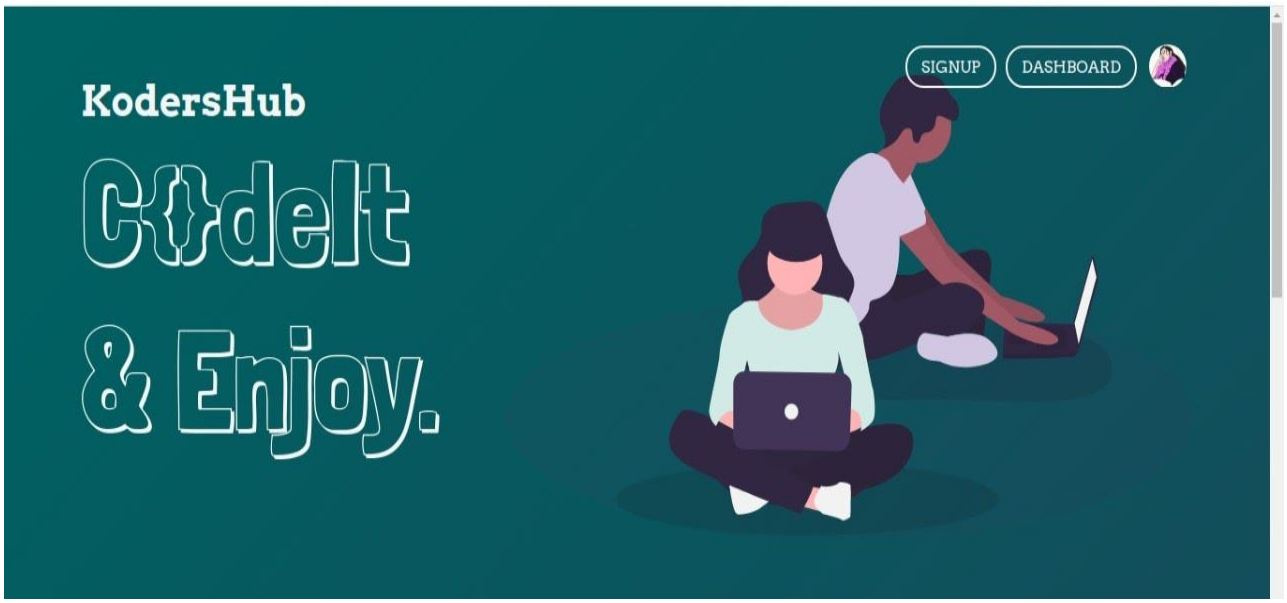
REPORT



KodersHub

Coding Webapp for Teens

Links:

- Github Project URL
- Live Preview

Made By:

- Syeda Tooba Ali (SE-53)
- Fareena Ehtesham (SE-58)
- M.Hasham Khalid (SE-79)
- Midha Tahir (SE-87)

TABLE OF CONTENT

Overview

This app provides a platform for our teens to practice coding skills with small challenges and compete in the IT world.

Our First Iteration:

For the completion of our project, we have taken one iteration. In which we promote communication, planning, modeling, construction, deployment and other umbrella activities.

Communication

Due to pandemic, our communication takes place in either virtual meetings or in groups.

Electronic focus group

First, we decided the domain of our project. All team members confirm this idea as we want to take this project to our final year.

We have done different meetings to discuss our project plan, framework and how we will be building this application.

Secondly we have done market research together.

Stakeholders:

- Students
- Teens want to learn the absolute beginner level of programming.

Functional Requirements:

- An attractive home page
- User Sign up and login functionality
- Challenging questions
- IDE for solving coding challenges
- User dashboard

Non-Functional Requirements:

- Application must be reliable
- Application must be secure
- Application should respond quickly
- Application should be available at any time

Planning

Decision of React Components: At First, we decided which and how many different code snippets we will be making into reusable components.

Decision of Data: We have planned where to put our main data and how to carry this data to other components.

Decision of Server-Side Routes: We have decided different routes for testing and signup. We have also given different routes for our frontend pages.

Decision of version control: We have decided to collaborate on github Repository for our code versioning.

Milestone Table:

#	Milestone Event	Start Date	Completion Date	Description
1	Ace Editor	10 June	15 June	Research and integrate ace code editor.
2	Informational Web App	16 June	30 June	Collection of requirements and placing them in components.
3	Backend Development	1 July	15 July	Backend library research for html, css, js testing.
4	Signup and Login Logic	15 July	20 July	Sign up and logic coding logic with database integration and passport js.
5	Integration of Frontend and Backend	21 July	30 July	Integration of react app with node backend along with mongo db.

6	Testing	1 August	3 August	Unit testing.
7	Deployment	8 August	13 August	AWS deployment.
8	Report Writing	13 August	15 August	Report Writing of our project.

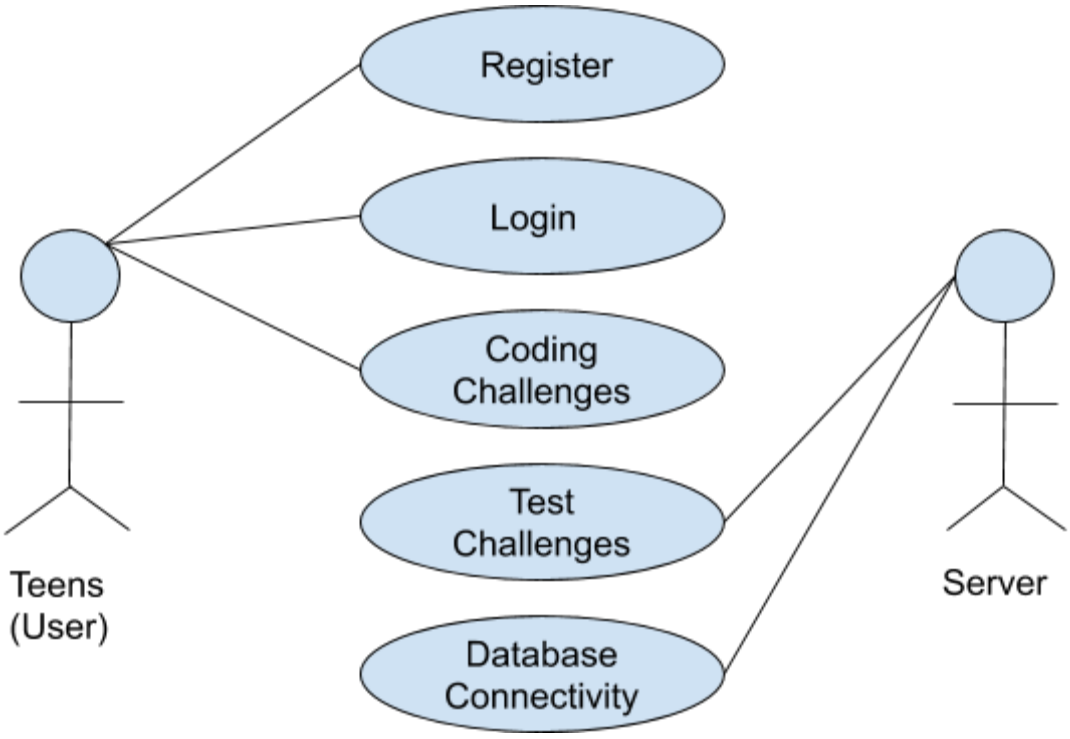
Modeling

Designing storyboard and UI for our app: [Figma Link]

Interface requirements: Meeting needs for our stakeholders and designing user-friendly and colorful UI for them.

Database requirements: We have decided that an unstructured database will be best suitable for our application.

Use Case Diagram:



Construction

We have designed our webapp using:

For frontend: React

For backend: Node and ExpressJS

For database: mongodb and mongoose

For UI: Material-UI and our own custom css

For Designing: Figma, Adobe Photoshop and Adobe Illustrator

For testing: Mocha , Chai, Lodash

We are testing the function expected from users in these files: using chai (for assertion) and mocha as the testing library.

Test for Add function

```
const { expect } = require("chai");
const resnap = require("resnap")(); // resnap to clear require cache

describe("Adder function", () => {
  beforeEach(resnap); // clearing require() cache

  it("Test # 1", () => {
    const program = require("../program");
    expect(program(10, 15)).to.equal(25);
  });

  it("Test # 2", () => {
    const program = require('../program');
    expect(program(100, 200)).to.equal(300);
  })
});
```

```
1) Adder function
     Test # 1:

    AssertionError: expected -5 to equal 25
    + expected - actual

    --5
    +25

    at Context.it (program_test.js:4:121)

2) Adder function
     Test # 2:

    AssertionError: expected -100 to equal 300
    + expected - actual

    --100
    +300

    at Context.it (program_test.js:4:203)
```

Test for Subtract function:

```
const chai = require("chai");
const resnap = require("resnap")(); // resnap to clear require cache

describe("Subtracter function", () => {
  beforeEach(resnap); // clearing require() cache

  it("Test # 1", () => {
    const program = require("../program");
    chai.expect(program(15, 10)).to.equal(5);
  });

  it("Test # 2", () => {
    const program = require("../program");
    chai.expect(program("a", 'b')).to.throw();
  });
});
```

```
0 passing (24ms)
2 failing

1) Subtracter function
     Test # 1:

    AssertionError: expected 25 to equal 5
    + expected - actual

    -25
    +5

    at Context.it (program_test.js:4:126)

2) Subtracter function
     Test # 2:

    AssertionError: expected 120 to equal 80
    + expected - actual

    -120
    +80

    at Context.it (program_test.js:4:206)
```

Testing of React Component

We used the jest testing library for testing react components in the development phase.

```
import React from 'react';

const title = 'Hello React';

function App() {
  return <div>{title}</div>;
}

export default App;
```

Test written in App.test.js

```
import React from 'react';
import { render } from '@testing-library/react';

import App from './App';

describe('App', () => {
  test('renders App component', () => {
    render(<App />);
  });
});
```

Deployment

We have used AWS EC2 Instance for deployment. Steps of deployment are as follows:

1. First we take a free tier from AWS.
2. Then we launch a new instance for our app.
3. We selected Ubuntu 18.04 LTS and t2.micro
4. Added http set them to: `ssh 0.0.0.0, http 0.0.0.0`
5. Downloaded a `.pem` key from AWS, changed chmod 400 and ssh into server.
6. Installed nginx server and our project dependencies along with git

7. Then we clone the project on the server and set some config files. Run server with pm2.
8. We have purchased the domain from name.com and set it up.

Next iteration:

We will be implementing process level isolation using docker in JS code.

We will calculate and show the score of each task on frontend. We will also show progress bar of languages progress.

We will combine other languages also e.g python etc.