**Day 1**

React Native is an open source framework for building mobile applications using JavaScript and React. It allows developers to build native mobile apps for iOS and android platforms using a single codebase.

**Installing dependencies**

To install dependencies for a React Native project, Node.js and npm (Node Package Manager) is needed to be installed on a system.

**Setting up the Server**

Setting up a server for a React Native project involves creating a backend that handles requests and serves data to the app. There are various ways to set up a server, and the choice depends on the project requirements and preferences.

**Modifying npm scripts**

 Modifying npm scripts is a common task when working on a project. We can add, edit, or remove scripts in the package.json file.  
/\* package.json \*/  
"scripts": {  
"dev": "node server.js",  
"build": "next build",  
"start": "NODE\_ENV=production node server.js"  
}.

**Day 1 Reflections**

React Native is an open source framework for building mobile applications using JavaScript and React. It allows developers to build native mobile apps for iOS and android platforms using a single codebase.

**Day 2**

React Native provides a set of pre-built, platform-specific components that are rendered as native UI elements. This allows for a highly responsive and performant user interface that looks and feels like a native app.

**More about React Native**

 React Native has a thriving community and a rich ecosystem of libraries, tools, and plugins that extend its capabilities and make it easier to integrate with other technologies.

**Building the Index Page**

Building an index page for a React Native project involves creating a component that serves as the main entry point of application. This component typically contains the structure and layout of the app, including navigation and the display of various UI elements.

**The Pusher App**

Pusher is a cloud-based platform that provides real-time communication APIs and services. It allows developers to easily add real-time functionality to their web and mobile applications without having to manage the underlying infrastructure.

**My Views on the Day**

 Building the index page and creating the pusher app are the two important and difficult activities of the day. They need a lot of time to be done and understood and to grasp its content.

**Day 2 Reflections**

 Pusher is a cloud-based platform that provides real-time communication APIs and services. It allows developers to easily add real-time functionality to their web and mobile applications without having to manage the underlying infrastructure.

**Day 3**

Building a chat feature in a React Native application involves several steps, including setting up a backend server, handling real-time communication, and designing the user interface.

**React Native Chats**

Building a chat feature in a React Native application involves several steps, including setting up a backend server, handling real-time communication, and designing the user interface.

**Building the chat component**

Building a chat component in React Native involves creating UI elements for displaying messages, input fields for composing messages, and logic for sending and receiving messages.

**Building the chat component: Part 2**

To enable real-time messaging, we need to set up a WebSocket server on a backend.

**Adding the chat component to the index page**

import React from 'react';  
import ChatComponent from './ChatComponent';  
const Index = () => {  
return <ChatComponent />;  
};  
export default Index;

**Adding the chat component to the index page**

 const Index = () => {  
return (  
<View style={{ flex: 1 }}>  
<ChatComponent />  
</View>  
);  
};

**My Views on the Day**

 Adding chat components to the index page and building the chat component are the important features of the day and beneficial, they were not easy and they need more time.

**Day 3 Reflections**

Building a chat component in React Native involves creating UI elements for displaying messages, input fields for composing messages, and logic for sending and receiving messages.

**Day 4**

**Displaying the chat message and deploying your chat app**

To display chat messages in React Native app, integrating the chat component that have been built is needed. Assuming that the previous steps have been followed, there should be a ChatComponent that handles sending and receiving messages.

**Chat App Reflections**

I have enjoyed adding the chatComponents more when creating the chat app, the challenges appeared when displaying the messages but solved it by doing more research on it. The process of creating a chat app can be enjoyable due to the immediate feedback I get when messages are sent and received in real time. It's satisfying to see a dynamic and interactive application come to life.

**My Views on the Day**

 Displaying messages on chat app is the important feature of the day and beneficial, it was challenging and needs more time.

**Day 4 Reflections**

The process of creating a chat app can be enjoyable due to the immediate feedback I get when messages are sent and received in real time. It's satisfying to see a dynamic and interactive application come to life.

**Day 5**

Working towards the week’s content and implementing them on our websites.