IT5007 Team Project: WeType

Group 1

Liu Zhixing A0268509R

Zhang Gaoli A0268264W

Zheng Yanan A0268565M

National University of Singapore

3 December 2023

Index

1 Introduction

In today's fast-paced digital environment, effective and efficient typing skills are essential for various tasks, including professional communication, coding, and content creation. WeType, a typing speed improvement web application aims to create a user-friendly and engaging environment for users to hone their typing abilities.

Here are some goals of the WeType project:

- Provide an intuitive and user-friendly interface for typing practice.
- Offer a range of typing exercises to cater to users of different skill levels.
- Create personalized user profiles to track individual progress.
- Deliver real-time feedback to users for continuous improvement.

In order to achieve the above objectives, the following design was carried out:

- There is a landing page. On entering WeType, a user needs to signup/login to his or her own account.
- After signing in, user will enter his or her account home page where some basic account information and this user's history of typing in this website could be accessed.
- Users can practice by clicking "Practice" button. User should type as much as possible within a fixed time (2 min). The more he or she types, the better grade he or she will get.
- When the time is up, the user can see his speed for this exercise.

In the subsequent sections of this documentation, we offer a comprehensive view of our work, examining various facets of the project. The second section delves into the intricacies of front-end design, while the third section provides insights into the back-end design. Moving forward, section 4 introduces the third-party libraries seamlessly integrated into this project. Section 5 gives an overview of our teamwork. Finally, section 6 provides an overview of potential improvements and enhancements that can be implemented in future releases of the project.

1

2 Front-end Development

2.1 Overview

The front end of WeType is responsible for creating user interface, handling client-side interactions, and exchanging data with back end. This section provides an overview of UI feature details, externally invoked resources, the technologies and architecture employed in the development of the front end.

2.2 UI Feature Details

2.2.1 Log in/Sign up Page

The Log in/Sign up page serves as the gateway for users to access the WeType platform.

- Authentication Forms: Intuitive and secure forms for user login and registration.
- User Feedback: Informative messages and feedback to guide users through the authentication process.
- Web Application Showcase: A visually appealing representation of WeType's purpose, conveying the application's focus on enhancing typing skills.

2.2.2 Home Page

The Home Page is the central hub where users can navigate to all other pages of the application.

- Navigation: Intuitive menus and navigation elements for easy access to the user setting page and different sections of the application.
- Practice Options: Give a way for users to start typing practice.
- Personalization: Customization options to tailor the user experience based on individual preferences.
- Dynamic Content: Dynamic content presentation based on user practicing time.
- Logout Option: Provide users with the flexibility to log out easily, enhancing overall user control.

2.2.3 Typing Page

The Typing Page is the core functionality of WeType, providing users with a platform to enhance their typing skills.

- Interactive Typing Interface: A responsive and interactive platform for users to practice and improve their typing speed and accuracy.
- Real-time Display of User Input: A dynamic interface that instantaneously reflects users' keypresses, showing the characters they have entered.
- Typing Result Display: At the conclusion of the typing session, display comprehensive results, showcasing metrics such as time, accuracy, and number of correctly typed characters.
- Progress Bar and Countdown Timer: Dynamically reflects time passing during practicing.
- Restart Button: Allows users to reset the typing exercise at any point, providing the flexibility to begin anew.

2.2.4 User Setting Page

The User Settings page empowers users to personalize their WeType experience by configuring preferences that extend across the platform. The customized settings will be stored securely in the user database, ensuring that their preferences are retained for future sessions.

- Avatar Selection: Users can select their preferred avatar from a range of options provided. The chosen avatar will be showcased on the Home Page, adding a personalized touch to their WeType profile.
- Theme Selection: Users can select their preferred website theme from the two provided. One is pink, the other is blue. This preferred information will also be stored in the database with the user's account.

2.2.5 Typing History Page

The Typing History page serves as a valuable tool for users to track and visualize their typing progress over time.

3

- Stage Summary: Displays the user's total exercise count for the past three months, providing an overall snapshot of their typing activity.
- Frequency Table: A frequency table akin to GitHub, with each day represented as a small grid cell. It can show an overall practicing frequency of the last three months. If a user practices on a specific day, the corresponding grid cell is colored. The color intensity deepens based on the frequency of exercises on that day. When hovering over a grid cell, information is displayed, including the date and the number of exercises performed on that day (if any).

2.2.7 Ranking Page

The Ranking page provides users with a competitive edge by displaying the top-performing users on WeType.

- Top Positions: Displays the usernames and the total practice of the top three users with their usernames and their practicing volume.

2.3 Technologies Used

2.3.1 React

React is a JavaScript library used for building user interfaces. In our project, we leverage React to organize and manage frontend development through component-based architecture, enhancing code maintainability and reusability.

2.3.2 Webpack

Webpack is a module bundler tool that packages multiple frontend resources into one or more files. We use Webpack to build and optimize our frontend application, as well as to handle modular development. *To support webpack, a version of npm at 11.0.0 or above is necessary.*

2.3.3 jQuery

jQuery is a fast, lightweight, and feature-rich JavaScript library designed to simplify DOM manipulation and event handling. In our project, jQuery is employed to facilitate DOM element manipulation, implement dynamic effects, and enhance interactivity.

2.3.4 Popper.js

Popper.js is a JavaScript library used for handling popovers and other positioning tasks. We utilize Popper.js to provide robust positioning and boundary detection, particularly in scenarios involving relative positioning of interface elements.

2.3.5 Bootstrap

Bootstrap is a popular open-source frontend framework that offers a set of tools for building modern, responsive, and mobile-friendly websites and web applications. In our project, we integrate the Bootstrap framework to streamline the creation of consistent and visually appealing user interfaces.

3 Back-end Development

3.1 Overview

The back end of WeType is responsible for handling server-side logic, managing data storage, and facilitating communication between the front-end and the database. This section provides an overview of the technologies and architecture employed in the development of the back end.

3.2 Technologies Used

MongoDB serves as the primary database system, providing a scalable and flexible NoSQL solution. MongoDB's document-oriented model allows for the efficient storage and retrieval of data, making it well-suited for our dynamic and evolving application needs.

GraphQL is employed as the query language and runtime for our API. GraphQL offers a more efficient and flexible alternative to traditional REST APIs by enabling clients to request only the data they need.

Apollo is utilized as both the server and client framework for GraphQL. The Apollo Server facilitates the creation of a robust GraphQL server with features such as data caching, real-time updates, and seamless integration with various data sources.

3.3 Components

3.3.1 API Endpoints

5

We defined and implemented GraphQL API endpoints to handle communication with the front-end.

Description	Type	Variables	Return
User login	Mutation	email, password	int: 1 if success, others if fail
User signup	Mutation	email, password, name	int: 1 if success, others if fail
Fetch user profile	Query	email	JavaScript object, empty if fail
Edit user profile	Mutation	email, name, preferred_style, avatar	int: 1 if success, others if fail
Add practice record	Mutation	email, time, words	int: 1 if success, others if fail
Fetch world ranking	Query		sorted array of JavaScript objects

Query/Mutation Format

```
// Query
  fetchUserInfo(email: String!, password: String!): UserProfile
  worldRanking: [User]
// Mutation
  logIn(email: String!, password: String!): Int
  signUp(email: String!, password: String!, name: String): Int
  editProfile(email: String!, name: String, preferred_style: Int,
avatar: Int): Int
  addRecord(email: String!, password: String!, time: Int!, words:
Int!): Int
```

3.3.2. Database Models

Define data models to represent users' account.

6

```
History = {
   key: [String]
   value: [Int]
   // user’s practice frequency
   // key: date, value: practice time on that date
}
Profile = {
   name: String,                 // user’s preferred name
   preferred_style: Int,         // enum
   history: History,             // user’s typing history
   avatar: Int,                  // enum
   total_practice_words: Int,    // number of user’s practiced words
   total_practice_time: Int      // number of user’s practiced time
}
User = {
   email: String,
   password: String,             // hashed, for authentication
   profile: Profile              // user’s profile data
}
```

3.4 Deployment

7

The back end of the WeType is designed to be deployed on the Docker container created by the image provided by IT5007 course. Before starting the server, developers need to ensure that the MongoDB, GraphQL, and Appolo dependencies are pre-configured in their environment.

4 Third-Party Libraries

OpenAI API to generate text - Davinci Model

In our typing website project, we have incorporated the "davinci" model from OpenAI API for generating text for typing practice. The "davinci" model is OpenAI's most advanced, universal language model, representing the pinnacle of the GPT-3 series. It is developed using deep learning algorithms and trained on a massive scale of data, enabling it to understand and generate highly natural and coherent text. This makes the "davinci" model exceptionally suitable for a wide range of text generation tasks including news articles, stories, poems, travel guides, lifestyle blog posts, and movie scripts.

Why We Chose the Davinci Model:

- Advanced Language Comprehension and Generation Capabilities: Thanks to its sophisticated algorithms and extensive training data, the "davinci" model can generate high-quality, engaging text, making it ideal for typing practice content.
- Versatility in Applications: It can handle various types of text requests, offering a diverse typing practice experience for users.
- Customizability and Flexibility: We can tailor the generated text to fit specific themes or difficulty levels through precise prompts and parameter settings, catering to the needs of different users.
- Continuous Optimization and Updates: As a model under OpenAI, "davinci" receives ongoing enhancements and updates, ensuring that the quality and relevance of the generated text remain at the forefront of the industry.

By utilizing the "davinci" model, our website is able to provide rich, high-quality materials for typing practice, helping users to improve their typing skills in an enjoyable and efficient way.

8

5 Teamwork

In the development of WeType, each team member played a crucial role, contributing our unique skills and expertise to ensure the project's success. Below is an overview of the key responsibilities shouldered by each team member:

	Zhixing	Gaoli	Yanan
Project scope design	✓	✓	✓
UI Design		✓	✓
Front-end development		✓	✓
Back-end development	✓		
Front-end and back-end interaction	✓		✓
Introduce third-party libraries			✓
Video, documentation, and report	✓	✓	

6 Future Enhancement

As our project evolves, we envision several opportunities for enhancement and expansion. While the current version delivers valuable features, our commitment to continuous improvement means that future releases may include the following enhancements:

Real-time Character Comparison

Instantaneous evaluation of the accuracy of users' input compared to the original text, resulting in the color of each character in the text dynamically changing. This feature ensures users are aware of any deviations and helps them correct errors on the fly. We had planned to implement this feature but postponed the plan due to lack of energy from team members.

Web Security

As part of our commitment to bolstering the security measures of the typing application, a future enhancement will focus on implementing robust password hashing techniques for user authentication. The introduction of password hashing aims to fortify the protection of user credentials stored in the system. By employing industry-standard hashing algorithms, such as

9

bcrypt, we will ensure that user passwords are securely transformed into irreversible, unique hashes. This enhancement is pivotal in safeguarding user accounts against potential security threats, including unauthorized access and data breaches.

Multi-thread server

To enhance the scalability and responsiveness of the typing application, a future improvement could involve the implementation of a multi-thread server. This addition would enable the application to efficiently handle concurrent user requests by dedicating separate threads to different tasks. By introducing multi-threading, the server could achieve improved performance, allowing for a smoother user experience even during periods of high traffic.

Friend system and race

Add a race with friend user function so that users can not only practice alone, but also compete with friends. The implementation of this feature involves real-time information interaction between the front and back ends due to the need to synchronise each other's typing progress on both front ends. Also, we can design a friend system to support the race function. Users could add or delete another account as his or her friend.

Achievement system

Add an achievement system to the web application to motivate out users and increase their engagement. This system aims to recognize and reward users for their accomplishments and milestones in typing proficiency. Users will have the opportunity to unlock various achievements based on factors such as typing speed milestones, consecutive days of practice, and mastery of specific typing challenges.

10