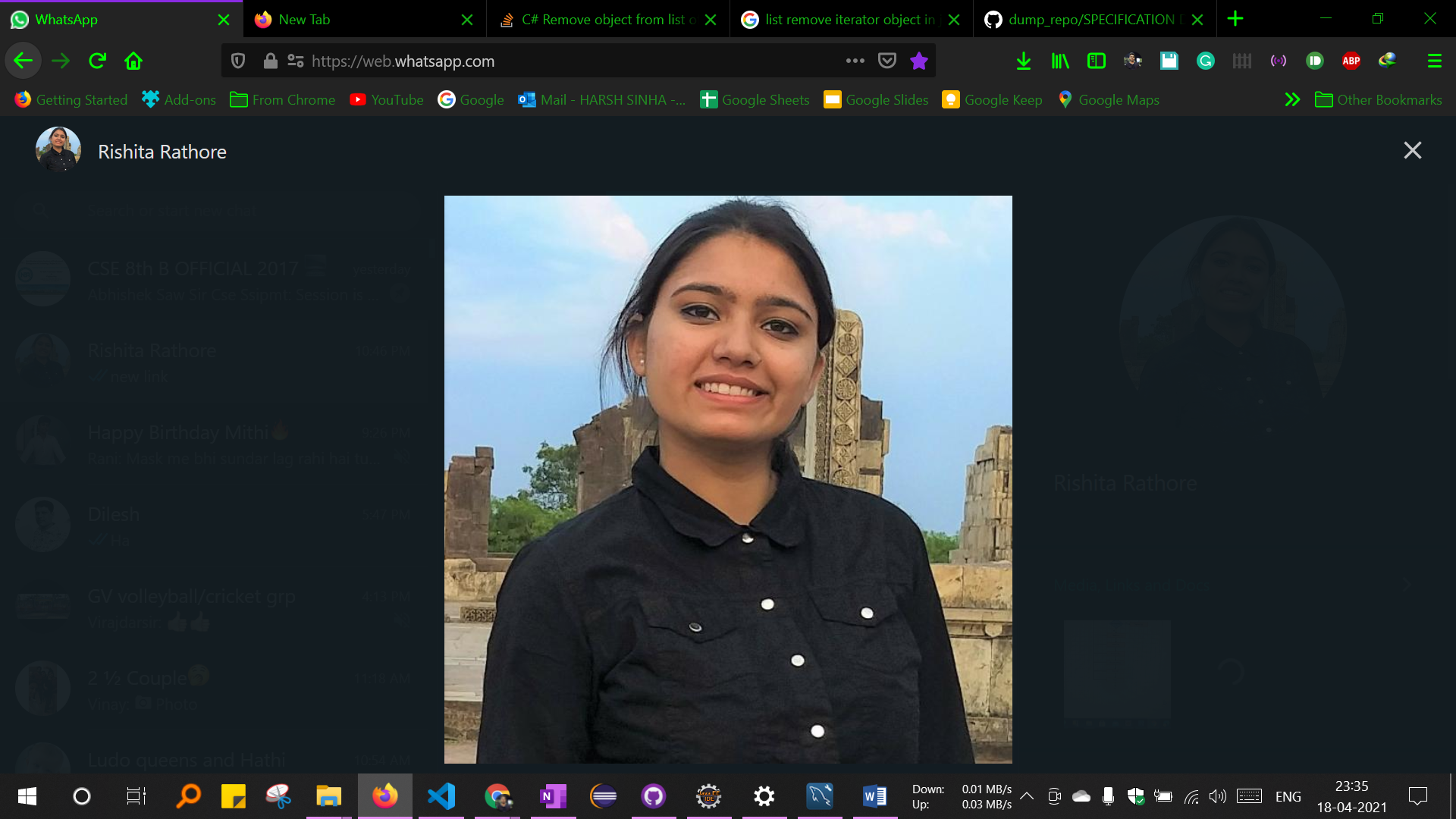
**ONLINE TEST APPLICATION**

**Submitted By: Rishita Rathore**



**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page No.** |
| I | Project and Developer Details  1.1 Project Objective | 3  3 |
| II | Project Requirement Analysis  2.1 Project Requirements  2.2 Software Requirements  2.3 Hardware Requirements | 3  3  4  4 |
| III | Quiz Features | 4 |
| IV | Core Concepts | 5 |
| V | Flow Chart | 7 |
| VI | GitHub Repository | 10 |
| VII | Conclusion | 10 |

**Chapter I: Project and Developer Details**

* 1. **Project Objective:**

The Online Test Application system creates an application that enables users to provide online tests, review them, and display the results.

This system contains three main modules: Quiz, Review, and Result. The quiz section of the online test application accepts the questions in JSON format. The JSON file can be easily shared from the server in the pre-defined format. The application renders the test at the client-side.

The “Review and display result” section allows users to declare the results immediately. You can simply call another JSON with the answers in it and evaluate and display the results immediately.

**Chapter II: Project Requirement Analysis**

**2.1 Project Requirements:**

**Features:**

* Multiple Quiz Options
* Quiz Page
* Review Page
* Result Page

**2.2 Software Requirements**

Operating System - Windows 7 or Later, macOS

IDE used - Visual Studio Code

Web Framework - Angular

Package Manager - npm (Node Package Manager)

**2.3 Hardware Requirements**

Processor - Intel core I3 or above

RAM - 4 GB minimum

Hard disk space - 1 GB minimum

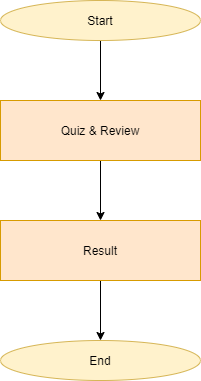
**Chapter III: Quiz Features**

* **Shuffle Questions:** shuffleQuestions property in config section is used to mention whether the questions should be randomized before showing or not. If true, the questions will be randomized. The default value is false.
* **Shuffle Options:** shuffleOptions property in config section is used to mention whether the options can be randomized before showing or not. If true, the questions will be randomized. The default value is false. This feature is configurable from the quiz json file.
* **Show Pager:** showPager property in config section indicates whether to show the pager element in quiz or not. If false, the numeric pager will not be shown. In this case, the user can still navigate via. first, prev, next and last button. The default value is true.
* **Auto Move:** autoMove property in config section indicates whether to move to next question automatically when the question is answered. If true, you don't need to press Next button to move to next question. The default value is false.
* **Load Quiz Dynamically:** In mock test/practice exams, user might want to change the quiz dynamically. To do so, you just need to call loadQuiz method and pass the URL of the new quiz. The attached sample or the demo link illustrates this. (In the sample app, you can load the quiz dynamically via changing the drop-down at top right corner.)
* **HTML Questions:** In many cases, you wish to have special characters, images, subscript, superscript, and other HTML elements. Adding HTML markup in question has been provided via angular sanitize. To add image or any HTML formatting in questions, just add the corresponding HTML tags in question.

**Chapter IV: Core Concepts**

* **HTML & CSS:** The front pages are made using HTML and CSS. The Quiz View contains the UI for quiz questions, previous-next button, paging and related stuffs. I have chosen the structure of keeping one question per page but if you would like to show more than one question, you can simply change the pager.size value accordingly. For buttons and pagination UI, bootstrap has been used. The UI for questions and options has been set in quiz.component.css.
* **JavaScript:** JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.
* **Angular:** Angular is a development platform, built on TypeScript. As a platform, Angular includes:
* A component-based framework for building scalable web applications
* A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
* A suite of developer tools to help you develop, build, test, and update your code.
* **NodeJs:** **NodeJS** gives you the tool npm that allows you to download libraries and packages you would **use in Angular** 11. From the shell you can go to your folder and type npm install to install dependencies you need to have installed to get your **angular** project going.

**Chapter V: Flow chart**

****

**Chapter VI: GitHub Repository**

[**https://github.com/rishitarathore/Learners-Academy.git**](https://github.com/rishitarathore/Learners-Academy.git)

**Chapter VII: Conclusion**

This project has three modules: Quiz, Review and Result. The user first opens the quiz page where he/she can select the subject of quiz. The questions appear sequentially where one question is displayed at a time. The quiz page has previous and next button as well which is used to navigate between questions. It also contains a review where a user can see how many questions they have attempted and can also jump to question directly from there. It also contains a timer. When the user clicks the submit button he is redirected to result page. The result page displays score and answers to all questions. Overall the project implements the requirements mentioned.