*A project report on*

**ONE AGENT PLATFORM**

*Submitted in partial fulfillment of the requirements for the degree of*

**Bachelor of Technology**

**in**

**Computer Science and Engineering with Specialization in Information Security**

*By*

**Kush Garg**

**(20BCI0007)**

Under the guidance of

**Mr. Aadesh Kumar**

**Manager**

**Fareportal India Pvt. Ltd**



**SCOPE**

**June 2024**

# DECLARATION

I hereby declare that the thesis entitled “ONE AGENT PLATFORM " submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering to VIT is a record of Bonafede work carried out by me under the supervision of Mr. Aadesh Kumar, Fareportal.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore Date :

**Signature of the Candidate**

# CERTIFICATE

This is to certify that the thesis entitled **ONE AGENT PLATFORM** submitted by **Kush Garg, 20BCI0007, SCOPE**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering with Specialization in Information Security* is a record of Bonafede work carried out by him under my supervision during the period, 2nd Jan 2024 to 2nd July 2024, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : **Signature of the Guide**

**Internal Examiner External Examiner**

**Head of Department   
Department of Analytics**

# ACKNOWLEDGEMENTS

I would like to express my special thanks to Dr G. Viswanathan, Chancellor, VIT University for creating an environment conducive for learning and growth academically as well as in extracurricular activities that enable every student to reach greater heights.

I would like to thank HOD Department of CSE With Information Security, Vellore Institute of Technology, for always encouraging and supporting, without which this project and the last four years would not have been possible.

I would also like to thank all the faculty and laboratory staff of the Department of Computer Science and Engineering.

I would like to express my gratitude to Rajat Bhatia SVP, People & Culture Fareportal India Pvt. Ltd for giving me an internship opportunity at Fareportal India Pvt. Ltd. Additionally, I would like to thank my managers Aadesh Kumar and Vikram Jha and team members Paul Tirkey, Lalit Saini, Arghya Sadhu and Pratiksha Vyawhare for guiding me throughout the internship.

Lastly, I would like to thank the entire OAP team at Fareportal India Pvt. Ltd. for welcoming me into the organization and helping me tirelessly during every point in the internship.

It is indeed a pleasure to thank my family and friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project

Place: Vellore

Date:

**Kush Garg**

# EXECUTIVE SUMMARY

I did my final year internship at Fareportal India Pvt. Ltd. from 3rd October 2022 to 3rd July 2023. I was a part of the OAP team responsible for developing solutions for the consolidated Agent platform. I completed my internship in the capacity of full-stack web developer where I used dot net core for backend along with HTML, CSS, Bootstrap and JavaScript, React, Redux, Hooks for frontend. I collaborated with the team remotely and was assigned tasks on a regular basis. I was also provided an opportunity to work on the projects everyone else were working on and some of the activities and features created by me were incorporated by the team in the production environment. I was given an opportunity to observe how software developers work on a daily-basis by working alongside them and see what issues and errors are faced by them while developing applications in the real-world. The focus of my work was to perform tasks and showcase my skills of C# and React as per the requirements of the projects.

 I used concepts of micro-frontends to create a standalone npm module that can handle all the shared functionalities among the various projects utilizing the same methods. I learned about creating production level web applications and using API’s, which were entirely new concepts to me. This was an important skill for me as most apps today are supposed to work on multiple platforms and thus should be designed in such a manner that they adapt to different devices without compromising on design quality. Besides the technical skills, I also learnt about the importance of collaborative work in a team while perfecting the individual task given to you. I saw people work on a single part of the program for hours trying to get a solution by using different methods. One major takeaway was that apart from syntax and coding skills, one should have problem-solving skills which will help them think of innovative ideas to create new apps and solve issues in an optimal and effective manner. In this report, I have mentioned the project specific details.

# Table of Contents

[DECLARATION ii](#_Toc134974432)

[CERTIFICATE iii](#_Toc134974433)

[ACKNOWLEDGEMENTS iv](#_Toc134974434)

[EXECUTIVE SUMMARY v](#_Toc134974435)

[Table of Contents vi](#_Toc134974436)

[List of Figures viii](#_Toc134974437)

[List of Tables viii](#_Toc134974438)

[List of Abbreviations viii](#_Toc134974439)

[ABOUT THE COMPANY ix](#_Toc134974440)

[CheapOair ix](#_Toc134974441)

[OneTravel x](#_Toc134974442)

[Fareportal Media Group x](#_Toc134974443)

[Travelong x](#_Toc134974444)

[Royal Scenic x](#_Toc134974445)

[Dukes Court Travel Ltd. x](#_Toc134974446)

[1. INTRODUCTION 1](#_Toc134974447)

[OBJECTIVES 1](#_Toc134974448)

[THEORETICAL BACKGROUND 1](#_Toc134974449)

[MOTIVATION 1](#_Toc134974450)

[2. LITERATURE SURVEY 2](#_Toc134974451)

[SURVEY OF THE EXISTING MODELS 2](#_Toc134974452)

[2.1.1 MICRO FRONTEND APPROACH 2](#_Toc134974453)

[2.1.2 MONOLITHIC DESIGN APPROACH 3](#_Toc134974454)

[SUMMARY OF THE SURVEY 3](#_Toc134974455)

[3. OVERVIEW OF PROPOSED SYSTEM 4](#_Toc134974456)

[3.1 ARCHITECTURE FOR THE PROPOSED SYSTEM 4](#_Toc134974457)

[3.1.1 MICROFRONTENDS 4](#_Toc134974458)

[3.1.2 USAGE OF MICRO FRONTENDS 5](#_Toc134974459)

[3.2 PROPOSED SYSTEM MODEL 5](#_Toc134974460)

[3.2.1 ARCHITECTURE OF A MICRO FRONTEND APPLICATION WITH REDUX IN MIND 6](#_Toc134974461)

[3.2.2 HOST 6](#_Toc134974462)

[3.2.3 REMOTE 7](#_Toc134974463)

[3.2.4 REMOTE STORE 7](#_Toc134974464)

[3.2.5 MIDDLEWARE 7](#_Toc134974465)

[4. PROPOSED SYSTEM ANALYSIS AND DESIGN 8](#_Toc134974466)

[REQUIREMENT ANALYSIS 8](#_Toc134974467)

[4.1.1 FUNCTIONAL REQUIREMENTS 8](#_Toc134974468)

[4.1.2 NON-FUNCTIONAL REQUIREMENTS 8](#_Toc134974469)

[5. TECHNICAL SPECIFICATIONS 9](#_Toc134974470)

[SOFTWARE REQUIREMENTS 9](#_Toc134974471)

[5.1.1 HTML 9](#_Toc134974472)

[5.1.2 CSS 10](#_Toc134974473)

[5.1.3 JAVASCRIPT 11](#_Toc134974474)

[5.1.4 REACT 12](#_Toc134974475)

[5.1.5 WEBPACK 13](#_Toc134974476)

[5.1.6 NODE.JS 15](#_Toc134974477)

[5.1.7 C# 16](#_Toc134974478)

[5.1.8 .NET 17](#_Toc134974479)

[5.1.9 MYSQL 18](#_Toc134974480)

[5.1.10 VISUAL STUDIO CODE 19](#_Toc134974481)

[5.1.11 MICROSOFT AZURE 20](#_Toc134974482)

[5.1.12 UNIT TESTING 21](#_Toc134974483)

[5.1.13 GIT AND GITHUB 21](#_Toc134974484)

[6. CONCLUSIONS 22](#_Toc134974485)

[5. REFERENCES 22](#_Toc134974486)

# List of Figures

Figure 1 Fareportal Logo x

Figure 2 Monolithic vs Microservice Architecture 2

Figure 3: Micro-frontend architecture 4

Figure 4: Generalized idea for how micro frontend is used in modern web Apps 5

Figure 5: A simple sketch depicting the implementation of micro frontends and redux. 6

Figure 6: Logo of HTML 9

Figure 7: Logo of CSS 10

Figure 8: Logo of JavaScript 11

Figure 9: Logo of React 12

Figure 10: Logo of webpack 13

Figure 11: Logo of Node.JS 15

Figure 12: Logo of C# 16

Figure 13: Logo of .NET 17

Figure 14: Logo of MySQL 18

Figure 15: Logo of Visual Studio Code 19

Figure 16: Logo of Microsoft Azure 20

# List of Abbreviations

* **UI/UX –** User interface and user experience
* **CSS –** Cascading Style Sheets
* **JS –** JavaScript
* **JSON -** JavaScript Object Notation
* **GDS –** Global Distribution System
* **OAP –** One Agent Platform

# ABOUT THE COMPANY

Fareportal is a technology company that powers leading hybrid travel agencies like CheapOair, OneTravel, and Travelong, a veteran corporate travel agency founded in Morristown, New Jersey in 1976. our strong knowledge and experience in the air travel booking vertical, including corporate travel management and automation of wholesale distribution of complex international airfares, today we operate in North America, Europe, and Asia. We are also proud to partner with over 500 airlines, over 1 million hotels, and hundreds of car agencies worldwide.

Fareportal’s unique hybrid business model bridges the gap between an online travel agency and a traditional travel agency by providing a convenient online booking capability as well as a 24/7 personalized trip booking experience arranged by hundreds of trained and certified travel agents in multiple countries and in multiple languages. By leveraging company owned and operated contact centers located worldwide, we can deliver one of the highest levels of customer service and support in the air travel business.

Travelers around the world can find and book their perfect trip on our websites, mobile and tablet apps, and by calling one of our hundreds of trained and certified travel agents.

Fareportal’s partner airlines benefit from the broad customer reach and strong customer value proposition as compared to the pure OTA model. Focused more on selling international and higher yield complex airfares, partner airlines obtain higher revenue on a per seat basis.

Fareportal’s constant innovation allows partner airlines to benefit from selling ancillaries on the booking path thereby further optimizing their revenues and making them more predictable. This also helps the Airlines in streamlining airport operations, having less people wait in line to pay for preferred seats and bags. Read more about how Fareportal has been first to sell ancillaries for American Airlines, Air Canada, and Spirit Airlines.

Figure 1 Fareportal Logo

## CheapOair

A picture containing font, logo, graphics, brand

Description automatically generated

CheapOair is the smartest way for savvy travelers to book their next trip. CheapOair allows customers to compare and book flights on over 450 airlines online, on award-winning mobile apps, or by phone with live travel agents.

## OneTravel



OneTravel simplifies the travel shopping experience by combining value-driven deals on flights, hotels, cars and vacation packages with 24/7 customer care only one mouse click or phone call away.

## Fareportal Media Group

A blue and green logo

Description automatically generated with medium confidence

Fareportal Media Group is the independent media division that manages the advertising operations, sales and business development for Fareportal’s travel websites. Through custom, integrated solutions, Fareportal ensures that its advertising partners get the best return on their investment.

## Travelong



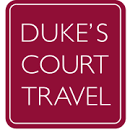
Travelong is a full service travel agency serving more than 100 corporate clients and 30 home-based agents across the United States. Established in 1933, Travelong’s innovative and highly experienced travel consultants have provided client satisfaction for over eight decades.

## Royal Scenic



Royal Scenic is a national wholesale supplier, air consolidator and preferred Fareportal partner with offices in Vancouver, Toronto and Montreal. Royal Scenic offers quality products and customer service exclusively to the retail travel community for both corporate and leisure markets.

## Dukes Court Travel Ltd.



Duke’s Court Travel is based in London and offers a wide range of travel services. Established in 1989, Duke’s Court Travel is an appointed agency for over 50 major international airlines. Licensed by the UK Civil Aviation Authority under ATOL, it offers full financial protection to any who book directly through the agency.

# INTRODUCTION

## OBJECTIVES

* To collaboratively work on different micro frontends that are individually capable of handling different functionalities
* To eradicate the monolithic architecture which has been causing development issues and bottlenecks and improvise performance of the project.
* Reducing Call handling time of agents
* Improve overall experience of customers

## THEORETICAL BACKGROUND

Monolithic architecture and microservice architecture are two commonly used approaches for designing software applications. Monolithic architecture is an older approach where all the application components are tightly integrated and deployed as a single unit. In contrast, microservice architecture breaks down the application into smaller, independent services that communicate with each other through APIs.

Monolithic architecture has been the traditional approach for designing software applications, where developers would write code for the entire application in a single codebase. This approach offers simplicity and ease of deployment but is limited in terms of scalability and flexibility. In contrast, microservice architecture offers greater flexibility and scalability, but is more complex to design and implement.

Microfrontends, on the other hand, are a newer approach for building web applications, where the user interface is broken down into multiple independent modules that are developed and deployed independently. This approach is like microservice architecture, but for the front-end portion of the application.

Microfrontends provide several advantages, such as improved scalability, faster development cycles, and the ability to easily update specific parts of the application without affecting the entire application. However, it also comes with its own set of challenges, such as increased complexity and a need for careful coordination between different teams.

## MOTIVATION

This architecture pattern enables developers to divide the frontend of the application into multiple, smaller, and independent parts that can be developed, tested, deployed, and maintained separately. This approach leads to increased agility, faster time to market, and improved scalability. By using micro frontend architecture, developers can easily create and deploy new features without disrupting the entire application. Moreover, this approach allows for a better user experience, as it enables faster loading times and more responsive interactions. With the adoption of micro frontend architecture, the Organisation can also achieve better fault tolerance and resiliency. In case of a failure in one part of the application, only the affected micro frontend will be impacted, while the other parts of the application continue to function seamlessly.

# LITERATURE SURVEY

## SURVEY OF THE EXISTING MODELS

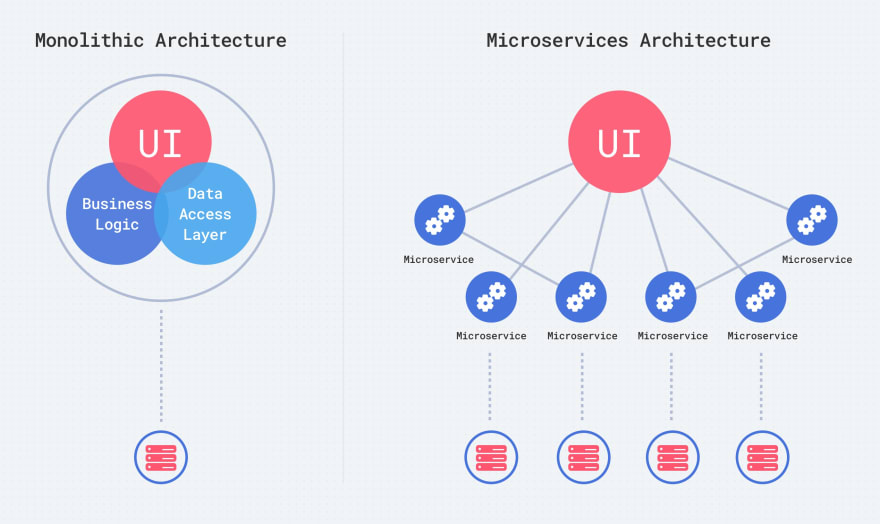


Figure 2 Monolithic vs Microservice Architecture

### MICRO FRONTEND APPROACH

The micro frontend approach is used to extend the idea of microservices to the frontend. Basically, when the frontend is decomposed into independent features managed and developed by different teams independent of each other.

In the case of Fareportal the OAP core UI repository needs to be decomposed so that different departments can work independently ensuring a workflow with CI/CD.

### MONOLITHIC DESIGN APPROACH

A monolithic architecture is a traditional model of a software program, which is built as a unified unit that is self-contained and independent from other applications.

In the case of Fareportal the old OAP repository has been developed in monolithic architecture.

## SUMMARY OF THE SURVEY

### Drawbacks of monolithic architecture

Monolithic architecture has been a widely used approach for designing software applications for many years. However, it has several drawbacks that have become more apparent as technology has evolved. Some of the drawbacks of monolithic architecture are:

**Scalability**: Monolithic architecture is limited in terms of scalability. As the application grows, it becomes increasingly difficult to scale the entire application. This can lead to performance issues and decreased user experience.

**Maintenance**: In monolithic architecture, all components of the application are tightly coupled. This makes it difficult to update or change any part of the application without affecting the entire application. As a result, maintenance becomes more complex and time-consuming.

**Flexibility**: Monolithic architecture is not flexible. All components of the application are designed to work together and cannot be easily separated. This limits the ability to integrate new technologies or change the application architecture.

**Testing**: Monolithic applications are difficult to test. As all components are tightly coupled, it is difficult to test individual components without testing the entire application. This makes it challenging to ensure that all components of the application are functioning correctly.

**Deployment**: Deploying monolithic applications can be complex. As all components are integrated into a single unit, deployment requires coordination between all components. This can lead to delays and downtime during deployment.

### Advantages of Micro-frontend architecture

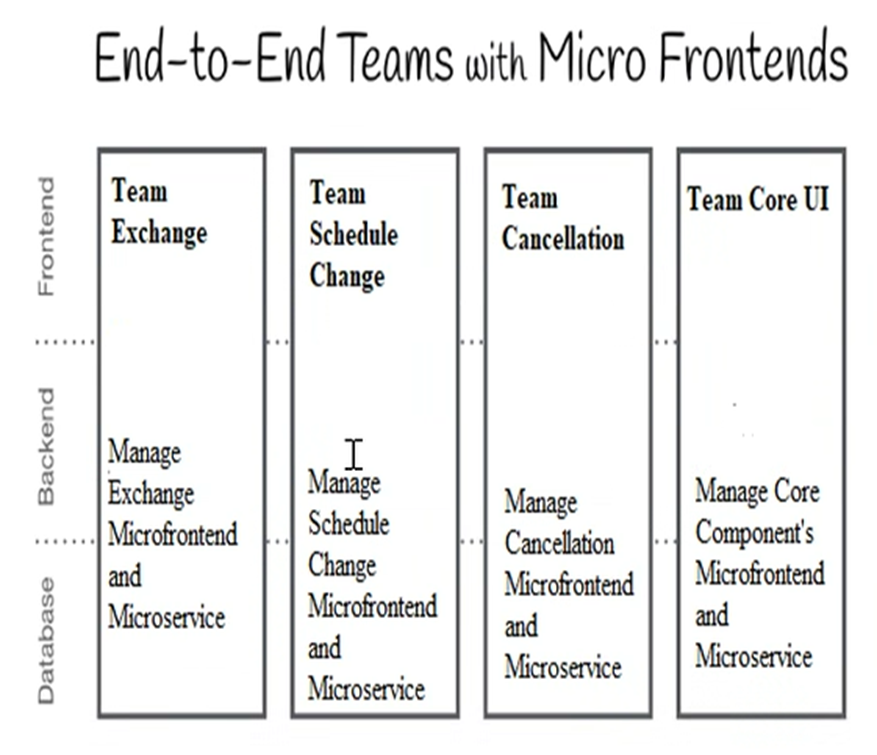
* Independent Development and Deployment: Different teams can work on separate micro frontends, allowing for independent development, testing, and deployment cycles. This leads to faster iteration and reduces coordination overhead.
* Enhanced User Experience: MicroFrontend Architecture enables faster loading times and improved performance as each micro frontend can be optimized individually. It also enables a more personalized and tailored user experience by providing specific functionalities for different user segments.
* Scalability and Resource Efficiency: MicroFrontends can be independently scaled, allowing for better resource utilization and scalability. This means specific components can be scaled up or down based on demand without impacting the entire application.
* Technology Flexibility and Innovation: MicroFrontend Architecture allows for the use of different frontend frameworks and technologies, empowering teams to choose the best tool for the job. It encourages innovation and exploration of new technologies.
* Fault Isolation and Resilience: Like Microservice Architecture, failure in one micro frontend does not affect the entire application. This promotes fault isolation, better error handling, and improved resilience.

### Advantages of Microservices architecture

* Scalability: Microservices allow for independent scaling of different components, enabling efficient resource allocation and better handling of varying workloads.
* Agility and Faster Time to Market: Microservices enable development teams to work independently on smaller components, leading to faster development cycles, quicker deployments, and the ability to respond rapidly to changing business needs.
* Flexibility and Technology Diversity: Each microservice can be built using the most suitable technology stack, allowing teams to leverage the best tools for different functionalities. This promotes technological diversity and reduces the risk of being locked into a single technology.
* Fault Isolation and Resilience: Failure in one microservice does not disrupt the entire system, as other services can continue to function independently. This promotes fault isolation and enhances the overall resilience of the application.
* Improved Maintainability: With focused codebases, developers can make changes to a specific microservice without impacting the entire system.

# OVERVIEW OF PROPOSED SYSTEM

## TEAM STRUCTURE AND RESOURCES



### OAP Cancellation

This application provides user to cancel the existing bookings. This application allow user to cancel the partial and full Cancellation of booking. In this application, User will create refund details and cancellation fee details on intent page when user processes for cancellation, we will show all the required step to the user which will require for cancel the booking. Till date 90% of cancellation process is Automated try to cover rest 10% in future. Previously our agents work manually as per customer requirements and it has taken more than 5 mins to perform cancellation process. Now user can cancel the ticket within 2 minutes.

### OAP Exchange

This application facilitates users in canceling their existing bookings efficiently. It allows users to changes in bookings either partially or in full. When a user initiates the cancellation process, they are directed to an intent page where they can provide refund details and cancellation fee information. The application guides users through the necessary steps to complete the cancellation successfully. Currently, 90% of the cancellation process is automated, significantly reducing the need for manual intervention. In the future, efforts will be made to address the remaining 10% of the cancellation process, aiming for complete automation. Previously, our agents had to handle cancellations manually, which typically took more than 5 minutes to complete. However, with this application, users can cancel their bookings within just 2 minutes. The implementation of this system has greatly streamlined and expedited the cancellation process, enhancing the user experience.

### OAP Schedule Change

This application allows users to make changes to their existing bookings. Users can modify their schedule and make adjustments to their reservations. The application provides a seamless process for users to request changes, including partial or full cancellations. Users can input their refund details and cancellation fee information on the intent page when initiating a schedule change. The application guides users through the necessary steps required to modify their bookings.

Currently, 90% of the schedule change process is automated, ensuring efficiency and reducing manual efforts. The application aims to further enhance its capabilities to cover the remaining 10% of scenarios in the future. Previously, our agents used to handle schedule changes manually, which often took more than 5 minutes to complete. However, with the introduction of this application, users can now make desired changes within just 2 minutes, streamlining the entire schedule change process.

### OAP Core

The booking summary view of the application provides users with an overview of their bookings, allowing them to see the details of their reservations and any associated cancellation information. This view provides a clear and concise summary of the booking status, helping users stay informed about their cancellations and refunds.

## ARCHITECTURE FOR THE PROPOSED SYSTEM

### MICROFRONTENDS

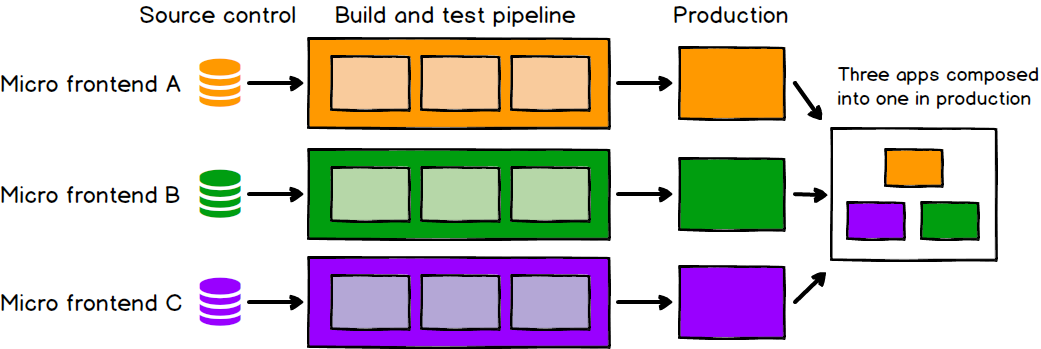


Figure 3: Micro-frontend architecture

Micro-frontend architecture is a design approach in which a front-end app is decomposed into individual, semi-independent “microapps” working loosely together. The micro-frontend concept is vaguely inspired by, and named after, microservices.

**The core ideas are as follows:**

* **Be Technology Agnostic -** Each team should be able to choose and upgrade their stack without having to coordinate with other teams.
* **Isolate Team Code -** Build independent apps that are self-contained.
* **Establish Team Prefixes-** Agree on naming conventions where isolation is not possible yet. Namespace CSS, Events, Local Storage and Cookies to avoid collisions and clarify ownership.
* **Favor Native Browser Features Over Custom APIs**
* **Build a Resilient Site -** We can use Universal Rendering and Progressive Enhancement to improve perceived performance.

**The Drawbacks are as follows:**

* **Can cause redundant dependencies. -** Can be fixed by identifying and consolidating all shared libraries to one micro frontend and reusing it.
* **Conflicting and overlapping styles of UI/UX -** resolvable by following strict UI/UX guidelines.
* **Decreased communication among different components -** resolvable using browser tools or Redux as shown in this document.

### USAGE OF MICRO FRONTENDS

For the usage of micro frontend, the most popular plugin is **Module Federation provided by webpack** in which the 2 most utilized features are.

* Remote – this specifies the location of remote repository from where the micro frontend is to be consumed.

remotes: {remote: "react\_remote@http://localhost:8081/remoteEntry.js"},

In order to use the functions of the remote directory we must ensure we have access to the remote entry file as that file is a bundle consisting of all exposed code.

* Exposes – this specifies all the components that need to be exposed to the network so that other repositories can consume it.

name: "react\_remote",

exposes: {

        "./AddTodo": "./src/AddTodo.jsx",

      },

The above snippet exposes the AddTodo component from the src folder to the sub path of AddTodo i.e. the import will look as follows

import AddTodo from "remote/AddTodo"

<AddTodo />

## PROPOSED SYSTEM MODEL

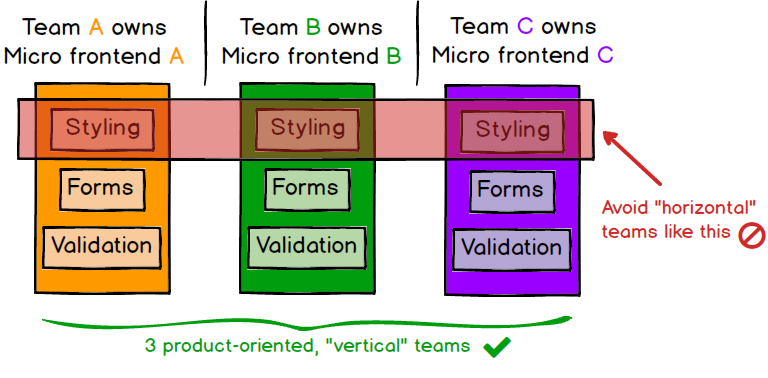


Figure 4: Generalized idea for how micro frontend is used in modern web Apps

### ARCHITECTURE OF A MICRO FRONTEND APPLICATION WITH REDUX IN MIND

This is a brief overview of micro frontends using redux is explained in this section.

The major parts that are needed to be considered in the architecture are:

1. Host
2. Remote
3. Remote Store
4. Middleware

Diagram

Description automatically generated

Figure 5: A simple sketch depicting the implementation of micro frontends and redux.

### HOST

This App is a web App that consumes all the micro frontends and compiles it in the browser. All runtime actions take place in this App.

### REMOTE

This is the micro frontend that is to be managed by separate teams/departments. Essentially a divisible component of the host app based on development roles.

The Remote Team A can work independently of remote Team B and whatever updates are required by team A are never going to affect the development process of team B. The remote teams are only concerned if there are any changes made to the remote store team, but it usually can be avoided if backward compatibility is kept in mind.

### REMOTE STORE

This is like the normal remote micro frontend. The only reason to separate it into a different section is because it solely deals with the redux capabilities of the entire project. This means that all the technical implementation of actions and reducers will be implemented here while the required functions will be encapsulated for easy use by other micro frontends. In the sketch above useStore is a custom method/hook that provides the actions, and the Provider component gives a Provider configured with the store.

Thus, other micro frontends need not separately implement anything related to redux.

### MIDDLEWARE

Middleware is software and cloud services that provide common services and capabilities to applications and help developers and operators build and deploy applications more efficiently. Middleware acts like the connective tissue between applications, data, and users.

Redux-Saga is a library primarily aimed to make application side effects like asynchronous data fetching and accessing impure browser cache. It is very easy to manage and efficient to execute. With Redux-Saga, it is easy to test and handle failure effortlessly.

**Why do we use saga middleware?**

In the context of Redux, a saga is implemented as a middleware because we can't use a reducer, which must be a pure function, to coordinate and trigger asynchronous actions (side effects). A saga manages processes that need to be executed in a transactional way, maintaining the state of the execution, and compensating for failed processes. In the context of Redux, a saga is implemented as a middleware because we cannot use a reducer, which must be a pure function, to coordinate and trigger asynchronous actions (side effects).

# PROPOSED SYSTEM ANALYSIS AND DESIGN

## REQUIREMENT ANALYSIS

### FUNCTIONAL REQUIREMENTS

#### PRODUCT PERSPECTIVE

The product is not entirely new but it is a major improvement over the previous version as the entire codebase is being broken down into multiple modules enabling many of the advantages of modern development that come with microservices.

#### PRODUCT FEATURES

The agent will be able to perform all the functionalities that were intended on OAP with the new microservice version too.

No additional features are added but the performance has been increased manyfold

Each micro-frontend acts as a dependency on each other so that the whole system can work fluidly.

#### USER REQUIREMENTS

Users must be registered in the departments of Fareportal and must have the required permissions and roles to perform activities in the portal.

They must also have a compliant device, network access and other configured devices provided by Fareportal

### NON-FUNCTIONAL REQUIREMENTS

#### EFFIECIENCY

The efficiency can be determined by its responsive time, loading speed and time to complete the allotted task. The loading speed will be quick since the app is lightweight and should be very responsive. Preference has been made to ensure the responsiveness of the app.

#### RELIABILITY

The application should perform the intended action with efficiency with no or very little error. In case of failure, the system which is failing must be isolated and rectified in fast and reliable manner

#### USABILITY

The sign of a good application is that it should be very easy to understand and simple to use. The users should not need any guide or help from the experts.

# TECHNICAL SPECIFICATIONS

## SOFTWARE REQUIREMENTS

### HTML



Figure 6: Logo of HTML

The Hypertext Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It is assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms are embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML is embedded and gets dynamically written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

### CSS



Figure 7: Logo of CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors,  and fonts. This separation improves content accessibility; provides more flexibility and control in the specification of presentation characteristics; enables multiple web pages to share formatting by specifying the relevant CSS in a separate.css file, which reduces complexity and repetition in the structural content; and enable the.css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

### JAVASCRIPT



Figure 8: Logo of JavaScript

JavaScript, often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. Over 97% of websites use JavaScript on the client side for web page behaviour, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js.

During the period of Internet Explorer dominance in the early 2000s, client-side scripting was stagnant. This started to change in 2004, when the successor of Netscape, Mozilla, released the Firefox browser. Firefox was well received by many, taking significant market share from Internet Explorer. Meanwhile, very important developments were occurring in open-source communities not affiliated with ECMA work. In 2005, Jesse James Garrett released a white paper in which he coined the term Ajax and described a set of technologies, of which JavaScript was the backbone, to create web applications where data can be loaded in the background, avoiding the need for full page reloads. This sparked a renaissance period of JavaScript, spearheaded by open-source libraries and the communities that formed around them. Many new libraries were created, including jQuery, Prototype, Dojo Toolkit, and MooTools.

### REACT

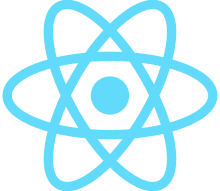


Figure 9: Logo of React

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

React code is made of entities called components. These components are reusable and must be formed in the SRC folder following the Pascal Case as its naming convention (capitalize camelCase). Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass the values between components through "props":

Lifecycle methods for class-based components use a form of hooking that allows the execution of code at set points during a component's lifetime.

* **shouldComponentUpdate** allows the developer to prevent unnecessary re-rendering of a component by returning false if a render is not required.
* **componentDidMount** is called once the component has "mounted" (the component has been created in the user interface, often by associating it with a DOM node). This is commonly used to trigger data loading from a remote source via an API.
* **componentWillUnmount** is called immediately before the component is torn down or "unmounted". This is commonly used to clear resource-demanding dependencies to the component that will not simply be removed with the unmounting of the component (e.g., removing any setInterval () instances that are related to the component, or an "eventListener" set on the "document" because of the presence of the component)
* **Render** is the most important lifecycle method and the only required one in any component. It is usually called every time the component's state is updated, which should be reflected in the user interface.

JSX, or JavaScript Syntax Extension, is an extension to the JavaScript language syntax. Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Facebook for PHP called XHP.

Hooks are functions that let developers "hook into" React state and lifecycle features from function components. Hooks do not work inside classes — they let you use React without classes.

React provides a few built-in hooks like useState, useContext, useReducer , useMemo and useEffect. Others are documented in the Hooks API Reference. useState and useEffect, which are the most commonly used, are for controlling state and side effects respectively.

### WEBPACK



Figure 10: Logo of webpack

Webpack is an open-source module bundler that is widely used by developers to manage and bundle their web application assets. It is a powerful tool that allows developers to organize their code into small, reusable modules, which can be optimized, transformed, and loaded on-demand as required by the application.

One of the key benefits of Webpack is its ability to handle a variety of assets, including JavaScript, CSS, and images. This makes it easier for developers to manage and optimize their codebase, resulting in faster load times and a better user experience. Additionally, Webpack provides a rich ecosystem of plugins and loaders that is used to customize the build process. These plugins and loaders enable developers to automate tasks such as code optimization, transpiling, and minification.

Another advantage of Webpack is that it supports hot module replacement, which means that changes made to the code can be reflected immediately in the browser without having to refresh the page. This can significantly speed up the development process and make it easier to iterate on code changes.

Webpack plays a crucial role in MicroFrontend Architecture by enabling the bundling of individual micro frontends into a single, cohesive application. In this architecture pattern, different development teams work on different micro frontends, and each of these micro frontends is developed, tested, deployed, and maintained independently.

Webpack is a module bundler that can take the different micro frontends and their dependencies, and bundle them into a single JavaScript file that can be loaded into a web browser. This enables the seamless integration of the different micro frontends, resulting in a unified user interface that offers a seamless user experience.

Webpack can also be configured to optimize the size of the final JavaScript bundle by removing duplicate code and compressing it for faster loading times. This approach can significantly improve the performance of MicroFrontend applications.

In addition to bundling, Webpack can also be used for other tasks such as code splitting, hot module reloading, and tree shaking. These features enable developers to optimize the development and deployment of Microfrontends by providing efficient ways to manage dependencies, improve performance, and reduce development time.

Overall, Webpack plays a critical role in Micro Frontend Architecture by enabling the bundling and optimization of multiple micro frontends into a single application, resulting in improved performance, a seamless user experience, and faster time to market.

### NODE.JS



Figure 11: Logo of Node.JS

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of modules' ' that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.

JavaScript is the only language that Node.js supports natively, but many compile-to-JS languages are available. As a result, Node.js applications can be written in Coffee Script, Dart, TypeScript, ClojureScript and others.

Node.js is primarily used to build network programs such as Web servers The most significant difference between Node.js and PHP is that most functions in PHP block until completion (commands execute only after previous commands finish), while Node.js functions are non-blocking (commands execute concurrently or even in parallel, and use call-backs to signal completion or failure).

Node.js is officially supported on Linux, macOS and Microsoft Windows 8.1 and Server 2012 (and later), with tier 2 support for SmartOS and IBM AIX and experimental support for FreeBSD. OpenBSD also works, and LTS versions available for IBM i (AS/400). The provided source code builds on similar operating systems to those officially supported or be modified by third parties to support others such as NonStop OS and Unix servers.

### C#

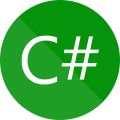


Figure 12: Logo of C#

C# is a general-purpose, multi-paradigm programming language. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. C# is a simple, modern, general-purpose, object-oriented programming language developed by Microsoft within its .NET initiative led by Anders Hejlsberg. C# is one of the most popular programming languages and can be used for a variety of things.it is a general-purpose, multi-paradigm programming language. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines.

By design, C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL), or generate any other specific format

### .NET



Figure 13: Logo of .NET

The .NET Framework (pronounced as "dot net") is a proprietary software framework developed by Microsoft that runs primarily on Microsoft Windows. It is the predominant implementation of the Common Language Infrastructure (CLI) until being superseded by the cross-platform .NET project. It includes a large class library called Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (in contrast to a hardware environment) named the Common Language Runtime (CLR). The CLR is an application virtual machine that provides services such as security, memory management, and exception handling. As such, computer code written using .NET Framework is called "managed code". FCL and CLR together constitute the .NET Framework. .NET Framework began as proprietary software, although the firm worked to standardize the software stack almost immediately, even before its first release. Despite the standardization efforts, developers, mainly those in the free and open-source software communities, expressed their unease with the selected terms and the prospects of any free and open-source implementation, especially regarding software patents. Since then, Microsoft has changed .NET development to more closely follow a contemporary model of a community-developed software project, including issuing an update to its patent promising to address the concerns. .NET Framework includes an implementation of the CLI foundational Standard Libraries. The .NET Framework Class Library (FCL) is organized in a hierarchy of namespaces.

NET platforms are encouraged to implement a version of the standard library allowing them to re-use extant third-party libraries to run without new versions of them. The .NET Standard Library allows an independent evolution of the library and app model layers within the .NET.

### MYSQL



Figure 14: Logo of MySQL

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyser. MySQL is offered under two different editions: the open-source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise share the version numbering system and are built from the same code base.

MySQL is a popular open-source relational database management system that provides a range of features to support efficient and reliable data management. One of its major strengths is its support for a broad subset of ANSI SQL 99, along with extensions that enable users to perform complex data operations. MySQL is also known for its cross-platform support, allowing users to work with the system across a range of operating systems and platforms. The system also offers support for stored procedures, using a procedural language that closely adheres to SQL/PSM, as well as triggers, cursors, and updatable views. MySQL supports online Data Definition Language (DDL) when using the InnoDB Storage Engine, and also provides an Information Schema and Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes. Additionally, a set of SQL Mode options are available to control runtime behaviour, including a strict mode to better adhere to SQL standards. These features make MySQL a powerful tool for managing complex data operations across a wide range of applications and use cases.

### VISUAL STUDIO CODE

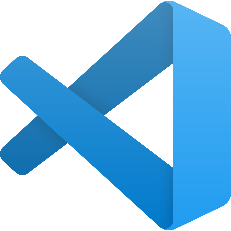


Figure 15: Logo of Visual Studio Code

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux and MacOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is free and open-source, although the official download is under a proprietary license. Visual Studio Code can be extended via [extensions](https://en.wikipedia.org/wiki/Plug-in_(computing)), available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new [languages](https://en.wikipedia.org/wiki/Programming_language), [themes](https://en.wikipedia.org/wiki/Theme_(computing)), and [debuggers](https://en.wikipedia.org/wiki/Debugger), perform [static code analysis](https://en.wikipedia.org/wiki/Static_code_analysis), and add [code linters](https://en.wikipedia.org/wiki/Lint_(software)) using the [Language Server Protocol](https://en.wikipedia.org/wiki/Language_Server_Protocol).

Visual Studio Code includes multiple extensions for [FTP](https://en.wikipedia.org/wiki/FTP), allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software. Visual Studio Code allows users to set the [code page](https://en.wikipedia.org/wiki/Code_page) in which the active document is saved, the [newline](https://en.wikipedia.org/wiki/Newline) character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

### MICROSOFT AZURE



Figure 16: Logo of Microsoft Azure

Microsoft Azure, often referred to as Azure is a cloud computing service operated by Microsoft for application management via Microsoft-managed data centres. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools, and frameworks, including both Microsoft-specific and third-party software and systems. Azure Web Sites allows developers to build sites using ASP.NET, PHP, Node.js, Java, or Python and deploy using FTP, Git, Mercurial, Team Foundation Server or uploaded through the user portal. This feature was announced in preview form in June 2012 at the Meet Microsoft Azure event. Customers can create websites in PHP, ASP.NET, Node.js, or Python, or select from several open-source applications from a gallery to deploy. This comprises one aspect of the platform as a service (PaaS) offering for the Microsoft Azure Platform.

Azure offers a suite of cloud-based services for identity and access management, data storage, and protection. Azure Active Directory (AD) is a popular tool for synchronizing on-premises directories and enabling single sign-on (SSO) for users across different applications. Azure AD B2C allows organizations to manage consumer identity and access in the cloud, while Azure AD Domain Services can be used to join Azure virtual machines to a domain without domain controllers. Azure Information Protection provides comprehensive protection for sensitive data, allowing organizations to control access and usage of critical information. Azure Storage Services offers a range of REST and SDK APIs for storing and accessing data in the cloud. Azure Table Service is a NoSQL non-relational database that enables structured text storage in partitioned collections of entities that can be accessed by partition key and primary key. These services provide a scalable, secure, and flexible platform for organizations to manage their data and identity needs in the cloud.

### **UNIT TESTING**

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff.

TODO use CHATGPT write prompt give me detailed information about <<fill>> testing framework with An example code and the test case for the same situation.  
TODO Put logo of the same

#### Moq

#### Jest

#### React Testing Library

#### Enzyme

### **GIT AND GITHUB**

TODO put logo

TODO documentation of the following commands

Add

Commit

Pull

Push

Branch

Pull request

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

GitHub is a web-based platform that provides a range of tools for coding collaboration and project management, utilizing the version control system Git. This platform is commonly used for tracking changes to code, collaborating with other developers, and managing projects with repositories. It allows users to clone projects to work on a local copy, stage and commit changes, and branch and merge to work on different parts and versions of a project. Additionally, users can pull the latest version of a project to their local copy and push local updates to the main project. It is important to note that Git and GitHub are not the same. Git is the version control system, while GitHub is a platform that makes use of Git to provide its services. As the largest host of source code in the world, GitHub is widely used by developers and has been owned by Microsoft since 2018.

# CONCLUSIONS

The following goals were achieved upon completion of the project

1. Understanding the entire C# and React knowledge required to work on production level applications and how apps are designed for improved workflow performance etc.
2. A single API and single UI based repository has been successfully decomposed into 4 API based and 4 UI based loosely coupled repositories that are individually managed by multiple teams and this has boosted productivity of individual teams without the need for reliance on other teams
3. Performance has been visibly improved and the API bottlenecking has been reduced to an optimal amount so that each request takes not more than 20 seconds to be responded to.
4. The overall unit test coverage was increased from 38.4% to well over 65% for each of the micro-frontend repositories.