

Front-end application using NextJS at Verifyme India / Ques10

A REPORT

submitted by

Sahil Arora (19BCE1366)

*in partial fulfilment for the award
of*

**B. Tech. Computer Science and Engineering
School of Computer Science and Engineering**



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

April 2022



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

I hereby declare that the project entitled “**Front-end application using NextJS at VerifyMe India / Ques10**” submitted by me to the School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai 600127 in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology – Computer Science and Engineering** is a record of bonafide work carried out by me. I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

Sahil Arora (19BCE1366)



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CERTIFICATE

The project report entitled “**Front-end application using NextJS at VerifyMe India / Ques10**” was prepared and submitted by **Sahil Arora (Register No: 19BCE1366)**. It has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology – Computer Science and Engineering** in Vellore Institute of Technology, Chennai, India.

Examined by:

Examiner I

Dr. Tulasi Prasad Sariki

tulasiprasad.sariki@vit.ac.in

Examiner II

Dr. Mirza Galib Anwarul Husain Baig

mirzagalib.anwarul@vit.ac.in

RH1-K/34, SECTOR 7
VASHI
NAVI MUMBAI - 400703
hello@verifyme.in

December 8, 2021

Internship completion letter

We are glad to inform you that **Mr. Sahil Arora** from **Vellore Institute of Technology - VIT Chennai**, has successfully completed his internship at Verifyme India from **8 October, 2021 - 8 December, 2021**.

During this internship, he worked on a front-end application using Reactjs.

We found him extremely inquisitive and hard working. He was very much interested to learn the functions of our core division and also willing to put his best efforts and get into the depth of the subject to understand it better.

His association with us was very fruitful and we wish him all the best in his future endeavors.

For Verifyme India

A handwritten signature in black ink, appearing to read "Yashbeer Singh", with a horizontal line drawn underneath.

Authorized Signatory
Yashbeer Singh | Founder

ACKNOWLEDGEMENT

I would like to take this opportunity to acknowledge Dr. Nithyanandam P, Head of the Department (HoD), B.Tech Computer Science and Engineering, SCSE, VIT Chennai, Dr. Ganesan R, Dean of the School of Computer Science & Engineering, VIT Chennai, and Dr. Geetha S, Associate Dean of the School of Computer Science & Engineering, VIT Chennai, for providing us the necessary guidance and assistance in learning the skills required to be applied in industry grade applications and products. Their teachings and mentorship will forever remain impactful and serve as a great foundation for us to build our careers and futures upon. Finally I would also like to thank my parents and friends for their constant encouragement and support, which motivated me to always give my best effort and utilise every opportunity presented to me.

CONTENTS

Chapter	Title	Page
	Title Page	i
	Declaration	ii
	Certificate	iii
	Industry certificate	iv
	Acknowledgement	v
	Table of contents	vi
	List of Tables	vii
	List of Figures	viii
	List of Abbreviation	ix
	Abstract	x
1	Introduction	01
2	Application Architecture	03
3	Authentication Patterns	04
4	Frontend Design	07
5	Development Environment	10
6	Frontend Design	11
7	Testing	12
8	Internship Schedule	14
9	Conclusion	15
10	References	16
11	Appendix - 1	17
12	Appendix - 2	18

LIST OF TABLES

Title	Page
Table 1 : Compare and Contrast between SSG, SSR and CSR	08
Table 2 : Weekly Internship Schedule	14

LIST OF FIGURES

Title	Page
Fig 1 : Steps for student verification	01
Fig 2 : Base Application Architecture	02
Fig 3 : JWT Authentication Pattern	05
Fig 4 : Frontend Design Architecture	07
Fig 5 : GitFlow Workflow Method	08
Fig 6: Mobile First Design	11
Fig 7: Cypress Testing, GUI based and Headless	12
Fig 8: Example passing Cypress Test run	13

LIST OF ABBREVIATIONS

Abbreviation	Expansion
SaaS	Software as a Service
BaaS	Backend as a Service
CEO	Chief Executive Officer
AWS	Amazon Web Services
JS	Javascript
CSS	Cascading Style Sheet
JWT	Json Web Tokens
HTTP	HyperText Transfer Protocol
XSS	Cross Site Scripting
CSFR	Cross Site Request Forgery
TOPT	Time based one time password
CSR	Client Side Rendering
SSR	Server Side Rendering
SSG	Static Site Generation
SEO	Search Engine Optimization
CI	Continuous Integration

ABSTRACT

Online education has experienced a massive boost due to the popularity of remote learning, digital courses, and most recently, the COVID-19 pandemic. As a result, online student identity verification has come to the fore as a simple & secure way to verify student identities & onboard both students & staff cost-effectively.

The manual & paper-based methods of student verification that are quickly being replaced are neither scalable nor convenient for educational institutions. They often result in lost student documents, long search times, errors in verification resulting in incorrect student documentation, low productivity for administrators, and massive expenses incurred due to insufficient storage.

Educational institutions that still rely on manual & paper-based methods of student verification take 5-10 days to verify a single student's ID documents, with administrators sometimes spending half their time simply searching for the right document.

Manual ID verification is, additionally, not scalable. As institutions increase the size of their student bodies, paper-based document verification simply can't keep up with the sheer number of IDs to be verified.

Verifyme India aims to streamline the process in which student verification takes place. Currently there are many services that require student verification to provide discounts or additional benefits but either they require an educational account that is linked to a database or manual verification. With Verifyme India, Ques10 wanted to provide faster and easier student verification for their service and others alike. Current implementation of student verification is filled with hassles and usually given for a specific period of time, example for 1 year. With the approach taken by Verifyme India, the student verification status would be allotted for the duration where the student is currently studying for. Additionally student verification with Verifyme India would take place as a whole. When a student verifies its status to a service, the

same student need not verify again for another service. The same student verification status is reproduced for the new service.

Verifyme India aims to be a Software as a Service (SaaS) while also working as a Backend as a Service (BaaS). They aim to provide both redirect based services for student verification and an API based service for other product partners where they would be able to implement this functioning on their end.

INTRODUCTION

Online education has experienced a massive boost due to the popularity of remote learning, digital courses, and most recently, the COVID-19 pandemic. As a result, online student identity verification has come to the fore as a simple & secure way to verify student identities & onboard both students & staff cost-effectively.

The manual & paper-based methods of student verification that are quickly being replaced are neither scalable nor convenient for educational institutions. They often result in lost student documents, long search times, errors in verification resulting in incorrect student documentation, low productivity for administrators, and massive expenses incurred due to insufficient storage.

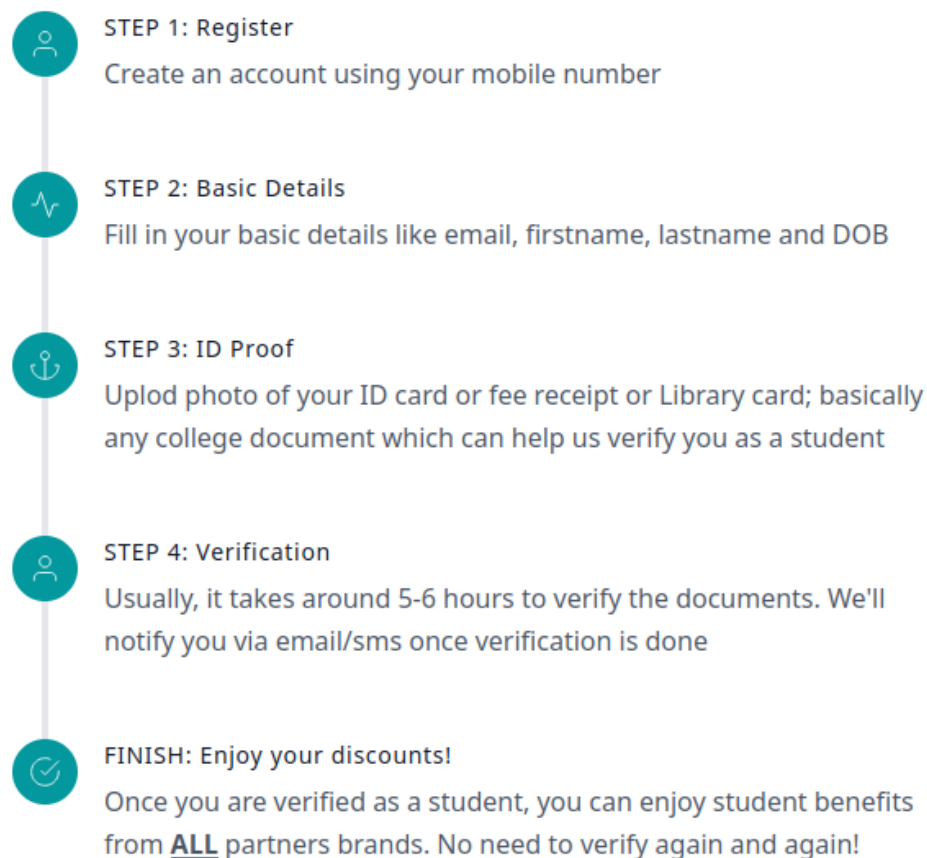


Figure 1. Steps for student verification

While working at Verifyme India my work was to create a frontend for the student facing website and the business facing website for the student verification service. The CEO and the current owner of the company was the lead developer for the backend which was written in Django (Python). The development of API was being done in parallel to the development of the frontend. The methodology of development was rapidly paced and we were releasing the product as soon as service was ready to be released. After the knowledge transfer had taken place we had a discussion with the whole team in which we were supposed to select the framework we would be using for creating the frontend. We had two options to choose from, one was using a pure react application which would be using Client Side Rendering and the other one was to use NextJS to create a combined application with Client Side Rendering and Server Side Rendering. To create such an application webpack as a bundler was chosen as the best choice. This was to improve the security and performance of the application.

APPLICATION ARCHITECTURE

The overall overview of the project consist of four items, these items are listed below-

1. AWS S3 Bucket - An AWS bucket was created to store the images of the student identification cards safely, so that they can be transmitted across the internet and there wouldn't be an issue of unauthorised access.
2. Postgres Database - A postgres database was used to create user details and other required items such as verification status. This same database was used to store all the details of the given user.
3. Django Backend - The Django backend was created to handle all the API requests. This includes all the functionality of the application. [2]
4. NextJS Frontend - The frontend that was built in NextJS which is a framework of ReactJS, was created to create an easy to use user interface that will allow for faster user verification. The design was meant to be created for mobile first use cases.

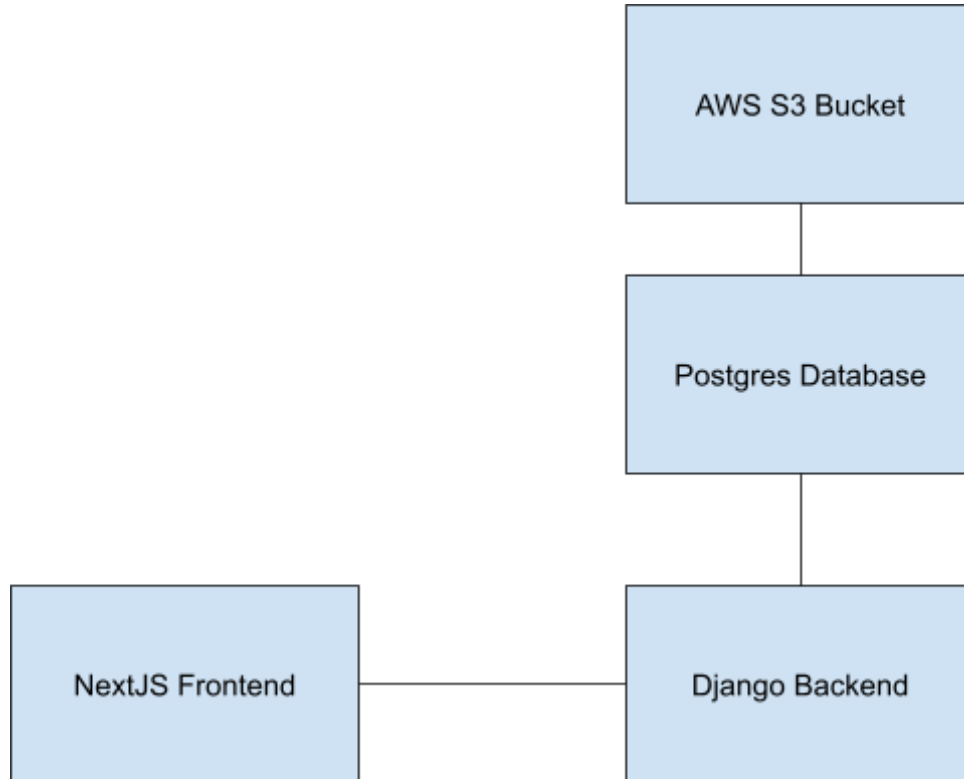


Figure 2. Base Application Architecture

AUTHENTICATION PATTERNS

Almost since the beginning of the Web there has been a need to secure websites to ensure not everyone on the Internet has access. The way that this is done is that a user authenticates with a username and password. The server then authorises them to perform certain actions based on their permissions. This distinction between authentication and authorization is important - one is verifying we are who we say we are and the other is a decision as to whether, if that is true, we are allowed to do the thing we are asking to do. In this project we were tasked to use a JWT based authentication pattern. [1]

JSON Web Tokens (JWT) -

Originally, the token was not intended to have any inherent meaning to the clients using it — it was meant to be a random hexadecimal that would be presented to the authorization server for validation as well as retrieval of any claims needed to process the request. However, the standard didn't say that it couldn't be something like JSON with the required data/claims right in it instead. This led to the development of JSON Web Tokens or JWTs as an alternative. [1, 3]

These JWTs not only embed the user details (claims) required by the resource server right in the token but then cryptographically sign the whole thing with its private key (which becomes a sensitive secret in this model — anybody with this key can sign JWTs which the resource servers will trust). This, together with an embedded expiry time, means that the process flow becomes much simpler as below: [3]

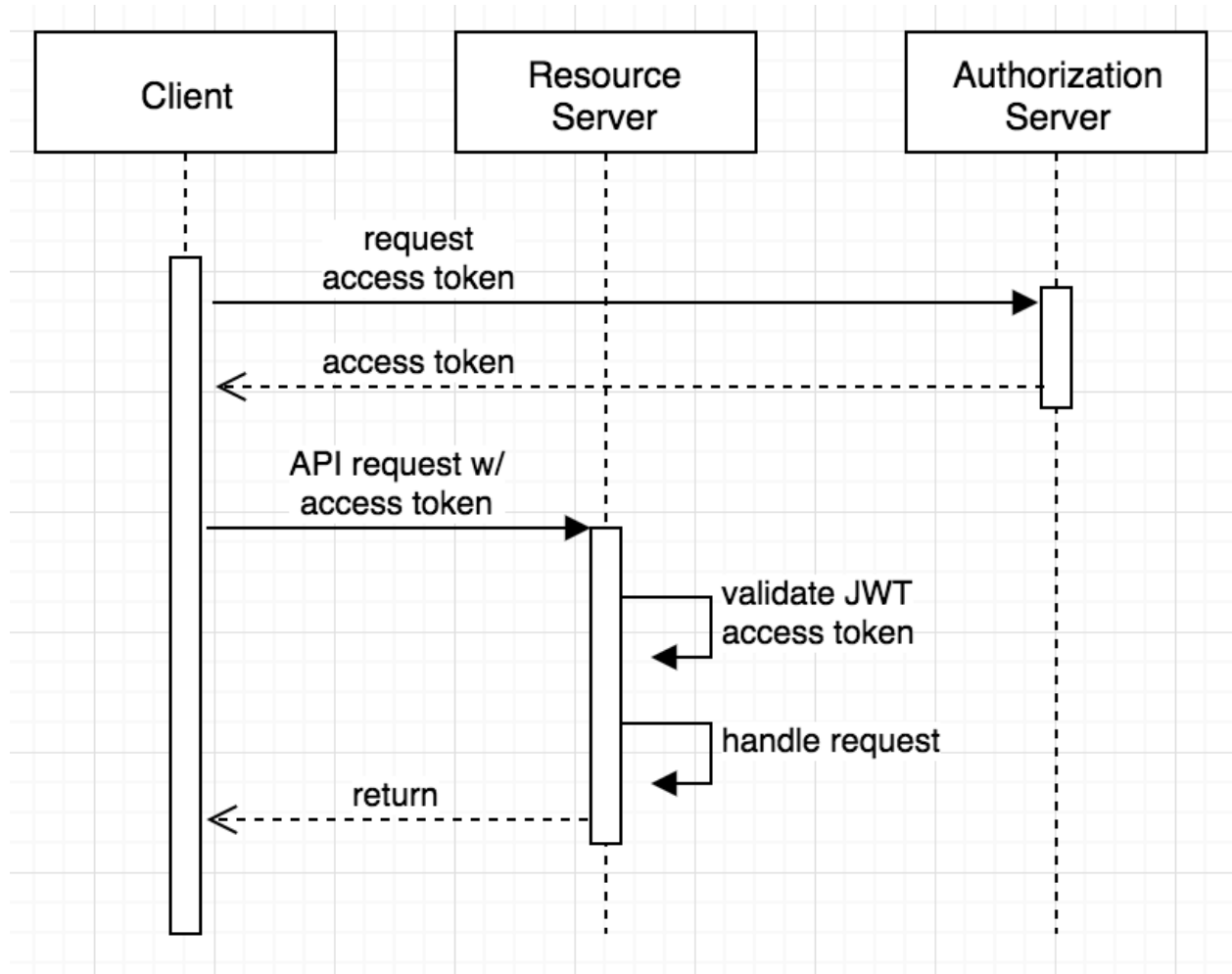


Figure 3. JWT Authentication Pattern

Note that in this scenario the resource server does not need to talk to the authorization server at all — it just verifies the cryptographic signature in the token and makes sure the expiry date is in the future. And, since the majority of the load in the old model for the authorization server was around validating all the requests, this is a huge reduction in the load on the system. Also, the data around the token and whether it is valid is offloaded to each client/browser, and it's localData, rather than a centralised session state database backing the authorization server.

For the purpose of this application, the security of the user isn't exactly the top priority. [3] Hence the JWT tokens were stored in local storage rather than using HTTP only cookies. HTTP only cookies are also another way to store tokens. Dealing with cookies has their fair share of

subtleties, but at a high level a cookie is a piece of data that your web server can set, that will be then stored by the user's web browser and sent back to the server on any future requests that browser makes to the same server as long as the cookie is valid and applicable to the request being made. Two of the most common attack vectors facing websites are Cross Site Scripting (XSS) and Cross Site Request Forgery (XSRF or CSRF). Cross Site Scripting attacks occur when an outside entity is able to execute code within your website or app. HTTP only cookies prevent that from happening. [4] If an attacker can execute code on your domain, your JWT tokens (*in local storage*) are vulnerable. Cross Site Request Forgery attacks are not an issue if you are using JWT with local storage. On the other hand, if your use case requires you to store the JWT in a cookie, you will need to protect against XSRF. [4]

For the purpose of this application a username / passwordless authentication method was used. Here we were using a TOPT based authentication service. The user inputs his/her mobile number and on submit receives an OTP which is valid for the next 5 minutes. On submit of a correct OTP, a JWT token is returned to the user and the user is redirected to the page that he/she is supposed to go to.

FRONTEND DESIGN

Verifyme India student sided application was built using NextJJS which is a framework built over ReactJS. ReactJS is a front-end javascript library which allows us to divide everything we make into a component based system. NextJS is a ReactJS framework built by vercel formerly known as Zeit. Next.js is one way that you can leverage React to support server-side rendering (SSR). Likewise, Create React App is one way that you can leverage React to support client-side rendering (CSR). [5]

There are other frameworks out there when it comes to either choice, but what we are really comparing in this post is how each rendering strategy impacts web application performance. We just happen to be using two of the more popular frameworks out there to make that comparison. Next.js gives you the best developer experience with all the features you need for production: hybrid static & server rendering, TypeScript support, smart bundling, route prefetching, and more. No config needed. [5, 6]

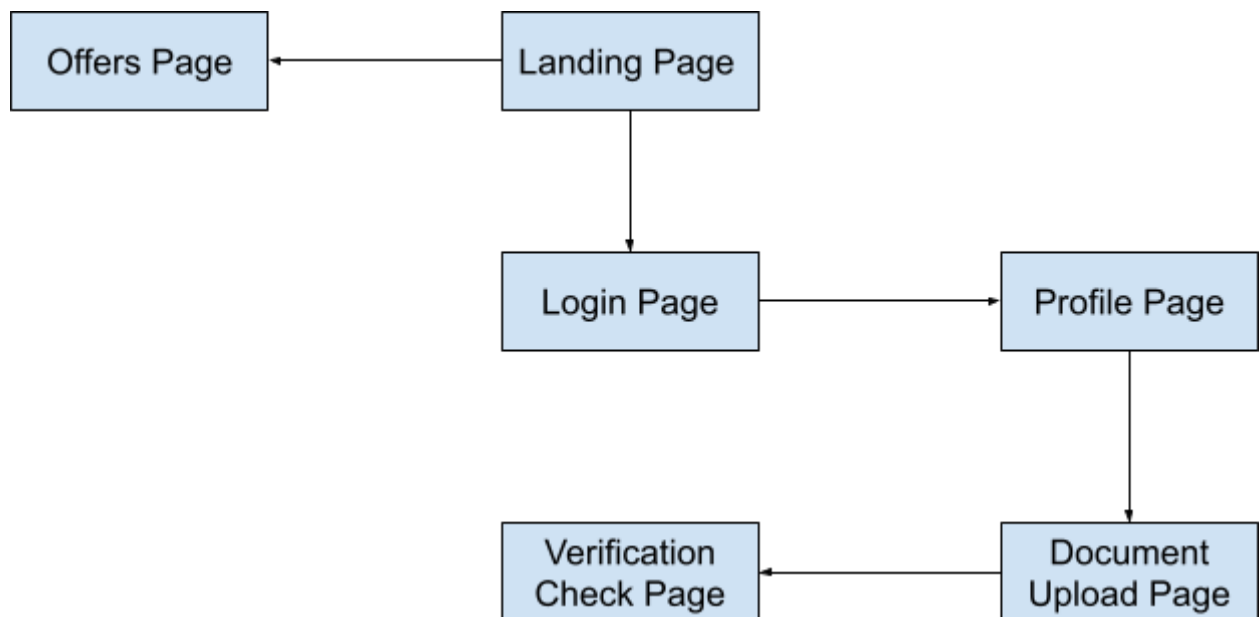


Figure 4. Frontend Design Architecture

Usage of CSR, SSR and SSG

A usual react application built only using create-react-app, defined standard by facebook can only use CSR. There are multiple issues with CSR that needs to be taken care of some of the issues with CSR are given below -

- For one, client-side rendering can increase the likelihood of a poor user experience. JavaScript can add seconds of load time to a page, and if that burden is fully on the client (website visitor), then they could get frustrated and leave your site.
- The second big downside of client-side rendering is its effect on search engine bots. For example, Googlebot has something called a second wave of indexing. In this process, they crawl and index the HTML of a page first, then come back to render the JavaScript when resources become available. This two-phased approach means that sometimes, JavaScript content might be missed, and not included in Google's index [8]

For pages that will see various on load changes, the best item to use is CSR, these types of pages are generally profile pages and don't require any SEO. If there are SSO requirements and the page changes according to the content that needs to be displayed SSR is the best way to go. On the other hand if a static site needs to be served with the addition of SEO. [9]

Method	Timing	Venue	Syntax
SSG	Build-Time	Server	getStaticProps() getStaticPaths()
SSR	Run-Time	Server	getServerSideProps()
CSR	Run-Time	Server	React Components

Table 1. Compare and Contrast between SSG, SSR and CSR

The website that was built was a combination of SSG, SSR and CSR. Pages like “404 not found”, “Terms & Conditions” were statically generated (SSG). Offer pages were generated

using (SSR), to improve performance and improve the space requirement. For the process of verification as no SEO was required , usage of CSR took place.

DEVELOPMENT ENVIRONMENT

All the development took place using Git's version control system. Here we were using Bitbucket for hosting our git repository. For the development model, "Gitflow Workflow model" was used. Gitflow is an alternative Git branching model that involves the use of feature branches and multiple primary branches. [10, 11]

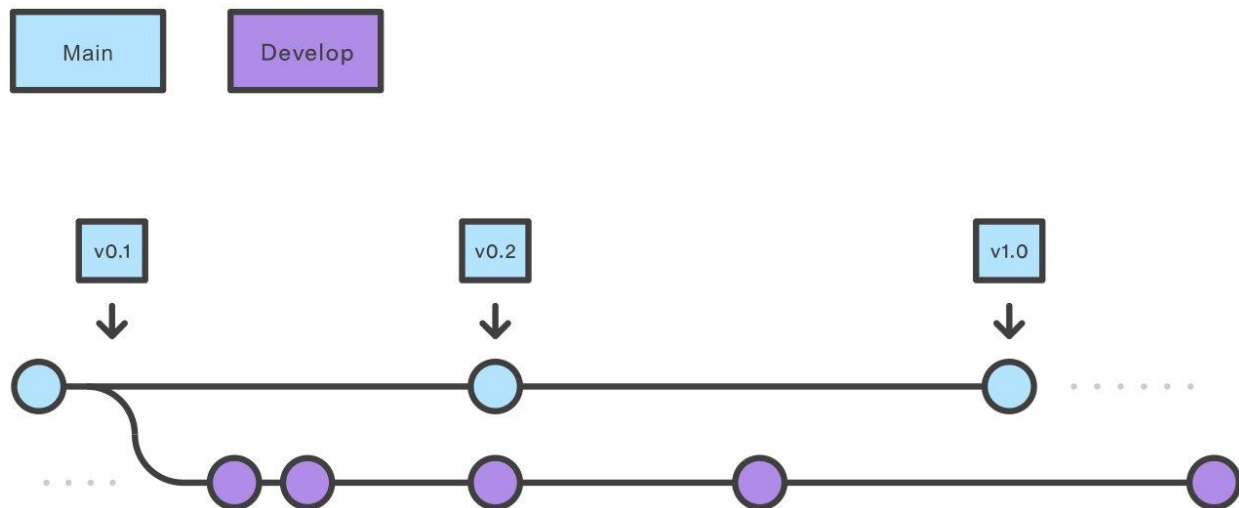


Figure 5. GitFlow Workflow Method

A development server was run on a statically deployed IP for testing purposes while the actual backend was deployed as "api.verify.me.in". The switching between these servers were handled using build time environment variables. Primary Code editor was visual studio code and we were using VSCode live share extension to improve productivity. All the task tracking was done using a trello board. Everyday a standup meeting would take place at 9:30 in the morning. The meeting would take place for 15 minutes and all the members working on the given project would connect online for a meeting. This meeting would consist of 2 tasks, one was task update where we would talk about what was done the last day, the other being what task would be done the day of the meeting. This would help set our goals and fix any issues that might arise. As the development for a website especially the frontend moves really fast and also as we have a smaller team. a daily standup meeting made a lot of sense.

FRONTEND DESIGN

For frontend design, two UI libraries were mainly used. TailwindCSS and MaterialUI React. Tailwind CSS provides utility classes to help you work within the constraints of a system instead of littering your stylesheets with arbitrary values. They make it easy to be consistent with colour choices, spacing, typography, shadows, and everything else that makes up a well-engineered design system. It uses the power of PostCSS. PostCSS is a software development tool that uses JavaScript-based plugins to automate routine CSS operations. MaterialUI React is used in some places to cut down on time.

Various practices were used to make the website more and more mobile friendly. Usage of mobile friendly elements, usage of mobile friendly touch gestures, mobile friendly keyboard support were some of them. (see appendix for lighthouse score)

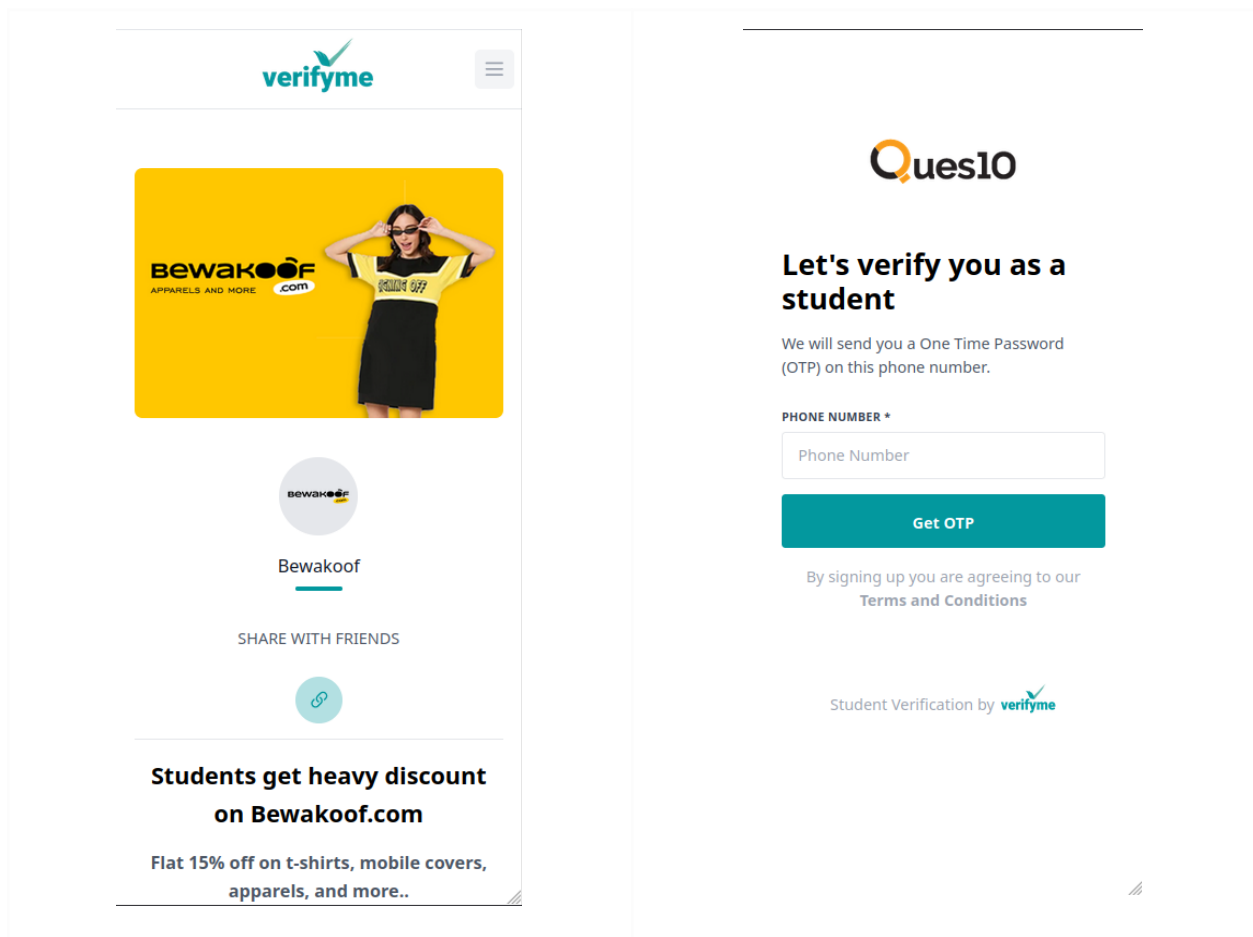


Figure 6. Mobile First Design

TESTING

There were two kinds of testing that were done for the system. Unit testing and Integration Testing. For both of these test cases, Cypress was used. Cypress can be run in two mode ie. GUI and headless. For the optimal functionality it can be added as a CI command and run under CI utilities such as Bitbucket CI or Gitlab CI. [12]

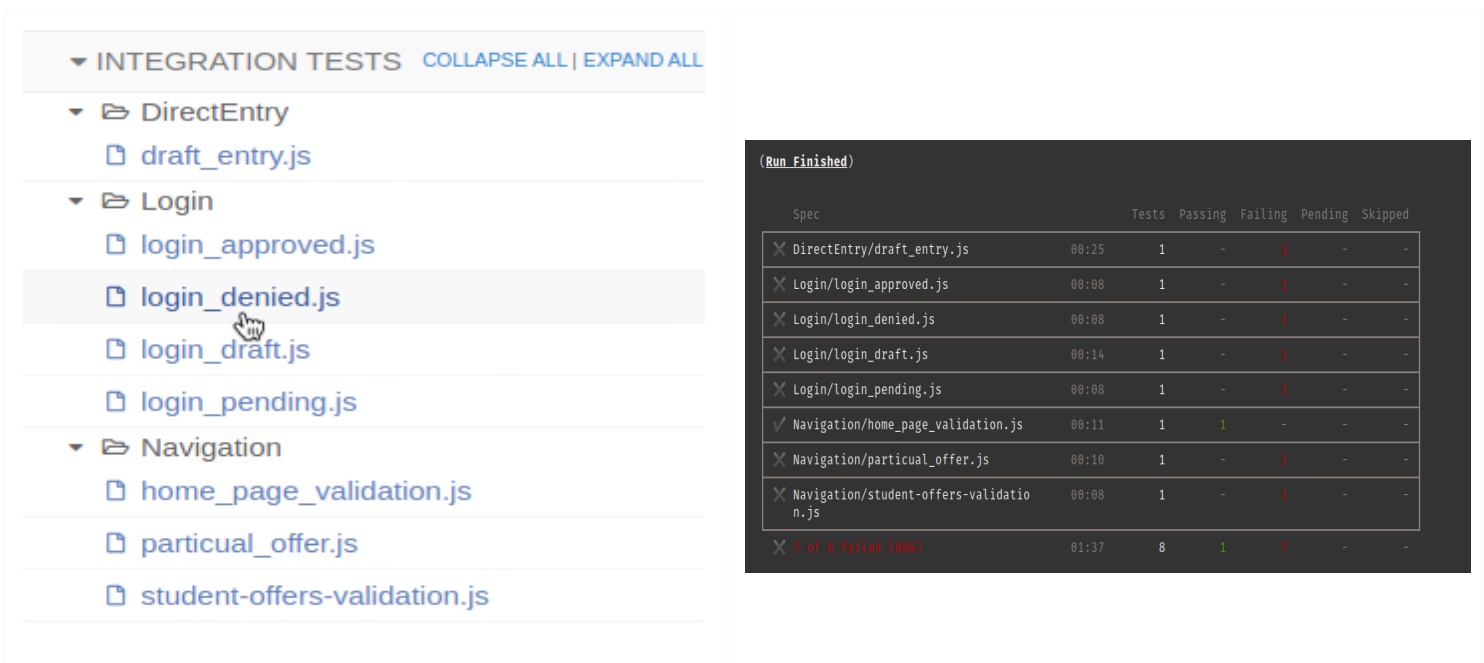


Figure 8. Cypress Testing, GUI based and Headless

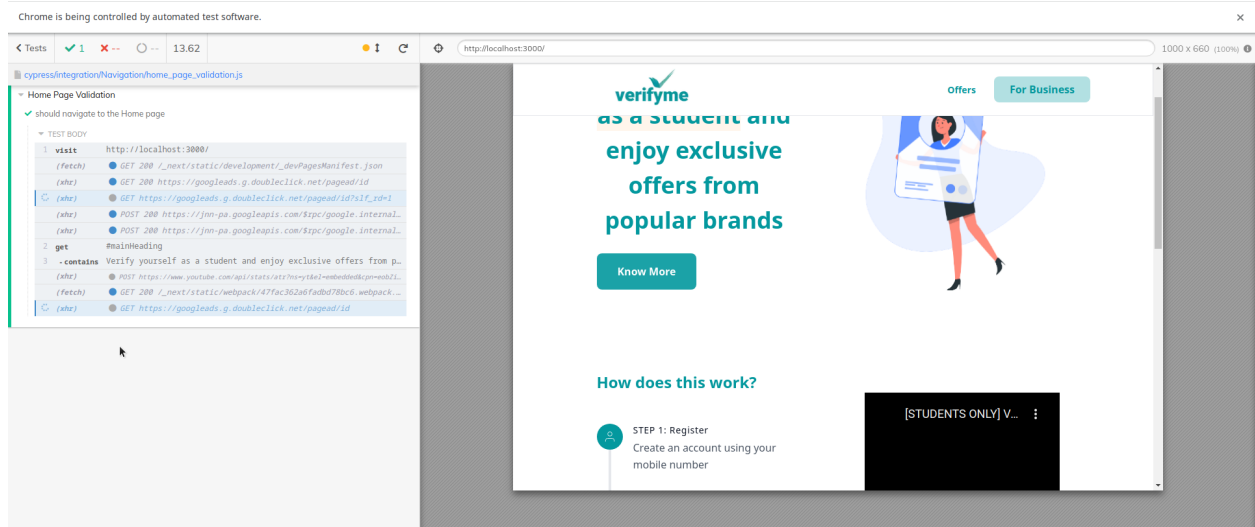


Figure 9. Example passing Cypress Test run

To deploy this build on a server, a docker based script was used. `next build` generates an optimised version of your application for production. This standard output includes:

- HTML files for pages using `getStaticProps` or Automatic Static Optimization
- CSS files for global styles or for individually scoped styles
- JavaScript for pre-rendering dynamic content from the Next.js server
- JavaScript for interactivity on the client-side through React

INTERNSHIP SCHEDULE

As the work done during the internship was fast paced, every week a huge amount of work was done. So to provide a schedule a table would be created to provide information about the work done and the schedule of the internship.

Time Period	Work Done
Week 1	Knowledge transfer and basic project setup
Week 2	Handling login and front page
Week 3	Adding Profile Page and Upload Document Page
Week 4	Creating the find institute dropdown and add institute modal
Week 5	Creating global environment and fixing build issues
Week 6	Creating final verification page and logout page / session expired page
Week 7	ESLint and SEO for promotional pages, helping out with the admin side web pages
Week 8	Unit and Integration testing using Cypress

Table 2. Weekly Internship Schedule

CONCLUSION

I was able to successfully complete the two month internships at Verifyme India. In a sense I got to learn how to work with a team and how to communicate in a way that is actually clear. The person managing me taught various ways in which working can become a lot easier in the industry and was very straightforward with everything he does. I believe this might have been one of the best first internships possible.

REFERENCES

- [1] Bihis, C. (1900). Mastering OAuth 2.0. United Kingdom: Packt Publishing.
- [2] Jain, Shayank. Designing Microservices Using Django. India: BPB Publications, 2020.
- [3] Varghese, Shiju. Web Development with Go: Building Scalable Web Apps and RESTful Services. United States: Apress, 2015.
- [4] Siriwardena, Prabath., Dias, Wajjakkara Kankanamge Anthony Nuwan. Microservices Security in Action. United States: Manning, 2020.
- [5] Konshin, Kirill. Next.js Quick Start Guide: Server-side Rendering Done Right. United Kingdom: Packt Publishing, 2018.
- [6] Dawson, Alexander. Future-Proof Web Design. Germany: Wiley, 2011.
- [7] Mohan, Mehul. Advanced Web Development with React: SSR and PWA with Next.js using React with advanced concepts. India: BPB PUBN, 2020.
- [8] Thakkar, Mohit. Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications. Germany: Apress, 2020.
- [9] Das, Subhankar. Search Engine Optimization and Marketing: A Recipe for Success in Digital Marketing. United States: CRC Press, 2021.
- [10] Loeliger, Jon., McCullough, Matthew. Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development. Germany: O'Reilly Media, Incorporated, 2012.
- [11] Krief, Mikael. Learning DevOps: The Complete Guide to Accelerate Collaboration with Jenkins, Kubernetes, Terraform and Azure DevOps. United Kingdom: Packt Publishing, 2019.
- [12] Mwaura, Waweru. End-to-End Web Testing with Cypress: Explore Techniques for Automated Frontend Web Testing with Cypress and JavaScript. N.p.: Packt Publishing, 2021.

APPENDIX - 1

Code Screenshots

Successful Linting using ESLint

```
(base) sahil@pop-os:~/Documents/verifyme-app$ yarn run lint
yarn run v1.22.17
$ next lint
info - Using webpack 5. Reason: Enabled by default https://nextjs.org/docs/messages/webpack5
✓No ESLint warnings or errors
Done in 3.25s.
```

Docker build File

```
# Install dependencies only when needed
FROM node:16-alpine AS deps
# Check https://github.com/nodejs/docker-node/tree/b4117f9333da4138b03a546ec926ef50a31506c3#nodealpine to understand why libc6-compat might be needed.
RUN apk add --no-cache libc6-compat
WORKDIR /app
COPY package.json ./
RUN npm install --frozen-lockfile

# Rebuild the source code only when needed
FROM node:16-alpine AS builder
WORKDIR /app
COPY . .
COPY --from=deps /app/node_modules ./node_modules
RUN npm run build && npm install --production --ignore-scripts --prefer-offline

# Production image, copy all the files and run next
FROM node:16-alpine AS runner
WORKDIR /app

RUN addgroup -g 1001 -S nodejs
RUN adduser -S nextjs -u 1001

# You only need to copy next.config.js if you are NOT using the default configuration
# COPY --from=builder /app/next.config.js ./
COPY --from=builder /app/public ./public
COPY --from=builder --chown=nextjs:nodejs /app/.next ./next
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/package.json ./package.json

USER nextjs

EXPOSE 3000

# Next.js collects completely anonymous telemetry data about general usage
# Learn more here: https://nextjs.org/telemetry
# Uncomment the following line in case you want to disable telemetry.
# ENV NEXT_TELEMETRY_DISABLED 1

CMD ["npm", "start"]
```

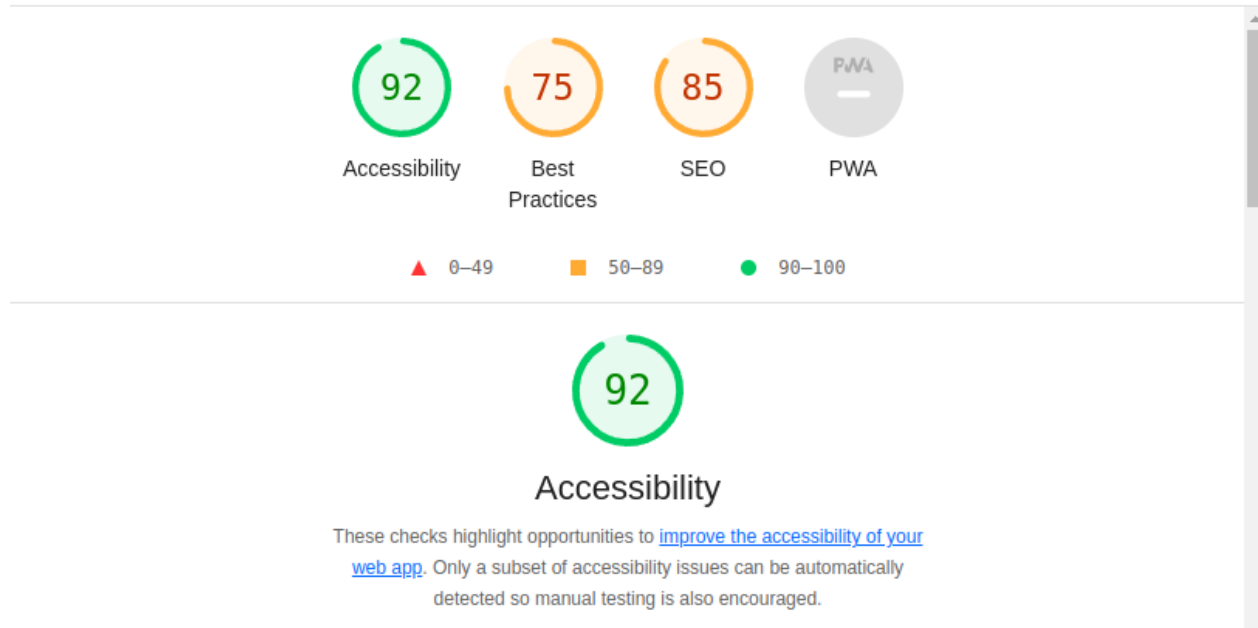
Google Analytics Routes Changed Handling

```
useEffect(() => {
  const handleRouteChange = (url) => {
    ga.pageview(url);
  };
  router.events.on("routeChangeComplete", handleRouteChange);
  return () => {
    router.events.off("routeChangeComplete", handleRouteChange);
  };
}, [router.events]);
```

APPENDIX - 2

Extra Screenshots

Lighthouse Score for the Web Application on Mobile



Bitbucket Repository

verifyme / Verifyme

verifyme-app

Student facing Reactjs app

master Files Filter files

Name	Size	Last commit	Message
/			
Components		2022-01-13	Linting fix
HOC		2021-11-02	added interaction and fixed redirects
Utils		2021-11-02	added interaction and fixed redirects
cypress		2021-12-07	UI fixes and help in offers, added borders around fields
lib		2021-11-12	added google analytics
pages		2022-01-13	Linting fix
public		2022-01-13	Improved offer listing page
styles		2021-11-14	Component Name changes, added youtube video
eslint.config.js	158 B	2021-11-09	added eslint and fixed all related linting issues, added new buttons for offers page and fixed me...
.gitignore	402 B	2021-10-26	Environment settings
Dockerfile	1.29 KB	2021-11-04	Changed port to 3000
README.md	1.41 KB	2021-12-24	note for environment variables
cypress.json	298 B	2021-11-30	fixed test and added the use of env variables
deploy.sh	611 B	2021-10-18	Deployment config

Repository details

Last updated: 2022-01-13

Open pull requests: 0, Branches: 1

Watchers: 2, Forks: 0

Version control system: Git

Access level: Read

Builds






















It looks like you haven't configured a build tool yet. You can use Bitbucket Pipelines to build, test and deploy your code.

Your existing plan already includes build minutes.

Set up a pipeline

Give feedback

Example Commit History

	Yashbeer	ea987e8	Linting fix	2022-01-13
	Yashbeer	a419729	Improved offer listing page	2022-01-13
	Yashbeer Singh	e3e274d	MERGED Merged in upload-loader (pull request #18) added loader and added an im...	2022-01-07
	sahilarora3117	79b575d	added loader and added an image size check	2022-01-06
	Yashbeer	c7909a9	bewakoof offer edit	2021-12-30
	Yashbeer	dab724a	Changed class to className for linter	2021-12-28
	Sahil Arora	e4ca193	MERGED Merged in Cupon-fix (pull request #17) fixed cupon grid Approved-by: Yas...	2021-12-28
	Sahil Arora	cd0d8a8	fixed cupon grid	2021-12-26
	Yashbeer	34bb80c	Bewakoof offer listed	2021-12-25
	Yashbeer	5ee0d2a	note for environment variables	2021-12-24
	Yashbeer	f93f9b2	Introduced port variable	2021-12-24
	Yashbeer	2923974	changed application port to 3000	2021-12-24
	Sahil Arora	9865c41	MERGED Merged in minorUIFixes (pull request #16) fixes for margins and terms an...	2021-12-10
	sahilarora3117	cd2a88d	fixes for margins and terms and conditions	2021-12-08
	Sahil Arora	16e12db	MERGED Merged in lintfix (pull request #15) Fixed Linting error Approved-by: Yash...	2021-12-07
	sahilarora3117	298845a	fixed linting error	2021-12-07
	Sahil Arora	b197674	MERGED Merged in UIFixes (pull request #14) UI Fixes Approved-by: Yashbeer Singh	2021-12-07
	sahilarora3117	5c5dafa	fixed margins and image	2021-12-07
	sahilarora3117	59ac612	UI fixes and help in offers, added borders around fields	2021-12-07
	Sahil Arora	f302d41	MERGED Merged in cypress-testing (pull request #13) Cypress Testing Approved-by...	2021-12-02
	sahilarora3117	0fb7925	added student-offers page testing validation	2021-12-02