**DEPLOYMENT**

**GitHub Pages Deployment**

After building and thoroughly testing the "Quiz Hub" website, the next step was deployment. Since this project was primarily for educational purposes and we wanted to make it publicly available, we decided to deploy it using GitHub Pages, a free hosting service offered by GitHub for static websites.

GitHub Pages allows developers to host HTML, CSS, and JavaScript files directly from a GitHub repository. The deployment process was relatively straightforward, as GitHub Pages integrates seamlessly with GitHub repositories. After pushing the final version of our project to the main branch, we simply enabled GitHub Pages from the repository’s settings and pointed it to the root directory of the project.

One of the major benefits of using GitHub Pages was the simplicity and ease of setup. We didn't need to configure a server, manage databases, or worry about scalability. The website could be accessed directly through the URL generated by GitHub Pages.

**Repository and Public Accessibility**

Once deployed, we made sure that the repository was publicly accessible on GitHub so that others could view and contribute to the project. This was a key part of the collaborative nature of the project, as it allowed other developers or students to fork the repository, suggest changes, and even contribute code improvements.

Our GitHub repository is publicly available at:

<https://github.com/Rohit-Sharma-1/interactivequizwebsite>

We included detailed documentation within the repository, including an updated README.md file, which explained how to set up the project locally, the technologies used, and the contribution guidelines. This documentation was essential for making the repository usable and easy to navigate for anyone interested in using or contributing to the project.

Having the repository on GitHub also provided version control, which was important for tracking changes, rolling back to previous versions when necessary, and coordinating with team members during the development process. The use of GitHub allowed us to stay organized and collaborate effectively on a single code-base.

**Continuous Integration and Deployment (CI/CD)**

Although we used GitHub Pages for deployment, we also set up basic continuous integration (CI) using GitHub Actions. GitHub Actions automatically triggered build and test processes whenever changes were pushed to the repository. This helped maintain quality control by ensuring that new changes didn’t break existing functionality.

We configured the action to run the tests after each commit, ensuring that the website's functionality was always intact before deployment. This automated workflow streamlined the development process, allowing us to catch issues early and reduce the chance of introducing bugs into the live version of the site.

**Deployment** refers to the process of taking a software application—especially a web application—and making it available for use on the internet or a private network. It's the bridge between **development** (writing code) and **delivery** (serving that code to users in a usable form).

Deployment was one of the most exciting stages in building Quiz Hub. After weeks of designing, coding, and testing, we were finally ready to put our project out into the world. But as we learned, **deployment isn’t just about clicking "publish"—it's about preparing the app for real-world use, making it accessible, ensuring it runs smoothly, and keeping it live .**

We chose **GitHub Pages** as our deployment platform because it’s free, reliable, and perfect for hosting front-end projects like ours.

**Challenges Faced During Deployment**

Even though deploying Quiz Hub using GitHub Pages seemed straightforward on paper, we encountered several real-world obstacles that made us better developers by the end of the process. These challenges taught us how crucial deployment is—not just from a technical standpoint, but also in terms of teamwork, configuration, and end-user experience.

**1. React Routing Not Working on Refresh**

One of the first issues we encountered was that React Router didn’t play well with GitHub Pages. When navigating between routes (like /quiz or /results), everything worked fine. But when a user refreshed the page or tried to access a route directly, they’d get a 404 error.

Why it happened: GitHub Pages serves a static index.HTML file, and doesn't understand dynamic client-side routes.

How we fixed it: We added a special 404.HTML file that contained a redirect script to index.HTML, ensuring that React could take over routing again after a refresh:

**2. Broken Relative Paths After Deployment**

After pushing our production build to GitHub Pages, we noticed that many assets like images and style-sheets weren’t loading correctly.

Cause: We hadn't set the homepage field correctly in our package.j son. As a result, the app assumed root-relative paths that didn’t exist on the GitHub Pages subdomain.

**3. Cache Issues Post-Update**

After pushing updates to the live site, users still reported seeing outdated content, which confused a lot of testers.

Why it happened: Browsers were caching old versions of the JavaScript and CSS files. Since the filenames didn’t change, browsers didn’t fetch the new ones.

How we solved it:

* We enabled file hashing during the build process (create-react-app does this automatically).
* We also advised our testers to clear their cache and hard refresh the site.
* Later, we added a small version number in the footer so users could verify if they were on the latest version.

**4. Deployment Automation Misconfiguration**

We wanted to use GitHub Actions for automatic deployment every time a push was made to the main branch. Our first few workflows didn’t run properly due to:

* Wrong YAML syntax
* Missing build steps
* Incorrect directory paths

Fixes:

We consulted GitHub’s documentation and examples to fix the YAML file.

It was a big win when our first successful deployment happened automatically with just a push to main.

**5. Domain & SSL Confusion**

Some team members wanted to link a custom domain to the project using a .tech domain they had. However, setting up a custom domain with GitHub Pages and HTTPS turned out to be trickier than expected.

Problems included:

* DNS settings not propagating
* GitHub forcing HTTPS which didn’t work immediately with the domain.