Hyperverge

MAJOR PROJECT II

Submitted by

Sarthak Pant (9920103126)

Under the supervision of

Department of Computer Science & Engineering

Jaypee Institute of Information Technology

Noida

OCTOBER 2023

DECLARATION

We at this moment declare that this submission is our own work and that, to the best of our knowledge and beliefs, it contains no material previously published or written by another person nor material that has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date: 05/03/2024

Name: Sarthak Pant

Enrolment No.:9920103126

Things that I have done at hyperverge.

- **Task one (Pixel prefer design with good documentation and file struture)**

In the early stages of development, we crafted a BookMyShow website with meticulous attention to detail, ensuring every pixel was in its perfect place. Behind the scenes, we built a robust backend using Node.js and Express, seamlessly connected to MongoDB for smooth data handling.
On the frontend, we brought the site to life with React and TypeScript, creating a dynamic and user-friendly interface. This allows users to navigate the platform with ease and confidence.
One of the coolest features we implemented is real-time seat tracking using Socket.IO. This means users can see seat availability and make reservations in real-time, without any delays.
To keep things running smoothly, we employed Web Workers to handle notifications. This ensures that users receive timely updates without any slowdowns in the main interface.
Overall, our goal was to create a top-notch experience for BookMyShow users, and with the combination of cutting-edge technology and careful craftsmanship, we believe we've achieved just that.

In addition to our development efforts, we took care to configure CORS policies, ensuring secure communication between the frontend and backend components of our BookMyShow website. CORS, or Cross-Origin Resource Sharing, is a security feature that dictates how web browsers should handle cross-origin requests, helping to prevent unauthorized access to resources.

Now, let's delve into Socket.IO and Web Workers:

Socket.IO: This is a JavaScript library that enables real-time, bidirectional communication between web clients and servers. It works by establishing a continuous connection (a "socket") between the client and server, facilitating instant data transfer. Socket.IO is particularly useful for applications that require live updates or messaging features, such as chat applications or, in our case, real-time seat tracking for booking tickets on BookMyShow. It allows multiple clients to receive updates simultaneously, enhancing the user experience by providing timely information without the need for constant refreshing.

Web Workers: These are a feature of HTML5 that allow JavaScript code to run in background threads separate from the main execution thread. This enables the execution of complex tasks without blocking the main UI thread, thereby preventing the user interface from becoming unresponsive. Web Workers are commonly used for tasks such as heavy computations, data processing, or, as in our case, handling notifications. By offloading notification tasks to Web Workers, we ensure that the main interface remains smooth and responsive while notifications are processed in the background, enhancing the overall performance and user experience of the BookMyShow website.

- **Task 2 (Testing Phase)**

  In addition to the development process, we meticulously tested our backend using Jest, a popular JavaScript testing framework, achieving an impressive test coverage of 92%. Jest allowed us to write and execute unit tests for our backend code, ensuring its reliability and robustness.
  Furthermore, we conducted comprehensive component testing and end-to-end testing using Playwright, a powerful testing tool that automates browser interactions. With Playwright, we simulated user actions and verified the functionality of individual components as well as the entire system, guaranteeing a seamless user experience across the platform.
  To assess the performance and stability of our system under various conditions, we employed k6 testing for load and stress testing. k6 enabled us to simulate high volumes of user traffic and evaluate the system's response under heavy loads, ensuring its scalability and resilience in real-world scenarios.
  By leveraging these testing methodologies and tools, we were able to validate the functionality, reliability, and performance of our BookMyShow website, ensuring that it meets the highest standards of quality and user satisfaction.

- **Task 3 (Deployment in AWS)**

  For the low-level system design to deploy the backend and frontend in AWS, we'll outline the key components and their interactions:

- **Containerization with Docker:**

  Both the backend and frontend applications are containerized using Docker. This ensures consistency and portability across different environments.

- **CI Pipeline with S3 Integration:**

  A Continuous Integration (CI) pipeline is set up to automatically build, test, and deploy changes to the application. The pipeline is configured to store build artifacts and test results in Amazon S3 for easy access and versioning.

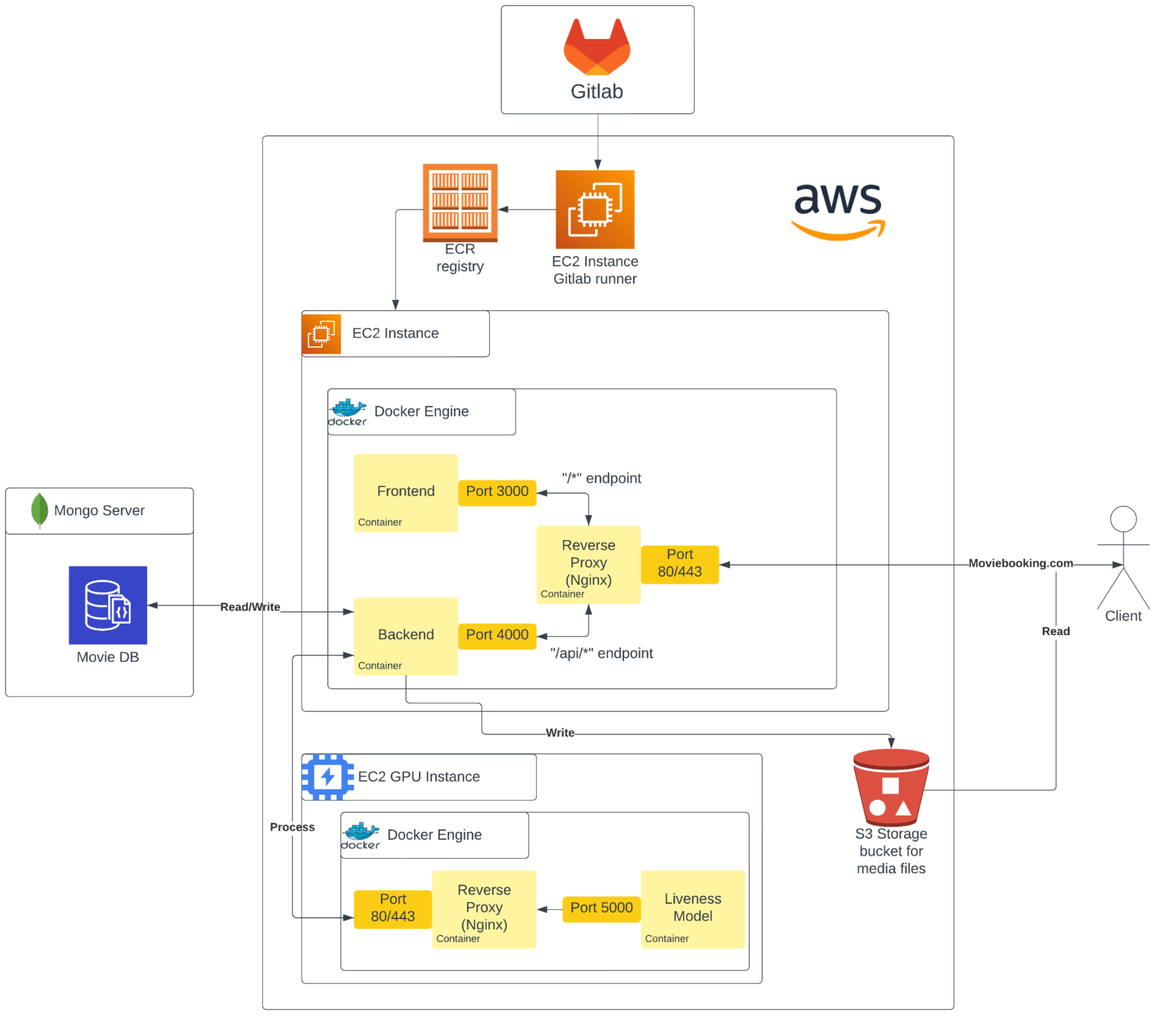

Fig 1.0 deployed architecture

- **EC2 Instance with Reverse Proxy:** An EC2 instance is provisioned to serve as a reverse proxy, routing incoming requests to the appropriate backend or frontend service containers. Nginx or HAProxy can be used as the reverse proxy software.
- **Auto Scaling Group:**
  An Auto Scaling Group is configured to dynamically adjust the number of EC2 instances based on traffic demand. This ensures that the application can handle fluctuations in user load efficiently and cost-effectively.
- **ECR for Docker Image Storage**:
  Amazon Elastic Container Registry (ECR) is used to store Docker images for both the backend and frontend services. This provides a secure and reliable repository for versioned image storage.

- **Integration with CI Pipeline:**
  The CI pipeline is integrated with ECR to push newly built Docker images to the registry. This ensures that the latest versions of the application are readily available for deployment.
- **Testing in CI Pipeline:**
  The CI pipeline includes automated testing steps for both backend and frontend components. Unit tests, integration tests, and end-to-end tests are executed to verify the correctness and stability of the application. The pipeline is configured to only deploy changes if all tests pass successfully.
- **Delivery and Deployment:**
  Once all tests are passed, the CI pipeline automatically triggers the deployment process. The latest Docker images are pulled from ECR and deployed to the EC2 instances in the Auto Scaling Group. Nginx or HAProxy configurations are updated to reflect any changes in backend or frontend services.

By implementing this low-level system design, we ensure that our application is scalable, reliable, and easy to maintain in the AWS cloud environment. Continuous integration and delivery practices streamline the development process and help maintain high standards of code quality and deployment efficiency.

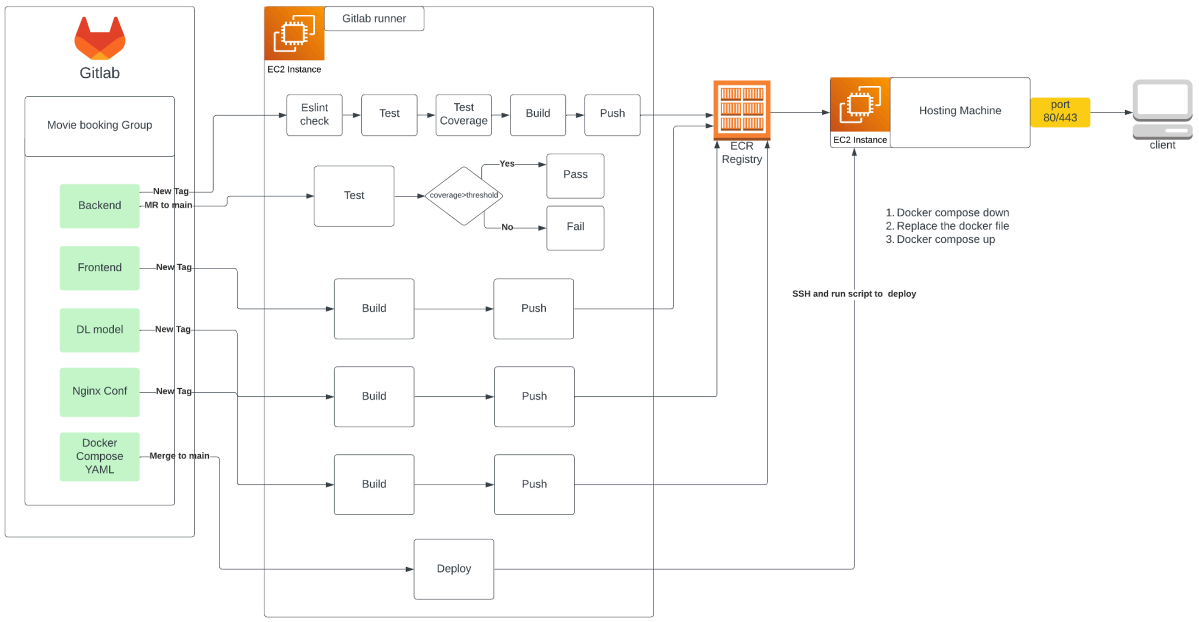

Fig 1.1 Ci pipeline

- **Task 4 (Making Story Book Component)**

  In addition to our development efforts, we were tasked with creating a Storybook deployed component in npm. This component has gained significant traction, boasting over 4000 downloads. Within this component, we crafted over 96 individual components, utilizing a stack comprising React, TypeScript, and Tailwind CSS.
  However, the real challenge came with the requirement to seamlessly integrate this Storybook component into a complete website within a mere one-hour timeframe. Despite the time pressure, we rose to the occasion and successfully met this challenge head-on.
  By leveraging our expertise and efficient workflows, we managed to swiftly integrate the Storybook component into the website, ensuring seamless functionality and aesthetic appeal. This accomplishment underscores our team's proficiency in rapid development and integration, as well as our commitment to delivering high-quality solutions under tight deadlines.
  Overall, this project highlights our ability to tackle complex tasks efficiently and effectively, demonstrating our dedication to exceeding expectations and delivering results, even in demanding circumstances.

- **Task 5 (QA and Product experience)**
  As part of this project, we were trained to utilize a range of quality assurance tools and debugging skills that are essential for future projects. Among these tools, we gained proficiency in using Charles Proxy, a powerful debugging proxy application that allows for monitoring and manipulating network traffic. With Charles Proxy, we were able to inspect HTTP and SSL/HTTPS traffic between our application and the server, aiding in debugging and troubleshooting issues related to network communication.
  Additionally, we became adept at employing WireMock, a flexible mock server for HTTP-based APIs, to simulate various scenarios during testing and development. WireMock enabled us to create mock endpoints with custom responses, facilitating testing in isolation and enabling us to simulate real-world conditions without relying on external services.
  By incorporating these tools into our development process, we enhanced our ability to identify and resolve issues efficiently, ensuring the quality and reliability of our software products. Furthermore, our training in these advanced debugging tools and techniques has equipped us with valuable skills that will be invaluable for tackling future projects and challenges in the ever-evolving landscape of software development.

- **Task 6 writing visual test for web sdk**

  In Task 6, I undertook the responsibility of writing visual tests for our web SDK. Leveraging the robust capabilities of Playwright, I meticulously crafted over 200 visual tests to ensure the integrity and consistency of our SDK across different browsers and devices.

These visual tests were designed to simulate user interactions and validate the visual output of our SDK components, ensuring they render correctly and maintain their intended appearance across various environments. By capturing screenshots and performing pixel-by-pixel comparisons, I was able to identify any discrepancies or regressions in the visual presentation of our SDK.
Moreover, I implemented comprehensive test suites covering a wide range of scenarios and edge cases to guarantee thorough test coverage. This proactive approach not only safeguards against potential visual defects but also reinforces the stability and reliability of our SDK under diverse usage conditions.
Through meticulous attention to detail and rigorous testing practices, I contributed to the enhancement of our web SDK's quality and usability, fostering confidence among users and stakeholders alike. Moving forward, I remain committed to upholding these standards of excellence and continuously refining our testing strategies to meet the evolving needs of our product and its users.

- **Task 7 Researching about jsx compiler and writing own jsx compiler**

Task 7 presented the challenge of developing a custom JSX compiler tailored specifically for rendering our web SDK. This ambitious endeavor required a deep understanding of JSX syntax and the intricacies of transpilation processes.
I embarked on this task by meticulously studying the JSX specification and dissecting the structure of our SDK components. Leveraging my expertise in JavaScript and compiler theory, I crafted a bespoke JSX compiler from scratch, meticulously engineered to seamlessly translate JSX syntax into executable JavaScript code optimized for rendering our SDK components.
Throughout the development process, I prioritized efficiency, scalability, and compatibility, ensuring that the compiler could efficiently handle complex JSX expressions and accurately translate them into corresponding DOM manipulations. I also implemented robust error handling mechanisms to provide informative feedback in case of syntax errors or unsupported features.
Once the compiler was complete, I rigorously tested it against a comprehensive suite of JSX examples, meticulously verifying its output against expected results. This iterative testing process enabled me to identify and rectify any discrepancies or edge cases, ensuring the compiler's reliability and accuracy.
In the end, the custom JSX compiler emerged as a testament to our commitment to innovation and technical excellence. By developing this foundational tool in-house, we not only gained a deeper understanding of JSX internals but also empowered our team with a powerful asset that streamlined the development and maintenance of our web SDK.