BOOKBOXED

Project report submitted in fulfillment of the requirement for the degree of Bachelor of Technology
in

Computer Science and Engineering/Information Technology
By

(Utsarg Saxena (181305))
(Anurag Sharma (181278))

Under the supervision of

(Dr. Vipul Kumar Sharma)

to

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report titled **" BOOKBOXED – A BOOK-ORIENTED WEB APPLICATION USING FIREBASE CLOUD MESSAGING"** in fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2021 to May 2022 under the supervision of **Dr. Vipul Kumar Sharma(**Assistant Professor) Computer Science and Engineering and Information Technology. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Utsarg Saxena, 181305

utsarg

Anurag Sharma, 181278

[illegible]

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Vipul Kumar Sharma
Assistant Professor
Computer Science and Engineering and Information Technology

Vipul Sharma

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our teacher and mentor **Dr. Vipul Kumar Sharma** for his immense support and valuable guidance without which it would not have been possible to reach at this stage of our final year project. I am also obliged to all my faculty members for their valuable support in their respective fields which helped me in reaching at this stage of my project. My thanks and appreciations also go to my colleagues who have helped me out with their abilities in developing the project.

Utsarg Saxena, 181305

Anurag Sharma, 181278

ABSTRACT

Books play an important role in everyone's life as they introduce them to a world of imagination, providing knowledge of the outside world, improving their reading, writing, and speaking skills, and raising their memory and intelligence. Our idea was to create a platform where we can not only buy and locate books but can also discuss them with a community of people with same interest, because reading books is just not enough, our goal was to create a platform where we could not only buy and locate books, but also discuss them with other people who shared our interests. Reading books isn't enough, discussing what we read is also important for learning in all disciplines because it allows us to process information rather than simply receive it.

Our project's main goal is to establish an online book store that not only allows users to search and purchase a book but also share the gained knowledge with a community of people who are also interested in that book. That way, we can understand the ideas given in any book in a much better way and we can also find other books that we might find interesting. We can also locate Bookstores in our locality.

Our Web App serves as a central database for all of the books, including their title, author, and price. The front end of this web project is built with React.js, and the back end is built with Firebase. The Google Books API keeps track of a variety of book-related information. The website offers a diverse selection of books, ranging from fantasy to fiction. The user can choose a book and see its price. The user may even search for specific books on the website.

The community space that we were talking about is chatting application integrated with our book store app. It acts as a discussion portal where we can log in using our Gmail id, and discuss literature with other people who are interested in that book. They can also help us with discovering other books of same genre. We also have our gallery of books, where we can explore books.

Firebase Cloud Messaging. can be used to send messages consistently and for free. It allows us to keep our app subscribers engaged by providing with contextually relevant messages that make them use major features. Messages can be sent using the Firebase Admin SDK or FCM server protocols. The Notifications composer can be used for testing as well as sending marketing or engagement messages utilizing robust built-in targeting and analytics, as well as custom imported segments.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Introduction

Web Services, Firebase Cloud Messaging, Google Cloud Messaging which superseded GCM, are all designed to send and receive messages over the Internet. In comparison to permanent hosts, where energy and data efficiency are minor considerations, energy is constantly constrained and network data throughput is usually capped in any mobile scenario accessible. In both circumstances, applications may be subjected to time limits. Smartphone applications are increasingly replacing websites in the Consumer-to-Business and even Business-to-Business industries as the number of mobile users connected to the Internet continues to rise. As a result, when choosing a messaging approach, it's no longer enough to consider how much energy is consumed; it's also necessary to consider message delivery time, which affects application reaction time. We built up a test bed employing mobile devices to measure one-way delay, round trip duration, and data usage in order to solve this problem.

FCM is a cross-stage application and messaging for Android, iOS, and web applications. Firebase, a firm currently possessed by Google, gives and keeps up with FCM. GCM's principal framework is acquired by FCM, but customer side improvement is streamlined (for instance, engineers at this point don't have to compose their own enlistment or membership retrying rationale in the customer application). Most of the thing has been said about GCM additionally applies to FCM. FCM works along these lines to GCM, then again, actually messages are steered through FCM servers rather than GCM association servers. The way that Google has not yet belittled GCM, engineers are profoundly urged to move their applications to FCM.

1.2 Problem Statement

Normal book shopping sites only provides us the option of buying and exploring books but our web app takes it a step further with providing its users a platform for not only buying and exploring books but also for discussing them with people with as much interest in that book, all in one place. That way we can actually discuss the ideas given in any book in a much deeper way as we'll be getting to know some other point of views regarding those books.

1.3 Objective(s)

1. Creating an online Bookstore.
2. Adding a Discussion tab.
3. Generating Location of nearest Bookstores.
4. Adding a Book gallery.

1.4 Methodology

Firebase Cloud Messaging. can be used to send messages consistently and for free. It allows us to keep our app subscribers engaged by providing with contextually relevant messages that make them use major featurs. Messages can be sent using the Firebase Admin SDK or FCM server protocols. The Notifications composer can be used for testing as well as sending marketing or engagement messages utilizing robust built-in targeting and analytics, as well as custom imported segments.

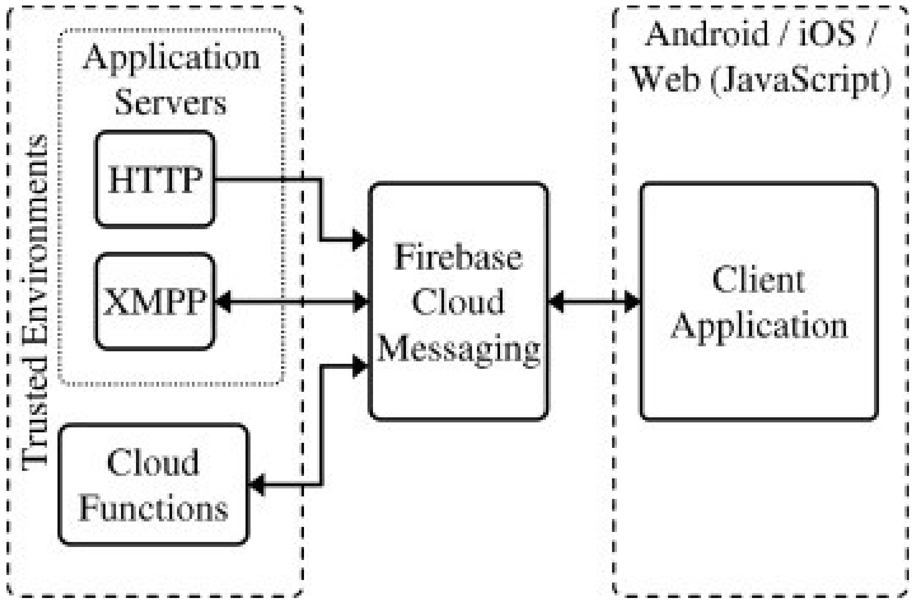


Fig 1.1 Working of Firebase

It Supports sending messages and notifications. They are automatically managed by firebase cloud manager SDK and it serves a purpose of showing the notification from the client application. It also helps in messaging with the help of built-in Keys, and messages contain key-value pairs and are automatically and fully controlled from the client application.

Firebase cloud messaging is almost similar to google cloud messaging, difference is that messages cross through firebase cloud messaging servers, but in the case of Google cloud messaging, it is not the same. Nowadays, most of the developers have changed their preference as firebase cloud messaging as it is much easier to use and it serves a purpose of showing the notification from the client application.

As a result, when choosing a messaging approach, it's no longer enough to consider how much energy is consumed; it's also necessary to consider message delivery time, which affects application reaction time. We built up a test bed employing mobile devices to measure one-way delay, round trip duration, and data usage in order to solve this problem.

1.5 Organization

This task report will incorporate compact and intensive substance on the different examination done to comprehend the different Processes in the point. Additionally, it will likewise incorporate the different advances taken to achieve the fulfillment of undertakings. Following is the series wherein the different exploration and steps are taken:

Part 1:

Manages the short outline of the theme selected. It gives a prologue to the point. Then, at that point, it moves to the issue proclamation of the undertaking alongside the goals that are should have been accomplished. Then, at that point, it continues on to the philosophy that should be selected. Lastly, it examines how the substance is coordinated in the remainder of the report.

Part 2:

Examines the summed up outline of the different exploration papers thought about alongside significant raw numbers finished with the necessary references used in those examination papers. It shows to perusers that you have a careful comprehension of your subject and that you appreciate how your review squeezes into and adds to a bigger assemblage of information.

Part 3:

Examines the framework advancement. In it different devices required are talked about alongside their concise subtleties. Further the improvement model is talked about which is to be utilized during the advancement of the task.

Part 4:

Gives the examination and differentiation between various strategies utilized till the current date and their exhibition correlations on different grounds.

Part 5:

Gives the end that has been gotten from the exploration and results got alongside the different uses of my task.

2. LITERATURE SURVEY

Firebase Cloud Messaging:

Firebase is a San Francisco-based cloud services provider and backend as a service provider. Firebase Cloud Messaging is real time database so it's a very popular for a lot of mobile applications and recently it has been growing in popularity in web applications as well as games. We are going to cover the introduction to firebase itself and then move on to covering the real time database. This report is a broad and general introduction to firebase real time database, meaning you would be able to understand the concepts behind it. The building blocks of it, the data types the structure what makes this a good database and how we can use it so this is not using it with any language.

Firebase has been growing popularity in recent years it was actually developed in 2011 by firebase Inc but it was acquired by Google in 2014. It's really popular in the mobile and web dev community so the reason for that is that it's just it's just a very simple and straightforward service that you can use it's technically a back-end as a service that provides a series of services and tools that you can use in your applications to help your application improve and grow and build upon what you already have so it's really good because using firebase is very powerful because a lot of the things the services it provides you with require very little code for you so instead of having to build so many things in your back-end from scratch. You just have to refer to firebase and then it would just perform a lot of things are already built in.

Now what do we mean when we say this like what are the different services provides so there are a series of services the first and the very popular one is the analytics one so this is kind of related to Google Analytics so we would use this today what to grow your app you would use all sorts of data analysis and data science tools that it provides you to make recommendations to test for your users to have and a series of things that would enable you to better understand the activity on your application and to better understand what makes your application fail so in case that your application crashes a lot. You could have bug reports as well you could have reports from your users when they're active what do they do how active they are so there is a lot of information you can gather using this you also have a wide variety of features that you can benefit from and use in your application. The first is actually the authentication features so authentication is for user management so this is what you would use to sign in with email and password and create an account with email and password but you could also sign in with Google sign in, with Facebook

sign in, with GitHub so this is you'd be using your different credentials from these different websites.
Services that we have on the internet and you can use those to sign into your application and firebase Brooke is sort of the gateway to that not that's what it helps you do use the authentication feature all right. Python actually so you can check that out if you're interested all right moving on we also have cloud storage so most applications or startups nowadays have some sort of media files on their applications so it will have images we would have videos we would have all sorts of things and for you to have to store them somewhere to store them on the cloud for your users to be able to access them and that's what we have cloud storage for so cloud storage was initially a Google product however it merges with firebase and provides the firebase cloud storage and you can use the storage to store or store documents photos videos anything your user is uploading or viewing from your application you can store it there and that media would sort of be directly available.
It also is authentication combined with storage to enable your users your application to keep track of which user has uploaded which photos so the way Instagram knows that these photos are yours because you uploaded them so these photos are associated with user X all right now. Another thing we have are the real-time database and the cloud fire stores so firebase actually provides two different database types cloud fire store was originally a Google thing but also merged with firebase upon acquiring the company we assign database used to be the actual firebase database but now it's called real time database so both of them actually do work in real time meaning they're always thinking across devices for any changes so you wouldn't have to refresh to be able to see any changes. You just see it in real time and they're both hosted on the cloud and they're both no SQL databases if you want more information on no SQL databases.
So, what you have to do to be able to use firebase we need a Google account the same way you do for any other service. Now there are some demo fire-based projects or some sample projects that we can access so you can just access them and you could also create a project. You could also view an old one so it depends on what you want to do with your work. If you're a Python programmer and you're interested so there is the dashboard that you have where your project overview is. It has all the sorts of data that you need for your project as well as all the features that you need to develop your project. Now here you can have some sort of statistics that keep track of downloads and uploads to your database after your storage at the same time so you could use those to keep track of how much data is being pushed into your database the whole time and what you can do about it. You have the authentication and the different features provides you with you have cloud forests or

which is another real-time cloud hosted no SQL database that firebase provides but it's not the real-time database so it just has a different name and it different features.
You can have your analytics here so you have performance and crash lytx meaning you want to be able to report you have your users report the bugs to you so you can go ahead and fix them in your code and all sorts of features you can have cloud messaging so this is used for using push notifications so if you want to notify people using your application you would use cloud messaging and send them a push notification a be testing as well as a series of other things so you can always add extensions as well so you can have shorter URLs. You could have translating tags you have resizing images so you can just add a huge number of things so the good thing about companies like Google is that they cover a wide variety of subjects meaning most of the time if you need something there will be an extension or something related to what you need all right so now we're in our tutorial we're actually going to go deeper into the firebase real time database. It's sort of a hierarchical format so we are just going to expand the data and this is what we have so this is some sample data for a bunch of people or users and here's how the data works.
Now we are going to talk about the data and the data types that we have and how we can add and remove data all right so let's just get started so it's a hierarchical database so if your form familiar with JSON this will be a breeze for you because essentially firebase is document oriented meaning the the data is stored within documents and these documents follow the sort of JSON format that we know. Here what we have we have is the database so it's the root node. The root node will always be the name of your database of your project all right then we have a child node for it and nuts users and within users. We have a series of all the English so if We are just going to collapse the data and open this up, there are the user IDs so these are different JSON objects within users that and each object from these so we have six objects each object represents one user and if you open it up the values within this object or a series of key value pairs that define the modifying the user itself an provide data and details about this user.
You can also change the way you have a layout so we can do this manually now so no coding required so here I can say name and then say Utsarg and then I can add it so now I have two child nodes for my route DB, so instead of having one parent node that's hierarchical here I just got a key value pair so this is also an object. We can also have another hierarchical format so we are just going to say hierarchy and then. We are going to use the plus about the graphical user interface here because we always tend to go for ads and then we can just say name and now we say Anurag and then now we can add, and here we have a hierarchy with that name and your hierarchy can go pretty

deep so if I add another thing here I can say some letters and then add another bunch of letters and then add and you can go infinitely deep with this, however it's not recommended because you will just be losing the sort of flexibility with fire based queries.

What do we mean by this fire based queries are more geared towards shallow data meaning you're better off not going more than probably two levels deep within nesting your data so you should stay within a good solid hierarchy but make sure your data isn't too far in a stood because that would cause problems when you're trying to query the data and you would just run into a bunch of issues so just keep that in mind. This is sort of the structure that we have we know that this is a new SQL database meaning the data points do not have to match each other so let me just delete this hierarchy and just forget it goes over there. We are going to delete this as well so let's just go to users and see what we are talking about. We are just going to expand so as you can see a user has address age

first-name and last name and a bunch of other users are the same however this user so Mark Johnson does not have an address. Why is that, this is mainly because it's a schema list database as most no SQL databases are meaning not all data points have to have the same attributes so if you familiar with SQL you know that within one. We have a table called person and we have columns called name last name not age something and these sorts have to have the sort of same structure meanwhile and no SQL.

We do not have to follow the same schema for every single data point within a document within a certain collection so this is what we have now like understand the no SQL part of it you understand this structure let's just talk about the datatypes, so we said key value pairs so here users are actually a key value pair where users is the key and the object itself is a series of objects so it's everything within it. It's a sort of array of objects. Let's just collapse it so the user is just the series of different user objects here each object is a also a key value pair this object has the key of this generated ID we'll talk about those in a minute and these are the values so the values are also a series of key value pairs and these key value pairs would define what this object really is. So where are these IDs?

These IDs are time-stamped generated by firebase when you push data into it you can choose to either set your own keys or use a generated key. Here the key is generated here the key is sucked because the key is users in this case alright here the key is set because the key is H and the value is 20 but here the key is not that it is generated using the timestamp and a bunch of other random things. Firebase uses to create these keys all so essentially this is what it is.

You have a node or a key value pair, you have a value that can be of another object. The object is a series of key value pairs like we know so what does this mean that you can have infinitely nested

objects however it's not as recommended because you rather want to have a shallower sort of format for you for easier queries, that’s what makes firebase and NoSQL database alright so these are the key value pairs now we know that the value can be another object so here users value is an object. The value can be an array of different objects so we have a sequence of objects separated by commas. They're within an array so same as JSON and that's another type of value so you have object adding the third type of value is a scalar value meaning you can have an integer like here so there are no quotation marks. This is actually an integer or you can have a string so these are strings right here they have quotation marks or you can have a series of other things such as bullion so we can create something like, let's just add something here and just say true and then we add it and as you can see it's does not have quotation mark it is a Boolean value.
So this is the same type of data types that we have as programmers but we also have the other parts which make it an asset such as arrays objects and then we have this scalar value. What happens from there on from then on we have to connect this database to a sort of application so it's either going to be an Android or iOS application or it's going to be a web application including Python which falls under the web for even if it's a desktop application, so you're connected to this application what you're going to do is that you're going to push and retrieve data from this database and you're going to work with it the same way you would work with any other database. So this is what makes the firebase real friendly database and sort of unique because it's part of a larger ecosystem of firebase products, however the firebase real time database on its own is really just a simple cloud hosted and real time no SQL database that syncs in real time. If your application requires NoSQL database and it's more of a sort of small-scale application, you may go for it because it's not expensive at all. You can use for free or you can maybe potentially want to stay and have larger pay money to keep benefiting from the fire-based services.

Neha Srivastava, Uma Shree, Nupa Ram Chauhan, Dinesh Kumar Tiwari Department of CSE FGIET, Rae Bareli, India. In their paper, “FIREBASE CLOUD MESSAGING” quotes

Firebase is a San Francisco-based cloud services provider and backend as a service provider. Firebase offers a unique framework for developing mobile and online applications. It can fabricate application and update it progressively. Firebase is exceptionally simple and it stores information in

JSON design. We don't have to arrange our server when we use firebase. All that will be dealt with by firebase consequently. So, server-side coding isn't needed. It will save time and will make us more useful. In this examination paper we will essentially disclose how to utilize firebase for our android application. Clearly, both server-side code and customer side code is needed for building a web-application. We'll need Ajax, an accessible API, and a place to host the backend. Many technologies, such as Ruby on Rails or services like Heroku, make backend development simple, but they require a significant amount of development work to implement. Because Firebase addresses the majority of backend concerns, it is referred to as Backend as a Service (BaaS). Web sockets are used by Firebase to push state to our application. It updates the data in real time, so we don't have to refresh the browser to see the latest information, making it more engaging.

FCM’s working:

Firebase Cloud Messaging (FCM) is a cross-platform interactive application that allows you to construct your app on a variety of platforms, including Android, iOS, tablets, and more, to deliver messages in a more dependable and cost-effective manner. You can use FCM to inform a client app when new email or other data is available to sync, all in real time. FCM allows many parties to transmit a message at the same time.

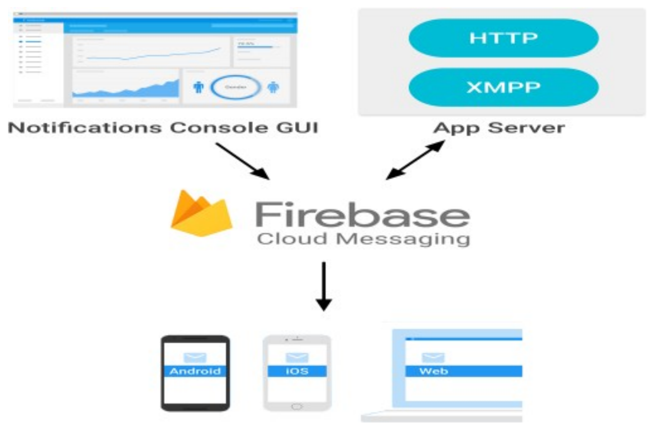


Fig 2.1 FCM Flow-Chart

A client app connects with FCM via HTTP or XMPP protocol, and an app server interacts with FCM via HTTP or XMPP protocol. Using the app server or the notification console, you can construct and send messages. Firebase Cloud Messaging enables us to create notifications using the same software development kit for client development as Firebase. Notifications can be used for testing or sending marketing or engagement messages with robust built-in targeting and analytics. Within nanoseconds, Firebase may refresh the data.

FCM Messages:

FCM provides a diverse set of messaging systems with advanced features and capabilities. Message type and Notification message are the two sorts of services covered in this section. The Hypertext Transfer Protocol (HTTP) is an application layer protocol enabling secure communication between sender and recipient over the internet. Extensible Messaging and Presence Protocol (XMPP) is an open XML technology and protocol for sending and receiving messages. X stands for extensible, which indicates it grows and adapts to changes in a defined way. The letter M stands for the message you're looking at. Prefers the presence indicator, which is used to inform the server whether you are available or not. Protocol is a collection of rules that allows systems to communicate with one another.

Types of Messages:

You can send two types of messages to clients using FCM:

1. Notification messages, also known as "display messages" in some cases.
2. Data messages, managed by the client app.

Notification messages:

The firebase SDK is in charge of handling notification messages. The notification message typically has a title, message, and icon, and it may be sent via the Firebase console UI. You won't have much control over the notification if you send this type of message. When the app is in the background, the notice will appear automatically. Here's an example of a JSON-formatted notification message in an instant messaging programme.

Data messages:

To transmit a data payload to the client app, set the data key with your custom key-value pairs. The maximum payload size for data messages is 4KB. Here's an example of a JSON-formatted message in the same IM app as before, with the data key encapsulating the information and the client app expected to interpret the content.

Using the Firebase console, we are unable to send Data messages. We'll need to use either the REST Client or the REST Server.

Features:

Real-time Database: A cloud-hosted database with no schema and structured in Java Script Object Notation.

Authentication: Authentication in Firebase is a fantastic tool that allows you to authenticate all of your users without having to write code.

Storage: Files can be securely uploaded and downloaded without being harmed by network quality.

Hosting: Firebase web apps can be statically hosted in a secure, fast, and efficient way.

Remote configuration: This functionality allows you to update the user application without having to deploy the most recent version.

Test lab: Applications are tested in a number of configurations and on a type of mobiles.

Crash revealing: This usefulness is utilized to make a report of all program crashes and blames.

Conclusion:

Large numbers of the Google highlights are conveyed forward alongside other progressed highlights like accident announcing, permitting designers to make basic and more practical applications giving a wide assortment of administrations. Firebase is a stage and device that is known for its speed and dependability as far as the time it takes to assemble constant applications with a profoundly more straightforward stage. In light of the variety of creative highlights that are acquainted with Firebase that give an assortment of administrations, it has a splendid future.

All of the previously mentioned would now be able to be accomplished utilizing Firebase, which is very similar and requires no convoluted setting. This saves a ton of time since information is refreshed progressively, which implies that all progressions are naturally refreshed to every single associated customer, making our applications/destinations novel. It has a straightforward and simple to-utilize dashboard. Set the Java Script Object Notation (JSON) like items inside security rules and change their default esteems to specific novel qualities to permit security in Firebase.

ReactJS:

When it comes to learning ReactJS there are a few high-level points that are really helpful to know for understanding what React.js really is how we could use it and how it might be different or similar to writing vanilla, JavaScript or working with other libraries and frameworks. The first thing to mention is that react is a user interface library that means that we use it for creating user

interfaces websites applications. Anything that the user is going to see in the browser and other places that react could go that is what it is primarily used to help with also react has a component architecture component is a small piece of code that fills a certain part of the user interface that you're building with react also data flow and react react has a one-way data flow which is becoming the standard today in JavaScript applications but it was one of the first to introduce it. This is very different from two-way binding which you may be familiar with in JavaScript. Finally we have component State in native react.js. What that means is that we can manage state data or changes that are happening in our application but they are controlled at a component level so a single part of the user interface would have state that could be shared with others and we'll talk about how this connects with 0.3 of the data flow flowing down in react so let's drill into each one of these in a little bit more depth starting with the user interface library.
Now ReactJS is an agnostic user interface library and what I mean by agnostic is that it doesn't care where your user interface will ultimately display we can see here an example of react in combination with react DOM a separate library that will have react code display in a browser so we have a function here or component name app and we're importing a header content and footer component as well these are also functions in other files and at the bottom. We could see that react passes this component that react makes into react DOM to render it to the page and we see good old-fashioned DOM selection here. Document get element by id so this will look for anything in the page where this is loaded and find an ID of route and load this component. Now this may make sense if you're used to working with JavaScript in the browser however the react library doesn't actually care that you're using it in the browser and we can see with this example here that we're integrating react with react 360 or their VR or 360 environment companion library so react on its own doesn't care where you send your final components and it always needs to be used in conjunction with a second library like react 360 react on it’s server. There are a number of them we'll be focusing specifically on react DOM because we're coding react for the browser but we want to point out here that react is agnostic. It could work in a lot of different environments and we just have to modify it or pass it into the companion library that makes the most sense for where we want to display our user interface. Now I want to make a point even further about this especially in the context of working on the web that when we create a react element or a piece of our user interface that is not actually a DOM element. So if we were to create a paragraph element here and put in the word paragraph this is react code we will learn and we log this out we would actually find that

we're getting a react element here that is a plain JavaScript object with a bunch of different properties and methods set and available but it is not a DOM element so react does not become DOM until it's passed through react DOM.

Now for almost all intensive purposes, this won't really matter but if you are used to working with DOM elements you have to switch over to remembering that in react an element is just a native JavaScript object and eventually it will become a DOM but you can't always do everything with those objects that you can with DOM elements and you could do a lot with react elements that you can't do with DOM elements so just want to draw that point in so to move on from the agnostic part. We want to point out again that react is a user interface library. What does that mean is it's going to be used to create interfaces whether they're charts or bars or search fields or entire websites whatever you're building with react commonly. It could be an entire user interface so an entire website all of its different components and pieces are an entire application all of its different markup and everything like that or it could be a little widget or a part of a site like a search part that you're only using from one little part and you're just creating the interfaces with that, Now ultimately it's going to kind of look and function like HTML and CSS and everything is you would expected to in JavaScript on the front end but we're going to be specifically using react for this so it's not as much for the data structure although we will see that there are ways to work with that in component state and props. However, react is a user interface library what is it used for building different interfaces parts of websites, parts of applications, anything that you might see and finally react is a library. This is in comparison with a framework. A framework often has a bunch of helper functions and a lot of structure over how everything has to be organized. React comes with a handful of functions that we could use and it comes with some suggestions on how to organize things like connected components and things like that but it's definitely not as robust and big as something like angular and you'll find in a lot of cases that you actually need to use react in combination with a bunch of other libraries. We already learned about react DOM. Just to use react on the web, we have to use it with a companion library. Sore act is pretty lightweight and just as functions for creating agnostic user interfaces all right so next up component architecture. Here we see an app .js file. This is commonly where you'll kick off an app. We import react and react DOM a header content footer. This is a very typical and what a high-level your main component would look like and then nested within our main div with a class name of app.

We have a header content and a footer; we can see at the bottom that we render this all to the page wherever root is and this is all pretty common conventions. Now you don't need to understand all this code at the moment but what we want to point out is that if we drill into one of these other components like header for example, we would find that it also imports react and it might import some other things like an add component or a logo and then it would be very similar to app it would export. A function of header and in react we could use functions to create components will learn. You could also use classes but functions are by far more popular we'll also see that rather than having to call header an app as you would normally call a function, we could use the component format and call them more like HTML, so in the react DOM render you could see that app is called more like a component a piece of an HTML than it is a function but you could also call it as a function that will work too because that is all that we have here we could also begin to see how we compose react applications and sites where we break up different parts of it into smaller components and as you build more of these you'll find that you could reuse them across things. If we keep them in this Lite modular small format alright, so let's imagine that we had an application with an app component or a function called app and then we had a header content and footer component that were all functions as well with similar names and then within those header, we had some more within footer, we had some more and so on and so on you could see here the naming conventions. You could see some of the organization of how this might be done and this is very common in a react app to have one main component and then the rest of your app nested along the way now you may be asking what do we do about conditional stuff or if we need to go here and load that instead and we'll get into all those but we want to start with this high-level simple thing to point out that we have these component trees now through those component trees data flows down one way and that is really important to emphasize and could be something that's maybe not tricky to grasp at first but can require some different patterns of coding that you might not be used to once you get the hang of it. It's a really simple really breakpoints in your application and with your data and this is very much becoming the standard with other libraries as well as well as just building with vanilla JavaScript a good approach to take with data architecture.

So what does this mean, data flows down one way through a react app. Let's come back to this component tree here where we have our app and all of these other components and imagine that we needed to get some data into our application doesn't matter could be users it could be content, it could be site info. Whatever it is, we go off and we make an API call and we get that data now we could pass this data down to the elements below it and then they in turn can pass it to the elements below that and this is how at the most basic level data flows through a react table now. This may seem pretty simple and that's good it starts off as a pretty simple concept. Now from here, we could also control what direction data flows and what gets it so maybe the footer doesn't need this data in order to function so we only pass these down to content and header and then they pass it down and how do we pass it. These are all functions so we pass it as parameters or props will learn about coming up but it's all pretty simple JavaScript when you look at it under the hood so data is flowing down from one component to another.

What happens when one piece of the component hierarchy here needs to update data, like let's say that you click on something in the main nav or there's a search component or something and you have to go off and fetch new data now in the react architecture we will learn about how main nav can actually trigger app to go back and go fetch new data and then automatically pass it back down. This means that we don't have to have two-way data binding where header content site info and app would all have to be tied to main nav and maybe other ones would have to have the ability to update it simply. We can call a function from app in any of its children components to trigger app to update now although this may look like we have a global State this is not exactly true and react has what's called a component level state and then this state or updated data is passed down.

So in the last example we saw app going and getting data so it looked like we had global state available everywhere but in this example header is going to go ahead and get some data and it's only going to pass it down to these two below it. It actually can't with the normal reactive data although we will learn about other methods of interacting with data that will allow sibling or parent child components to talk to each other in different ways but in the most basic fundamental way and react header can't actually pass this state data into content or footer because they're not nested within it that state and that data only belongs to header and any of its children that it passes data down into now depending on your experience with state management. This may start to get a little complicated but that's all we really need to mention. Now in order for us to really start digging into the code and learning how all of this works, so again at a high level react is a user interface library. It has a component architecture where different functions control different parts of that. User interface data is only going to flow one way and that's down from a component into its children and every component in react has the ability to manage its own state and pass it down to its children.

Naimul Islam Naim in her Research paper ReactJS: An Open Source JavaScript Library for Front-end Development quotes

ReactJS is a prominent open source front-end JavaScript library created by facebook,. Because of its simplicity and simple but effective development process, React is very popular among developer communities. React simplifies the creation of interactive user interfaces. It efficiently updates the application by presenting the exact components to each state's view and making data changes. Every component in ReactJS controls its own state and composes it to the user interfaces. With the use of components rather than templates in JavaScript, a large amount of data may be simply provided to the app, keeping the state out of the DOM. Server-side rendering is also possible with Node React. We can utilize React Native to create mobile apps in addition to web apps.

Benefits of learning ReactJS:

Easy to learn: Unlike certain JavaScript libraries, where learning the frameworks takes a long time, React requires little effort to get started constructing an application. React has a lot of powerful features. React's readability is one of its strongest assets. Even people who are unfamiliar with it will find it simple to read. While other frameworks necessitate learning many framework ideas while ignoring language principles, React does the exact reverse. Consider how different the rendering of a part of an employer's list is in React and Ionic (AngularJS).

It’s quick and sharp: One-way unidirectional data flow across states and layers in an application is a characteristic of ReactJS. Data flows in a single direction between the application states and layers in this case. When using two-way data binding, such as Angular, when a model is altered, the view changes as well, and vice versa. React is a considerably smaller package that renders updates in the DOM much faster than other frameworks. The document object model is abbreviated as DOM. As a result, selecting the instruments needed to complete the task is simple.

It introduced JSX: JSX is a programming language that allows you to declare DOM elements before components in JavaScript files. This implies that the logic and graphics for the components are all in one location. When other frameworks use queues to place them, this is a fantastic concept.

React Virtual DOM:

DOM implies Document Object Model. Current intelligent web innovations depend intensely on DOM control. It's generally alluded to be the "heart" of the advanced web. It's a portrayal of an organized text. In any case, it is more slow than other JavaScript activities on the grounds that most JavaScript structures update the DOM regardless of whether it isn't needed. In other words, those updates aren't needed to complete the activities, yet they do as such of course. Accept that nine merchandise have been set in a shopping basket in an internet-based web store. How about we imagine you just need to purchase the main thing and you're prepared to look at. Most advances would remake the full rundown that was set in the container for this situation. This implies the system needs to work multiple times harder than it needs to. As a result of a solitary change, the framework should reproduce the rundown precisely as it was beforehand.

Respond didn't begin Virtual DOM, yet it utilizes it and makes it accessible to designers free of charge. Virtual Dom is just a HTML DOM reflection. Each DOM object, like a reporter or a lightweight duplicate, has a virtual DOM object in React. A virtual DOM has properties that are like those of a genuine DOM. It can't, in any case, make any immediate adjustments to the presentation. Control of the DOM consumes a large chunk of the day. Control of Virtual DOM, then again, is quicker in light of the fact that it doesn't have anything to do with the view and makes no changes to the screen.

Advantages:

1. The React diffing algorithms are extremely quick and efficient.
2. We were able to create various frontends for the same application by including JSX and typescript.
3. It is extremely light and may be used on any mobile device.
4. There are a lot of tractions and mindshare.
5. It can also be used as a standalone engine without React.

Disadvantages:

1. It takes up a lot of memory as the DOM is copied completely in memory.
2. There isn't much of a difference between static and dynamic parts.

Data Flow Management in React:

Web users all across the world are clamoring for richer, quicker, and more interactive web apps. In this day and age, the internet is an extremely crowded area. Millions of websites and applications constantly saturate the internet with massive amounts of data. Many frameworks have already been developed to provide smoother and smarter services to all users. However, most recent JavaScript frameworks are difficult to master, occasionally frightening, and difficult to maintain.

ReactJS is thought to be less complicated, faster, easier to understand, and useful for maintaining states. It enables us to design reusable user interface components and introduces the virtual DOM in place of error-prone manual DOM manipulation. It allows you to create the user interface in a way that is dependent on the condition of the data. To speed up the process, only a few DOM changes were implemented.

<u>WebRTC</u>

Webrtc or Web Real-time communication is a free open source project that provides web browsers mobile applications with real time communication. Our goal was to design a protocol that connects peer to peer. That's the shortest possible lowest latency path and we'll also provide an API that is simple for everyone to use and once we put it in the browser, it becomes a standard, and once it's a standard the friction goes away. It finds all possible communication it and also the security parameters medium from operation and then it does is exactly the same thing and somehow they communicated to each other.

WebRTC is basically a collection of api's that allows direct connection between browsers and this allows them to exchange files, information or any type of data. So looking at this write direct connection between browsers. It sort of sounds familiar, like WebSockets but not really, so the key word here is direct as for WebSockets, what happens is there's a connection or communication between only the client and the server. There's a lot of things going on in between but that's irrelevant as if client wants something it makes a request to the server and the server responds. Now if there are multiple clients, they make multiple requests and server responds to all of it, and since it's a socket it looks more like that so now the client doesn't need to keep making requests every time it's looking for something, so if one of the client changes something or makes an update, let's say they type in text or upload a file they send it to the server, the server processes the information and immediately pushes update to all the other clients and what this process does is that there's actually a sort of delay so between the sending client and the server and the server processing all of the information, these two clients on the right will have to sort of just sit there and wait normally. It's probably only a second at most but this is a huge issue if it comes down to a voice chat or live streaming where one second can change a lot of things so let's take a look at WebRTC.

Working

So with WebRTC the clients can actually directly communicate with each other and completely bypass the server so this decreases the latency by a lot because now receiving client does not have to wait for the sending client to the server and then the server to itself. Normally a client is really just a browser, it's HTML and it makes requests to the server and it waits for a response so that it can update itself and can contain new information but with WebRTC this changes fundamentally. The way we think of browsers because now browsers can communicate to each other without the server so how do we even know about each other, and how does it know which client is a connected to. Answer is signaling. SIgnaling is how it knows, so let’s take an example, there is a server and it's clients, so client A would signal to the server and say something along the lines of, “Hey, I need to talk to client B. Here's my information.” The server will then take that to client B and say, “Hey, client A wants to talk to you, do accept” and client B says, “Sure, here's my information”. The server then takes that back to client A and says, “Okay”. Client feeds information, it accepted the connection and it starts the connection between the two clients and with this connection. The client A can now directly talk to type B and the server will know nothing about it, so how does this work and what information is client A and B sending to the server and receiving. So this is sort of what happens behind the scenes, first, of course, everything goes through the API, so the developer interacts with the API. The client and server uses this API to set up the connection. Then there's identifying the client, so how does the server know this is the client and it's not just someone pretending to be the client then there's the type of data that it's sending over. Is it a video? Voice? Is it just some random files? What's being sent over, then there's Matt traversal so that stands for network address translation and that alone can be attack talk.

Basically what it is, is it's a technique that establishes a connection between the clients. It's commonly used in peer-to-peer sharing and transferring, and it transfers sort of like metadata from the browser such as browser information, IP addresses ports and many many more.Then there's security so identity is already sort of like a type of security but this type of security is encryption so when data is being sent over and transferred over WebRTC automatically encrypts any type of data, so that way if someone were to somehow listen in into this transfer of data, they won't be able to obtain any useful information. Finally, its codec so this just determines how the data is going to be compressed and sent over so WebRTC sounds great as it allows direct communication.

WebRTC uses UDP so this in itself isn't really an issue but this type of protocol isn't reliable for transferring important data so all it does is it sends data really really quickly. It just constantly sends data but it doesn't check whether or not the data is being received, so this could be useful for something like video on video chat because you can lose a few frames and it's not going to be noticeable but if a file being transferred loses a few bytes of data the entire file can be corrupted. Another challenge is that web RTC does not have any standard signaling protocol, so all the different developers, different companies will have their own methods of implementing different protocols and lastly it's not compatible with all of the browsers. Popular browsers such as Chrome, Firefox and Oprah are fully supported, no plugins needed. It just works straight out but other browsers such as a Microsoft edge and Apple's Safari require external plugins so that WebRTC can sometimes work.

Usage

So why WebRTC? With WebSockets, live streaming is possible but as as mentioned above, the streaming is way too slow and there's a very noticeable latency, very noise, a very noticeable lag but with WebRTC, streaming can be much quicker and this allows or rather this makes it so that we have no need for extra apps such as Skype or even zoom, all of it can be done directly from the web browser and companies such as Google and Facebook already implemented this technology in their chat services. WebRTC is embedded in web technologies so since there's already a connection between client to client, the server does not need to use any more resource to process incoming data and then transfer it over. So what this allows is that now this company or this server can just say, “okay here's the connection, you guys deal with it” and basically they don't have to waste any resources but they're giving their client extra features and there are many many development kits tools and open source libraries that can give us as developers many new interfaces. WebRTC also solves many security issues because encryption is mandatory for all WebRTC components and since it's not a plug-in or an extra app it runs inside of the browser's sandbox without creating a new process, so no spyware malware or anything can get into your system and also since it's using your browser security, camera and microphone access has to be granted explicitly so that way you know your face won't show up on someone else's computer by accident.

So, lastly WebRTC is still a pretty new tool for the browser but it has a lot of potential as its name states for real-time communications.

MRTC or web real-time communication is a technology that exists in most modern browsers that allows users to communicate with one another in real-time from within their browsers without actually having to use a server. What's kind of really neat about this is that the actual communication that's happening from one pair to the other pair all happens without a server now that's not to say that there are no servers involved in web real-time communication, for example, the project we're working on a chatting application, and there in fact will be a server, and we're going to actually be using an express and socket IO server on our back end but that service sort of just responsible for actually facilitating the connection between the two peers but once the two peers actually have a proper connection with one or another the server is no longer used. They are not communicating with each other directly without the actual use of a server.

Creating Client Folder

To create this app, we're going to go ahead and create a client folder that we are going to be bootstrapping in react application, so in other words we're going to be using the create react app to go ahead and build a new react application, that's going to serve as the client-side part on this application so we're going to do terminus 8 MPX create active client and then what this is going to do is, there have the inside the root of our application then we call there's going to be a folder called clients that will effectively be the sort of client-side part of our application that will just be a real defecation bootstrapped and using Korean taps. Then we are going to open up another terminal, navigate over to the actual client folder so now we effectively have to Terminal two terminals open point into the same project. The first one is going to be in the root of the project just at a video chat. Tthis wil basically serve as the terminal that's responsible for all of our back-end code and then in this terminal here we'll be pointing to the actual client folder. This is going to be responsible for all of our actual clients that are the actual react application part of this application so for dependencies in the actual root of the application we're going to need to install Express as well as Sicario so we're going to say yarn add Express and socket IO for the client application. We're going to install socket AO client and react without our socket IO client. It is because we're going to be having that sort of socket IO on the server so we actually

we need to have our clients application to be able to communicate with our server so we have to install socket little client on the client. we're going to be installing rack router table just so we can actually navigate between the different screens that are going to exist within our clients and application so let's go ahead enter and install these dependencies

Now the first thing that we're going to want to do is actually start building up our server-side code, so basically the whole point again is that our server-side code is read because what about to see in reality just for the actual sake of communicating one peer with another. They don't in fact require a server, the whole purpose of our server here is to actually allow the sort of connection between the different pairs but the kind of flow that our applications going to have effectively, it's just going to be a URL in the sort of clients and application in our case, it'll be like localhost/room/term ran, that might be when two people find themselves within the same URL at the same time. These people are automatically in a go into an actual conversation with each other on an actual video chat application and that's kind of going to happen automatically so the way this would work is let's say user actually creates a room so they're going to have like a button that they can press that says create room when they do that, they're going to then get navigated over to room ID that they've just created at this point. They're the only person here so all they're really going to see is just going to be their own video but they're not actually going to be in any kind of conversation just yet. Finally, when they actually take this URL and share it with somebody who they want to have a chat with that other person will then click on that link join the same room at this point the second person is joining effectively use your link.

Connections using WebRTC

We have to actually create the connection between the two peers and without this connection they don't actually have data flowing between one another and then there's not going to be any sort of WebRTC goodness happening, so to kind of facilitate this actual handshake what we need to do is we need to take this offer that's going to be created by user be send it down to the socket IO server which will then send it back up to use rate and then in return user able to go ahead and formulate with known as an answer in response to this offer take this answer, send it back down to the server and then a servers should go ahead and send it back out to you. Once this sort of round-trip process is done, you know sent each other an offering an answer now they've actually created a proper handshake now they can start communicating with one to another. That's what our server-side code is going to be doing right now.

First thing we have to do is we have to import the Express module. We are then importing HTTP. There's no need to install HTTP, it's a built-in module. We're then going to go ahead and create an app object which is basically our Express app. We do this by calling the Express function finally over here, we're going to say consumer is equal to http that creates server passing in our Express app which we've just created. We're then going to go ahead and import socket IO and we're going to create this variable called socket by importing socket IO and then going to create this instance of IO by calling socket passing in a newly created server so effectively. Now we've taken our Express app and we've attached it to our HTTP server which we then take and attach to our actual socket IO instance and then the last thing we now need to do right before we connect to start writing code but just as far as boilerplate is concerned is we just need to go ahead and tell our server to listen on port 8000 and then we're going to have a simple call back saying that the server isn't back running on port 8000. The basic idea is going to be that a person will be able to sort of create a room that's going to be defined by some sort of arbitrary URL or by some sort of arbitrary ID that exists within the URL and then another person that will then go ahead and join the room would then effectively be in the room together with this first person and now they're going to be in a sort of video chat application with one another. When a person finally actually connects to our socket IO server this connection event is going to fire us academic then go ahead and generate the socket object for this individual person and then what we're going to do is we're going to go ahead and attach an event listener object particular socket set that's going to be called join room so this join room events will then get fired off from within the client side code once we actually get to the client side code, you'll see how this gets fired off but basically when it does in fact get fired up what's going to happen is we're going to be pulling the room I'd the out of the URL and then sending it down to the server. Now what the server's going to do is going to have a basic check if this room's our room ID. If this room already exists in our rooms collection then basically that means that at that particular key in the rooms collection or in the rooms object that means we already have an array of socket IDs rights in others or we can simply say rooms a rumor they that push take the socket that ID so basically every single time that you create a new socket with socket.

They

will automatically put in a generator diabete onto the image sokka to kind of identify this socket, so in this room we already have a user with a socket ID. Now the new person is just joined to take their stock going to be and push it as well into that same array. This all assumes that this room already exists however if this room doesn't yet exist what we're going to do with the first socket ID. We just kind of created the room, they're the first person here so we're going to go ahead and take their ID and put it into a newly created array at this particular room. We'd be inside of the rooms object finally and what we're then going to go ahead and trying to find the other user so basically now what we want to do is check to see if there is already somebody else within this room. In other words, currently joining or our user be this current and joining right so the way that we're going to do this is we're pretty much going to go to the room of the room ID right to know they're going to go to this particular room at the inside of the rooms collection and they're going to go to that array, perform a simple find and the way that this file is going to work as they're pretty much going to try to see if there’s an ID inside of this array. That's not my own right so the assumption here is that since at any given point in time there's only going to be two people within a room because we're only just having a bunch of
one-on-one conversations happening but at any given point in time only two IDs will exist within one array and so then the basic idea is so long as we find an ID that is not equal to our own. That means if someone else is there, you can actually go ahead and get that ID of the other person because we are going to know that someone else is here and we need to know who we are not trying to call by sending our offer to them. So what we're going to do here is we're going to have a basic if check, if other user does in fact exist, we're going to omit an event back to ourselves with the events called other user which basically is telling ourselves that yes, there is another user right here and this iscthat users ID and then to the other

user, the one who’s already here, we're going to be telling them by the way somebody else's joint and this is their user or this is the user ID so basically what's happening now is when user B joins the room where user a already is, what's not happening is user people kind of get notified that user A exists, and then usually be able to get user A’s ID and then the the flip side of that user A who has now been sitting in the room waiting for somebody else and join get notified that somebody did in fact join and will also be notified of what that other person user ID actually is well suffer all this code pretty much does handles a logic.

Handshaking between Pairs

Now that we've actually enjoined in the room, we need to start reading the logic of what it takes to actually facilitate the handshake between the two pairs. There was the actual process of sending an offer, we're sitting an offer and so on and so forth so what we've just done basically is we've created this event called offer basically saying socket did not offer when the offer event gets fired it's going to be accepting this payload as an argument and we're pretty much going to be calling on our i/o object. We're going to say go ahead and send it an event. Sending an event to payload that target right so now there's a person that we’re trying to call. So in other words, we are user B and we are trying to call user A. We’re going to be sending an event to use A’s ID, so this target over here will represent the socket idea of the person we’re trying to send an event to, We’re going to be meeting the event called offer with the payload. Now this payload would pretty much include two very important things the first one is Who am I, as the caller as well as the actual sort of offer objects.

In other words when you're actually going to create an offer, some sort of arbitrary data sounds like that arbitrary. It's actually quite meaningful to ever RDC but us as developers don't need to really worry about too much about what the data actually is, we just need to know that given this data we have to actually go ahead and send it off to the other user. So, this data now will be what we're going to call the offer, so effectively we're going to be taking the offer and then sending it to the other. so again, payload will include actually Who am I, the caller, our ID, as well as the offer that I'm not trying to send to the other user.

Answer Event

So now what we've done is we've pretty much created another event called answer. The answer event will pretty much happen when user A is calling user B and now user B's actually answering, so now we're going to be basically sending back to use your ain't event. Think by the way you're receiving an answer and so the basic idea is we have not received the offer and we are not going to go ahead and formulate our answer. So just like the offer is some sort of bid up there that needs to get sent to the other peer, the answer is another bit of data that has to get sent back to the original peer that it's been trying to call so we're going to be listening out for this answer event and then for this answer event, we're pretty much going to be getting this payload argument. This payload object is going to include the target of who we're trying to send the event to and in the actual payload which is of course going to include the actual answer data that we need to actually send back to the calling peer.

Now finally the last thing that needs to happen in our server-side code is they're pretty much going to have this event called ice candidate so very briefly without kind of getting into too many details about what a nice candidate is when the two pairs are trying to sort of have a connection between each other of course and I realized snare there's going to be firewalls and we're also going to have to figure out like which IP address would try to connect to and sort of low-level networking stuff. Now there's a concept known as ice servers which basically is either a stun server or a turn server and these are basically responsible for trying to scrape with known as an ice candidate then the sort of most basic definition. It's just a way for sure to peers to kind of agree upon a certain proper connection that they can both agree on that's going to work for them and so each sort of peer will kind of come up with a bunch of different candidates until they have exhausted all their options until they finally reach a sort of mutual agreement about which can that kind of make sense for them and so then what's going to happen is this event will basically get us best sort of both peers because basically peer A will come up with the candidates send at the peer B. B will kind of come up with its own kin and send them back to peer A at some point in time. They're pretty much going to come to some kind of conclusion about which kind of makes sense for them and then they're going to pretty much finish their handshake and have a proper connection and that pretty much includes all the server-side code. We can now go ahead and jump into the actual client-side code, so one of the first things we're going to want to do in the client-side now is we're going to want to go to our source folder and inside of that we're going to want to create a new folder called routes.

<u>Creating Rooms</u>

Now with interests, we pretty much want to create two files, the first one is going to be called room bjs and then the second one's going to be called to preach at DJs. Let's first take a look at what create room needs to do, so in create room we are importing react which is going to be used to sort of generate the unique ID fo the individual room. we’re then creating a component called create room which is going

to export the component. It will define a function called create. Within create, we’re pretty much creating an ID by calling the wit function and then we’re going to go and navigate over to the room ID. So in the actual room the sort of local O's, actual conversations are going to basically be happening so any two people that find themselves under that room ID is going to represent in the actual room that they are both tied to and that therefore if they're both within that same room at the same time they can actually have a conversation with each other. So basically what's going to happen is if any one person wants to kind of go and create a new room, they can then take that URL to share to somebody els. The first twins are going to want to be met with create room when they click on that button they're going to go ahead and call this function create create an ID and then they're going to navigate over to sort of room where the actual chats going to happen. They're going to take that URL, send it to some printer. The one actually have a chat with when that person actually joins that same URL. They're now both within the same room and then this sort of signaling process of actual WebRTC handshaking will start happening between their peers and within a few seconds they'll be in an actual video chat application. If it all goes well, this now bring us in the actual room file where the actual video chat application logic is going to actually exist. peer rep the soccer riff of the user and user stream all these reps are going to get used and as we start to kind of build up.
We're going to build up our JSX where we’ll be returning to video tags. One video tag is referencing our use a video ref and then the other video tag is referencing our partner video tag video ref and the basic idea is we kind of want to have one video tag to display our own video and then one other video tag display our partner video and we're going to basically be using the refs to kind of attach the stream which we're soon going to see how this is actually done to the individual ref so that we can

actually start displaying our own video as well as our partners video on the screen. Now we’ll be defining this use effect over here and as you can see by fact we have the empty array here.. For the dependency array, we're pretty much treating this use effect as a basic did mount and then what we're doing is asking the browser to grant us access to the users video as well as their audio so when the browser is going to go ahead and ask you to kind of give us access. When the users can actually hit allow then this promise is going to go ahead and resolve and then when it resolves, we're actually going to get this stream object that's going to actually include both the audio and the video because that's what we actually went ahead and asked. Then we're going to do take the stream and attach it to the source object of our user video ref. That's going to allow us to actually go ahead and display the actual video of ourselves in the actual video tag in the browser and then finally we'll be just taking this sort of stream and attaching user stream wrap over here for later use because we might need it in the future. We are going to connect it to our socket IO server and then in middle e apps and emit an event back down to the server saying that we're trying to join the room and if we come back to the server we have the event that server's listening out on. It's called the join room so this is where we're going to actually emit that event and then we're going to be passed now in the room ID by pulling it up at the URL by saying props that matched up params that room ID. That’s the sort of react route or Dom way we've actually pulling out parameters from the URL. We're going to grab the room ID out and then send it down to the server. Server can go ahead and put us in to a specific room within the sort of rooms we kind of have created within the server. So now we have this other user event that the server is going to either fire back to us or it won't depending if somebody else is already here right so in other words in this case let's imagine that user A already is in this room and now you should be in the one that's joining so in the case you should be the one that's joining this other user event is going to impact fire and what we're going to get as usual are these going to be user A’s IDs so you should be the one that's currently joining. User A already is in the room so when user B joins the server we'll go ahead and send them back in messaging by the way this is the other user and this is their ID and then immediately we're going to go ahead and call the function call to call user which of course doesn't exist yet we're going to fill that function in very soon and then what it does is it actually takes us here and pass it down to the call user function. Also, we are going to take this user ID that we just got and we're going to throw it in the other user ref because we're going to need it for safekeeping for a little bit later.

SocketIO events

Three more events that we're going to be listening out for within our socket IO connection here and use effect so the first one is going to be the offer right. When we are in fact receiving an offer,that's going to go ahead and call the handler receive call function which of course doesn't exist yet we're going to fill that in very soon then what happens when we're getting we're going to be getting an answer rights. In other words, we received an offer and then we're studying that we're sending an answer back. So now the person who was the one that made the original call is now getting an answer in return so they have to be none for that answer. This event is going to get fired and it's going to call the handle answer function which we are not noticing is and then finally we have the ice candidate events. Different peers will be kind of exchanging ice candidates with each other so they're both going to be listening for this ice candidate event and then when this event gets fired, we're going to call the handle ice candidate message function, but for now we'll see what does this call user function actually do. What this function is really doing it's accepting these are at needs and arguments which are passing in right over here. We're then going to say PRF that occurrence we do in fact have a ref that we're calling PRF that's going to pretty much be equal to whatever this creates. Pure function returns, so obviously this function doesn't exist yet they're going to have to go build it up but suffice it to say this is the one it's going to actually build an actual WebRTC pure object and then we're going to return that from within the function which we're then it going to store inside the PRF ref.

The next thing it’s going to do is it's going to go to our stream and as we actually took the stream and stored it inside of this youth stream ref for safekeeping therefore using it later is one such instance we're going to be using that actual ref and then we’re going to call the get Trax method that exists on the stream that's really going to return an array of all the tracks that we have on this particular stream. In this case it's going to be an array of two because we've got one track for the video, one track for the audio. We're going to call the add track method that exists on our peer object. The add check method basically is a functional but except it is going to be the individual track as well as the overall stream that this track is included as part of right and the reason why we're doing this is pretty much taking our stream and attaching it to our peer because once we actually have the sort of peers both connected to one another, we want our pair to actually be able to have the capability of sending our stream to the person, we're trying to have a video chat with. In order to do that, we have to sort of give our peer sort of access to our individual stream. That's what you can kind of think of this as that'll have actually seen the create peer function. We’ll build that up and see what that entails. We've got to create pure function which is accepting user ID as an argument. This user ID represents the person who are trying to call so we are the ones who are currently making the initial call, so represents the person is going to be receiving our offer and then all we're doing is creating this pair object by calling the peer connection constructor and they're pretty much the only configuration knobs that we're passing into. It is going to be the leads ice servers so we're pretty much going to be the ID servers. Just an array of objects we're going to be setting up one for the stun one for the turn. Turn servers will allow us to kind of figure out a proper path for our actual peer connections to be able to connect with one another and in fact these are the ones they're responsible for actually generating. These ice candidates are pretty much going to be getting sent from one pair to the other pair so they can come up with some sort of agreement on what they can actually use as a proper connection method and start streaming data between each other back and forth without the actual use of a server. Then we’ll be attaching three event handlers on this peer object. Their person won't be on ice candidate so

<u>**Browsing through Candidates**</u>

whenever the browser with the fad decide that it wants to go ahead and send, another ice candidate is going to go ahead and raise his event in which case where it's going to go ahead and call our functions. We're pretty much going to have a function called handle ice candidate event then we're going to have another event called on track this basically will represent whenever we are receiving the remote appear so they finally have a proper connection with another peer and we have a video chat application happening.

So now the new remote peer sending us their extreme so we grab the stream and display it on-screen. It's going to have an inside handle track event function so when we're actually receiving a remote stream, the on track event will fire it and then call our handle track event and like this is where we can actually go ahead and grab that stream and then attach it to the partner video ref which we have already defined. You can see in JSX that we have the partner video where there is going to be a video tag. That's where this is going to get populated and then finally the last event that we're going to be listening out for is going to be on negotiation. We needed this z the target of the person who are trying to send our offer. We're going to send their own ID as well by pulling it out the sock arrest. The ID that's going to represent the caller who is actually calling you and then the ID represents the actual offer data which were then going to go ahead and call the offer event passing in this actual payload, which once again includes who we are as well as what our actual offer entails and that should any of that fail, we're in a pretty much come down and just simply do a console.log.

Processing UID:

So now, as we've pretty much seen what it takes to make the call, now let's go ahead and handle what happens when actually we're receiving a cost as you can see here I mean we already hadn't have an offer. Let's go ahead and call the handler receive call function and fill in that particular function of what happens when you are in fact receiving a call. Tthere you can see we have the receive call function, and it's accepting this incoming payload as an argument and then the first thing we're doing is say peer reps at current and call create peer function for these sort of receiving peer so now we are not the initiator. We now are the ones who are receiving the call so we're just going to go ahead and call crate peer without passing the UID because obviously we're not the ones that are initiating the call so we're not sending out an opportunity to anybody. So there's no need to pass in the user ID we're then you're going to go ahead and create a description object here by passing the incoming data SDP. The SDP represents our actual offered data so we're going to go ahead and call the RTC session description of constructor passing in. This SDP objec,t the one that we've got at the offer is going to create the sort of remote description and so as you can see here are pretty much then I'm going to go and remote description whenever you have an offer the offer that's local to you then set local description or the answer and then whatever you are receiving from the sort of remote peer you then set as q remote description so in this case, promise resolves a reference back to our theme and we're going to go ahead and do the same thing. We're going to just attach our tracks to our appear so remember in this case we're now actually not the caller, we are now actually the ones who are the ones who are receiving the call, so in this case we have to actually also go ahead to our stream, take our stream and attach it to our peer so that our peer can take our stream and send it back to the person who we're trying to have a chat with then finally we're going to create the answer. We are going to be handle what happens when you are in fact receiving an answer from the person who you have called as we actually do have an event set that says stack a ref that occurred that on answer when you're actually receiving an answer. We are going to call the handle answer function and populate the handle answer function. Now we’ll have a function called handle answer which will pretty much just accept this message. This message would represent the sort of did that were getting back from the person who we called and as you can see we're once again calling the RTC session description constructor passing the message that SCP in this case, the message that FTP will not be an offer rather it's going to be an answer but nonetheless we're still using it to kind of create this description object which were then going to pretty much as path to our set remote description function on our own peer and this pretty much completes the sort of handshake cycle right because we pretty much have sent an offer out to the person that we're trying to have a phone call with and they have been pretty much taken that answer. They set it as their route both description and they have now in turn cribbed their answer which they have been set up their local description and then also sent it back to us.

3. SYSTEM DEVELOPMENT

The method involved with characterizing, making, testing, and executing another product application or program is known as frameworks advancement. Inside improvement of custom-tailored frameworks, information base framework creation, or the acquisition of outsider produced programming are generally conceivable outcomes. All data frameworks should be directed by composed guidelines and methods. The management of the organization must create and implement standards, as well as adopt an acceptable strategy.

The process of creating, acquiring, implementing, and maintaining a system is governed by the system development life cycle methodology. Computerized information systems and related technologies must be maintained.

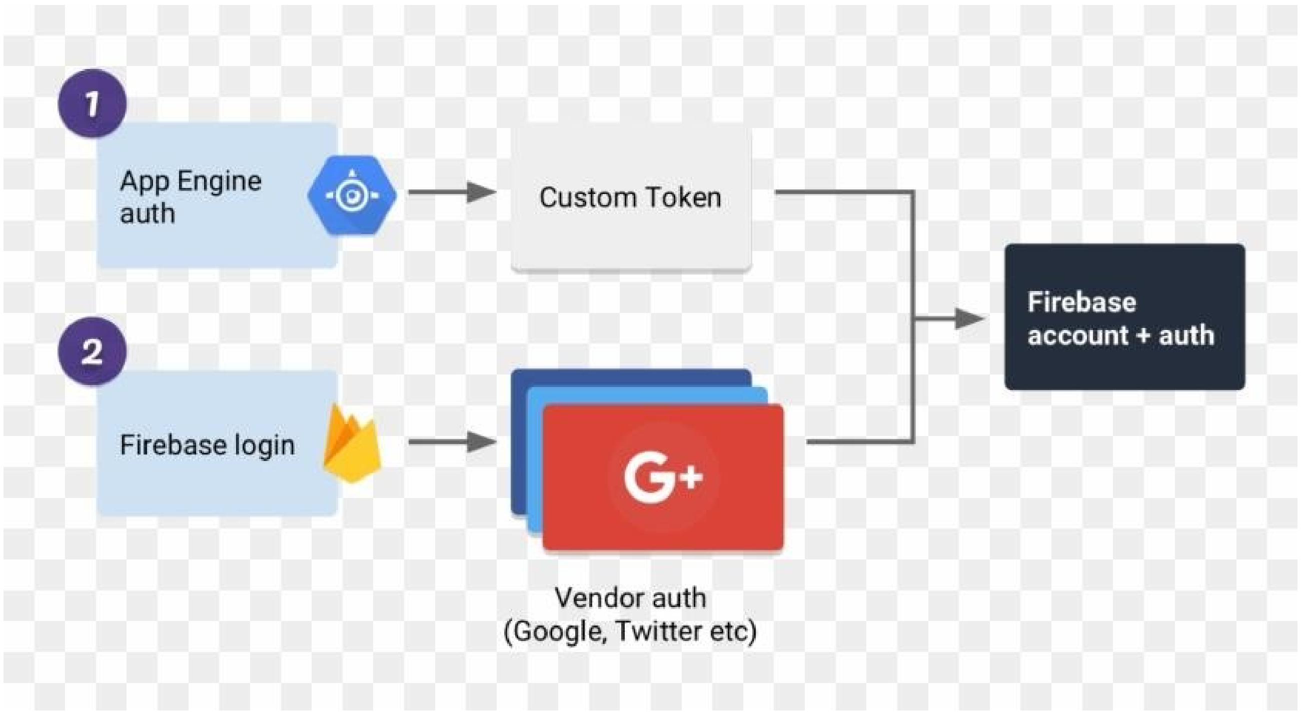


Fig 3.1 FCM Authorization

3.1 Use-Case Diagram

A use case diagram can help you outline the details of your system's users (also known as actors) and how they interact with it. You'll need a collection of specialized symbols and connectors to construct one. A good use case diagram can assist your team in discussing and representing:

1. Interactions between your system or application and people, organizations, or external systems

2. Goals that your system or application assists those entities (sometimes referred to as actors) in achieving

3. Your system's capabilities

A utilization case outline's primary objective is to portray a framework's dynamic viewpoint. It gathers the prerequisites of the framework, which incorporate both inside and outer variables. It alludes to individuals, use cases, and an assortment of different things that allude to the entertainers and variables answerable for use case chart execution. It portrays how an element from the rest of the world can connect with a framework part.

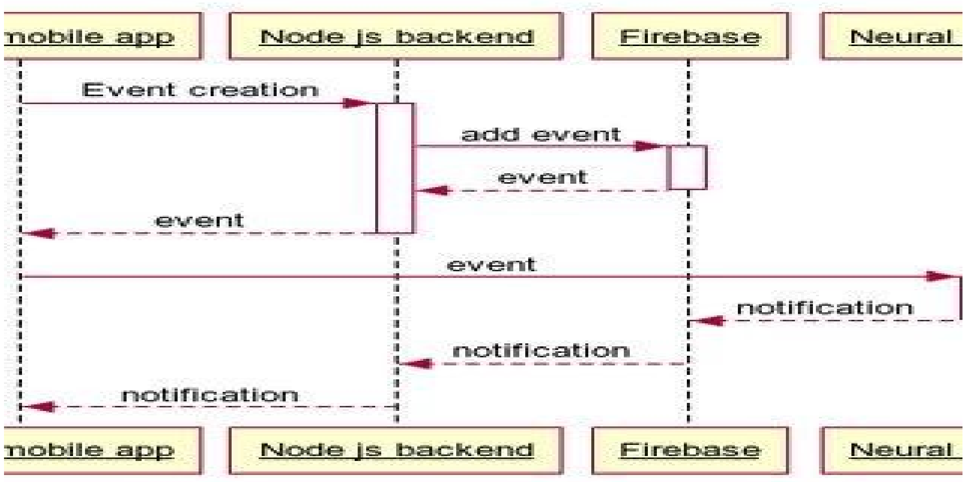


Fig 3.2 FCM notifications

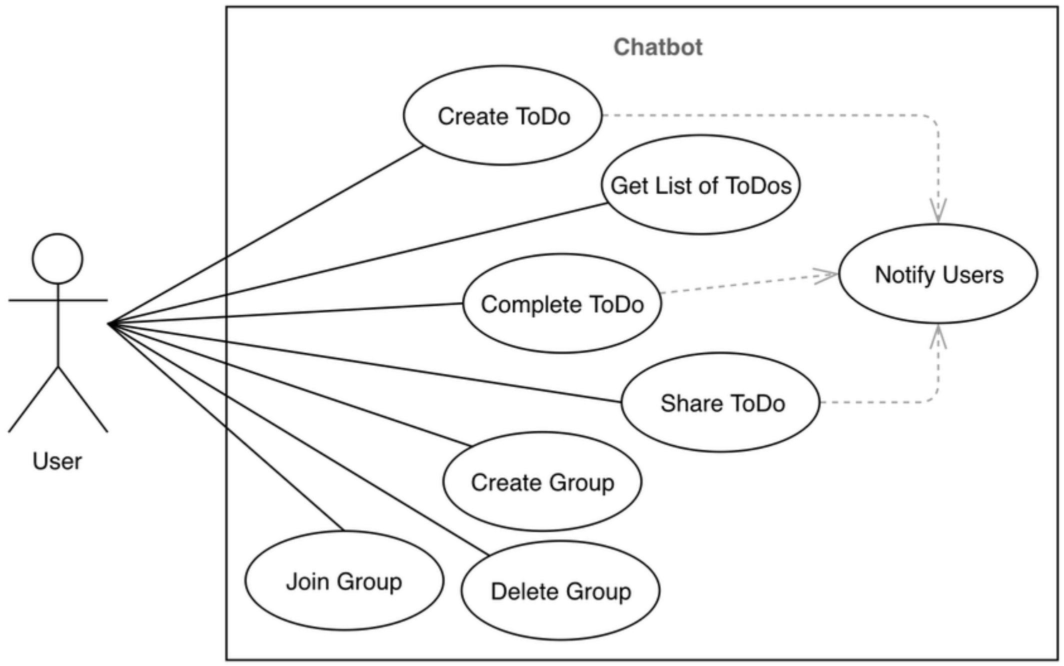


Fig 3.3 Use-Case Diagram

3.2 State Transition Diagram

State-progress graphs show each of the states that a thing can have, just as the occasions that make it change state (advances), the prerequisites that should be met before the change can happen (monitors), and the exercises that happen over the article's lifetime (activities). State-progress graphs are amazingly valuable for delineating the conduct of specific items over an expansive scope of utilization situations. State-change outlines are ineffectual at portraying the cooperation between the things that create advances.

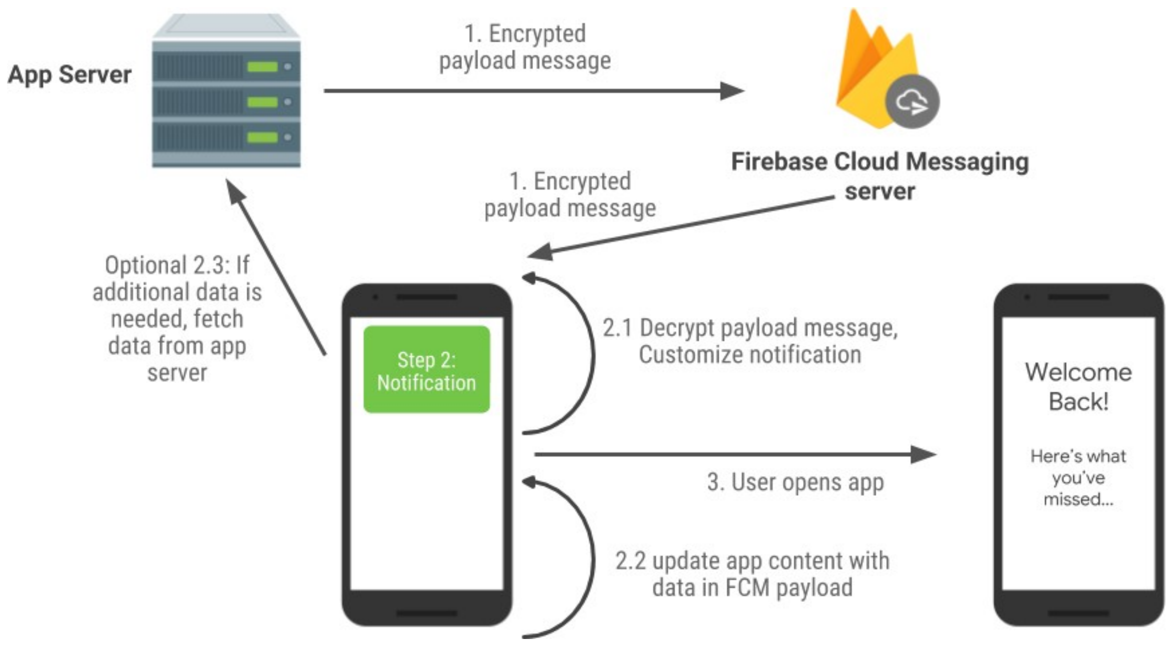
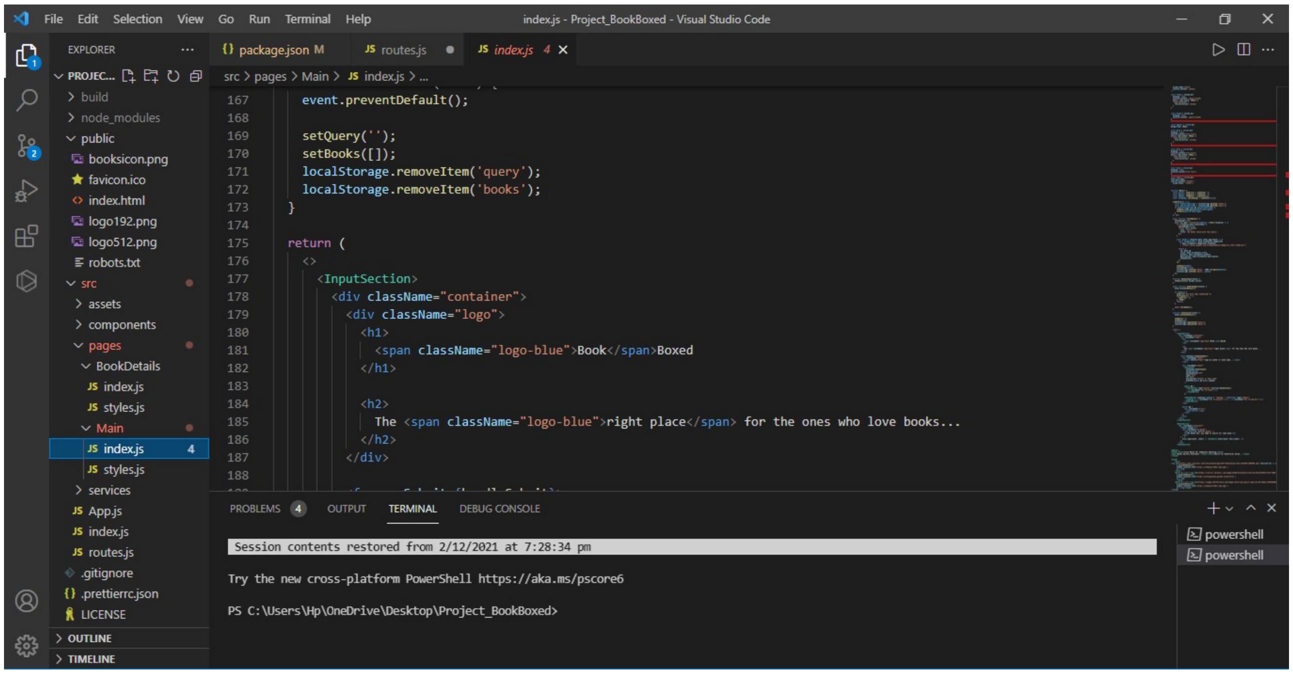

App Server
1. Encrypted payload message
Firebase Cloud Messaging server
1. Encrypted payload message
Optional 2.3: If additional data is needed, fetch data from app server
Step 2: Notification
2.1 Decrypt payload message, Customize notification
Welcome Back!
Here's what you've missed...
3. User opens app
2.2 update app content with data in FCM payload

Fig 3.4 State Transition Diagram

3.3 <u>React.js as a Front-end</u>

The most important advantages of ReactJS for front-end development:

1. React makes it easier to create sophisticated user interfaces that can be quickly tested.
2. The learning curve for React is steep.
3. React.js is extremely adaptable.
4 . React.js is a fast-rendering framework.



```
    event.preventDefault();

    setQuery('');
    setBooks([]);
    localStorage.removeItem('query');
    localStorage.removeItem('books');
  }

  return (
    <>
      <InputSection>
        <div className="container">
          <div className="logo">
            <h1>
              <span className="logo-blue">Book</span>Boxed
            </h1>

            <h2>
              The <span className="logo-blue">right place</span> for the ones who love books...
            </h2>
          </div>
```

```
Session contents restored from 2/12/2021 at 7:28:34 pm

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Hp\OneDrive\Desktop\Project_BookBoxed>
```

Fig 3.5 React Front-end Code Snippet

A growing range of front-end development tools is available. Some of these tools will be more appropriate for your project than others, which is understandable. For some, a simple frontend built using HTML, CSS, and JavaScript will sufficient. However, as the programme grows, you'll need more sophisticated tools — frameworks and libraries – to create complicated user interfaces. As a result, knowing the most up-to-date industry trends is critical when dealing with such a huge and continuously changing ecosystem.

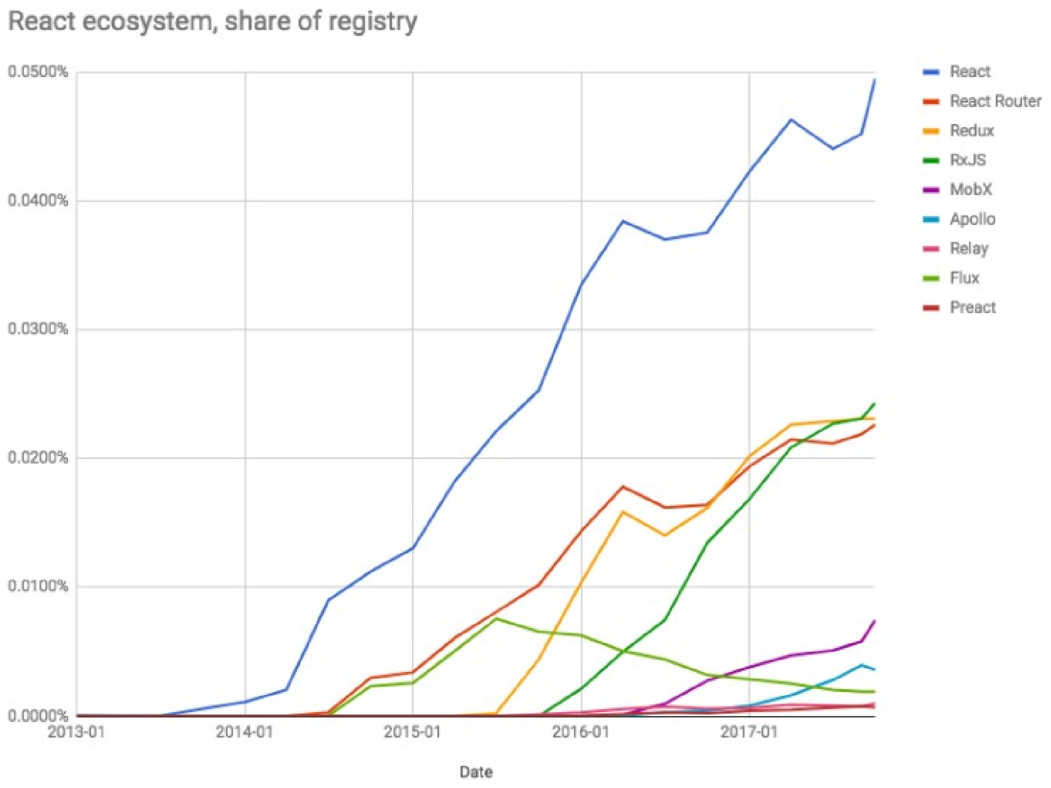


Fig 3.6 React Growth Trend

3.4 <u>Firebase as a Back-end/Database</u>

Firebase Cloud Messaging. can be used to send messages consistently and for free. It allows us to keep our app subscribers engaged with providing thm contextually relevant text that make them to use major featurs. Messages can be sent using the Firebase Admin SDK or FCM server protocols. The Notifications composer can be used for testing as well as sending marketing or engagement messages utilizing robust built-in targeting and analytics, as well as custom imported segments.

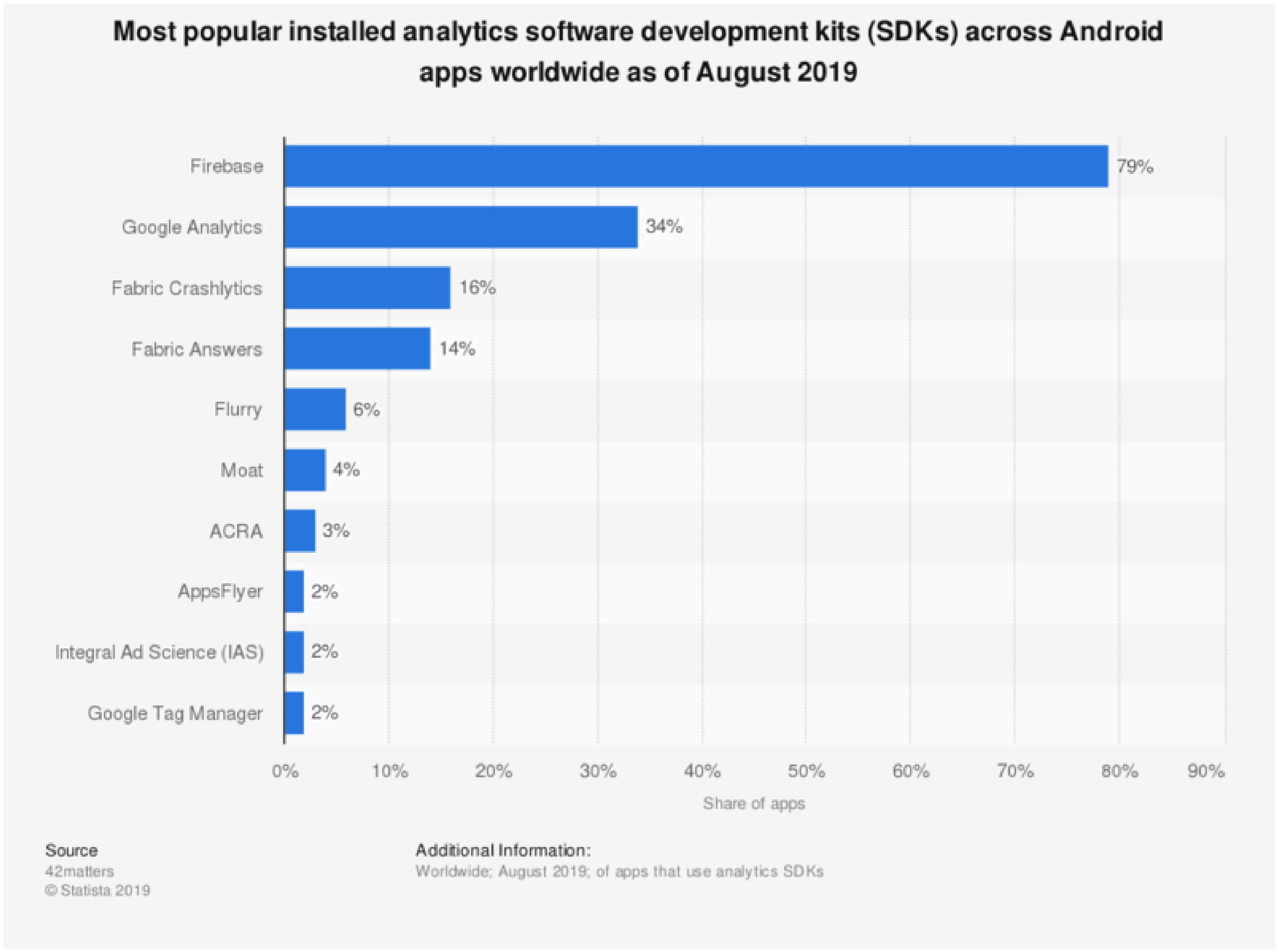


Fig 3.7 Firebase Growth Trend

Firebase cloud messaging is almost similar to google cloud messaging, difference is that messages cross through firebase cloud messaging servers, but in the case of Google cloud messaging, it is not the same. Nowadays, most of the developers have changed their preference as firebase cloud messaging as it is much easier to use and it serves a purpose of showing the notification from the client application.

As a result, when choosing a messaging approach, it's no longer enough to consider how much energy is consumed; it's also necessary to consider message delivery time, which affects application reaction time. We built up a test bed employing mobile devices to measure one-way delay, round trip duration, and data usage in order to solve this problem.

4. Performance Analysis

Traces are used in Performance Monitoring to collect data about these processes. A trace is a report that contains data collected in your programme between two points in time.

Metrics are the performance data collected for each trace and vary based on the type of trace. When an instance of your app makes a network request, for example, the trace captures information like response time and payload size that are crucial for network request monitoring.

When an instance of your app runs a monitored process, the trace for that app instance automatically captures characteristics data. When an Android app makes a network request, for example, the trace records the device, app version, and other information for that unique app instance. These attributes can be used to filter your performance statistics and see if specific user segments are having problems.

Inactivity, application size, the measure of DOM hubs, the quantity of asset demands made, JavaScript execution, CPU load, and different factors all affect execution. Making the experience as accessible and intuitive as doable,

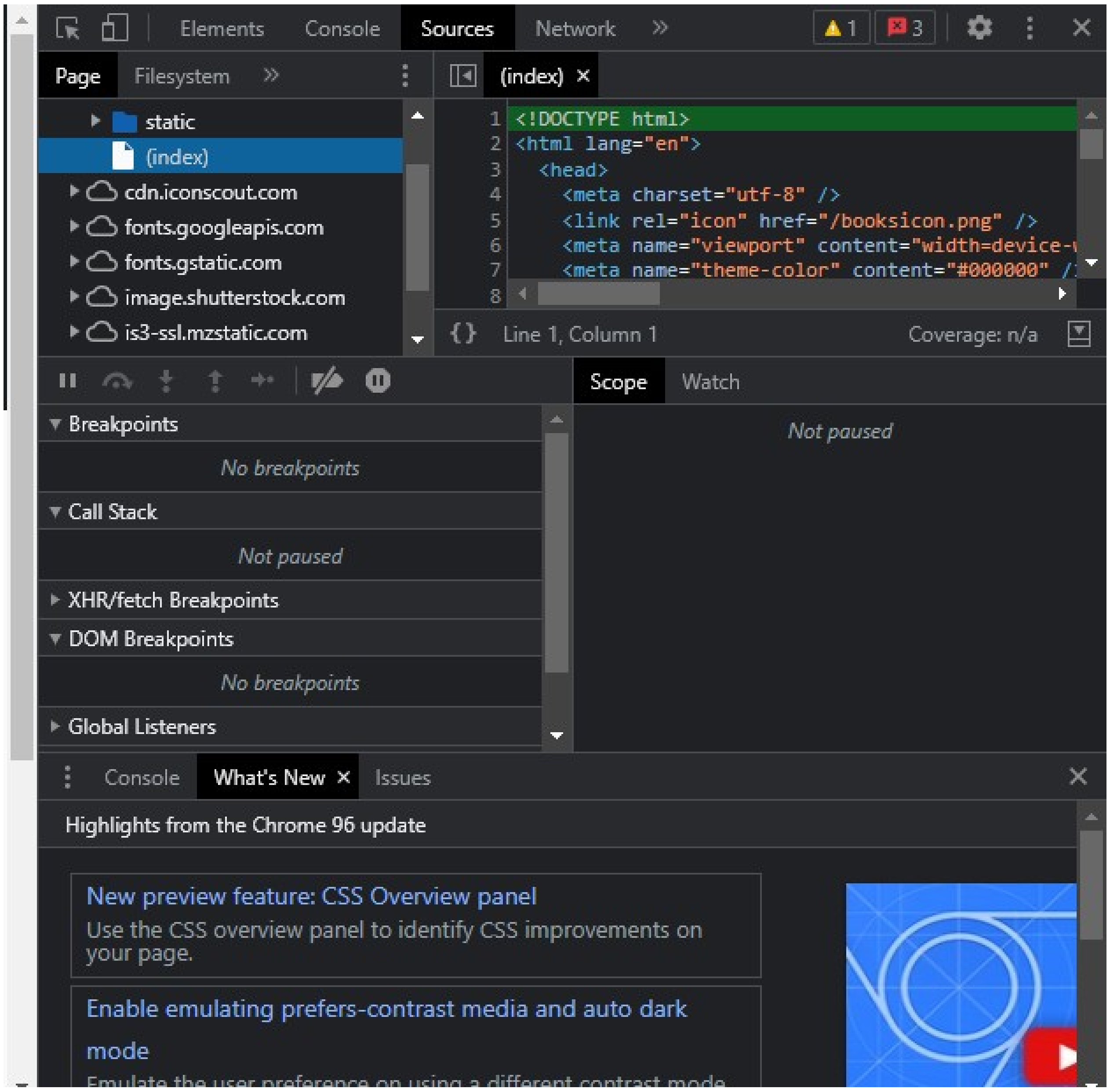
as quick as could be expected, while non-concurrently stacking in the more drawn out tail components of the experience, while limiting stacking and response times, and adding extra elements to cover inactivity, is basic.

The goal of online performance is to make websites as quick as possible, which includes making slow processes appear to be faster. Does the site load quickly, allowing the user to begin interacting with it right away, and provide reassuring feedback if something takes a long time to load?

We've grown accustomed to great website performance over time. This puts small-time website owners at a disadvantage: when someone visits your page for the first time, they're comparing it to the average of all the websites they've visited, even big-name brands with entire teams dedicated to performance optimization.

As a result, when choosing a messaging approach, it's no longer enough to consider how much energy is consumed; it's also necessary to consider message delivery time, which affects application reaction time. We built up a test bed employing mobile devices to measure one-way delay, round trip duration, and data usage in order to solve this problem.

The goal of online performance is to make websites as quick as possible, which includes making slow processes appear to be faster. Does the site load quickly, allowing the user to begin interacting with it right away, and provide reassuring feedback if something takes a long time to load?



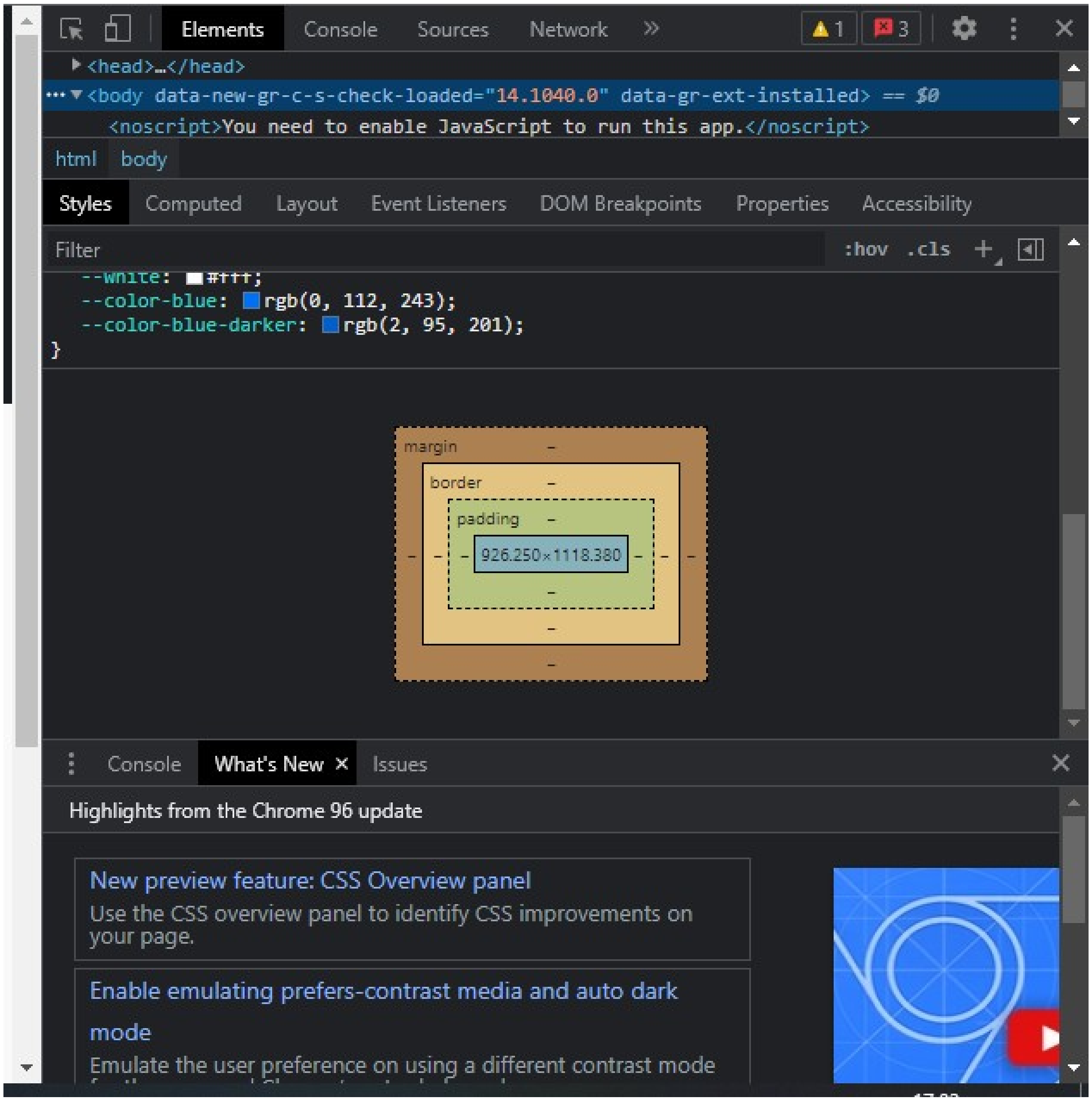
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="/booksicon.png" />
    <meta name="viewport" content="width=device-
    <meta name="theme-color" content="#000000" /
```

Highlights from the Chrome 96 update

New preview feature: CSS Overview panel
Use the CSS overview panel to identify CSS improvements on your page.

Enable emulating prefers-contrast media and auto dark mode
Emulate the user preference on using a different contrast mode

Fig 4.1 Console Output



```
▶<head>…</head>
••• ▼<body data-new-gr-c-s-check-loaded="14.1040.0" data-gr-ext-installed> == $0
    <noscript>You need to enable JavaScript to run this app.</noscript>
```

```
  --white: #fff;
  --color-blue: rgb(0, 112, 243);
  --color-blue-darker: rgb(2, 95, 201);
}
```

Highlights from the Chrome 96 update

New preview feature: CSS Overview panel

Use the CSS overview panel to identify CSS improvements on your page.

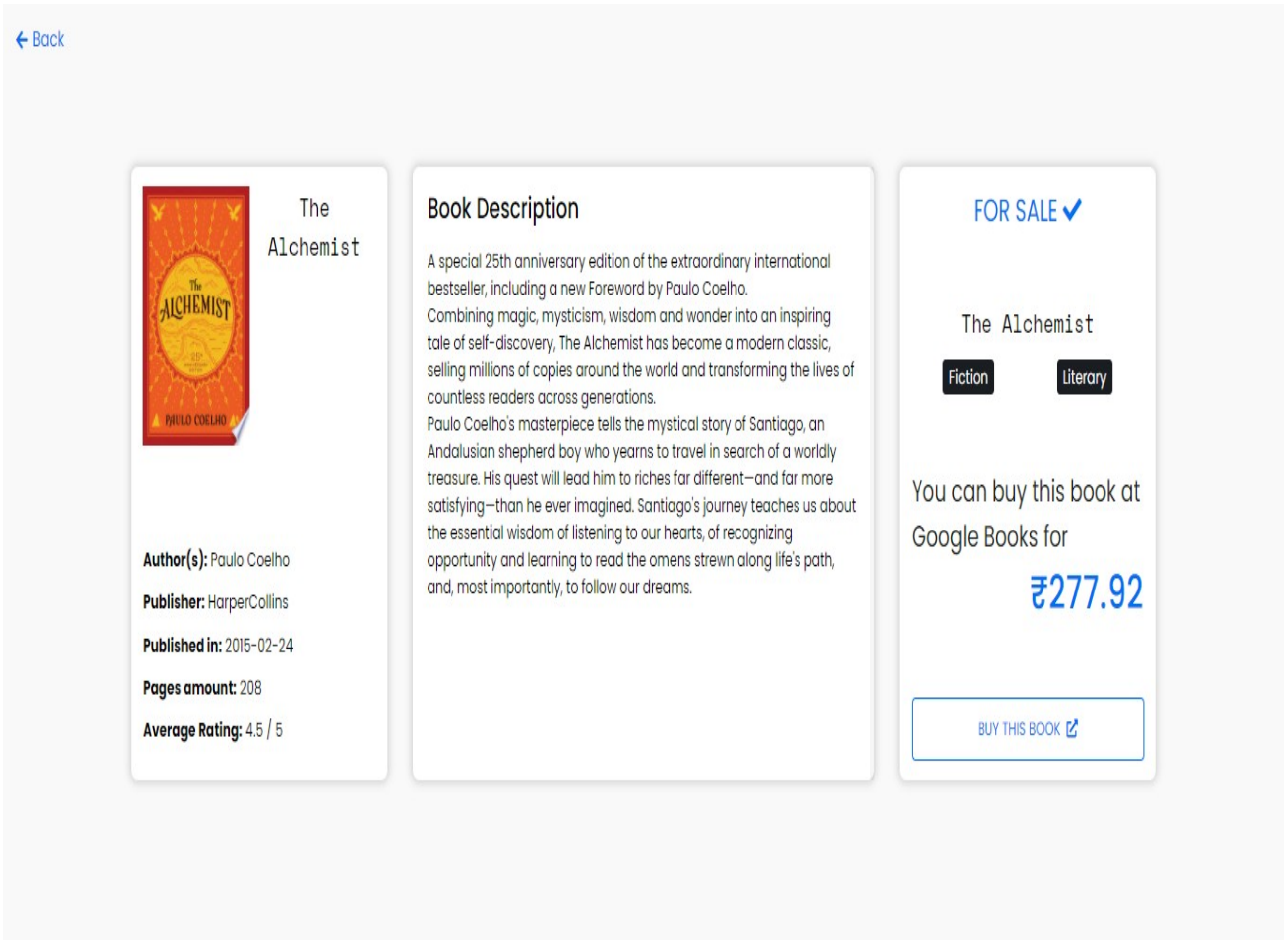
Enable emulating prefers-contrast media and auto dark mode

Emulate the user preference on using a different contrast mode

Fig 4.2 Webapp Layout

Output

1)



The Alchemist

Author(s): Paulo Coelho

Publisher: HarperCollins

Published in: 2015-02-24

Pages amount: 208

Average Rating: 4.5 / 5

Book Description

A special 25th anniversary edition of the extraordinary international bestseller, including a new Foreword by Paulo Coelho.
Combining magic, mysticism, wisdom and wonder into an inspiring tale of self-discovery, The Alchemist has become a modern classic, selling millions of copies around the world and transforming the lives of countless readers across generations.
Paulo Coelho's masterpiece tells the mystical story of Santiago, an Andalusian shepherd boy who yearns to travel in search of a worldly treasure. His quest will lead him to riches far different—and far more satisfying—than he ever imagined. Santiago's journey teaches us about the essential wisdom of listening to our hearts, of recognizing opportunity and learning to read the omens strewn along life's path, and, most importantly, to follow our dreams.

FOR SALE ✓

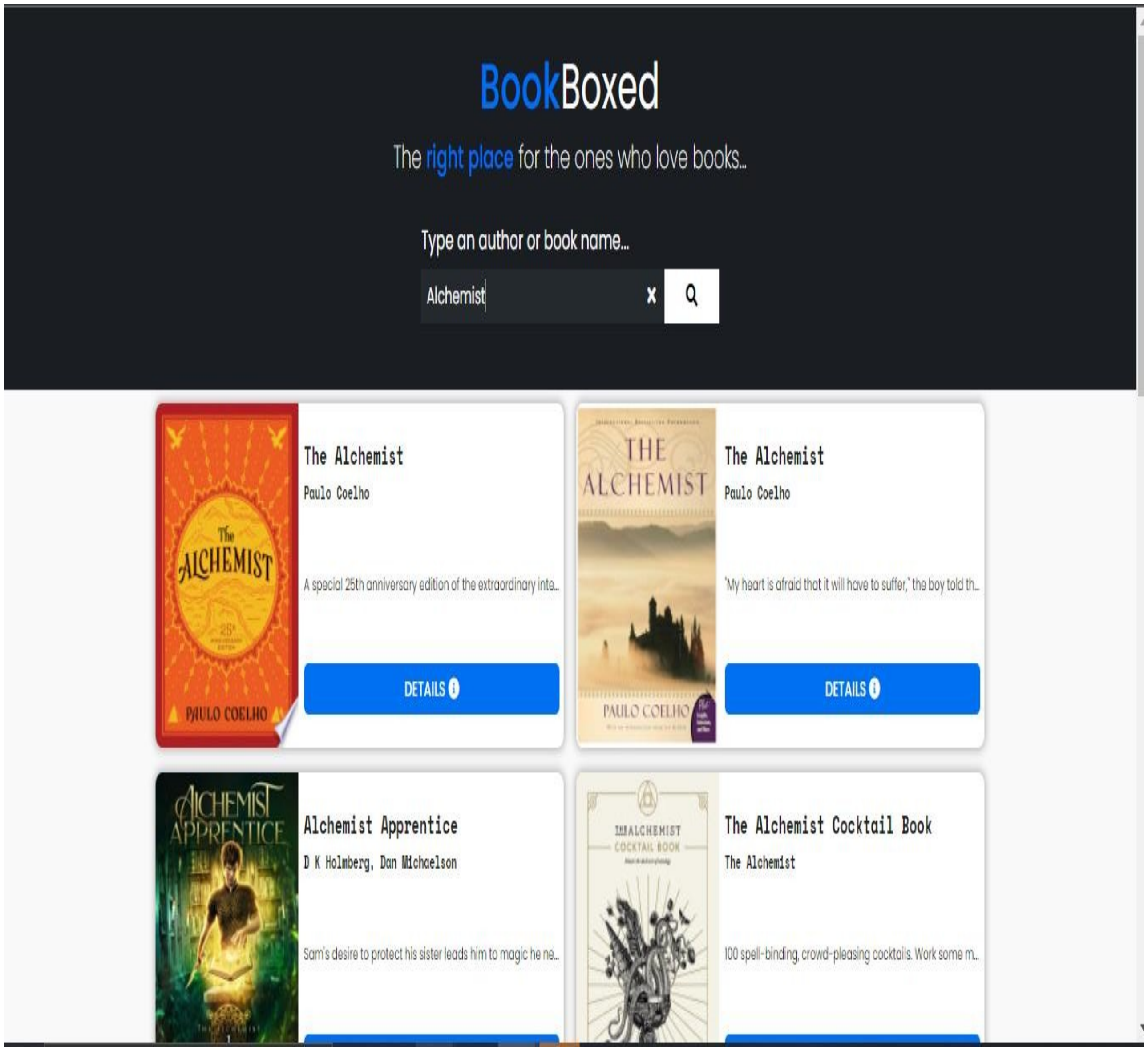
The Alchemist

Fiction Literary

You can buy this book at Google Books for

₹277.92

Fig 4.3 Book Details Output



BookBoxed
The right place for the ones who love books...
Type an author or book name...
Alchemist
The Alchemist
Paulo Coelho
A special 25th anniversary edition of the extraordinary inte...
DETAILS
The Alchemist
Paulo Coelho
'My heart is afraid that it will have to suffer,' the boy told th...
DETAILS
Alchemist Apprentice
D K Holmberg, Dan Michaelson
Sam's desire to protect his sister leads him to magic he ne...
The Alchemist Cocktail Book
The Alchemist
100 spell-binding, crowd-pleasing cocktails. Work some m...

Fig 4.4 Book Search Results

3)

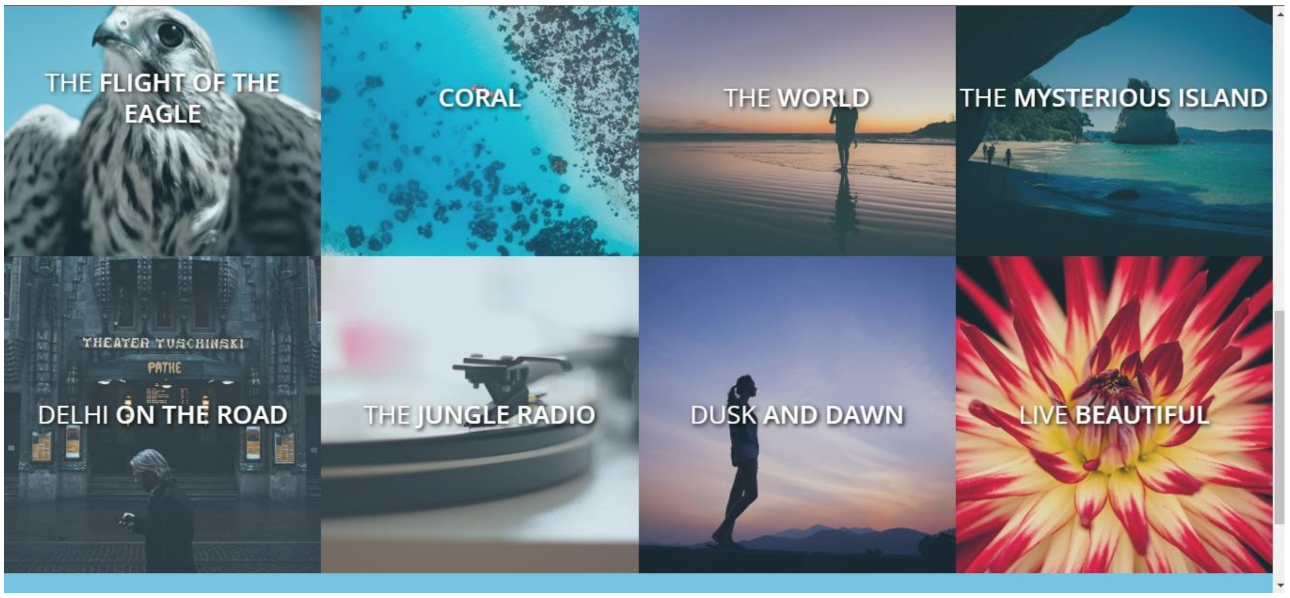


Fig 4.5 Books Gallery

4)

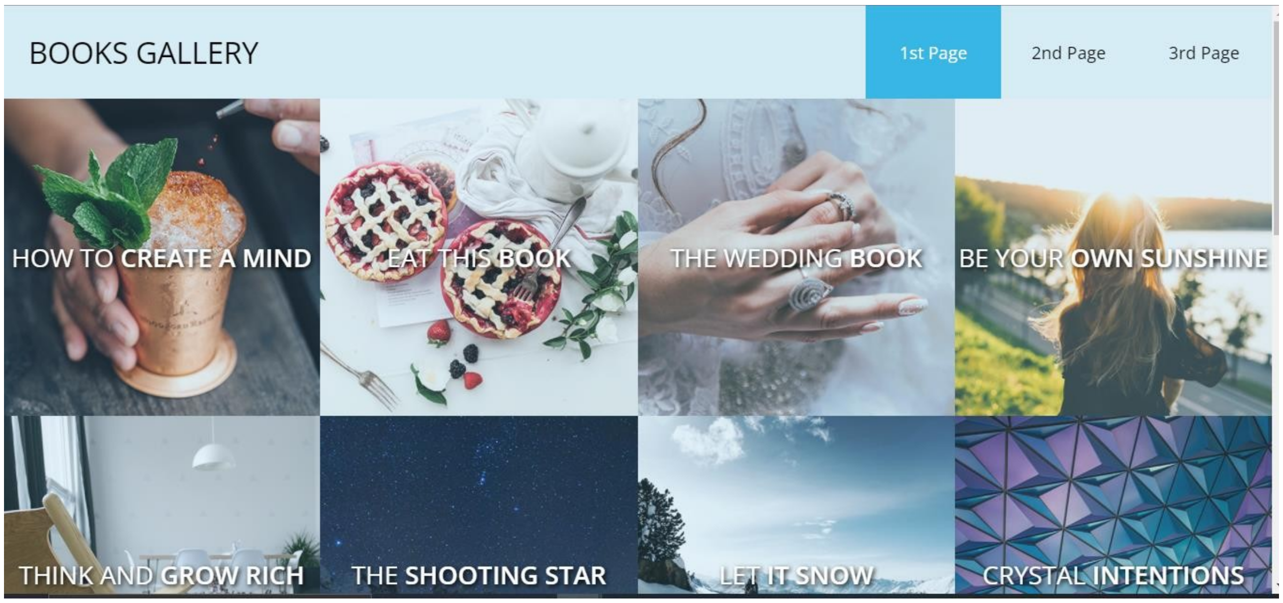
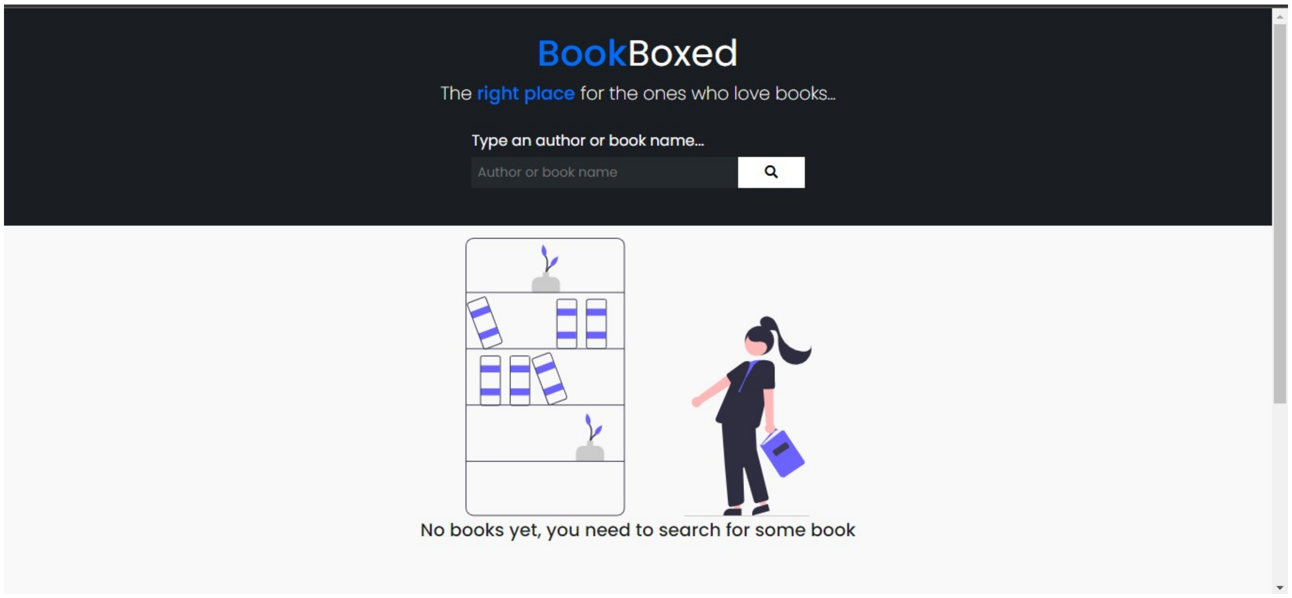


Fig 4.6 Books gallery page

5)


BookBoxed

The right place for the ones who love books...

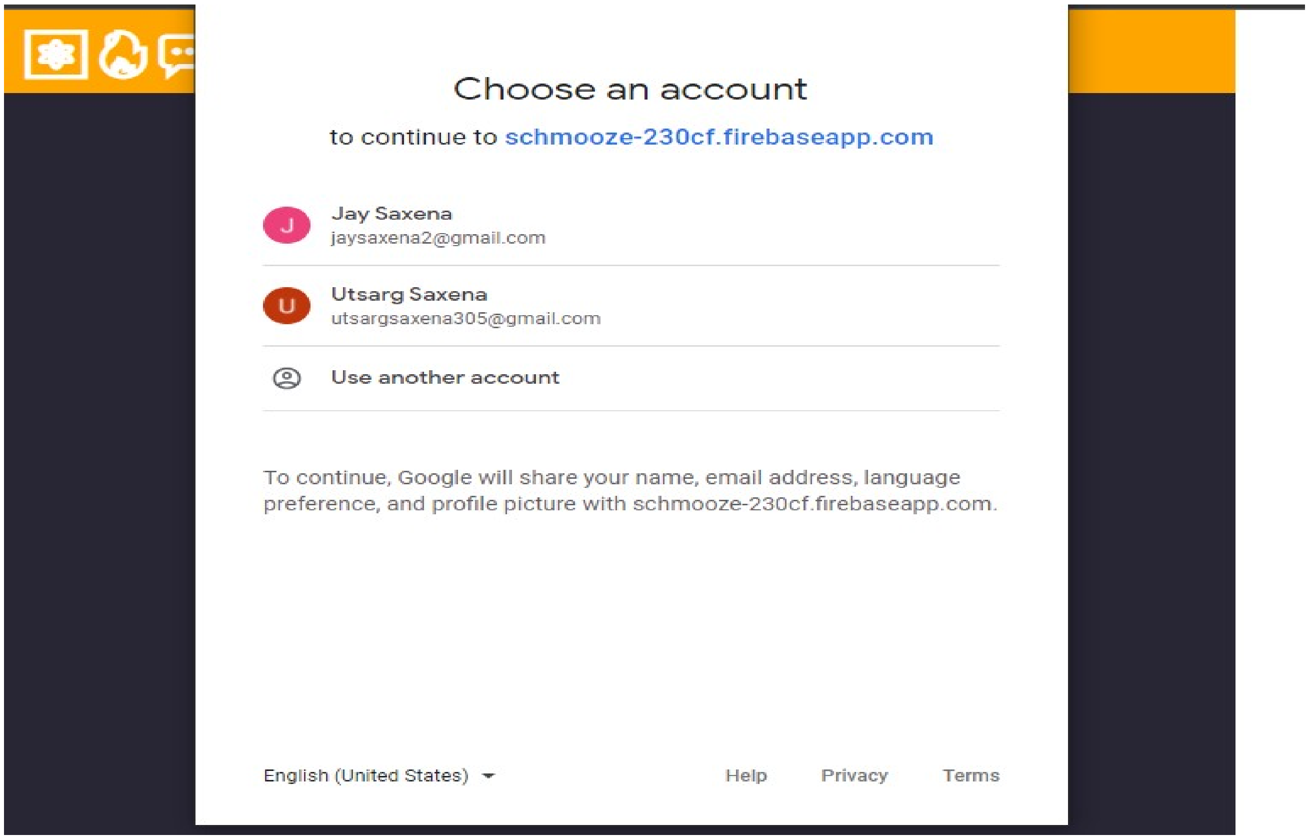
Type an author or book name...

Author or book name

No books yet, you need to search for some book

Fig 4.7 Homepage

6)



Choose an account

to continue to schmooze-230cf.firebaseapp.com

Jay Saxena
jaysaxena2@gmail.com

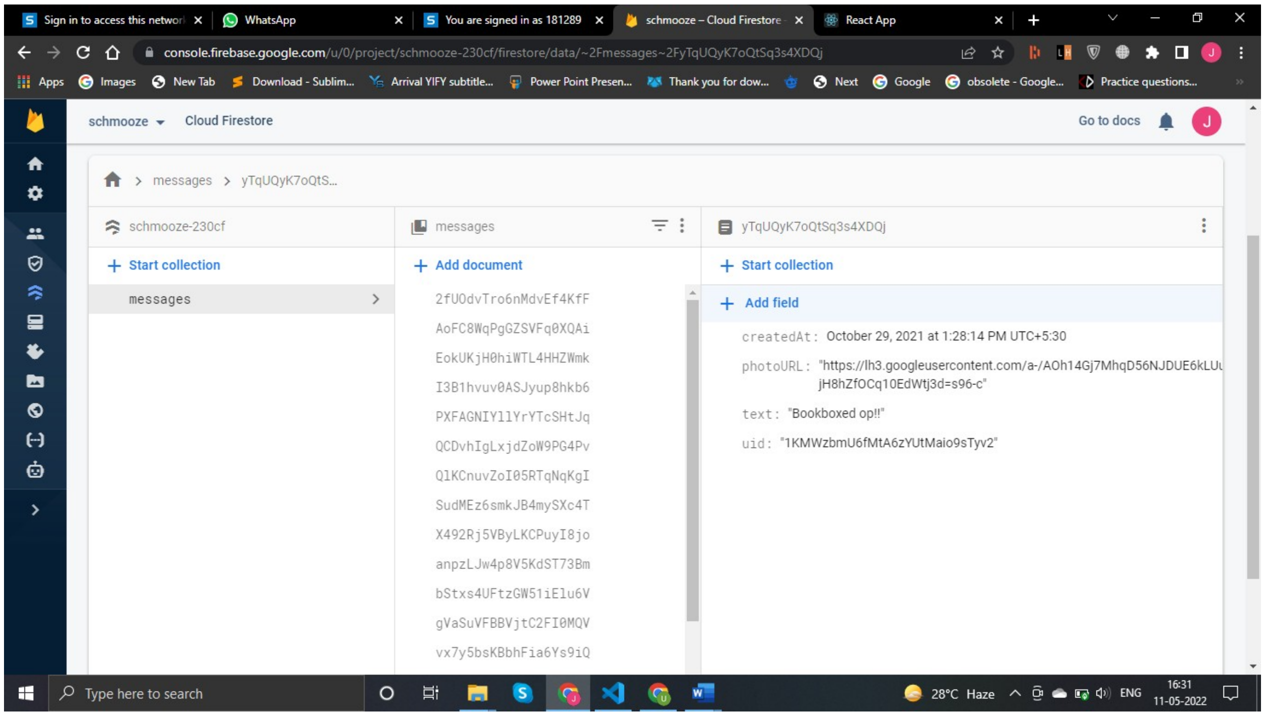
Utsarg Saxena
utsargsaxena305@gmail.com

Use another account

To continue, Google will share your name, email address, language preference, and profile picture with schmooze-230cf.firebaseapp.com.

Fig 4.8 Chat App interface

7)



```
messages

2fU0dvTro6nMdvEf4KfF
AoFC8WqPgGZSVFq0XQAi
EokUKjH0hiWTL4HHZWmk
I3B1hvuv0ASJyup8hkb6
PXFAGNIY1lYrYTcSHtJq
QCDvhIgLxjdZoW9PG4Pv
Q1KCnuvZoI05RTqNqKgI
SudMEz6smkJB4mySXc4T
X492Rj5VByLKCPuyI8jo
anpzLJw4p8V5KdST73Bm
bStxs4UFtzGW51iElu6V
gVaSuVFBBVjtC2FI0MQV
vx7y5bsKBbhFia6Ys9iQ
```

```
createdAt: October 29, 2021 at 1:28:14 PM UTC+5:30
photoURL: "https://lh3.googleusercontent.com/a-/AOh14Gj7MhqD56NJDUE6kLUu
          jH8hZfOCq10EdWtj3d=s96-c"
text: "Bookboxed op!!"
uid: "1KMWzbmU6fMtA6zYUtMaio9sTyv2"
```

Fig 4.9 Firebase Database

5. **Conclusion**

React.js serves as a front-end tool for creating Ui for the webapp, as it has reusable components i.e., Components get re-rendered instead of getting refreshed, therefore decreasing the page loading time. It also has Virtual Dom, which helps in presentation as Ui is stored in real memory and synced.

Firebase serves as a Back-end for the chatting application as it helps in creating an application without the Backend Server, A user can create Backend server without spending money, data also gets synced inside the application only in real time itself. It is mostly faster than most of the backend web services available right now, moreover it is a No SQL database, which increases the searching speed in database.

Firebase cloud messaging is almost similar to google cloud messaging, difference is that messages cross through firebase cloud messaging servers, but in the case of Google cloud messaging, it is not the same. Nowadays, most of the developers have changed their preference as firebase cloud messaging as it is much easier to use and it serves a purpose of showing the notification from the client application.

The Combination of React.js and Firebase is great as React is mainly in use to display components in Browsers, while Firebase serves the purpose of authorization, authentication and keeping store of a Real-time Database, while creating a full – stack chatting app based on Firebase Cloud messaging.

6. REFERENCES

[1] **Neha Srivastava, Uma Shree, Nupa Ram Chauhan, Dinesh Kumar Tiwari Department of CSE FGIET, Rae Bareli, India "FIREBASE CLOUD MESSAGING".**

[2] **Naimul Islam Naim ReactJS: An Open-Source JavaScript Library for Front-end Development**.

[3] **Research and Implementation of WebRTC Signaling via WebSocket-based for Real-time Multimedia Communications Cui Jian, Zhuying Lin**

[4] **WebRTC Peer to Peer Learning H. Fateh Ali Khan 1 , A. Akash 2 , R. Avinash 3 , C. Lokesh 4 1,2,3,4 Department of Information Technology, Valliammai Engineering College, Chennai, India**

[5] https://reactjs.org/docs/getting-started.html

[6] https://firebase.google.com/docs

[7] https://www.digitalocean.com/community/tutorial_series/how-to-code-in-react-js

[8] https://firebase.google.com/

[9] https://www.npmjs.com/package/firebase

[10] "Google Announces Firestore, a Document Database"

[11] ""Firebase is launching Cloud Firestore, a new document database featuring realtime sync, no-hassle scaling, and offline support"

[12] "Google acquires LaunchKit to make life easier for Android developers"

[13] https://en.wikipedia.org/wiki/Firebase#cite_ref-16

[14] https://searchmobilecomputing.techtarget.com/definition/Google-Firebase

Report

by Utsarg Saxena and Anurag Sharma

Submission date: 11-May-2022 04:33PM (UTC+0530)
Submission ID: 1833745642
File name: Report_181305.docx (4.54M)
Word count: 17386
Character count: 80419

Report

ORIGINALITY REPORT

6%
SIMILARITY INDEX

6%
INTERNET SOURCES

1%	%
PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ijirset.com Internet Source	2%
2	www.coursehero.com Internet Source	1%
3	docplayer.net Internet Source	1%
4	www.edrawmax.com Internet Source	<1%
5	yalantis.com Internet Source	<1%
6	firebase.google.com Internet Source	<1%
7	dev.to Internet Source	<1%
8	pdfcoffee.com Internet Source	<1%
9	en.unionpedia.org Internet Source	<1%
		<1%

10	en.wikipedia.org Internet Source	<1%
11	www.ir.juit.ac.in:8080 Internet Source	<1%
12	"Practical DWR 2 Projects", Springer Science and Business Media LLC, 2008 Publication	<1%
	www.semanticscholar.org Internet Source	
13	www.digitalocean.com Internet Source	<1%
14	www.ijraset.com Internet Source	<1%
15	www.theseus.fi Internet Source	<1%
16	handwiki.org Internet Source	<1%
17	dlib.ptit.edu.vn Internet Source	<1%
18	ir.unimas.my Internet Source	<1%
19		<1%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | [✓ B.Tech Project Report] | Paper

Name: ANURAG SHARMA, UTSARG SAXENA Department: COMPUTER SCIENCE Enrolment No 181278, 181305

Contact No. 7590831491, 9034335699 E-mail. 181278@juitsolan.in , 181305@juitsolan.in

Name of the Supervisor: Dr. VIPUL KUMAR SHARMA

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): BOOKBOXED

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 64
- Total No. of Preliminary pages = 6
- Total No. of pages accommodate bibliography/references = 1

[illegible] Utsarg

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ......6......(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible] 13/05/22

(Signature of Guide/Supervisor)

[illegible]

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
			Character Counts	
Report Generated on		**Submission ID**	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com