VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Shaleem John

CHAT APP WITH REACT JS AND FIREBASE

CASE: CHAT APP FOR A COMPANY

School of Technology

2010

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Department of Information Technology

ABSTRACT

Author	Shaleem John
Title	Chat App with ReactJS And Firebase
Year	2010
Language	English
Pages	62
Name of Supervisor	Timo Kaankaanpää

The objective of this thesis was to build Chat App with React JS and Firebase in order to chat among friends, colleagues, family and teachers. Also, the motivation of this thesis project was to design and develop a Chat App for the company Solution Online that uses it for demonstration to students in high schools in a developing country.

The Chat Application was implemented to be used with the help of popular front-end technology, React JS and Firebase hosting services, which offers cost free backend for the chat message and data. The study showed that Firebase backend is a powerful real-time database with high security. The whole application was built in Microsoft´s well known IDE Visual Studio Code and the command line prompts and power shell in Windows 10. All the tools showed the accuracy of the project.

The thesis work was completed with the complete Chat App, meeting all the set requirement by the company Solution Online. Chat App was tested in different web browsers, which showed the expected results.

Keywords Chat App, React JS, HTML, CSS, New Technologies and Firebase

Table of Contents

LIST OF FIGURES

LIST OF ABBREVIATIONS

API - Application Program Interface

AUTH – Authentication

CLI – Command Line Interface

CSS - Cascading Style Sheets

DOM – Document Object Model

ES6 – European Commission Management

HTML – Hyper Text Markup Language

ID – Identification

IDE – Integrated Development Environment

JS – Java Script

JSON – JavaScript Object Notation

JSX – Java Script XML

MVC – Model View Controller

NoSQL – Non-Structured Query Language

NPM – Node Package Manager

PC - Personal Computer

SRC – Source

UI – User Interface

UML – Unified Model Language

URL – Uniform Source Locator

VAMK – Vaasan Ammattikorkeakoulu

VSC – Visual Studio Code

XML – Extensible Markup Language

1 INTRODUCTION

In this chapter, the experience of work placement and over all tasks of the Chat App is discussed.

1.1 Organization the thesis was written to

The requirement for a Bachelor of Engineering´s degree in Information Technology is that student must do training in order to gain experience necessary in the technological field. The work must be theoretical and practical work.

The thesis project was carried in at Solution Online company on 1st May 2018 and practical training was completed officially starting on 15th March 2019 for about 10 months. The experience and skills that are gained and challenges which are faced in solving problems during the training. My training included various activities but mainly were learning and practicing, building React JS applications and a Mobile Online Payment system.

I express my gratitude to Solution Online for giving me this opportunity to do Information Technology training with a partial fulfillment of the requirements for the Bachelor of Engineering in Information Technology.

Throughout this training, I was very honored and lucky with the encouragement and guidance from the firm´s manager, Maqsood Ahmed. I thank the staff of Solution Online for being guiding and sharing the knowledge in technology. I also thank to other staff for creating a good learning environment.

1.2 Overall task to be done

Chat Application with React JS and Firebase consists of three main components on the Chat App´s main page such as sign-up, log-in, message sender bar and, chat panel and sidebar The URL of the Chat App: https://decapp-9d394.firebaseapp.com/. Firebase offers backend components to Chat App.

A new user will sign-in with a valid email address and a six-character password. Firebase Authentication for user´s sign-up and log-in features and chat history are found at https://console.firebase.google.com/?pli=1.

After a successful signed-up, the user enables to sign-in and selects the favorite chatroom the user is also able to create new chatroom if necessary. The Chat Application is useful for students, colleagues, friends, companies and customers.

The Chat Application (Chat App) with React JS and Firebase is available for the users in desktop, laptop, tablet, IOS and Android smart mobiles.

2 USED TECHNOLOGIES

In this chapter, all technologies of the Chat App are mentioned with the background.

2.1 Chat Background

In this section, the background history of the chat is discussed.

2.1.1 SMS

Text message is the short electronic message which is called as SMS. It allows user to send and receive small text messages with little cost. It is a more secure and reliable way to send and receive short message that consists of around 160 characters. For SMS does not require an App and the internet and it is available on all mobiles while MMS (multimedia services) require the internet or a WIFI service. MMS is the enhanced form of the SMS, which enables users to send and receive video and pictures images /1/2/.

SMS is being used for the last 20 years on GSM standard as communication technology enhanced to 3G, 4G and 5G SMS developed to MMS which allows the user to send and receive video and pictures messages. While Chat App is a free of cost, it occurs via a desktop or a mobile app /1/2/.

2.1.2 Instant Messaging

Instant messaging platform was created in the 1960s by MIT allowing up to 30 users to log in at a time. Instant messaging is a one-to-one conversation-based platform which allows user to connect with another person´s device for sharing the text and picture image. In other words, instant messaging is a sort of real-time online chat which is enabled via a special software application. Since technology has improved the instant messaging has gained popularity and it is used in every life. The downside of instant messenger or IM is that the other user must have the same device or software program for the communication. Examples of some famous instant messenger applications are shown in Figure 1. /1/2/.

Figure 1. Instant message categories /3/.

2.1.3 Chat App

The word communication has been derived from the Greek word, “Communicate”, meaning the exchange or sharing of ideas or expression.

Chat was created at the University of Illinois in 1973. Chat App is a platform that enable messaging either one-to-one or to group members usually occurring in a chat room where multiple people can join and shares their interests. In other words, Chat App is also called “Real- Time” chat because it is on-going message service which processes the hundreds of messages between the sender and the receiver without any delay.

Some examples of popular Chat Apps are WhatsApp, Viber, LINE, Skype, WeChat, IM, tt,, C, Tango, N, kik, TALK, talk and ebuddy in Figure 2

Figure 2. Chat App programs /4/.

Chat App figure is different than the old tradition instant messaging applications because it requires only a desktop and a laptop, for example, Yahoo messenger and Hotmail messenger, while Chat App is enabled via mobile apps on smart phones such as Android, IOS as well as a desktop.

2.1.3.1 Benefits of Chat

Benefits of chat includes many companies and schools rely on live chat today because it has the following benefits.

1 Faster support: It is easy to approach students, colleagues and customers by live chat. Generally, it takes less time to solve the customer issue when using live chat.
2 Real-time text examine: It is one of the benefits between customer and agent. A customer sends the request for a solution and an agent views and think about the solution of the issue and replies to the customer immediately.

3 Instant customer feedback: Customer can rate the company service right after using the chat service.
4 No waiting queues: Customers can reach the company agent immediately.
5 On a browser site: The customer and agent conversation take place on the website. During a chat, the agent can see where the customer is currently looking the issue so the agent can easily guide the user by copying the link or screenshot image.
6 Purchasing solution: On online shopping point of view, is one of the important advantages of live chat. A customer can ask the agent about the product cost in the middle of online shopping.
7 File transfer: A customer can provide file, document and images of the questioned issue upon asking by the company agent.
8 Mobile Messaging Integrations: Now, companies add Live Chat messenger like Viber, WhatsApp and Facebook to a company website. Customers are connected with the company´s central live chat system via advance live software programs.
9 24/7 Support Service: A company is available for the customers 24/7 without any time zone restrictions. Customers can talk to a chatbot even out of office hours.
10 Less costs: Live chat is cheaper than the traditional phone call for the customer to access the company agent for questioning and solving issues /5/.

2.1.3.2 Disadvantages of Chat

Disadvantages of Chat are:

1 Internet Access: The user needs constant access to the internet during a chat conversation with colleague or company´s help desk. Also, users have internet access difficulties in developing countries
2 Communication difficulties: Some users have difficulties in chat communication with the help desk, particularly elder and disable people
3 Virus Risk: Companies can have virus risks attached files, pictures or images when transferring them from the user to company during chat
4 User Unfriendly: Live chat is unfriendly for the novice computer users
5 Unfriendly: It is observed unfriendly on mobile platforms /5/.

2.2 HTML

HTML is a programming language which was developed by Berners-Lee in 1980 but standardized in 1995. HTML stands for Hyper Text Markup Language. It is one of the most popular languages on the planet used for developing web pages and websites. HTML is used for the structure of web documents i.e. headlines, paragraphs and images etc. HTML is known as the basics of all programming language in the globe. HTML is highly used in other programming languages such as Javascript and CSS for the its display attributes. It consists of opening and closing tags as <html> </html> with the file extension .html /6/7/8/.

2.2.1 Advantages of HTML

1 HTML is easy to learn and use without any prerequisite's knowledge in programming.
2 HTML has no license fees.
3 HTML is supported by all browsers on the web and supported by all platforms.
4 HTML is lightweight code so that fast to download.
5 HTML is very easy to edit in any text editor such as Notepad or Notepad ++.
6 HTML is the most search-friendly programming on the globe /6/7/8/.

2.2.2 Disadvantages of HTML

1 HTML script is easy to write while it can take much time to write a single webpage,
2 HTML is very easy to write as compare to other programming languages while it needs unaccountable words to develop even a single webpage on the internet,
3 HTML needs a server for saving a file,
4 HTML is easy to write but it is difficult to find an error when a webpage does not open /6/7/8/.

2.3 CSS

CSS stands for Cascading Style Sheets. It is the advanced form of HTML language and little easier to use than HTML, and it is also compatible with HTML. CSS is used for the presentation of web pages i.e. design, colors, layout and fonts etc. CSS has a special standard of a sector and declaration such that selector, property, declaration block and values. CSS has three ways of inserting style sheets into HTML code i.e. External CSS, Internal CSS and Inline CSS. CSS file extension is file.css /9/10/11/12/13/.

2.3.1 Benefits of CSS

1 CSS has more attributes than HTML that make a better-looking web page style format than HTML,
2 Its faster webpage speed.
3 It has less lightweight code that makes webpage takes less downloading time
2 It makes updates, modification and maintenance a lot easier,
3 It is compatible with multiple devices as well as platform environment,
4 CSS is extremely used in E-commerce, social media and web-based online services,
5 It is easy to handle image files by CSS. Files such as jpeg, png and gif can be easily modified by using the CSS frameworks,
6 It is easy to handle to Dynamic Website Templates with the help of CSS frameworks,
7 It is easy to handle Flash Animation and Effects in web pages with the help of CSS Flash files such as button effects, loaders and animated background,
8 It is used for the client-side and server-side styling display which gives better display in the websites,
9 CSS has three types of inserting styles i.e. Internal CSS, External CSS and Inline CSS. Internal CSS is a ruleset which needs to add <style> tag in the <head> section of the HTML and it is mainly used a single page. External CSS which links to an external .css file extension. It is extremely useful for creating a large webpage.

Inline CSS means adding CSS into the HTML file and located in the head of the document between style tags /9/10/11/12/13/.

2.3.2 Disadvantages of CSS

1 CSS have many types like CSS, CSS 1 and CSS 3. These different kinds of CSSs create confusion among developers and web browsers in different platforms,
2 CSS web pages is not compatible on all browsers,
3 CSS is an open text-based code which does not have built-in security. It means anyone can read and write the CSS file and change the website contents which can cause the major issues in the websites /9/10/11/12/13/.

2.4 Bulma

Bulma is a cost free and open-source modern CSS framework which is based on modern Flexbox. Flexbox is a new layout model which was built in CSS 3.

It was designed as a one-dimension layout method because we set only either columns or rows. Flexbox contains Flex-container property, Flex-direction property and Flex-item property. Flex-container property in which items are arranged either rows or columns. Flex-direction property contains the elements which are arranged either as row, column, row-reverse or column-reverse items. Flex-item property in which item are arranged via Flexbox /14/.

2.4.1 Benefits of Bulma

1 It is simple to learn and use
2 It has lightweight code
3 It is flexible and powerful framework
4 It provides easy to write e.g. write the Large word which displays Large Button
5 It is used by front-end developers
6 It has responsive frameworks like the Bootstrap CSS
7 It is well-documented and ready to use for the software applications
8 It has solid foundation of all properties such as buttons, forms, tables etc.

9 It has various components which include vertical and horizontal alignments and other media object features
10 Bulma makes easy to build web sites and application because developers do not need many codes and many required features are ready-made in the Flexbox /14/.

2.4.2 Disadvantages of Bulma

1 Bulma has slow response in Internet Explorer browser
2 It is constantly being developed by developers
3 It is still not used by majority of developers.

Installation in npm:

Bulma is installed in npm by: npm install bulma

In Visual Studio Code:

Bulma is in .src/bulma/css/bulma.**css** /14/.

2.5 Visual Studio Code

Visual Studio Code is the part of visual studio family which is developed by Microsoft in November 2015. It is based-on Electron framework which is used for Node.js (node java script). It is written in TypeScript, JavaScript and CSS /15/16/.

2.5.1 Features

Features of Visual Studio Code are as follows:

1 It is open-source and Freeware text editor for private and commercial purposes
2 It is cross-platform source code editor debugger
3 It supports many different programming languages while only proper installation of the extension is required for React JS, Java, JavaScript, C++, C#, Python etc.
4 GitHub is built-in
5 Products availability as it explains the definition and show opening and closing of the brackets etc.

6 It provided unique customizations
7 Provides Visual Studio Keymap extension for using Key binding
8 It provides in the portable mode that means it keeps data and settings in the same location of installation is possible even on a USB drive
9 It is available in many different language services
10 It is also available in Remote Development mode /15/16/.

Installation of Visual Studio Code:
All platforms installer is available, and the developer can pick the proper one for the project.
https://code.visualstudio.com/download

2.6 React JS

React JS was created by Jordan Walke and deployed first at Facebook in May 2013. It is based on open-source JavaScript library. In other words, ReactJS is a great platform which is developed for the front-end mobile and web apps. It is also known as React JavaScript library, which is used for the developing of User Interfaces (UI).

It consists of independent, isolated and reusable components which are pieces of UI. Every React JS project has a main root component called as Application (App). Application root component contains other related child components. React JS application comprises of React element and React component. React component is the function or class which renders React element or HTML element /17/18/19/.

2.6.1 Key Features of React JS

The React JS offers plenty of benefits but below there are some key benefits.

1 Easy to learn: It is easy to learn as compare to other front-end frame works technologies
2 JSX: JSX is a shorthand of JavaScript XML. React JS uses JSX instead of HTML which enables the user to read and write the components easily and clearly.
3 Speed: It allows the developers reuse the individual components on both client-side and server-side without effects on the main application

4 Flexibility: React JS code is easy to maintain and flexible due to its framework structure as compared to other front-end framework technologies
5 Performance: React JS uses virtual DOM and server-side rending thus it makes complex web apps runs on high speed
6 Code stability: It follows unidirectional data flow or downward data flow which means parent structure unaffected by any modifications in its child structure
7 Integration: React code easy to integrate with other frameworks like Angular
8 SEO- friendly: SEO stands for Search Engine Optimization. React offers less page load time and fast rendering speed.
9 It is easy to test UI cases with React code
10 React renders at client side while SEOs do not use the data /17/18/19/.

2.6.2 Virtual DOM and Real DOM

DOM stands for Document Object Model, which is programming language platform interface for HTML and XML documents. It is a tree-like structure which enables dynamically accessing and managing the elements. It has three main types:

1 HTML DOM
Standard model for HTML documents
2 XML DOM
Standard model for XML documents
3 Core DOM
Standard model for all types of documents
In addition to, DOM have two types Real DOM and Virtual DOM. React JS uses Virtual DOM /20/21/.

Real DOM
1. Its manipulation or updates is slower and costly.
2. It is heavy weight code.
3. It can update the HTML in the browser /20/21/.

Virtual DOM
1. It is not created by React, but it is used as for free.

2. It is a copy of Real DOM.
3. Its manipulation or updates is faster and easier.
4. It is light weight code
5. It cannot update the HTML directly in the browser /20/21/.

2.6.3 Life Cycle of React Components

In life, we humans as well as tree have a life cycle like we are born, grow and finally we die. It is the same phenomena in software application´s life cycle, which takes birth as initialization, it grows by updating and changes occurs during the different stages by mounting and unmounting. React JS has four life cycle classifications as below /22/25/

1 Initialization: This is first and most important phase in which developer setup or construct the component with the given props (properties) and default state.
2 Mounting: In this second phase, after creating state and props by developer. React JS component is ready to mount in the browser DOM.
3 Update: This is third stage, once the component gets added in browser DOM. React JS components updates it when a state or prop changes in the application.
4 Unmounting: This is fourth and final phase of the life cycle of React JS component in which the component is removed from browser DOM when it is not needed anymore /22/25/.

2.6.4 User Interface

User Interface (UI) is the interaction and communication between human and any device. Some common examples are:

- Command Line User Interfaces (CLI)
- Graphical User Interfaces (GUI)
- Touch screen Interfaces
- Menus Interfaces
- Voice User Interfaces (VUI)
- Form-fill Interfaces

• Natural-Language Interfaces /23/24/.

2.6.5 ES6

The European Computer Manufacturers Association (ECMA) created ECMAS to standardize JavaScript in 2015. ECMAScript (ES6) is a script language. ES6 is generally used for the client-side scripting also server-side application, especially with Node.js /26/.

ES6 offers you the possibility to write the code more readable and in a more advance way. In addition, ES6 introduces many advance features like arrows, classes, template strings and class destruction /26/.

2.7 Flux

Flux is an architectural pattern which was developed by Facebook to work with React in 2011. FLUX´s application architecture is more suitable for building user interface application also client-side web-based applications /27/28/29/.

2.7.1 Features of Flux

Flux´s architectural pattern is based on four segments in unidirectional data flow. Its values and objects can be mutated or changeable without creating any new values and objects. It is integrated with TuxedoJS, Fluxxor and React as shown in Figure 3 /27/28/29/.

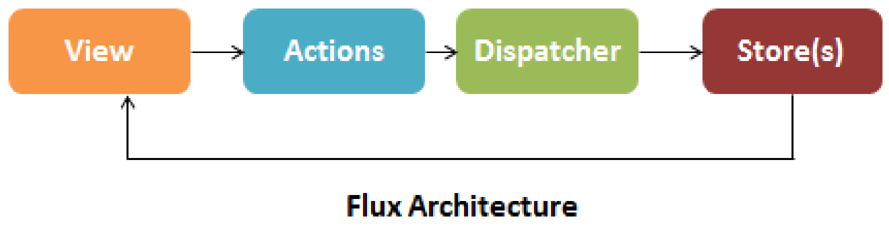


Figure 3. Flux Architecture /31/.

Flux´s four segments are as following:

1 Actions: Action is a simple JavaScript object in Flux pattern which must pass through dispatcher.
2 Dispatcher: Dispatcher is the central hub of the application in Flux patter which dispatches all the data and send it to the stores.
3 Stores: Flux pattern has multiple stores for each application while each store is a singleton type object.
4 View: View is the element of the Flux pattern which sends the data from the store to the application /27/28/29/.

2.8 Redux

Redux was developed by Dan Abramov and Andrew Clark in 2nd June 2015. It is a library in React. Its whole application based on a single state object which is immutable or unchangeable directly. Redux library is used for React, but it also works well with other front-end libraries like Angular JS, Backbon.js, Ember, Meteor and Polymer /27/28/29/.

2.8.1 Features of Redux

Redux´s features as shown in Figure 4:

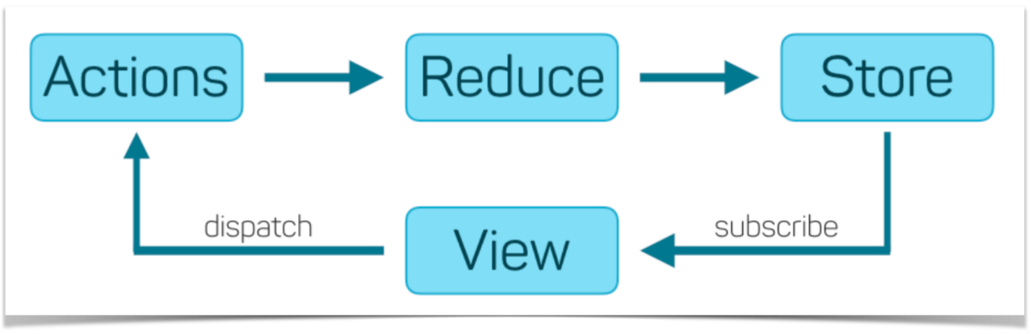


Figure 4. Redux structure /30/.

The four parts of the Redux are as below:

1 Actions: Actions are the simple JavaScript objects having special identity to the objects in Redux
2 Reduce: Reduce / Reducer is a simple function in Redux which receives the copy of the old state and apply the new changes as updates it by the components
3 Store: Redux library has single store per application
4 View: View is the element of the Redux library which sends the data from the store to the application /27/28/29/.

2.9 MVC

MVC stands for Model View Controller. It is an architectural design of an application. All these three components are essential for creating and working together for all web applications /31/.

2.9.1 Features of MVC

MVC´s three components are as shown in Figure 5:

Model: It is the basic part of the architectural design that controls the logic and behavior of the application,
View: It displays the model´s data for the user in the application,
Controller: It is the major part of the architecture design because if offers the interface between view and model.

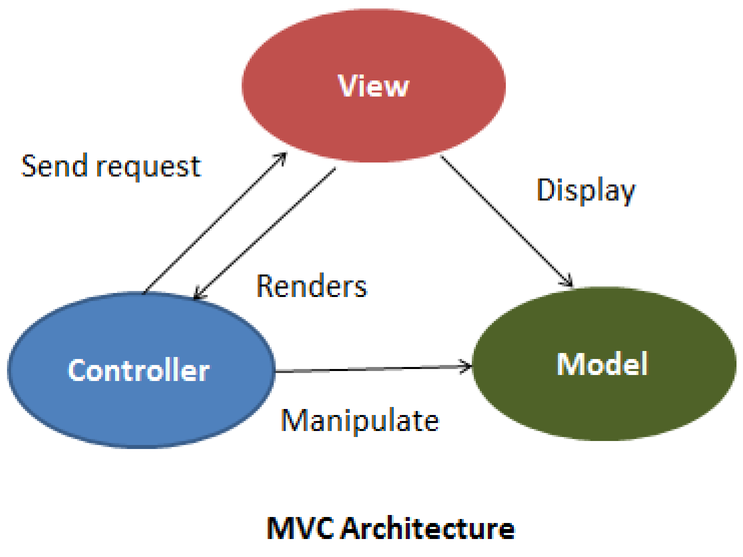


Figure 5. MVC Architecture /31/.

MVC´s key features are as:

1 Architecture: It is architectural design for web application and developing UI
2 Data Flow Direction: It provides bidirectional data flow
3 Store: It has no store concept as Flux and Redux with multiple stores and single store Logic
4 Logic: In MVC, controller controls the whole logic while in Flux, all logic is handled by Store, but Reducer controls the logic in Redux
5 Debugging: In MVC architecture design, debugging is difficult because of its bidirectional data flow
6 Applications: MVC works fine in both client-side and server-side frameworks. It supports all front-end frameworks e.g. AngularJS, Ember, Backbone, Sprout and Knockout /31/.

2.10 Firebase

Firebase is the Firebase Cloud messaging software cross-platform. It is also known as Google Cloud Messaging it was developed by Andrew Lee and James Tamplin in 2011. Firebase provides cloud service, server and real-time backend server. API for the clients also helps to develop high quality applications. Most importantly, Firebase is a NoSQL database in which data is stored in a cloud /32/33/34/35/37/.

When full-fledged application is built, you have to rely on the backend service as well. Therefore, Firebase is a good choice because it saves time and allows you to focus on development. It stores the data in JSON file or JSON objects and we do not need to write any server codes, but it is ready-made API for developing and implementation of applications /32/33/34/35/37/.

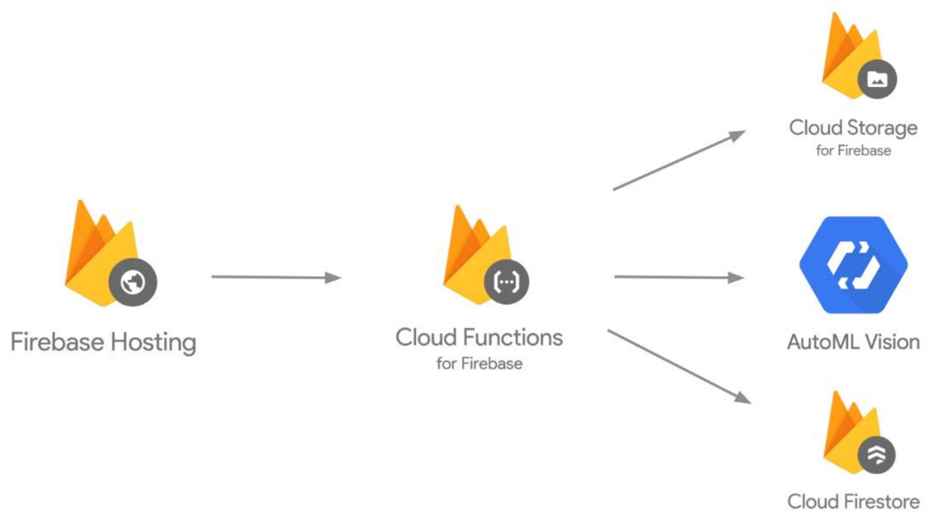


Figure 6. Firebase backend structure /36/.

Figure 6 demonstrates that the Firebase hosts the services to cloud functions which provides three different types of cloud services as below:
1 Cloud storage which stores data
2 AutoML vision is the Google´s machine learning model which is used for building the image recognition
3 Cloud Firestore is the Google´s NoSQL cloud platform which provides database for mobile, web and server development /36/.

2.10.1 Features of Firebase

The features of Firebase are as below:

1 Real-time Database: It synchronizes the sender´s data to Firebase which updates instantly, and client receives the updated data on real-time on any connected device. It is initialized by the Firebase´s initialize () function as implemented in my Chat App as below:

```
Const firebaseApp = firebase.initializeApp(config);
```

2 File storage: Firebase enables the file storage directly from the client even in offline mode. It is implemented in Chat App as below:

```
export const messageRef = firebase.database().ref('messages');
```

Data in the cloud is highly protected by Firebase´s own security system.
3 Authentication: Firebase provides its own pre-built or custom UI authentication or sign-in method either email / password, phone number, google id, Facebook id, yahoo or Apple id. The email and password authentication method are used in Chat App as fellow:

```
auth.createUserWithEmailAndPassword ('rs@rs.com', '123456')
```

And

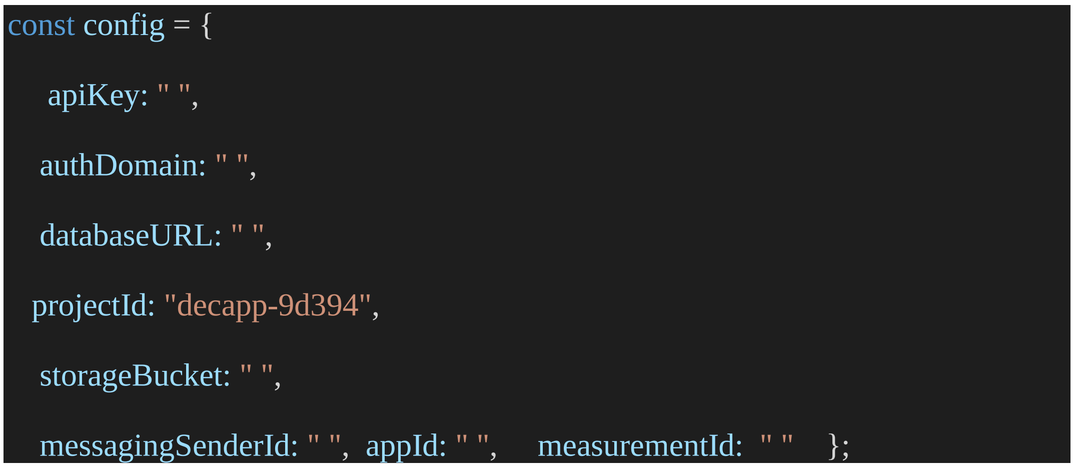
```
const firebaseApp = firebase.initializeApp(config);
```

And

```
export const auth = firebaseApp.auth();
```

4 Hosting service: Firebase offers the implementation of static websites and single page applications which is built with HTML, CSS and JavaScript. It uses HTTPS and SSL protocols for the security of files and data delivery. It does not require Content Delivery Network (CDN), which is built in function in Firebase.

5 Free of cost: Firebase does not require payment for providing databases service to client to some extent. On the other hand, the client has to pay if the database memory is not enough for the specific services /32/33/34/35/37/.



```
const config = {
    apiKey: " ",
   authDomain: " ",
   databaseURL: " ",
  projectId: "decapp-9d394",
   storageBucket: " ",
   messagingSenderId: " ",  appId: " ",     measurementId:  " "    };
```

Figure 7. Screenshot of Firebase.js in Chat App

The screen shot in Figure 7 shows my Chat App project in the Firebase: It has unique digits of ApplicationKey, authenticationDomain, databaseURL, projectID, storageBucket (data storage), messagingSenderId, ApplicationID and measurementID (gtag () calls and Firebase).

2.10.2 Disadvantages of Firebase

The disadvantages of Firebase are as below:

1 Firebase does not support traditional SQL data because it is NoSQL, which is built on JSON file,
2 Firebase has limited querying capabilities,
3 Firebase is a real-time database which is a single file data structure and impossible to deploy relations with the data items,
4 Firebase databases less supports to iOS applications /32/33/34/35/37/.

Installation of Firebase in npm module:

Firebase is installed either in command line (CLI) or power shell in Windows: npm install firebase

In Visual Studio Code:

The Chat App project is created with firebase.js, firebase. json and. firebaseerc are in source (src) folder of the project.

3 CHAT APP SYSTEM ANALYSIS

This chapter explains the system analysis of the Chat App in details.

3.1 Environment Setup

The environment setup of Chat App requires the installation npm first.

3.1.1 Installation of npm

NodeJS is JavaScript runtime open source environment which helps developers to build web applications with the help of frameworks like web pages, images, video clips, social network sites and application forms. It is based on Chrome's V8 engine which runs on cross-platform such as windows, Linux and Mac. The V8 engine compiles the JavaScript code into machine code that makes our program code runs smoother and faster /38/39/.

NodeJS is an open source library which is currently used by unaccountable developers around the globe. NodeJS uses a special programming technique which is called asynchronous programming. Asynchronous programming means that developers can execute multiple codes at a time without finishing the current execution can execute the parallel code. NodeJS offers popular services like Netflix, PayPal, Uber and eBay /38/39/.

Npm stands for Node Package Managers. It is the world´s largest open source library which is used for Nodejs. It was developed by Rebecca Turner, Kat Marchan in 2010. It is written in JavaScript itself and free for use /38/39/.

It is highly recommended to download NodeJS directly from its official website i.e. http://nodejs.org. NPM comes with the installation of NodeJS.

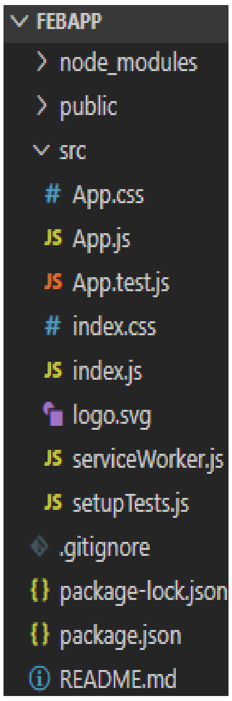
```
C:\Users\John> npm -v
6.11.3
C:\Users\John> node -v
v12.12.0
```

The above commands show that npm and node is installed with the latest version.

3.1.2 Create React App

The official way to create an app in React JS is “create-react-app”, which was first deployed in Facebook for testing the real-time chat of a user. It helps to save a lot of time, so we do not need to install or configure extra tools like, for example, web packs, Babel and other dependencies.

A simple command is only needed to install either in command prompt or power shell in windows.
1 npm install -g create-react-app (g stands for globally installation)
Create a project name then install a new app inside the project
2 C:\Users\John\Desktop>mkdir testingapp
3 C:\Users\John\Desktop\testingapp> create-react-app chatapp
4 Installing packages of dependencies such as
Installing react, react-dom, and react-scripts. Wait for the couple of minutes.
We suggest that you begin by typing:
cd chatapp
npm start
Happy hacking!
The Chat App file structure in VS code created by create-react-app as shown in Figure 8



```
FEBAPP
  node_modules
  public
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    serviceWorker.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Figure 8. Chat App file structure in vs code created by create-react-app

3.1.3 Installing Visual Studio Code

The IDE stands for Integrated Development Environment. It is used for the text editor for building software applications, compiler and debugger with help of various integrated tools. In other words, IDE is a software tool which can perform multi functions such as to write code, modify code, compile and debug code etc. /15/16/.

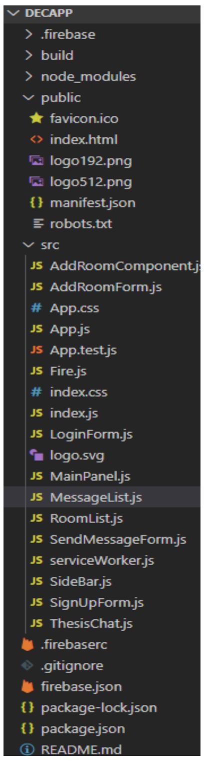
Now, there are many IDEs available in the market which are used for the different performs. Most important, Visual Studio Code (VSC) is the best choice for Chat App. It is free and open-source and it works perfectly on cross-platforms. It is recommended to install visual studio code from its official website https://code.visualstudio.com/download /15/16/.

3.2 User Story Case

The user story of the Chat App explains the file structure of the application and how the user can use the application.

3.2.1 Project Structure

In this section, the deployment of this project is discussed. The whole project structure and different components of the project will be explained in Visual Studio Code. All component files of the project are in the source (.src) folder on the left-hand side as shown in Figure 9.



```
DECAPP
  > .firebase
  > build
  > node_modules
  v public
      favicon.ico
      index.html
      logo192.png
      logo512.png
      manifest.json
      robots.txt
  v src
      AddRoomComponent.j
      AddRoomForm.js
      App.css
      App.js
      App.test.js
      Fire.js
      index.css
      index.js
      LoginForm.js
      logo.svg
      MainPanel.js
      MessageList.js
      RoomList.js
      SendMessageForm.js
      serviceWorker.js
      SideBar.js
      SignUpForm.js
      ThesisChat.js
    .firebaserc
    .gitignore
    firebase.json
    package-lock.json
    package.json
    README.md
```

Figure 9. Screenshot of project in Visual Studio Code

In source (.src) folder, App.js is the main app file of the project, SignUp.js and LoginForm.js component for creating a user in chat application and SendMessageForm.js component is for writing the message to another user, which is in the main panel on right side. While, the SideBar.js component contains Logout button and during a chat it enables creating a New chat Room. Fire.js component is for backend and hosting service of a user´s chat and firebase. json contains the json file of the backend hosting service of firebase while the firebaserc file shows the default chat project.

3.2.2 User Story

In this section, a user´s access in the chat application UI is discussed in detail. How a user sign up and sign in Chat App and also is shown how two different users chat with each other in real-time.

A browser is needed by the user on any operating system to access Chat App´s URL such as **https://decapp-9d394.firebaseapp.com/** **.** At first, a valid email id like sj@vamk.fi and a six-character long password strength is needed in order to sign up as shown in Figure 10.

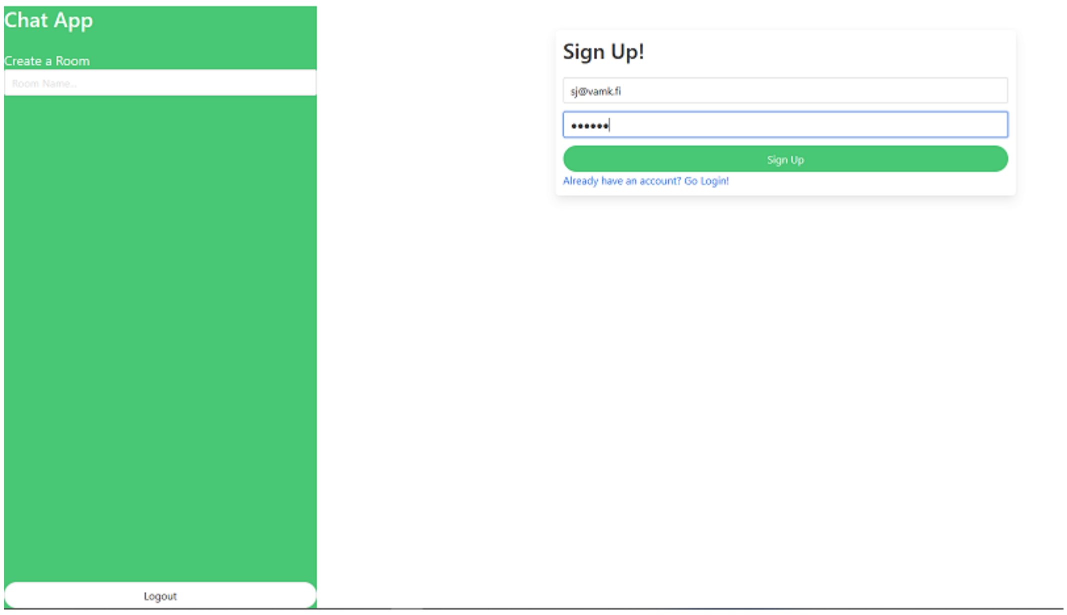


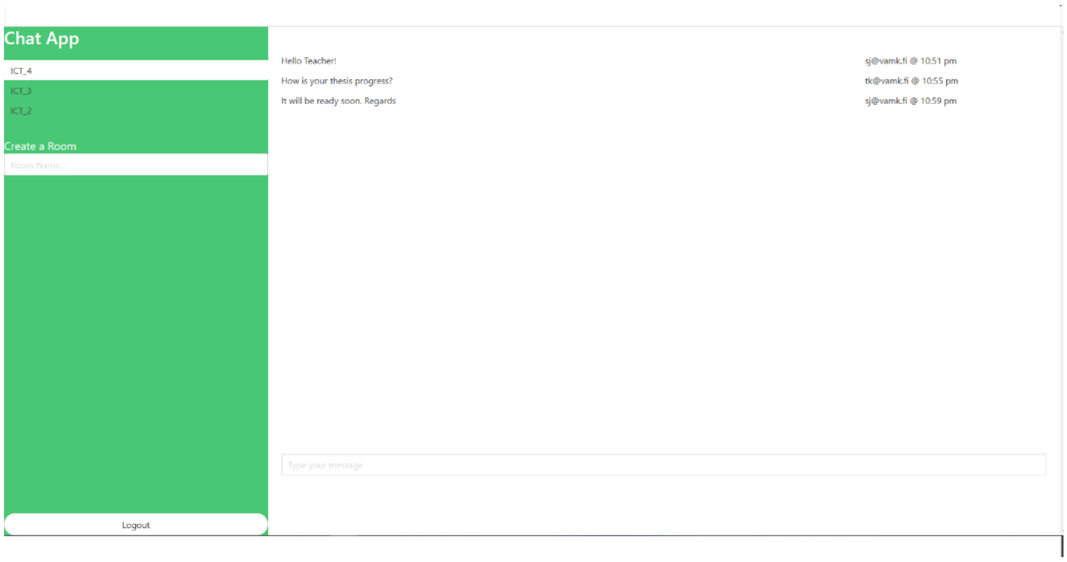
Figure 10. Screenshot of Sign Up form

A browser is also needed by another user on any operating system to access Chat App´s URL **https://decapp-9d394.firebaseapp.com/**. At first, a valid email id like tk@vamk.fi and a six-character long password strength is required to sign up then a user can login in the Chat App.

The user message is written in the message bar at the bottom of the message panel and enter button is pressed. The sender´s message is displayed with the user´s valid email and time stamp in the chat panel. The receiver could view the sender´s message and reply to the message by writing in the message panel and pressing the enter button.

Chat Application is a full-fledged application that demanded backend and hosting services providing a real-time database and storage of the user chat´s for all back and forth messages. Firebase is the database hosting service which is used for backend functionality for the Chat App, but the application does not respond without backend service. A chat session is ended just simply pressing log-out in the chat panel.

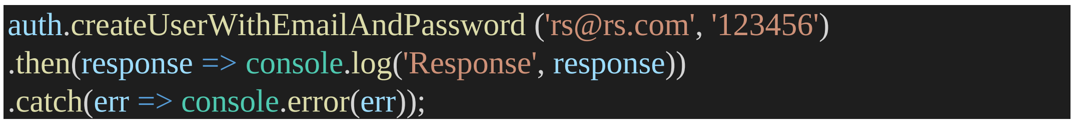
Both chat users could chat in real-time and the user chat with a valid email and the chatting time is displayed in the chat session as shown in Figure 11.



Hello Teacher! sj@vamk.fi @ 10:51 pm
How is your thesis progress? tk@vamk.fi @ 10:55 pm
It will be ready soon. Regards sj@vamk.fi @ 10:59 pm

Figure 11. Screenshot of two users chat in the chat session

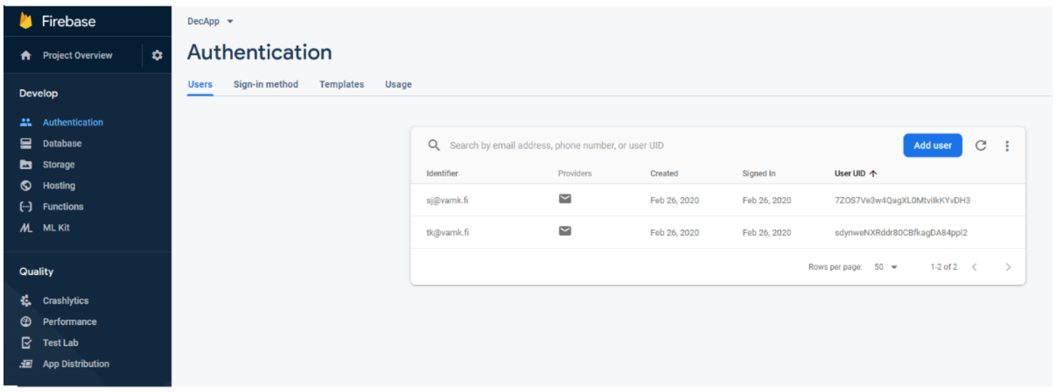
It is assumed that, Firebase works as a backend in the chat application project. Firebase uses auth () method for a creating user id with email and password purpose as shown in Figure 12.



```
auth.createUserWithEmailAndPassword ('rs@rs.com', '123456')
.then(response => console.log('Response', response))
.catch(err => console.error(err));
```

Figure 12. Screenshot of the Firebase auth () method

It is observed that, Firebase works as a backend the Chat Application project. Firebase auth () function fetches the user´s email and saves it in the Firebase Authentication, which is shown in the newly created user date and specific UID as in Figure 13.

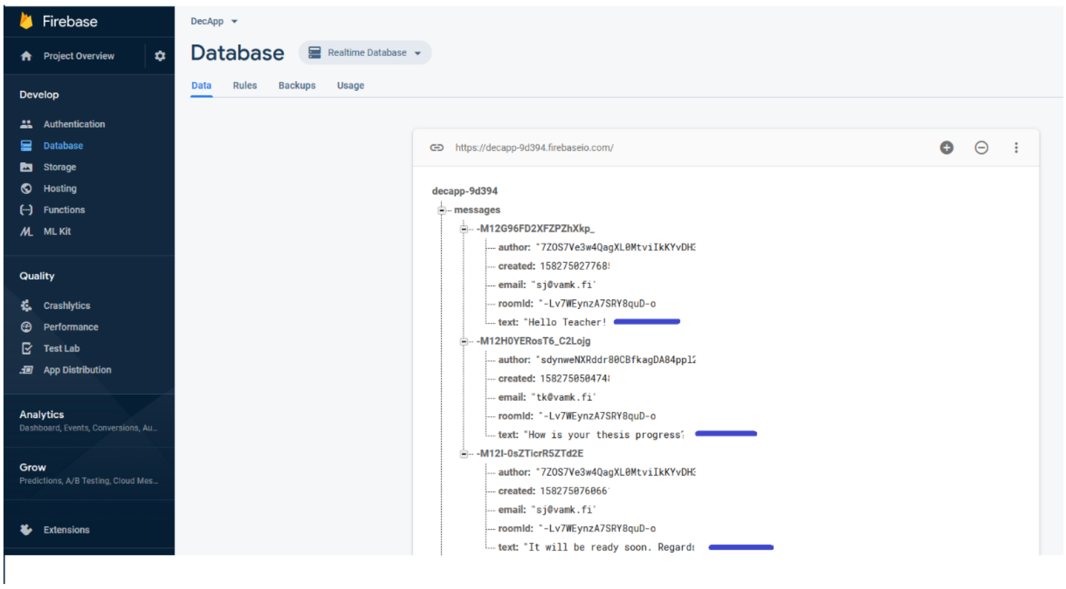


Identifier	Providers	Created	Signed In	User UID ↑
sj@vamk.fi	✉	Feb 26, 2020	Feb 26, 2020	7ZOS7Ve3w4QagXL0MtvilkKYvDH3
tk@vamk.fi	✉	Feb 26, 2020	Feb 26, 2020	sdynweNXRddr80CBfkagDA84ppl2

Rows per page: 50 1-2 of 2

Figure 13. Screenshot of the Firebase Authentication

Most importantly, it is recommended never to save the passwords in the Firebase or any other databases. In addition to this, Firebase auth () function is discussed in detail in the next chapters. Firebase is a real-time database and backend for the chat app. For that purpose, it receives the chat data from the users´ chat UI and sends it to the Firebase. In Firebase database, data or messages are shown by the author id, which is assigned for all users, a created id means date and time of the chat, email of the user, roomId in which chat is written and finally the chat text. It is shown as in Figure 14.



```
https://decapp-9d394.firebaseio.com/

decapp-9d394
 messages
   -M12G96FD2XFZPZhXkp_
      author: "7ZOS7Ve3w4QagXL0MtviIkKYvDH
      created: 158275027768
      email: "sj@vamk.fi"
      roomId: "-Lv7WEynzA7SRY8quD-o
      text: "Hello Teacher!
   -M12H0YERosT6_C2Lojg
      author: "sdynweNXRddr80CBfkagDA84ppl
      created: 158275050474
      email: "tk@vamk.fi"
      roomId: "-Lv7WEynzA7SRY8quD-o
      text: "How is your thesis progress
   -M12I-0sZTicrR5ZTd2E
      author: "7ZOS7Ve3w4QagXL0MtviIkKYvDH
      created: 158275076066
      email: "sj@vamk.fi"
      roomId: "-Lv7WEynzA7SRY8quD-o
      text: "It will be ready soon. Regard
```

Figure 14. Screenshot of chat in Firebase database

3.3 User Case Diagram

Use Case Diagram is one of the diagrams of UML (Unified Model Language) which demonstrates the interaction of the users (called Actors) with each other within the system. In other words, a use case diagram represents the activity of a system between actors, elements and their roles. The use case diagram consists of a system which is a whole scenario in which all events and flows of the different elements are interacted with each other. Actors can be users or internal elements in the system and the used cases that made connection with elements in the system that is always represented by an oval shape in Figure 15 /40/41/42/.

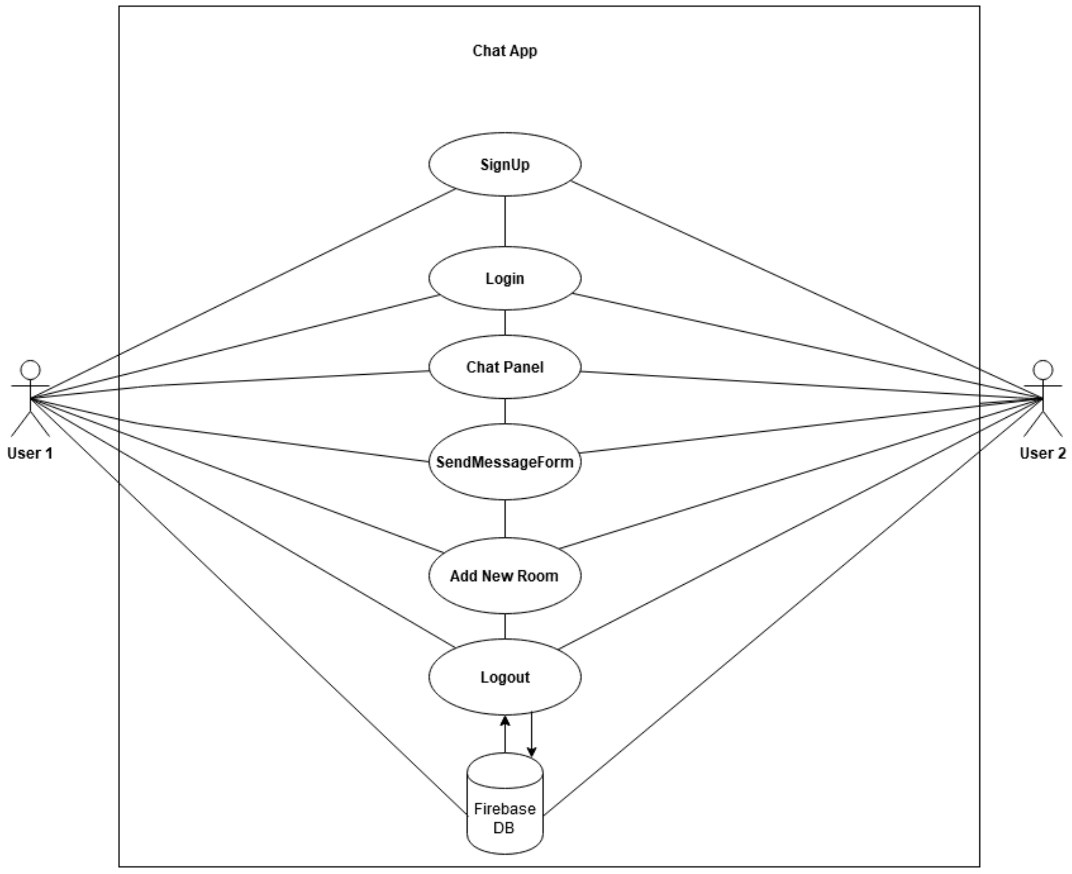


Figure 15. Use Case Diagram of Chat App

3.4 Test Plan

In this chapter, how to test Chat App in different browsers for example Firefox and Google Chrome is discussed.

For Chat App testing purposes, a valid email and password are needed for the user as well as browser and internet service.

1 Testing in Firefox browser:

User email id:
Password:

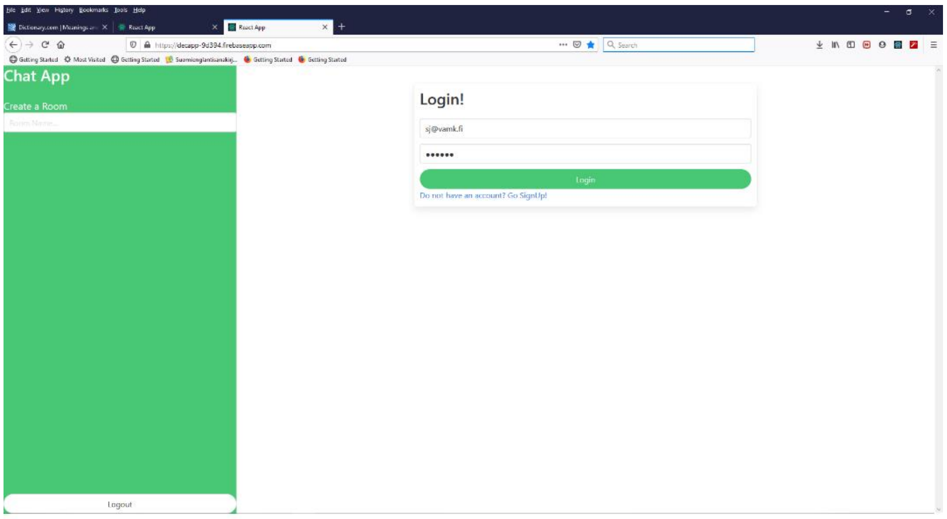
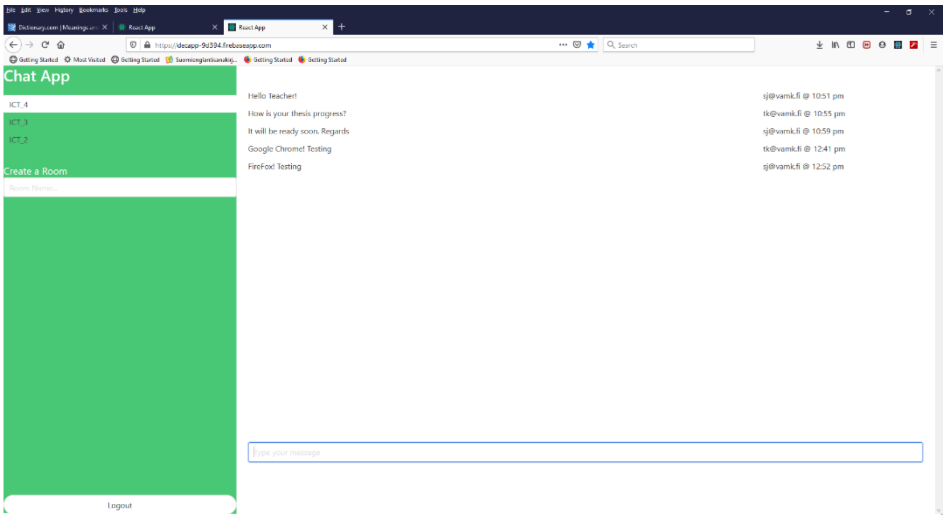


Figure 16. Screenshot Login page in Chat App

Result:

User successfully logged in the Chat App page in Firefox browser.



Hello Teacher! sj@vamk.fi @ 10:51 pm
How is your thesis progress? tk@vamk.fi @ 10:55 pm
It will be ready soon. Regards sj@vamk.fi @ 10:59 pm
Google Chrome! Testing tk@vamk.fi @ 12:41 pm
FireFox! Testing sj@vamk.fi @ 12:52 pm

Figure 17. Screenshot of chat panel

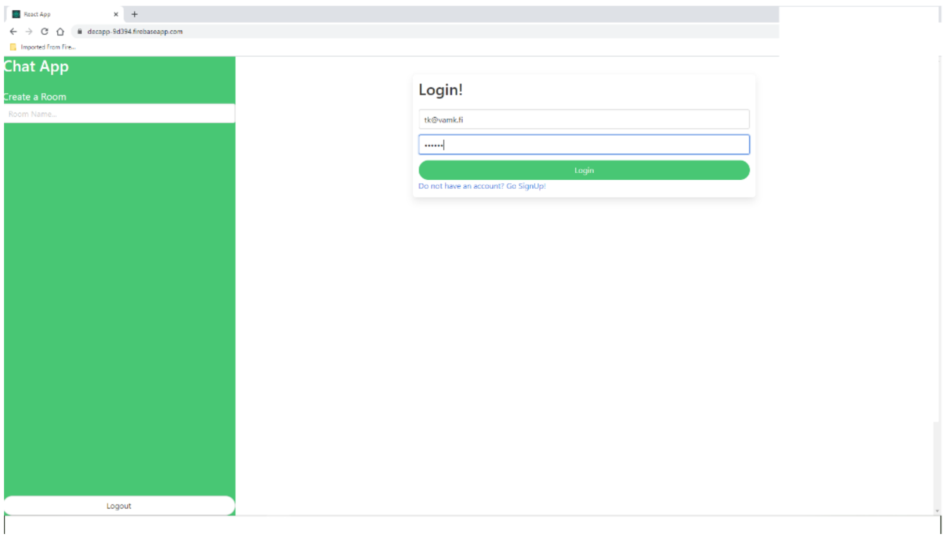
Result

User successfully chat with another user in Chat App panel.

2 Testing in Google Chrome:

User email id:

Password:



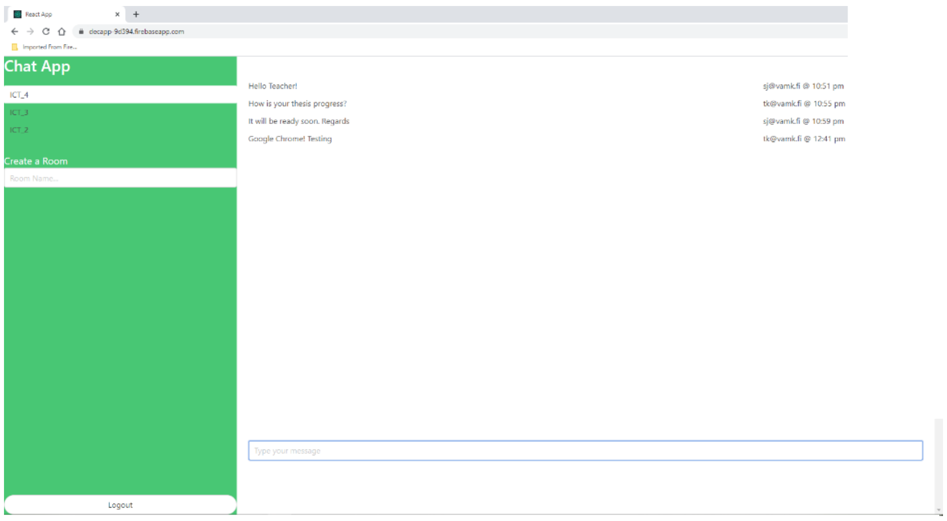
tk@vamk.fi

••••••

Figure 18. Screenshot Login page in Chat App

Result:

User successfully logged in the Chat App page in Google Chrome.



Hello Teacher! sj@vamk.fi @ 10:51 pm
How is your thesis progress? tk@vamk.fi @ 10:55 pm
It will be ready soon. Regards sj@vamk.fi @ 10:59 pm
Google Chrome! Testing tk@vamk.fi @ 12:41 pm

Figure 19. Screenshot of chat panel

Result: User successfully chat with another user in Chat App panel.

4 CHAT APP SYSTEM DESIGN

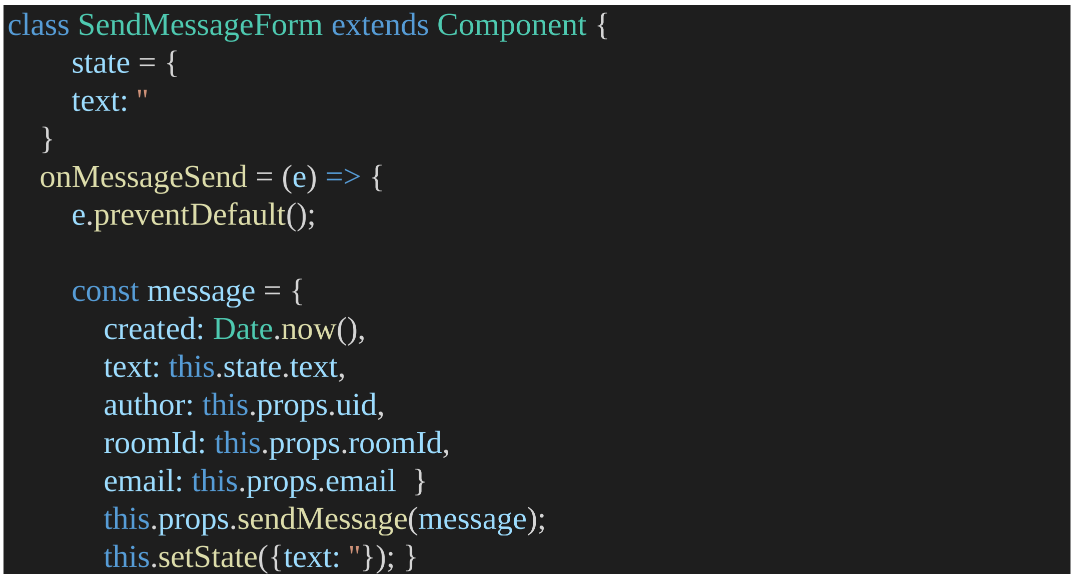
In this chapter, the Chat App system design is explained in detail.

4.1 Architectural Design

The Chat App Architecture and how the user sends and receives messages with the support of backend, or a chat server is explained in this chapter.

Chat App splits into two main parts that are a chat user or a client which is front-end that means it can be the web, a mobile device, a desktop or any other device. The other parts are the chat server or server or the backend where all chat takes place in real-time communication. In this project, Firebase database acts as a chat server or the backend.

Chat App and backend both are handled as the essential parts of the chat architecture. Chat App consists of Chat UI, which displays the UI for signup as well as login, chat panel, creating new room and logout. Firebase backend is synchronized with the sender´s data to Firebase, which is updated instantly, and the client receives the updated data in real-time on any connected device from the backend server as in Figure 20.



```
class SendMessageForm extends Component {
      state = {
      text: ''
   }
   onMessageSend = (e) => {
      e.preventDefault();

      const message = {
         created: Date.now(),
         text: this.state.text,
         author: this.props.uid,
         roomId: this.props.roomId,
         email: this.props.email  }
         this.props.sendMessage(message);
         this.setState({text: ''}); }
```

Figure 20. Screenshot of SendMessageForm

The SendMessageForm screenshot is presented, the user is sent a message onMessageSend (e) where e is the argument method for chat message submission, creating and displaying the chat message time, the author is showing the sender of the chat message, roomId as the selected room of the user for the chat, the email is showing the user´s assigned email, this.props.sendMessage(message) shows the property of the message and this.setState {text} in setState updates the text message. The method e. preventDefault () prevents the default action from occurring on the "submit" button.

It is assumed that, Firebase acts as backend for the project, so that Fire.js file is in source folder of the project. Firebase provides its own pre-built or custom UI authentication or sign-in method either as email / password, phone number, google id, Facebook id, yahoo or Apple id. App.js is the main JavaScript file in which all other components files are exported. The email and password authentication method is implemented in Chat App as:

```
auth.createUserWithEmailAndPassword ('rs@rs.com', '123456')
```

Exporting Firebase to App.js:

```
export const auth = firebaseApp.auth();
```

For initializing App:

```
const firebaseApp = firebase.initializeApp(config);
```

In App.js:

```
import firebaseApp from './Fire';
```

4.2 Module Design

This topic explains the functionality of all the components of Chat App and Firebase backend.

Chat Application with React JS and Firebase consists of three main components on the Chat App´s main page including sign-up, log-in, send message form components while Side Bar component consists of rooms, adding new rooms and log-out component. The URL of the Chat App: https://decapp-9d394.firebaseapp.com/. Firebase components are acted as backend service for Chat App.

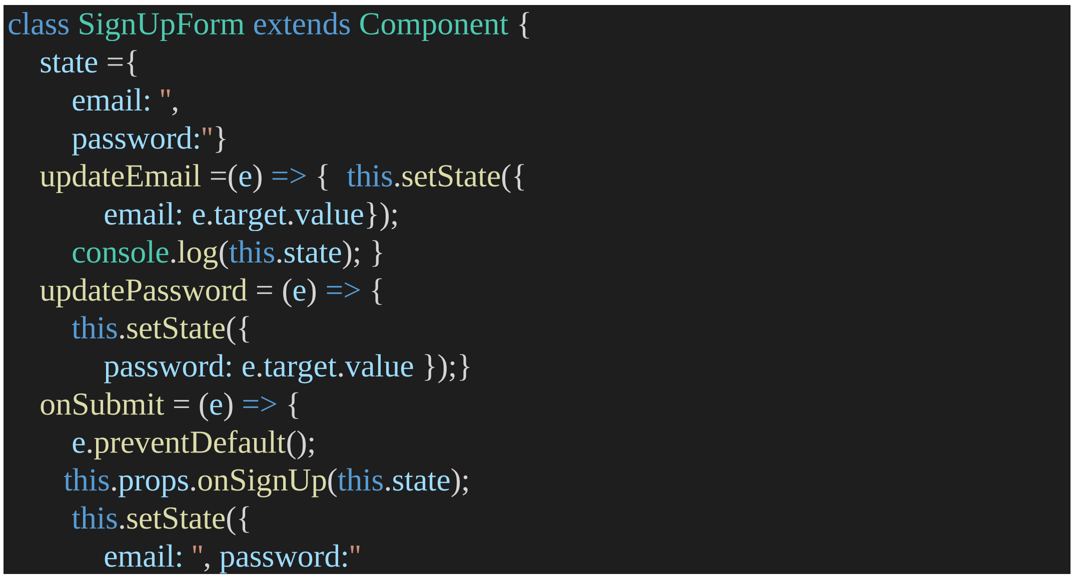
4.2.1 Components

In this section, different components of the Chat App are explained.

1 Sign Up component

Sign Up is the major component which is required by any new user to sign up first then login in the Chat App.

In Chat App, Sign Up component requires the user´s email and password. Chat App updates the email of a new user in the backend by updateEmail (e) method where e is the event argument method for SignUp submission. The password of new user is updated in the backend by updatePassword (e) function where the event handler is the email and password are handled by onSubmit (e) method and with the property (props) of onSignup by this. props.onSignUp. The method e. preventDefault () prevents the default action from occurring on “submit” button in Figure 21.



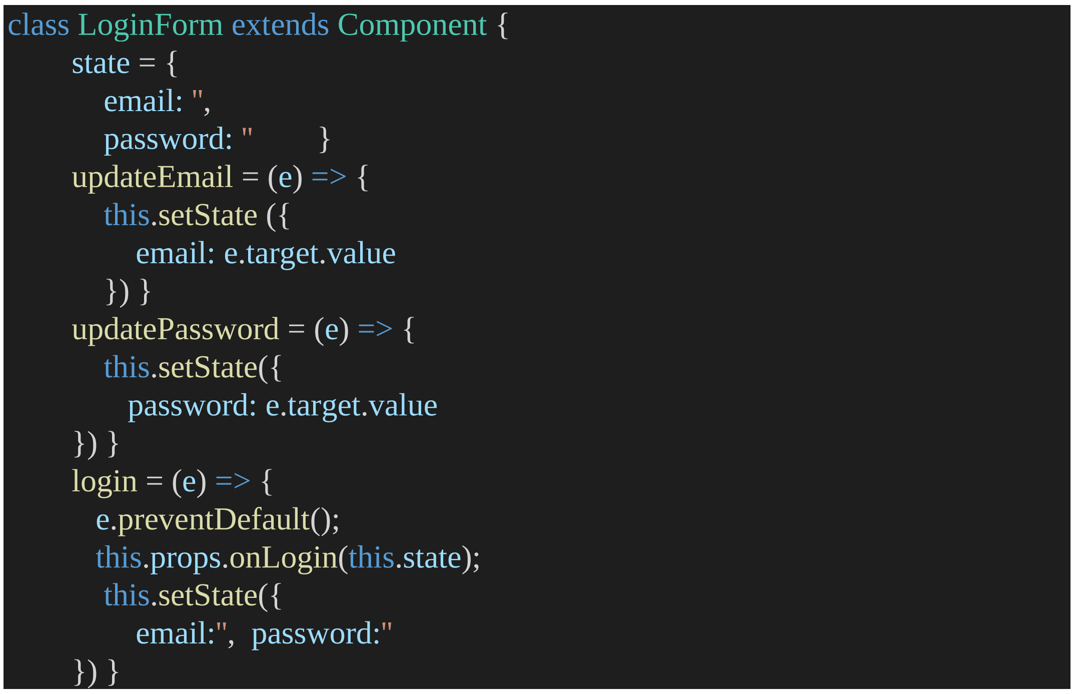
```
class SignUpForm extends Component {
   state ={
      email: '',
      password:''}
   updateEmail =(e) => {  this.setState({
         email: e.target.value});
      console.log(this.state); }
   updatePassword = (e) => {
      this.setState({
         password: e.target.value });}
   onSubmit = (e) => {
      e.preventDefault();
     this.props.onSignUp(this.state);
      this.setState({
         email: '', password:''
```

Figure 21. Screenshot of Sign Up component

2 Login Component

In Chat App, Sign Up component and Login component are quite similar, but there is a slight difference on onSubmit (e) and Login (e) event method.

The email and password are demanded in Login component. The email of a new user is updated in the backend by updateEmail (e) function where e is the event method. On the other hand, the password of new user is updated in the backend by updatePassword (e) function, where e the event method. A user´s email and password are handled by onLogin (e) method and with property (props) of onLogin by this. props. onLogin. The method e. preventDefault () prevents the default action from occurring on “login” button.



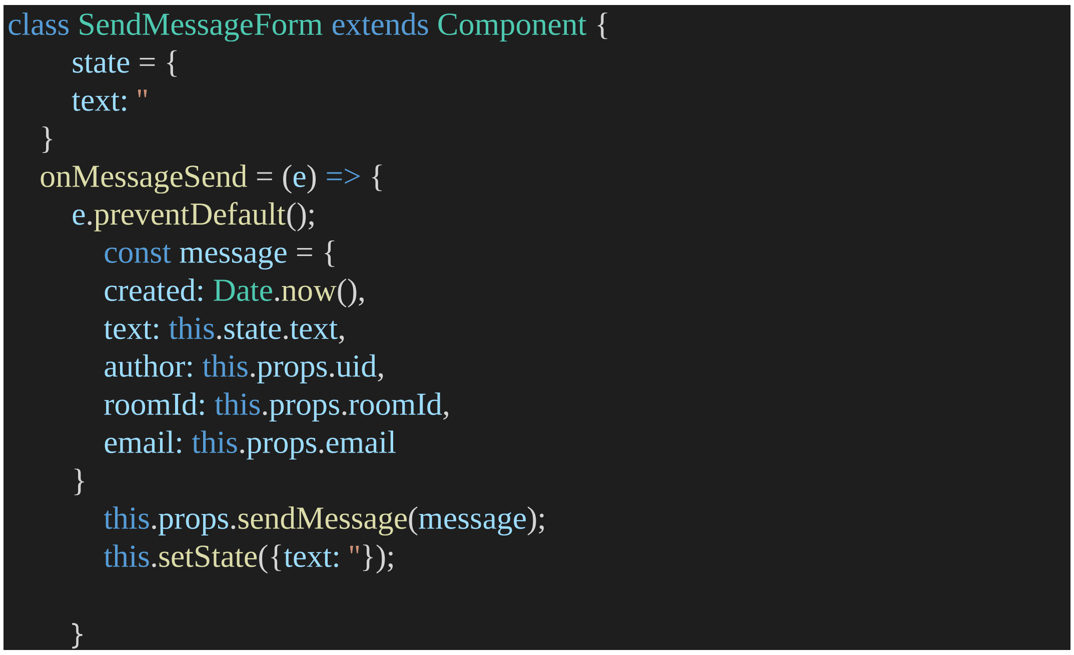
```
class LoginForm extends Component {
      state = {
         email: '',
         password: ''       }
      updateEmail = (e) => {
         this.setState ({
            email: e.target.value
         }) }
      updatePassword = (e) => {
         this.setState({
            password: e.target.value
      }) }
      login = (e) => {
         e.preventDefault();
         this.props.onLogin(this.state);
         this.setState({
            email:'',  password:''
      }) }
```

Figure 22. Screenshot of Login component

3 Send Message Form component

After the successful sign up and login, the message is written in the Send message Form.

A chat message is written in the Send message form, which is then handled by onMessageSend (e), where is e is the method for chat message submission, chat message time is displayed as created, the sender of the chat is shown as the author, the selected room of the user for the chat is shown as roomId, the user´s assigned email is shown as email, the property of the message is shown by this.props.sendMessage(message) while this.setState {text} in which setState updates the message in the output field. The method e. preventDefault () prevents the browser from refresh by entering the message in the message input field.



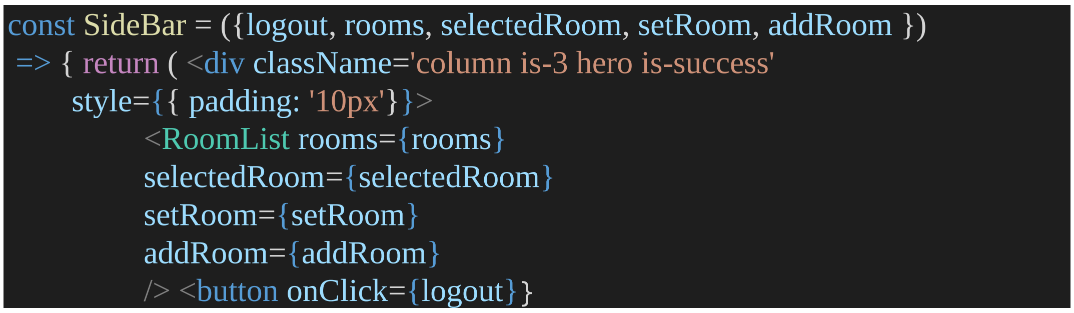
```
class SendMessageForm extends Component {
        state = {
        text: ''
    }
    onMessageSend = (e) => {
        e.preventDefault();
            const message = {
            created: Date.now(),
            text: this.state.text,
            author: this.props.uid,
            roomId: this.props.roomId,
            email: this.props.email
        }
            this.props.sendMessage(message);
            this.setState({text: ''});

        }
```

Figure 23. Screenshot of SendMessageForm

4 Side Bar component

Side Bar is other important component in Chat App. It comprises of a room list, adding a new room and log out component. All chat rooms are seen by all the users in this component as shown in the screenshot of the chat panel.

After a successful sign up and login, the chat room is selected, or a new room can be added by the user for the individual or group in order to privately chat as mentioned in the screenshot. Finally, a chat is quitted simply by pressing logout button, which is handled by onClick function in Figure 24.

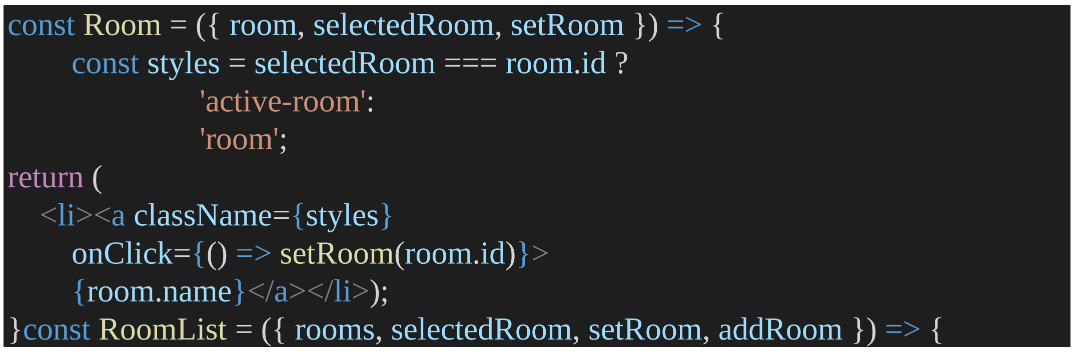


```
const SideBar = ({logout, rooms, selectedRoom, setRoom, addRoom })
 => { return ( <div className='column is-3 hero is-success'
      style={{ padding: '10px'}}>
            <RoomList rooms={rooms}
            selectedRoom={selectedRoom}
            setRoom={setRoom}
            addRoom={addRoom}
            /> <button onClick={logout}}
```

Figure 24. Screenshot of Side Bar component

5 Room List component

Room List is also a significant component because it allows the user to select the favorite chat room based on personal interest. It is dealt with onClick on desired setRoom with room id and room name as shown in the screenshot in Figure 25.

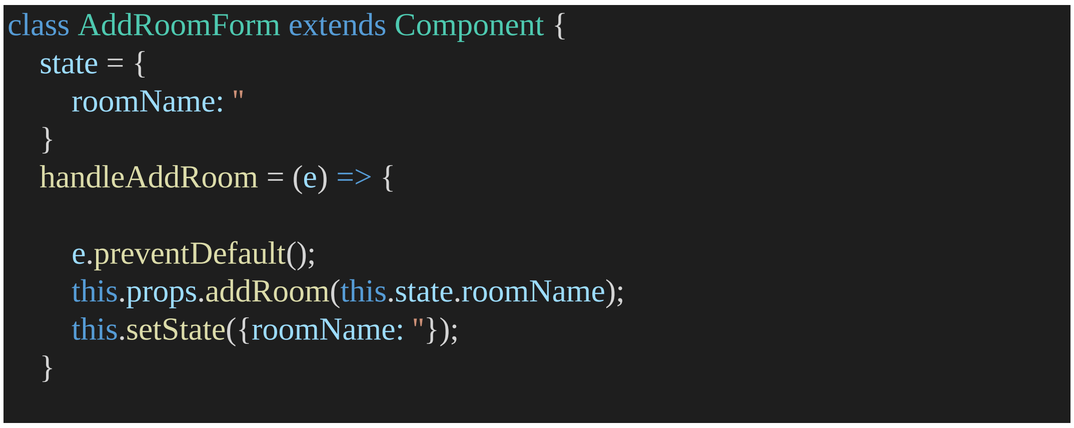


```
const Room = ({ room, selectedRoom, setRoom }) => {
      const styles = selectedRoom === room.id ?
                  'active-room':
                  'room';
return (
   <li><a className={styles}
      onClick={() => setRoom(room.id)}>
      {room.name}</a></li>);
}const RoomList = ({ rooms, selectedRoom, setRoom, addRoom }) => {
```

Figure 25. Screenshot of Room List component

6 Add Room Form component

Add New Room is other essential component which enables user to create a new room for the individual or a group for chat purposes. It is controlled by handleAddRoom (e) method and new room name is created by property (props) of addRoom. The method e. preventDefault () cancels the itself action on “message send” button as shown in the screenshot of Add Room Form in Figure 26.



```
class AddRoomForm extends Component {
   state = {
      roomName: ''
   }
   handleAddRoom = (e) => {

      e.preventDefault();
      this.props.addRoom(this.state.roomName);
      this.setState({roomName: ''});
   }
```

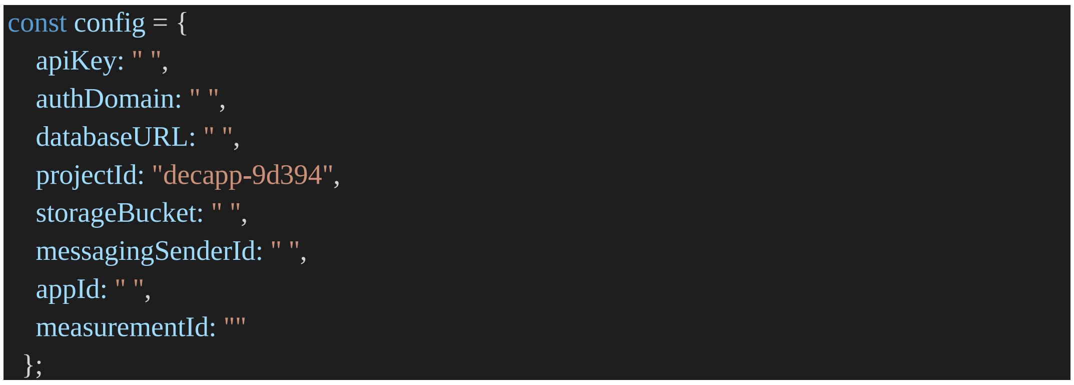
Figure 26. Screenshot of Add Room Form

4.2.2 Backend component

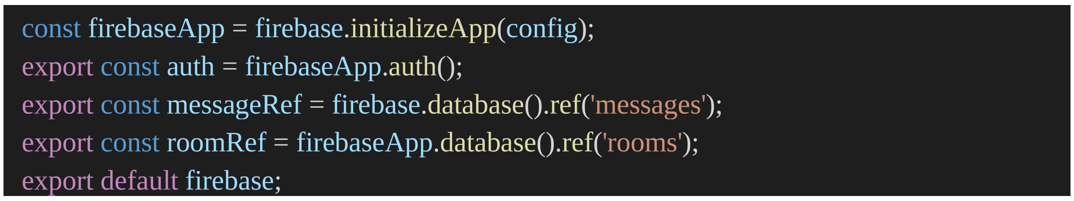
Firebase is the Firebase Cloud messaging software cross-platform. When a full-fledged application is built a backend, service is required for the purpose. Firebase was proven a good choice in the Chat App project because it saved time, which allowed me to focus on development. Server code is not required because data is stored in JSON file or JSON objects in the Firebase, but it is a ready-made API for developing and implementation of applications. It consists of three vital components such as fire, firebaserc and firebase.json.

1 Fire.js

The screenshot in Figure 27 shows Firebase backend in the Chat App project. It has unique digits of ApplicationKey, authenticationDomain, databaseURL, projectID, storageBucket (data storage), messagingSenderId, ApplicationID and measurementID.



```
const config = {
   apiKey: " ",
   authDomain: " ",
   databaseURL: " ",
   projectId: "decapp-9d394",
   storageBucket: " ",
   messagingSenderId: " ",
   appId: " ",
   measurementId: ""
 };
```



```
const firebaseApp = firebase.initializeApp(config);
export const auth = firebaseApp.auth();
export const messageRef = firebase.database().ref('messages');
export const roomRef = firebaseApp.database().ref('rooms');
export default firebase;
```

Figure 27. Screenshot of Firebase backend

Firebase backend is initialized Chat App with function as:

```
const firebaseApp = firebase.initializeApp(config);
```

User email and password are fetched with below function by Firebase:

```
export const auth = firebaseApp.auth();
```

Chat data is stored with below function by Firebase:

```
export const messageRef = firebase.database().ref('messages');
```

Selected rooms and room are created with below function by Firebase:

```
export const roomRef = firebaseApp.database().ref('rooms');
```

2 Firebaserc.js

Project and Chat App unique id is served by Firebaserc.js in VS code as below:

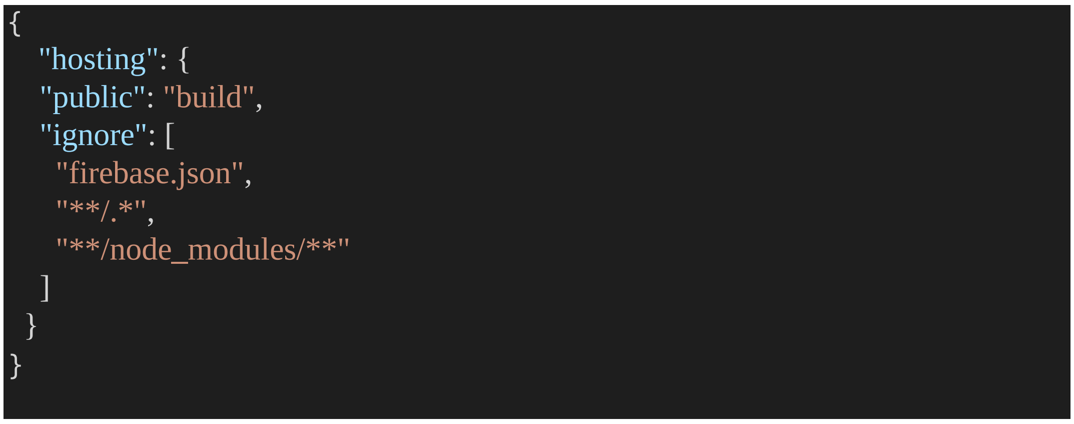
{

"projects": {

"default": "decapp-9d394”}}

3 Firebase.json

Firebase backend is NoSQL, so that data is stored in JSON file or JSON objects, Chat App is hosted by Firebase backend services, which is shown as “hosting” and “build” is shown the production mode of the project in Figure 28.



```
{
  "hosting": {
  "public": "build",
  "ignore": [
    "firebase.json",
    "**/.*",
    "**/node_modules/**"
  ]
 }
}
```

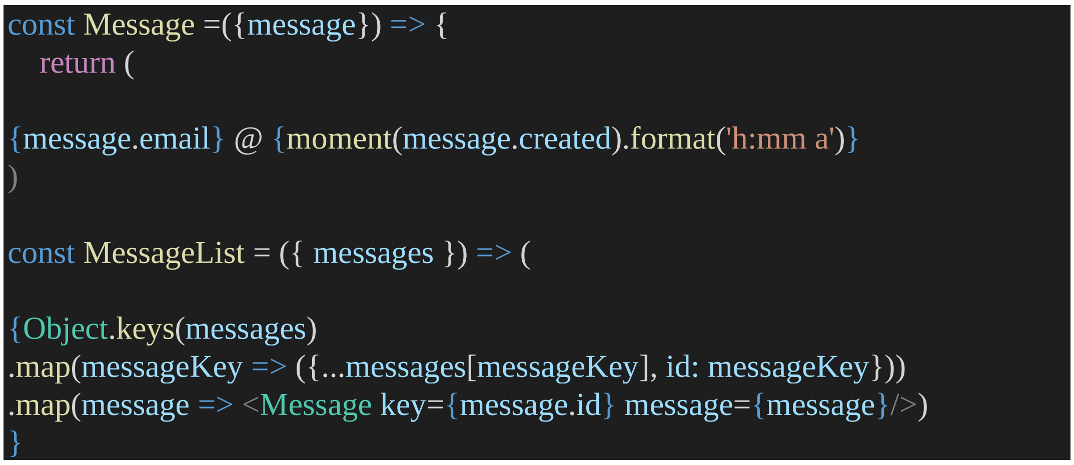
Figure 28. Screenshot of Firebase.json

5 IMPLEMENTATIONS

The implementations of the Chat App are discussed with the codes in this chapter.

5.1 Send Chat

This topic discusses how users sends chat message in Chat App. For better understanding, it is recommended to learn first React Map, which is a JavaScript library function, the map method represented as. map () method creates a new array by calling a provided function on every messageKey in project. The Key is a special string attribute which is required to include creating lists of messages in a project. The important feature in JSX is the spread operator attribute. The object such as message ´s property is provided by JSX spread operator attributes in the project which is represented by … (three dots).



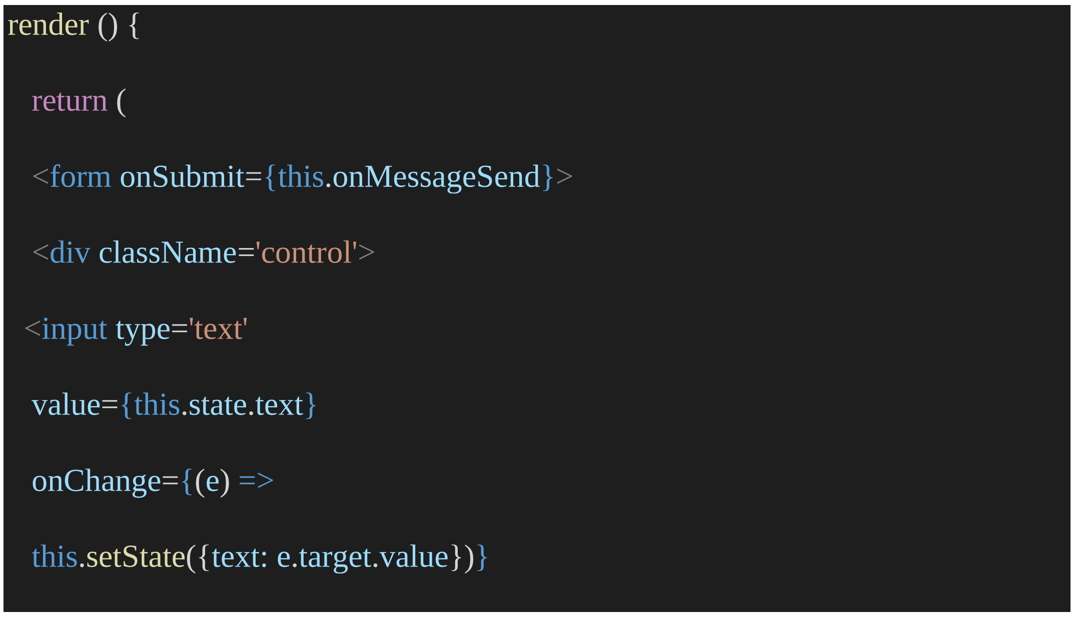
```
const Message =({message}) => {
   return (

{message.email} @ {moment(message.created).format('h:mm a')}
)

const MessageList = ({ messages }) => (

{Object.keys(messages)
.map(messageKey => ({...messages[messageKey], id: messageKey}))
.map(message => <Message key={message.id} message={message}/>)
}
```

Figure 29. Screenshot of MessageList

In order to reach this goal, two constructors are built namely as const MessageList and const Message as in the above screenshot in Figure 29 of Chat App also onChange (e) event argument is modified in the render () function of SendMessageForm. At first, a constructor is built as MessageList, in which message is sent by a user as an object like a Key string attribute value and applying map () method on every messageKey then spread Key string attribute (…) is applied on every messageKey and id. Again, map () method is applied over the messages such as id of the message and message text.

Another constructor is built as Message by which message is returned with email id, what time message is created with the proper time format. Time format is accomplished with moment.js, which must be installed first in npm.

Now, onChange (e) the event handler is modified in SendMessageForm. Whenever the change in input field as text is changed, the event of target value is changed which is updated the state by this. setState as shown in the below screenshot in Figure 30.

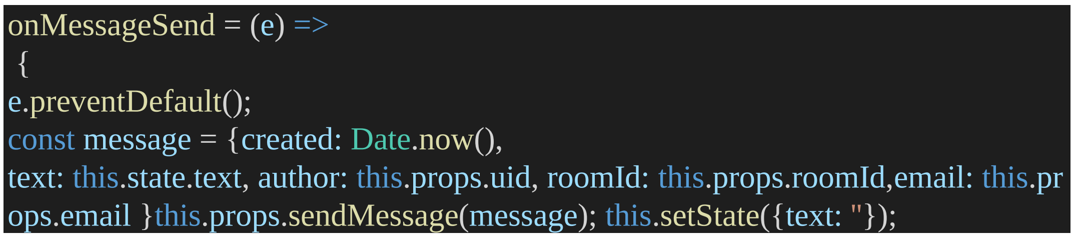


```
render () {

  return (

  <form onSubmit={this.onMessageSend}>

  <div className='control'>

 <input type='text'

  value={this.state.text}

  onChange={(e) =>

  this.setState({text: e.target.value})}
```

Figure 30. Screenshot of onChange function

Finally, onMessageSend (e) an event handler is built, which is prevented from refreshing the whole browser by simply pressing Send Message From.

Const message is created as a constructor, showing the user´s message created time, text state, author property of uid, the roomId where chat is occurred, the property of email by this function, the property of sendMessage as message and text is updated by setState as shown in the screenshot in Figure 31.



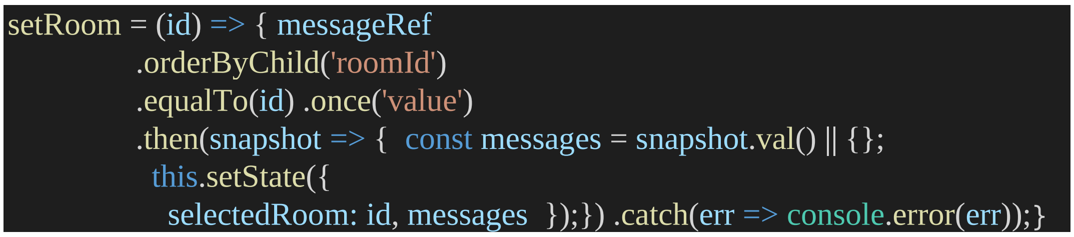
```
onMessageSend = (e) =>
 {
e.preventDefault();
const message = {created: Date.now(),
text: this.state.text, author: this.props.uid, roomId: this.props.roomId,email: this.pr
ops.email }this.props.sendMessage(message); this.setState({text: ''});
```

Figure 31. Screenshot of onMessageSend function

5.2 Receive Chat

This section explains how the user´s chat is refreshed automatically. In order to achieve this purpose, Firebase´s listener method “on” and “child_added, orderByChild and key” methods are used. These methods are implemented and only adding some conditions in the selected room is used in the project.

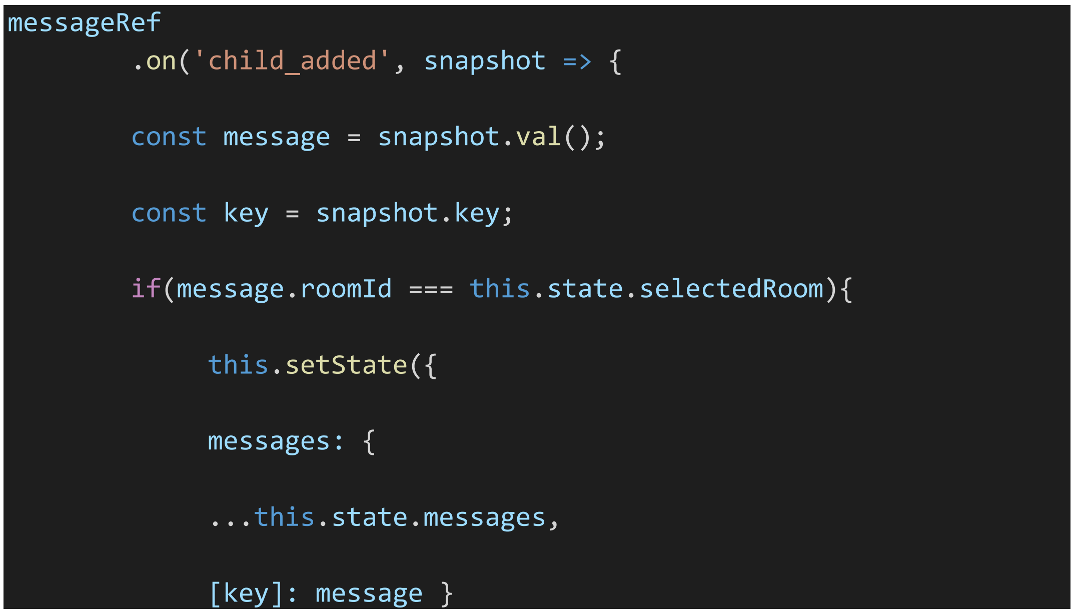
The first step is: setRoom (id), .orderByChild which is roomId, .equalTo (id) for the room, .once (‘value’) for one time message, .then(snapshot) is a snapshot of the value or message is taken an empty {} object means that a value can be nothing, state of the selectedRoom and corresponding messages is updated by this.setState while .catch(err => catches error in the browser console as shown in Figure 32.



```
setRoom = (id) => { messageRef
            .orderByChild('roomId')
            .equalTo(id) .once('value')
            .then(snapshot => {  const messages = snapshot.val() || {};
              this.setState({
                selectedRoom: id, messages  });}) .catch(err => console.error(err));}
```

Figure 32. Screenshot of snapshot setRoom

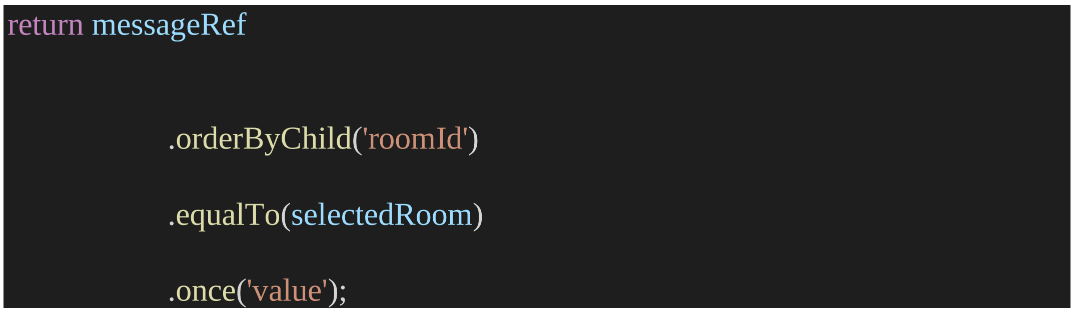
The second step is configuring the messageRef. It is accomplished by adding Firebase´s “on” listener, which takes child_added method and snapshot, snapshot of val () is taken by message constructor while a snapshot of the key can be a value or a message of the user which is taken by the key method constructor. MessageRef is updated by setState, message state and key such as message as in Figure 33-34.



```
messageRef
        .on('child_added', snapshot => {

        const message = snapshot.val();

        const key = snapshot.key;

        if(message.roomId === this.state.selectedRoom){

             this.setState({

             messages: {

             ...this.state.messages,

             [key]: message }
```

Figure 33. Screenshot of messageRef

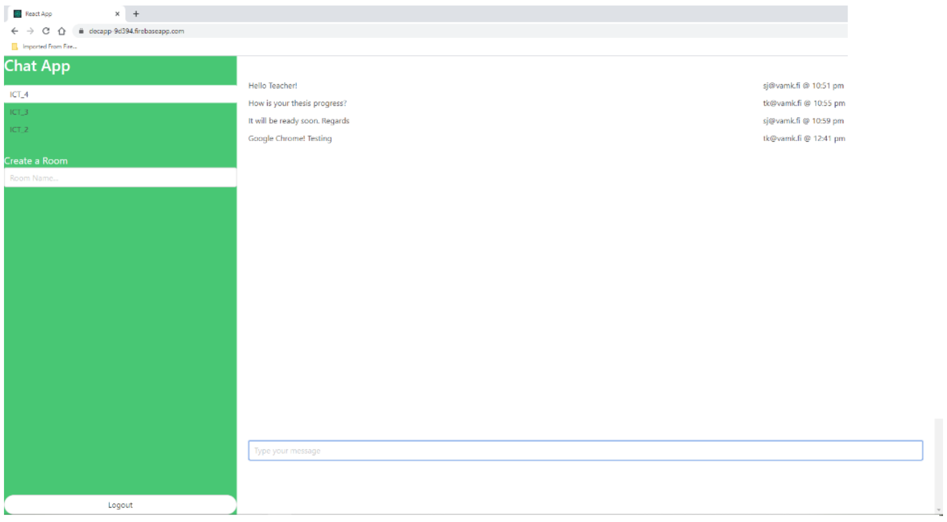
The third step, return message, on refresh is promised by Firebase backend as below in Figure 34.



```
return messageRef


          .orderByChild('roomId')

          .equalTo(selectedRoom)

          .once('value');
```

Figure 34. Screenshot of Firebase backend promise

The final screen shot of the sender´s and the receiver´s chat message with user email and time stamp is shown in Figure 35.



Hello Teacher! sj@vamk.fi @ 10:51 pm
How is your thesis progress? tk@vamk.fi @ 10:55 pm
It will be ready soon. Regards sj@vamk.fi @ 10:59 pm
Google Chrome! Testing tk@vamk.fi @ 12:41 pm

Figure 35. Screenshot of Receiver chat message

6 TESTING

In this chapter, test cases of the application are examined.

6.1 System Testing Cases

1 Sign Up new User

• Test Step

Go to the Chat App page and Sign Up with a valid email and a six-character password then click “Sign Up”

• Expected Result

It should allow the user and enter the chat panel

• Actual Outcome

A new user is entered in the chat panel.

2 Check New Created User in Firebase database

• Test Step

Go to the Firebase console project with Gmail and password. Check the Firebase´s Authentication users

• Expected Result

Newly Created User email, created date and unique uid in Firebase

• Actual Outcome.

A new Created User is available in Firebase Authentication user list.

3 Login new User without Sign Up

• Test Step

Go to the Chat App page and Login with a valid email and a six-character password and click “Sign Up”

• Expected result

It should not allow the user to enter the chat panel

• Actual Outcome: User is not allowed to enter chat panel.

4 User messages with email and time

• Test Step

Logged in user writes the message in Send Message Form

• Expected Result

Users writes any message and receiver views the message with sender´s email and time

• Actual Outcome

Sender´s email id with time is viewed by receiver.

5 Logged in User chat data in Firebase

• Test Step

Go to the Firebase console project with Gmail and password. Check the Firebase´s real-time database data

• Expected Result

User´s chat messages data available in the real-time database

• Actual Outcome

Chat messages data of user are available in Firebase database.

6 Logged in User able to create New Chat Room

• Test Step

User assigns a name to New Chat Room in empty field under the Rooms

• Expected Result

User creates a Room

• Actual Outcome

New Room is created by user.

7 Sign Up new User with less than six-character password

•Test Step: Go to the Chat App page and Sign Up with a valid email and less than six-character password and click “Sign Up”

• Expected Outcome

Chat App does not allow user to sign up

• Actual Outcome

Sign up is not successful.

8 Sign Up new User with mobile number

• Test Step

Go to the Chat App page and Login with mobile number and six-character password and click “Sign Up”

• Expected Result

User should not able to sign up because it is defined in Firebase Authentication user can only sign up with a valid email and password

• Actual Outcome

Sign up is not successful.

9 Delete the user chat data in Firebase database

• Test Step

Go to the Firebase console project with Gmail and password. Check the Firebase´s real-time database data. Delete the chat history under the selected room

• Expected Result

Administrator allows to delete user chat messages or data that will be also deleted in the selected room of the chat panel

• Actual Outcome

The messages or data is lost by deleting in the Firebase real-time database.

10 Logged in user Refreshes chat automatically

• Test Step

Logged in user writes the message in Send Message Form

• Expected Result

Chat panel windows Refreshes automatically and is also observed by user

• Actual Outcome: Window is Refreshed in the chat panel automatically.

7 CONCLUSIONS

React JS is one of the most popular and powerful front-end technology around the world, today. It offers excellent performance in any application. In addition to this, React JS offers good compatibility on different platforms, browsers and devices. After developing the Chat App, it is clear that React JS is an easy to learn and convenient to implement.

Firebase played a crucial role in providing real-time database backend service. It is easy to use because extra code writing for the server is not required while it is a ready-made API. Visual Studio Code was an excellent IDE in this project.

I conclude by saying that this thesis application is a simple project but needs development in the future if the developer gets an opportunity to put effort into it.

REFERENCES

/1/ SMS, Instant message – Last access 27/02/2020
https://www.lifewire.com/difference-between-chat-and-instant-messaging-3969422

/2/ SMS, Instant message – Last access 27/02/2020
https://blog.mysms.com/sms-vs-messenger-why-you-should-not-have-to-decide.html

/3/ SMS, Instant message – Last access 27/02/2020
https://www.trickstrend.com/popular-instant-messenger-apps/

/4/ Chat Apps pictures – Last access 27/02/2020
http://xvdvx.blogspot.com/2014/08/en-iyi-mobil-chat-sohbet-programi.html

/5/ Chat Features – Last access 27/02/2020
https://www.userlike.com/en/blog/live-chat-advantages-disadvantages

/6/ HTML – Last access 27/02/2020
https://www.educba.com/advantages-of-html/

/7/ HTML – Last access 27/02/2020
https://www.quora.com/What-are-the-benefits-of-using-HTML

/8/ HTML – Last access 27/02/2020
https://www.babujitechnolife.in/2019/05/html-ladvantages-and-disadvantages/

/9/ CSS – Last access 27/02/2020
https://www.educba.com/uses-of-css/

/10/ CSS – Last access 27/02/2020
https://www.educba.com/html-vs-css/

/11/ CSS – Last access 27/02/2020
https://skillcrush.com/blog/css/

/12/ CSS – Last access 27/02/2020
https://www.hostinger.com/tutorials/difference-between-inline-external-and-internal-css

/13/ CSS Features – Last access 27/02/2020
https://connectusfund.org/6-advantages-and-disadvantages-of-cascading-style-sheets

/14/ BULMA – Last access 27/02/2020
https://mobiosolutions.com/what-is-bulma-advantages-disadvantages-of-using-bulma-2/
/15/ Visual Studio Code – Last access 27/02/2020
https://code.visualstudio.com/
/16/ Visual Studio Code – Last access 27/02/2020
https://en.wikipedia.org/wiki/Visual_Studio_Code
/17/ React JS Features – Last access 27/02/2020
https://curatti.com/pros-cons-reactjs/
/18/ React JS Features – Last access 27/02/2020
https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html
/19/ React JS Features – Last access 27/02/2020
https://mindmajix.com/reactjs-interview-questions
/20/ DOM – Last access 27/02/2020
https://www.educative.io/edpresso/what-is-dom
/21/ DOM – Last access 27/02/2020
https://www.w3schools.com/js/js_htmldom.asp
/22/ Life Cycle of React JS – Last access 27/02/2020
https://hackernoon.com/reactjs-component-lifecycle-methods-a-deep-dive-38275d9d13c0
/23/ User Interface – Last access 27/02/2020
https://www.w3computing.com/systemsanalysis/types-user-interface/
/24/ User Interface – Last access 27/02/2020
https://searchapparchitecture.techtarget.com/definition/user-interface-UI
/25/ Life Cycle of React JS – Last access 27/02/2020
https://www.freecodecamp.org/news/how-to-understand-a-components-lifecycle-methods-in-reactjs-e1a609840630/
/26/ ES6 – Last access 27/02/2020
https://www.javatpoint.com/es6
/27/ FLUX And REDUX – Last access 27/02/2020
https://medium.com/@sidathasiri/flux-and-redux-f6c9560997d7

/28/ FLUX and REDUX – Last access 27/02/2020
https://www.educba.com/redux-vs-flux
/29/ FLUX and REDUX – Last access 27/02/2020
https://www.dropsource.com/blog/immutable-data-in-react-and-flux/
/30/ REDUX picture – Last access 27/02/2020
https://blog.codecentric.de/en/2017/12/developing-modern-offline-apps-reactjs-redux-electron-part-3-reactjs-redux-basics/
/31/MVC And FLUX picture – Last access 27/02/2020
https://www.clariontech.com/blog/mvc-vs-flux-vs-redux-the-real-differences
/32/ FIREBASE – Last access 27/02/2020
https://howtofirebase.com/what-is-firebase-fcb8614ba442
/33/ FIREBASE – Last access 27/02/2020
https://firebase.google.com/
/34/ FIREBASE – Last access 27/02/2020
https://en.wikipedia.org/wiki/Firebase
/35/ FIREBASE – Last access 27/02/2020
http://www.itinfotech.in/database/difference-between-firebase-mysql-and-sqlite-database/
/36/ FIREBASE picture – Last access 27/02/2020
https://hackernoon.com/make-predictions-on-a-custom-automl-model-with-cloud-functions-and-firebase-ec3868a1671b
/37/ FIREBASE Advantages – Last access 27/02/2020
https://www.altexsoft.com/blog/firebase-review-pros-cons-alternatives/
/38/ NodeJS – Last access 27/02/2020
https://medium.com/codingthesmartway-com-blog/async-programming-with-/javascript-callbacks-promises-and-async-await-980e3f144185
/39/ NodeJS – Last access 27/02/2020
https://www.educba.com/why-use-node-js/
/40/ Use Case Diagram – Last access 27/02/2020
https://www.lucidchart.com/pages/uml-use-case-diagram
/41/ Use Case Diagram – Last access 27/02/2020
https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

/42/ Use Case Diagram – Last access 27/02/2020

https://www.draw.io/