our backend.
10. Type "cd ../frontend", navigating to our frontend codebase.
11. Type "npm i", installing other required dependencies at this new location.
12. Type "npm start", connecting you to our web application via your default browser.

The code should now be running and viewable within the web browser, address "localhost:3000"

1.3 Subsequent runs

After the initial setup the steps required to run the code reduce to:

1. Run git bash & use its terminal to...
2. Type “cd file/path/to/typemania.github.io/backend”, navigating to the backend folder
3. Type “npm run dev”, starting the developer server.
4. Type “cd ../frontend”, navigating to the frontend folder.
5. Type “npm start”, connecting you to our web application via your default browser.

The code should now be running and viewable within the web browser, address "localhost:3000"

2. File Structure & Frameworks

2.1 Frameworks

There are two major folders within the project, backend and frontend. The backend deals in all things related to our database. The frontend is concerned with the web application itself.

Our backend database server is outsourced to the cloud-based MongoDB service. We do not own or maintain any physical server computer, but are able to access & control the database via Mongoose, an object data modeling library for javascript. This documentation will not cover how to use the library, but you can retrieve such information from https://mongoosejs.com/docs/index.html.

Our frontend web application exists on a single html page. Utilizing React, we are able to convert our web page into blocks or components that can each be freely added and removed from view at a click. This documentation will also not be covering the functions and classes of this framework, but you can retrieve such information from https://reactjs.org/docs/getting-started.html.

The frontend also makes use of a framework for game creation known as Phaser. The framework deals heavily in graphics and animations and is yet another library that our documentation will not be covering. You can access its documentation at https://photonstorm.github.io/phaser3-docs/.

There are more frameworks and dependencies in use that affect the project to a lesser degree than the above stated. You can get more information on this from the package.json files found in the frontend and backend folders.

2.2 backend

Inside of the backend folder lies these subfolders:

Subfolder	Description - What types of files to expect inside
config	Database connection, configuration settings for the database and what can access its resources.
controllers	API calls: user creation, deletion, and authentication in the database.
middleware	Error handling & logging, json web token verification.
models	Data schematics for users, statistics, & songs.
public	N/A
routes	Differentiates registered and guest user navigation paths, authentication routes.
views	Error pages.

Chart 2.2: backend subfolders

2.3 frontend

Inside of the frontend folder lies these subfolders:

Subfolder	Description - What types of files to expect inside
public	Default items when the user's browser has javascript disabled.
src	The meat of the web application. This folder makes up the bulk of our visible features.

Chart 2.3: frontend subfolders

2.4 frontend/src

Inside of the frontend/src folder lies these subfolders:

Subfolder	Description - What types of files to expect inside
app	Connections to backend/api.
assets	Frontend images & song files until our database is functional.
components	The building blocks of our webpage which can be switched in and out at will.
data	Dummy data for leaderboard until our database is functional.
features	API slices concerning user login authentication and unused user page and functions.
hooks	Function components and their states

Chart 2.4: frontend/src subfolders

3. Essential Files, Functions, & Classes

3.1 Backend

Essential Files	Overview of purpose
/controllers/usersController.js	Handles user creation, deletion, and modification, api calls.
/config/dbConn.js	Hooks up our system to the database.
Essential Classes	**Properties**
/models/User.js	● username - String ● password - String ● email - String ● Functions of User found in /controllers/usersController.js

/models/Song.js	• user - /models/User.js • title - String • artist - String
/models/Stat.js	• user - /models/User.js • song - /models/Song.js • score - Integer • errorPerc - Integer
Essential Functions	**Precondition & Postcondition**
connectDB() found in /config/dbConn.js	Precondition: Receives nothing. Postcondition: Awaits a connection to the MongoDB database.
createNewUser() found in /controllers/usersController.js	Precondition: Requests a response from the database and receives inputs from user registration fields. PostCondition: A new user is added to the database if it responds and all registration fields are filled properly.

Chart 3.1: Essentials to understanding backend.

3.2 Frontend

Essential Files	Overview of purpose
/src/components/public/PubHome.js /src/components/Home.js	The building block of the web page that houses all of the other building blocks. The PubHome.js variant is our guest user page layout while the Home.js displays the registered user page layout.
/src/components/Game.js	The heart of our web application. The game canvas itself where graphics and animations take root and give way to the interactivity a game ought to have.
/src/components/SongSelect.js	Provides the user with an interface to control the music.
Essential Classes	**Properties**
/src/components/Game.js	• componentDidMount() - Method • shouldComponentUpdate() - Method • render() - Method
Essential Functions	**Precondition & Postcondition**
componentDidMount() found in /src/components/Game.js	Precondition: The web application begins running. Postcondition: The gameplay scene renders to the web application.

create() found in componentDidMount() method of /src/components/Game.js	Precondition: componentDidMount() triggers. Postcondition: The initial graphical elements are rendered to the gameplay scene.
update() found in componentDidMount() method of /src/components/Game.js	Precondition: create() function finishes. Postcondition: Calls itself continuously in a loop, updating the position of graphical elements according to their animation details.

Chart 3.2: Essentials to understanding frontend.