**Assessment Application**

**Requirements**

Stack Used **–** MEAN stack (MongoDB, Express, Angular, NodeJS)

MongoDB - Version 4.2.2 – Link : <https://www.mongodb.com/download-center/community>

Visual Studio Code - Text Editor – Link : <https://code.visualstudio.com/>

Express-“4.17.1”- $ npm install express

Mongoose-“5.8.0”- $ npm install mongoose

Body-Parser-“1.19.0” - $ npm install body-parser

Cors-“2.8.5”- $ npm install cors

NodeJS- “12.13. 1”

1. **Roles and responsibilities**.

The team members includes K.R. Arun Kumar and Anish Raghavendra.

K.R. Arun Kumar - in charge of creating the admin module i.e.. to create a MongoDB database to store and retrieve questions, options(answers) and correct answer.

Anish Raghavendra -in charge of creating the user module i.e. to create a module to allow users to login and to display stored questions and options in the form of a multiple choice quiz.

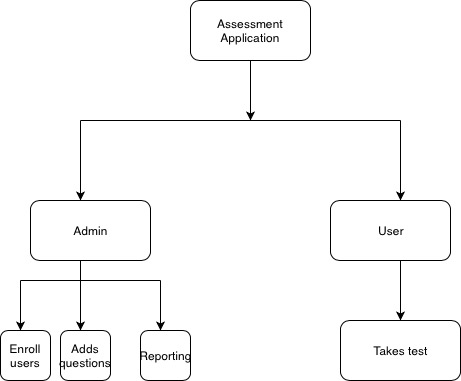
1. **Team goals and business objective**.

The main goal was to create a multiple choice quiz application.

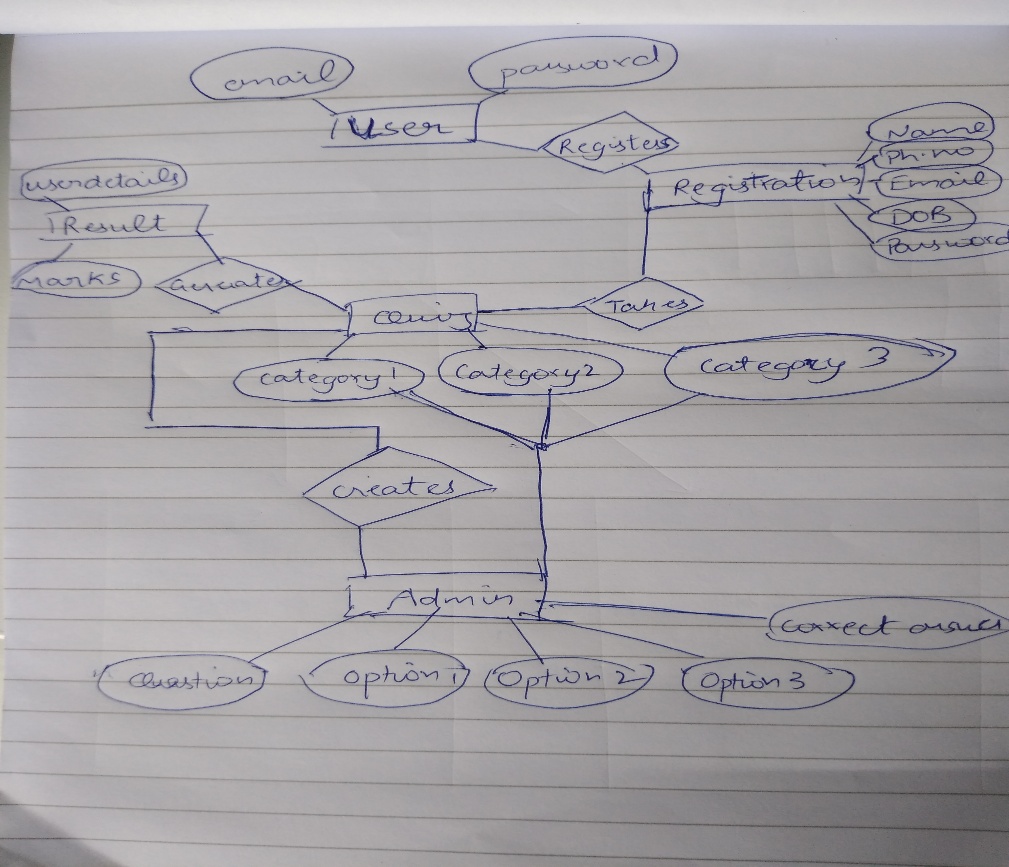
1. **Background and strategic fit**.

The reason behind the development of the web application is to allow Perficient to test potential recruits and their employees. The application was to be structured to include multiple individual categories/topics each with a separate quiz.

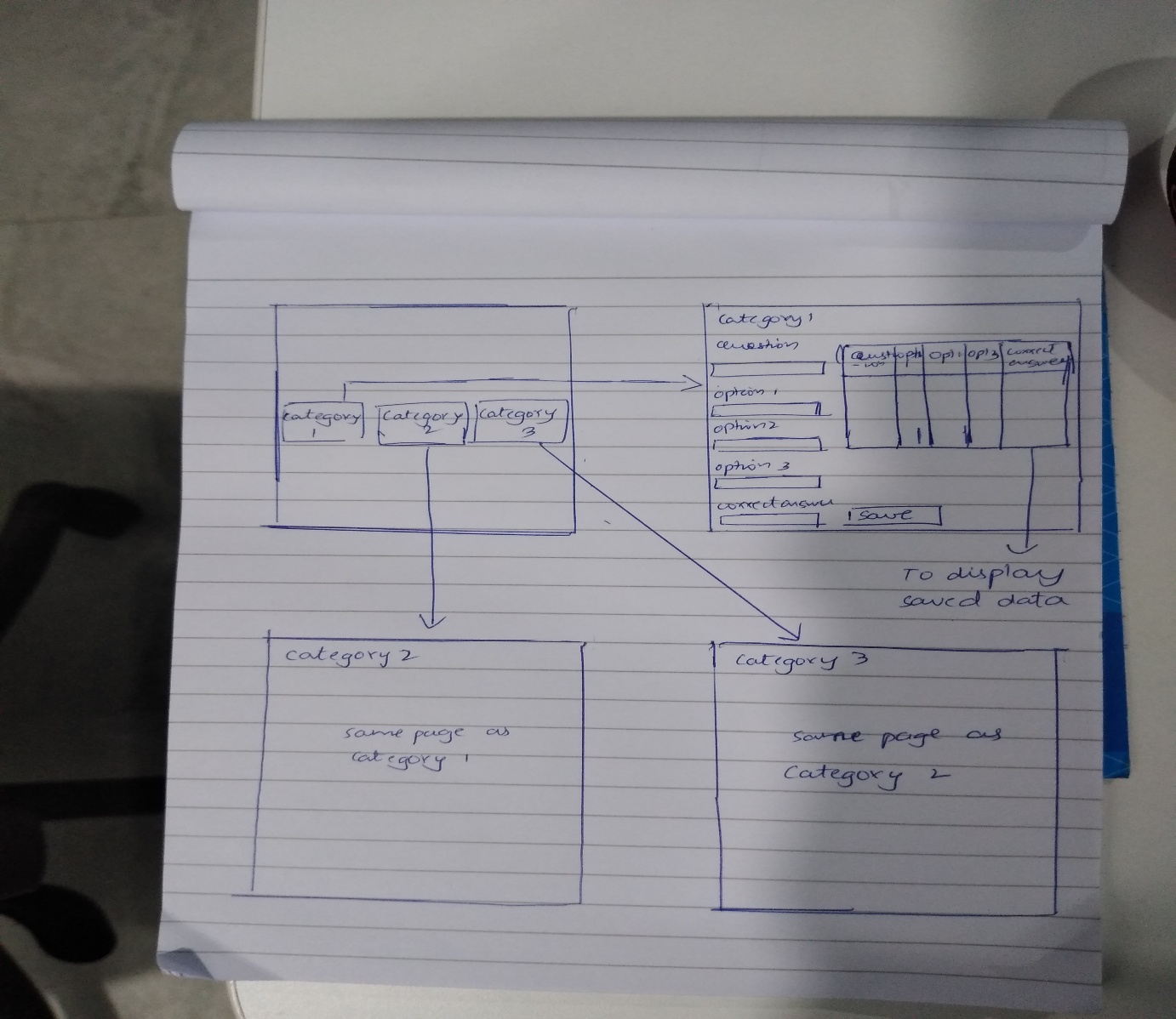
1. **User interaction and design**.



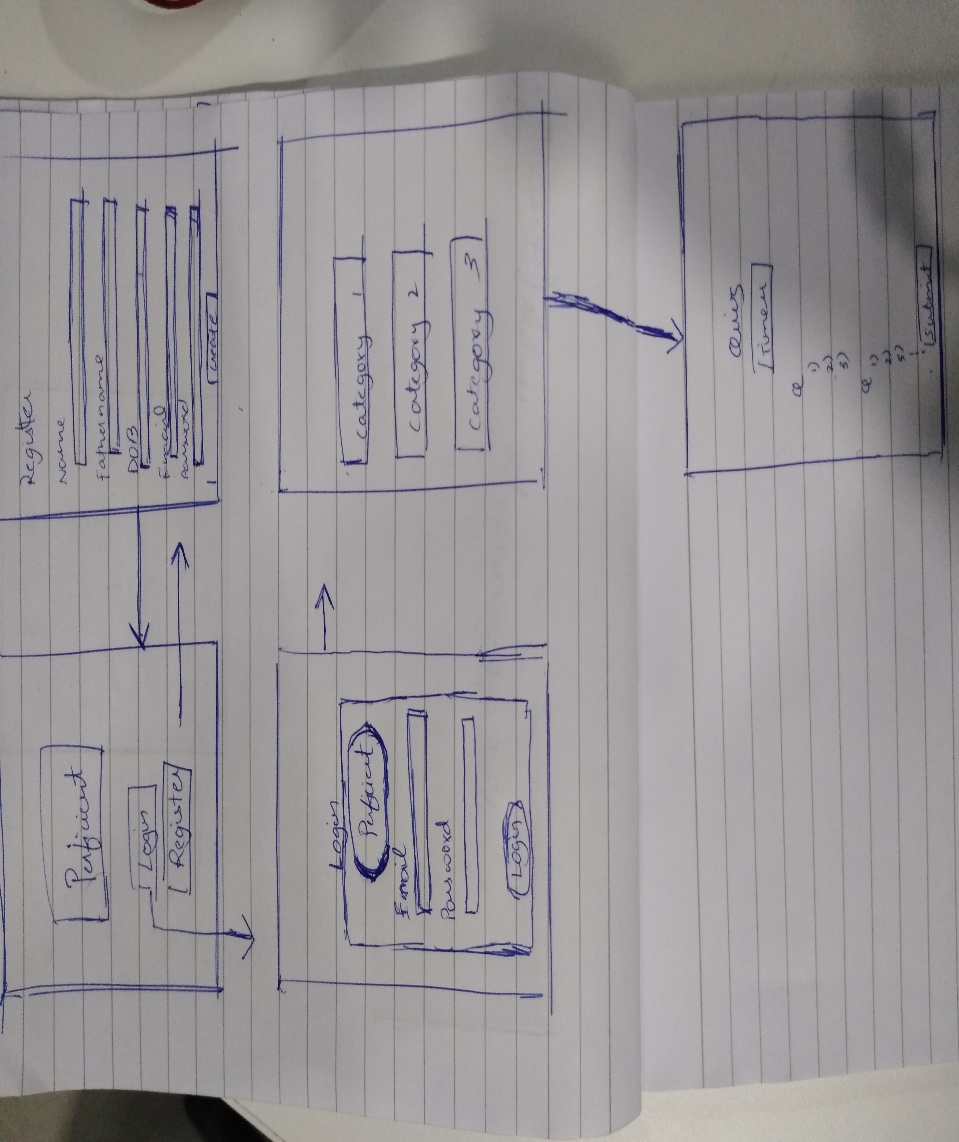
BASIC FLOWCHART FOR APPLICATION



E.R. DIAGRAM FOR THE APPLICATION



BASIC DESIGN FOR THE ADMIN MODULE



BASIC DESIGN FOR THE USER MODULE

1. **Completed**

Admin Module –

* Can perform CRUD operations on database. Allows the admin to store, edit and delete questions, options(answers) and correct answer.

User Module –

* Allows the user to select the category for the test. Test questions and options are displayed in MCQ format and score is generated upon clicking submit.

1. **To do**

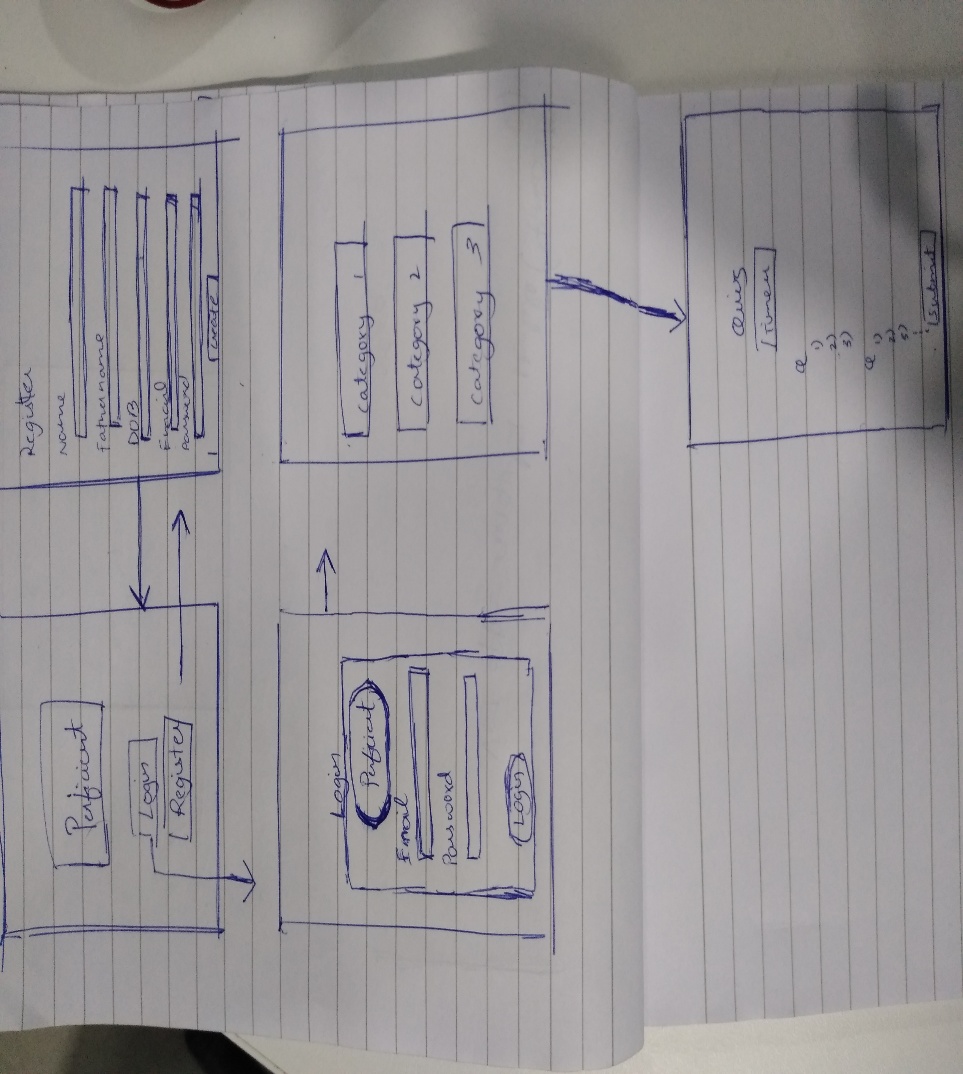
Admin Module -

* Once the questions, options and correct answer are submitted and stored, the page must be refreshed in order to submit/store once again.
* Allow the admin to create multiple categories.

User Module –

* Create a database for registration details and one for storing the outcome of the test.
* Creating a time automated test.
* Allowing registered users to login

User Module



The components included in the user module as in source code are –

1. Home component – This component consists of a login and register button which routes to the loginpg and register component respectively.
2. Register component – This component consists of a form with personnel details of the user. And once the form is submitted it routes back to the Home component.
3. Loginpg component – This component consists of the login page with email and password and routes to the category component.
4. Category component – This component displays the various categories available in the form of buttons. Category1, Category2 and Category3 route to components quiz, quiz1 and quiz2 respectively.
5. Quiz component – This component retrieves the questions, options and correct answer and displays it in MCQ format.
6. Quiz1 component - This component retrieves the questions, options and correct answer and displays it in MCQ format.
7. Quiz2 component - This component retrieves the questions, options and correct answer and displays it in MCQ format.

NodeJS folder –

* Db.js – Connections to mongoose.
* Index.js – Start server at port 3000 using express, mongoose and body-parser.

./models/test.js – establish schema of the collection and to use tests collection

* Employeecontroller.js – specifies get, put and post functions.

NodeJS1 folder –

The contents of this folder are the same as NodeJS folder with the exception that is used to establish a connection to a different collection db1.

NodeJS1 folder –

The contents of this folder are the same as NodeJS folder with the exception that is used to establish a connection to a different collection db2.

Shared folder –

* Test.model.ts – used to export class test.
* Test.service.ts – contains functions post, get, put and delete to perform the necessary functions.

Shared1 folder –

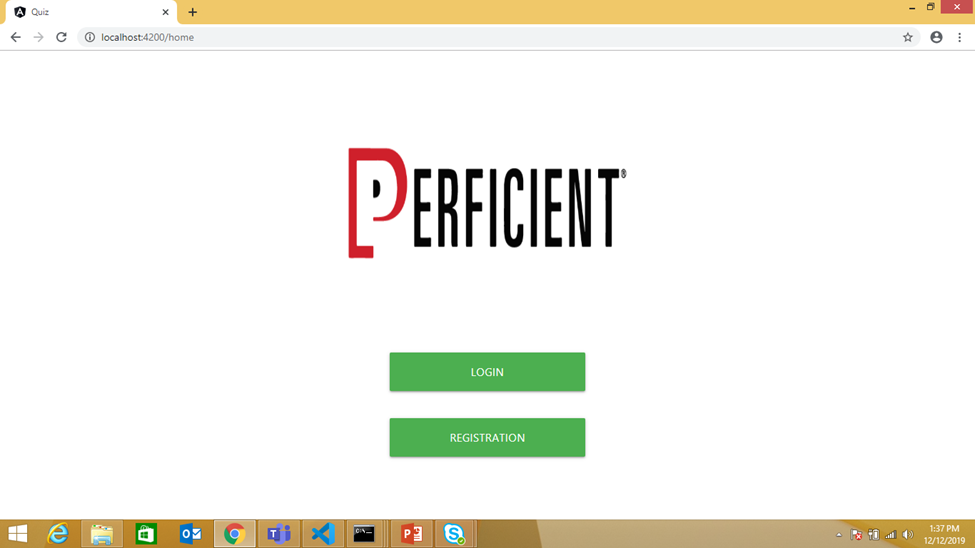
The contents of this folder are the same as the Shared folder.

Shared2 folder –

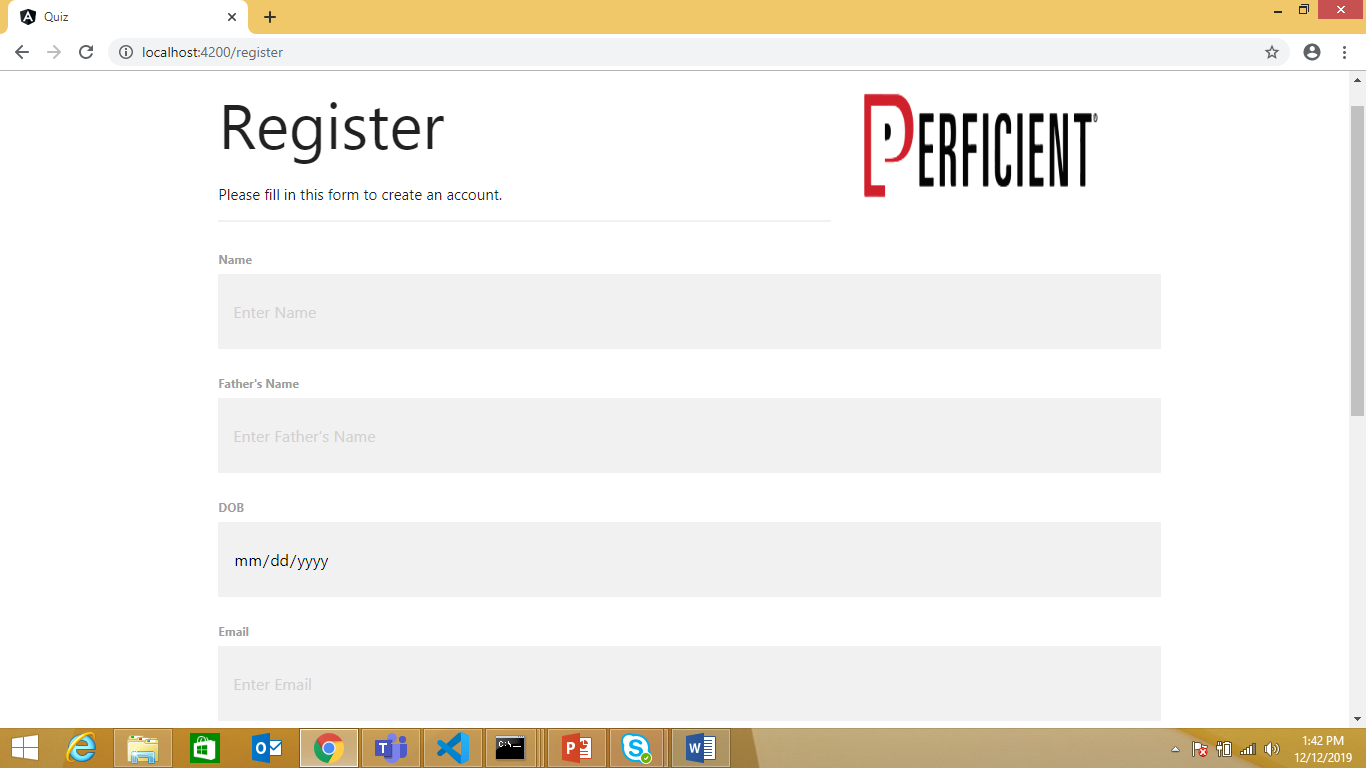
The contents of this folder are the same as the Shared folder.

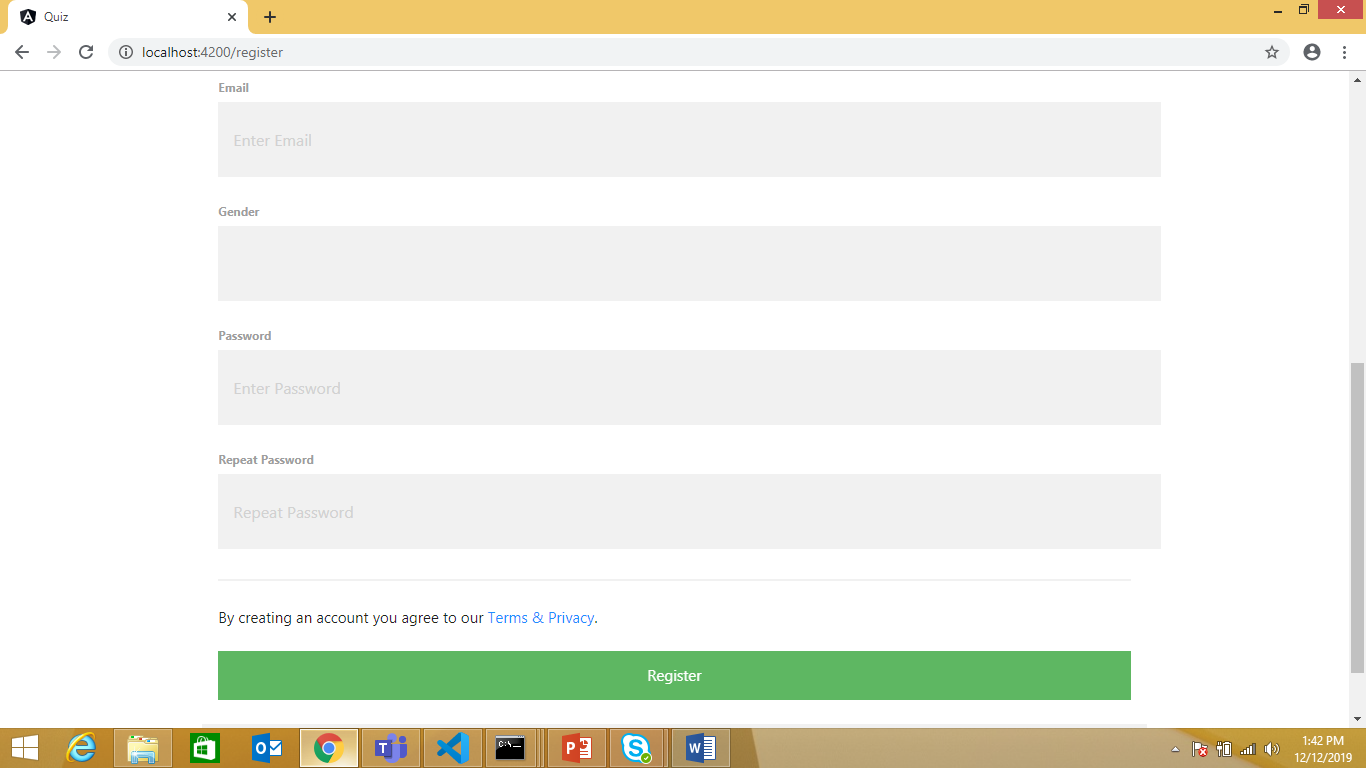
**User Experience design**

Home component

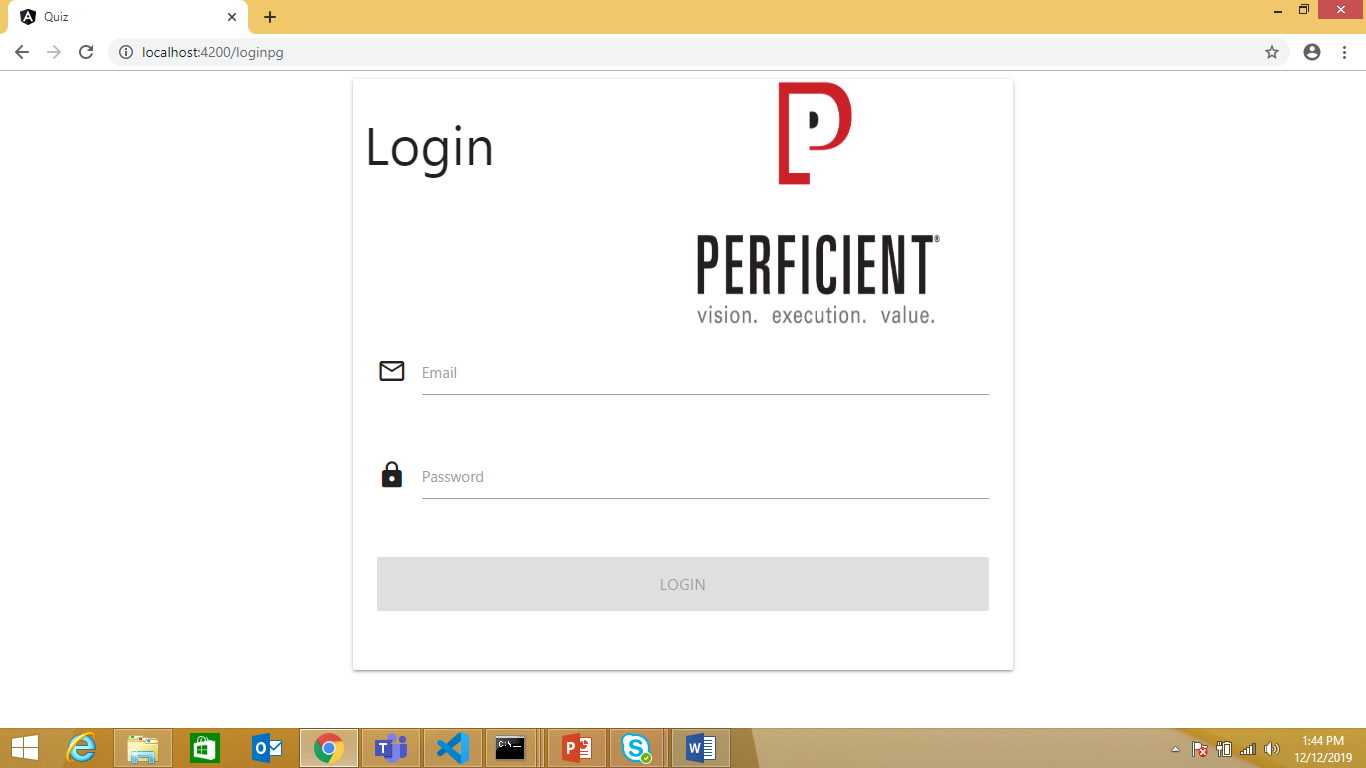


Register component

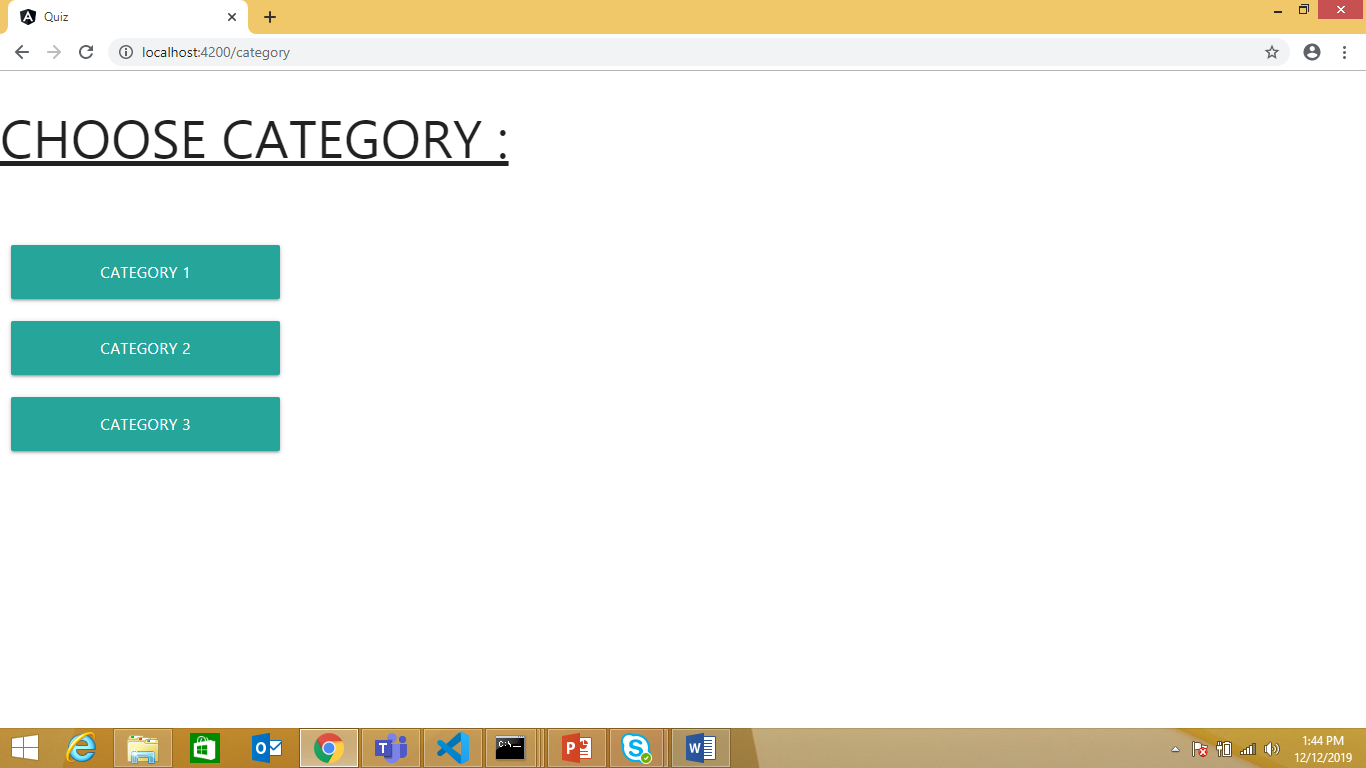




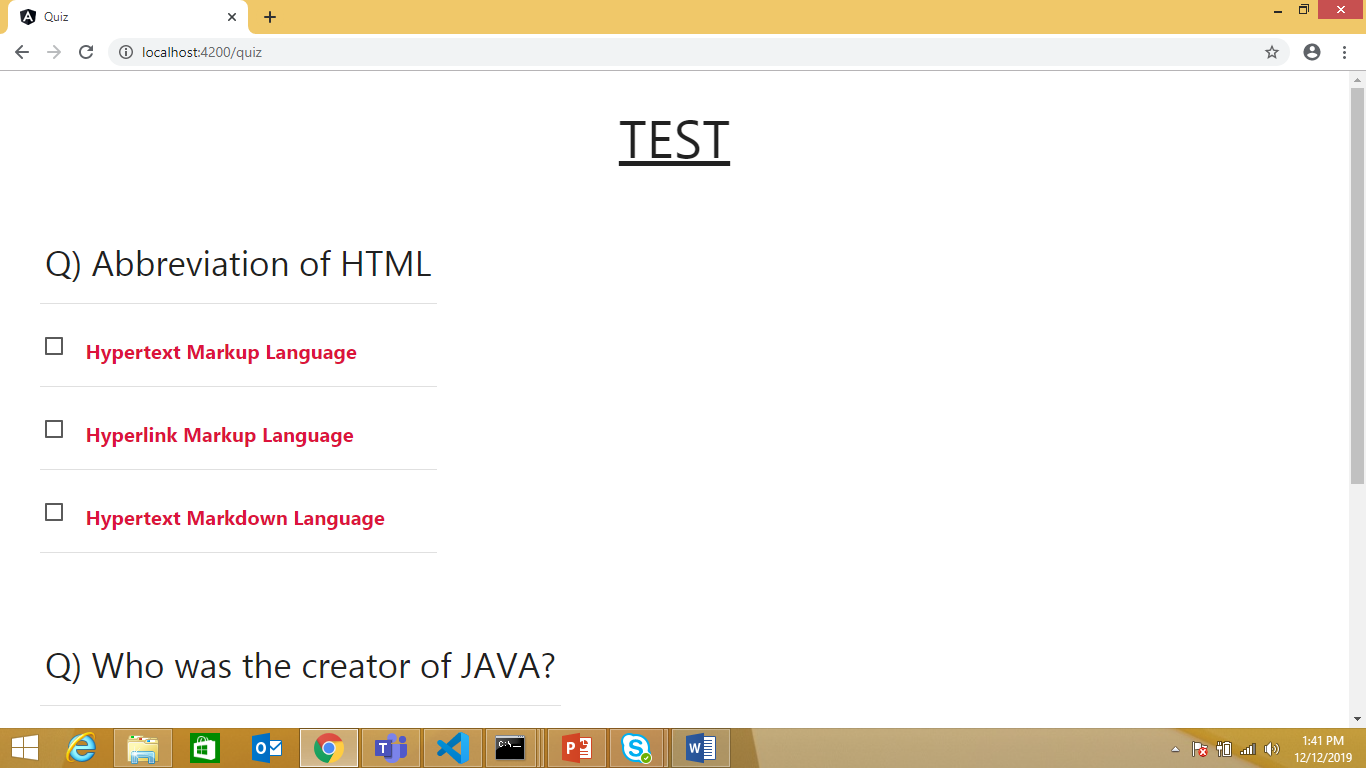
Loginpg component



Category component



Quiz component



Admin module

Home component – This component displays three buttons category1, category2 and category3 which routes to components test, category1 and category2 respectively.

Test component – The test component is where the quiz for category 1 is prepared. It consists of a form to enter the question, option 1, option 2, option 3 and the correct answer. A table displays the entered data and allows the admin to edit or delete it. The test component uses the tests collection.

Category1 component – The test component is where the quiz for category 2 is prepared. It consists of a form to enter the question, option 1, option 2, option 3 and the correct answer. A table displays the entered data and allows the admin to edit or delete it. The test component uses the db1 collection.

Category2 component – The test component is where the quiz for category 3 is prepared. It consists of a form to enter the question, option 1, option 2, option 3 and the correct answer. A table displays the entered data and allows the admin to edit or delete it. The test component uses the db2 collection.

NodeJS folder –

* Db.js – Connections to mongoose.
* Index.js – Start server at port 3000 using express, mongoose and body-parser.

./models/test.js – establish schema of the collection and to use tests collection

* Employeecontroller.js – specifies get, put and post functions.

NodeJS1 folder –

The contents of this folder are the same as NodeJS folder with the exception that is used to establish a connection to a different collection db1.

NodeJS1 folder –

The contents of this folder are the same as NodeJS folder with the exception that is used to establish a connection to a different collection db2.

Shared folder –

* Test.model.ts – used to export class test.
* Test.service.ts – contains functions post, get, put and delete to perform the necessary functions.

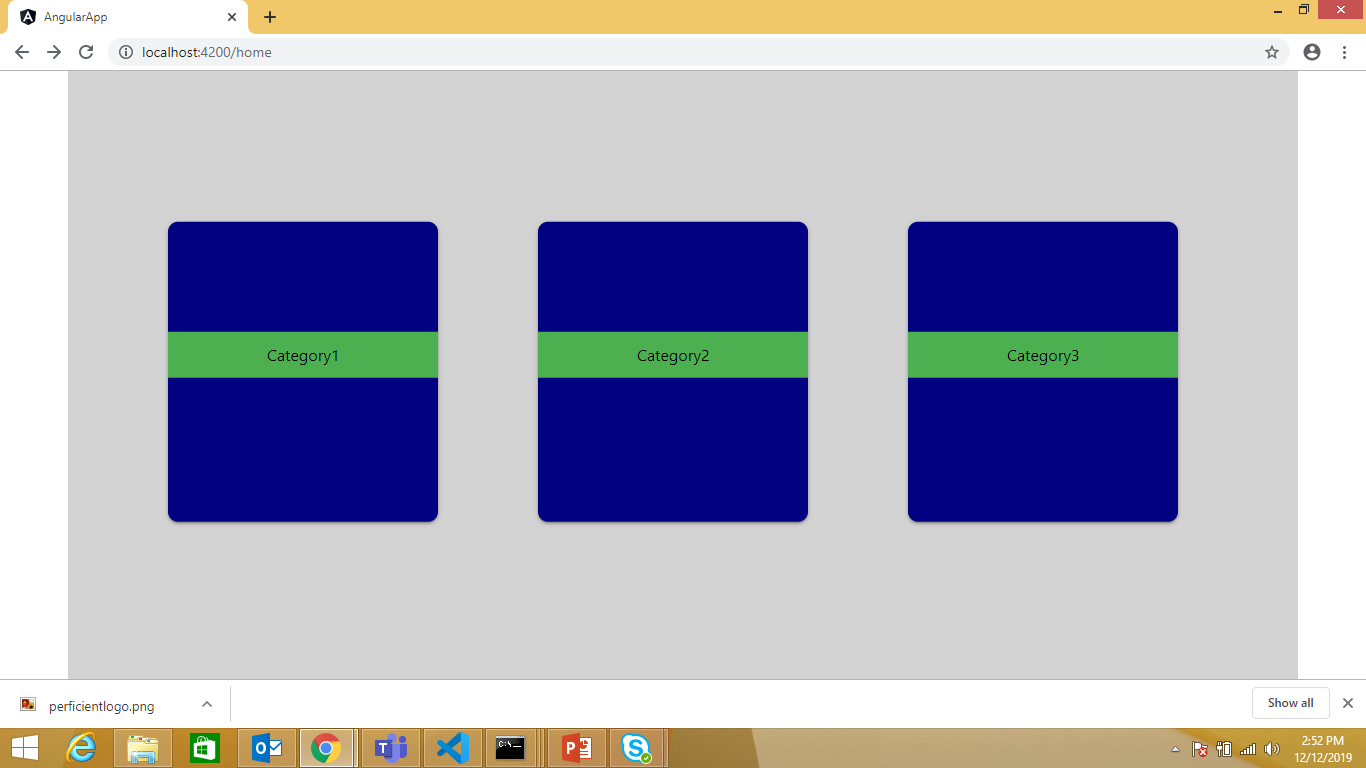
Shared1 folder –

The contents of this folder are the same as the Shared folder.

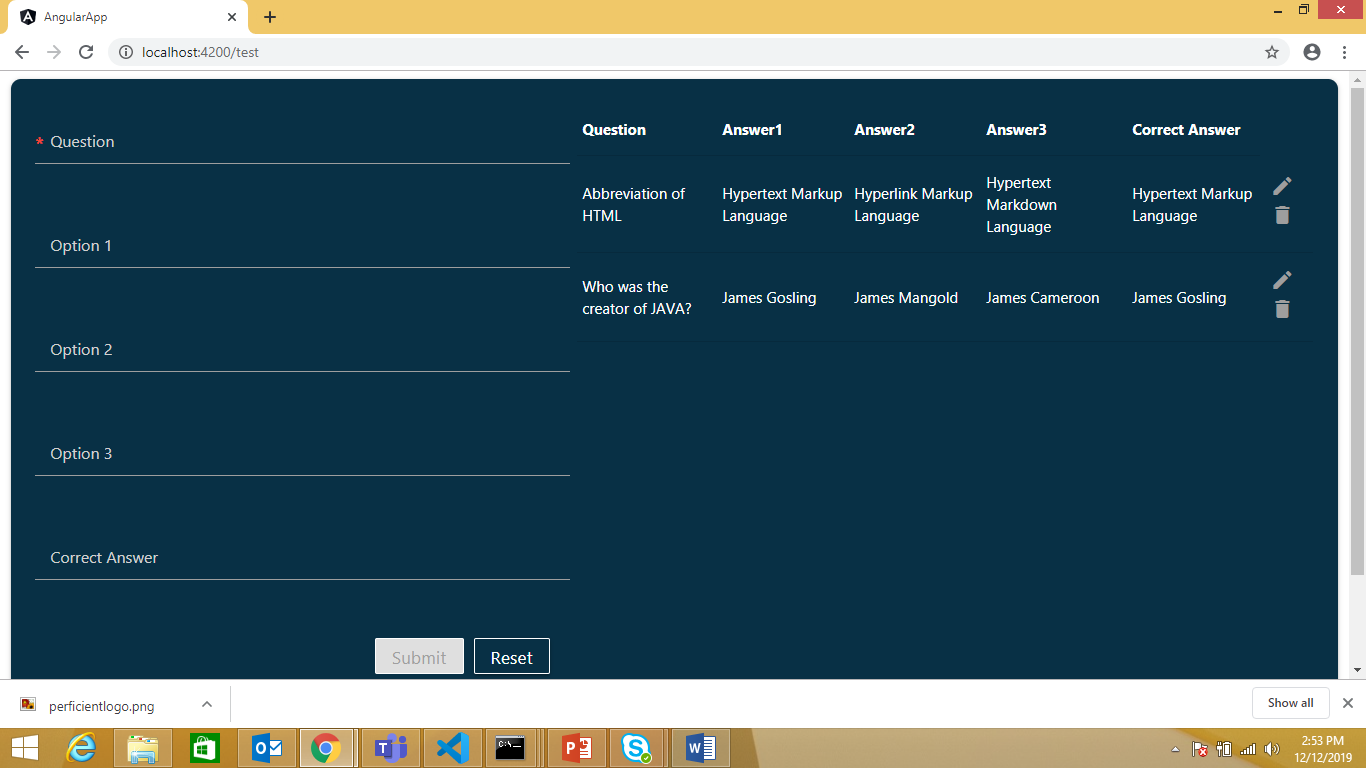
Shared2 folder –

The contents of this folder are the same as the Shared folder.

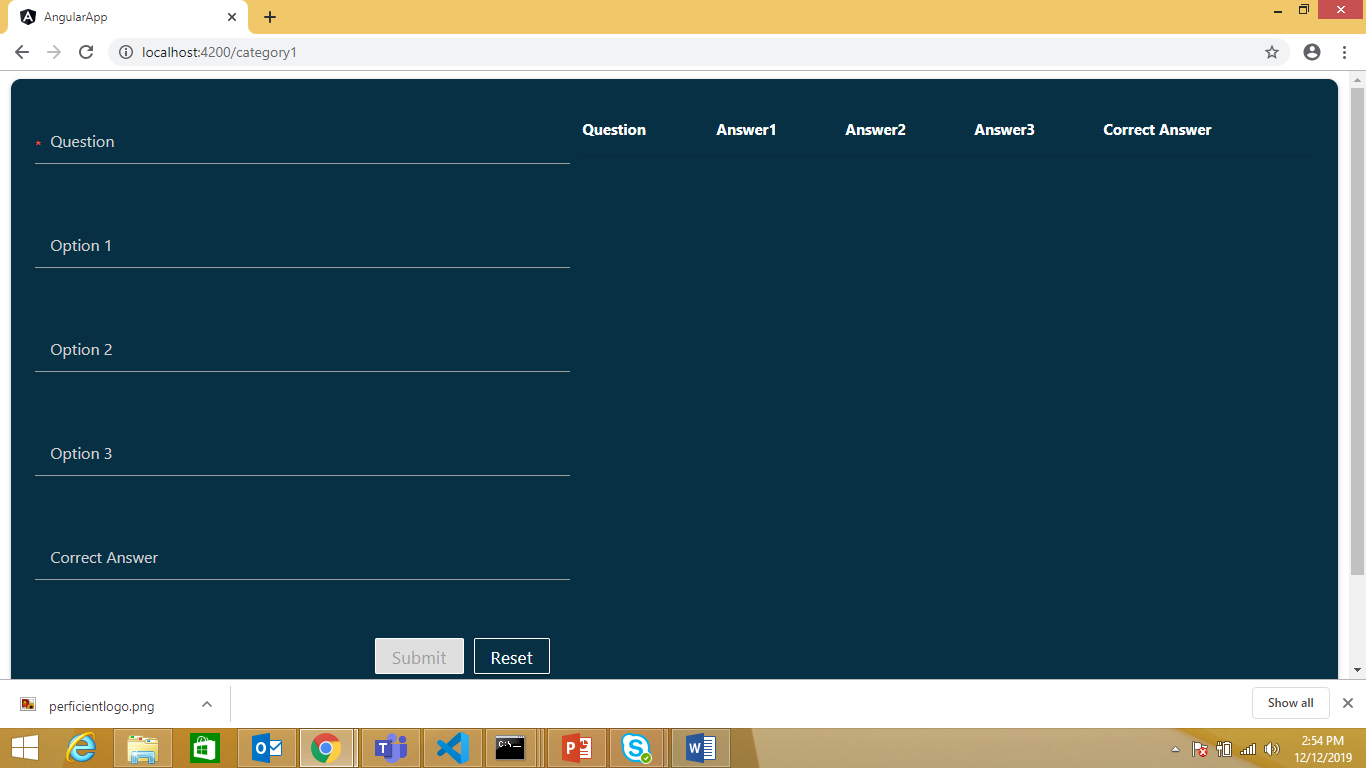
**Home Component**



**Test Component**



**Category1 Component**



**Category2 Component**

