Department of Engineering Technology

Foundation University Islamabad

School of Science and Technology

Course: Mobile Application Lab

Lab Manual 9

Name: Fatima Iqbal

Roll no: Fall-23-A-010

Topic: **Open ended lab**

Performance			Communication		
Description	**Total Marks**	**Marks Obtained**	**Description**	**Total Marks**	**Marks Obtaine d**
Implementation of CRUD Operation	5		Documentation and Reporting	5	
Total Marks obtained					

Lab Task Description:

You are hired by a startup to create a simple mobile app prototype for managing posts and users.

The backend is not yet ready, so your team must use a fake REST API for now (such as

https://jsonplaceholder.typicode.com/, https://reqres.in/, or any other test API).

You are also encouraged to create your own custom fake APIs using tools like Mocky.io,

Beeceptor, or JSON Server to better reflect your app&#39;s specific use case.

Your goal is to:

Explore the fake API using Postman to understand endpoints.
Create a React Native app that fetches and displays a list of posts and users.
Implement features to add, edit, and delete posts using POST/PUT/DELETE requests.
Handle API loading states and errors gracefully.

You are free to design the UI layout, choose how you show data (e.g., FlatList, cards, modals),

and handle forms. Creativity and proper API usage will be assessed.

React Native App for Managing Posts and Users (Using Fake REST API)

1. Introduction

In this lab task, we are given a real-world scenario where we are hired by a **startup** to design a **mobile app prototype** for managing posts and users. Since the actual backend is not ready yet, we will use a **fake REST API** like JSONPlaceholder or Reqres for testing and development.

The main goal of this task is to learn how to:

- Connect a **React Native app** with a fake API.
- **Fetch data** (GET) to show posts and users.
- **Add** new posts (POST), **edit** existing ones (PUT), and **delete** them (DELETE).
- Handle **loading** indicators and **error** messages properly.
- Build a **modern and responsive UI** using components like `FlatList`, `Card`, `Modal`, and forms.

To make this app functional and user-friendly, we will explore the fake API endpoints using **Postman** to understand how the API works (like what kind of data it sends and receives). Later, we'll use that API in our React Native app with `axios`.

This lab task helps us practice working with **real-world APIs**, managing **state** and **UI** in React Native, and applying **CRUD operations** (Create, Read, Update, Delete) using simple HTTP requests.

By completing this task, we gain experience in:

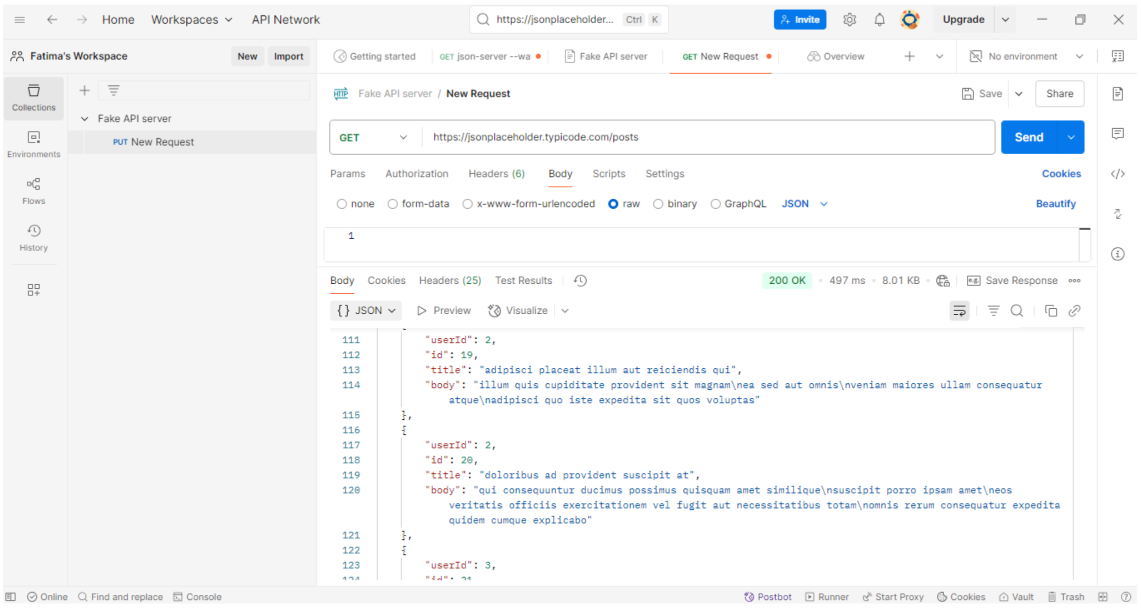
- API integration
- React Native development
- Error handling
- UI/UX design using React Native Paper
- Practical use of modal forms and interactive lists

In the next sections, we'll break down the project with code snippets and simple explanations to show how each feature works.

1. Explore the Fake API

- Use https://jsonplaceholder.typicode.com/
    - o `/posts` → GET all posts, POST new post, PUT to update, DELETE to remove
    - o `/users` → GET list of users
- Test these endpoints in Postman to understand responses.

1.1. GET:-

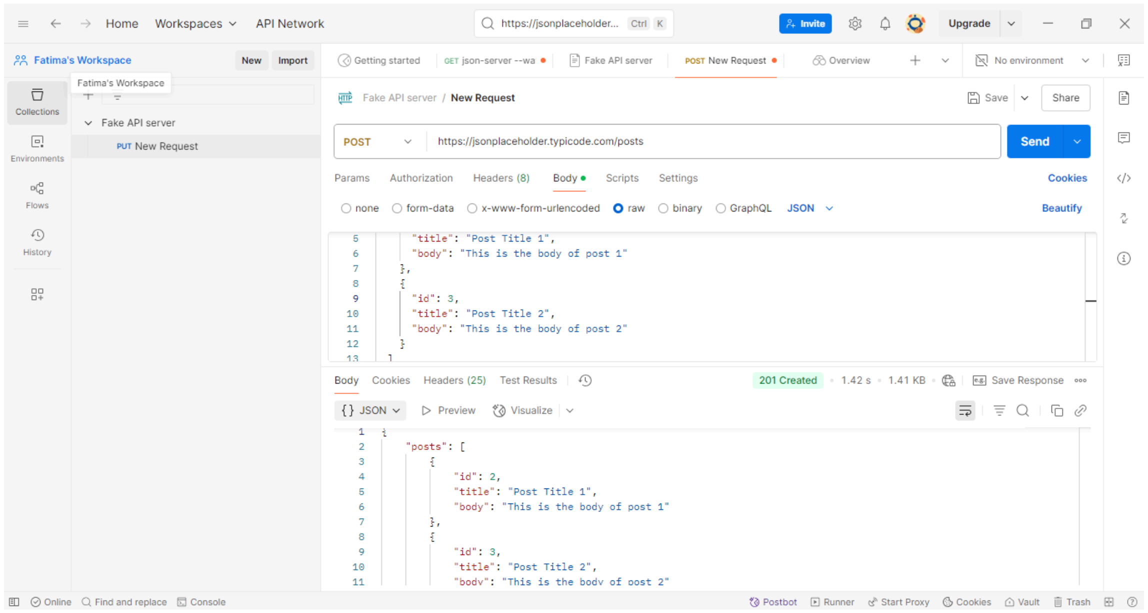


GET https://jsonplaceholder.typicode.com/posts

200 OK • 497 ms • 8.01 KB

```
        "userId": 2,
        "id": 19,
        "title": "adipisci placeat illum aut reiciendis qui",
        "body": "illum quis cupiditate provident sit magnam\nea sed aut omnis\nveniam maiores ullam consequatur atque\nadipisci quo iste expedita sit quos voluptas"
    },
    {
        "userId": 2,
        "id": 20,
        "title": "doloribus ad provident suscipit at",
        "body": "qui consequuntur ducimus possimus quisquam amet similique\nsuscipit porro ipsam amet\neos veritatis officiis exercitationem vel fugit aut necessitatibus totam\nomnis rerum consequatur expedita quidem cumque explicabo"
    },
    {
        "userId": 3,
```

1.2.POST:-


Fatima's Workspace — Fake API server / New Request

POST https://jsonplaceholder.typicode.com/posts

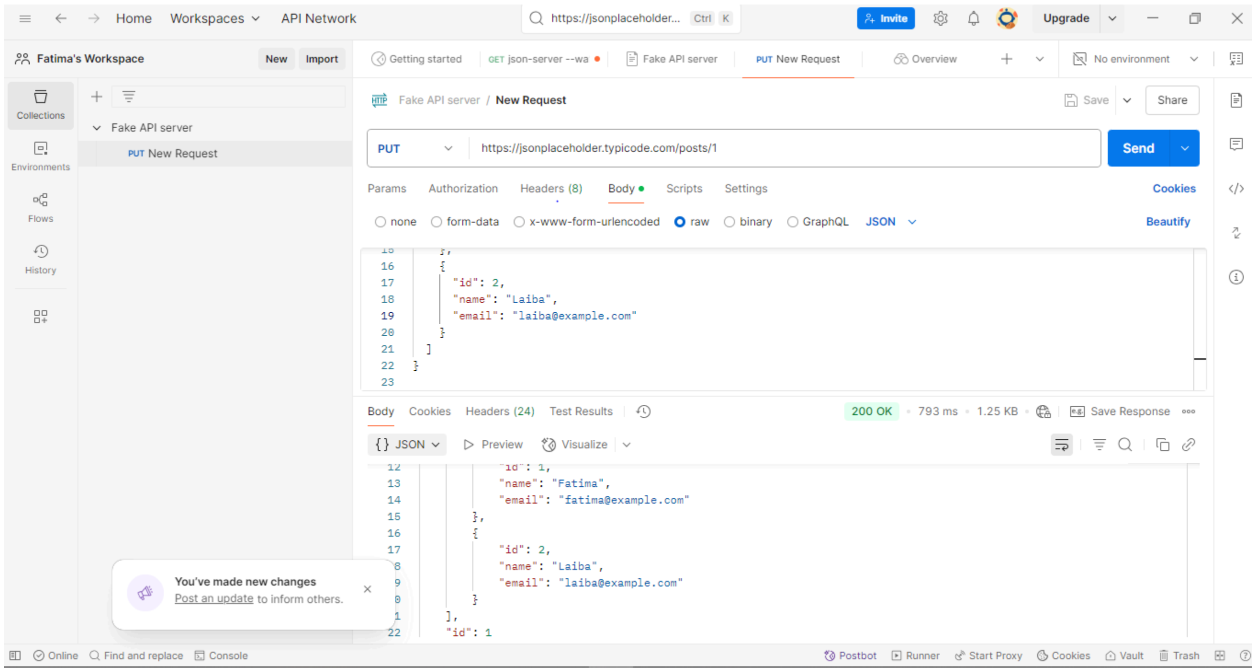
Body: raw JSON

```
    "title": "Post Title 1",
    "body": "This is the body of post 1"
  },
  {
    "id": 3,
    "title": "Post Title 2",
    "body": "This is the body of post 2"
  }
]
```

201 Created · 1.42 s · 1.41 KB

```
{
    "posts": [
        {
            "id": 2,
            "title": "Post Title 1",
            "body": "This is the body of post 1"
        },
        {
            "id": 3,
            "title": "Post Title 2",
            "body": "This is the body of post 2"
```

1.3.PUT:-


Fatima's Workspace — Fake API server / New Request

PUT https://jsonplaceholder.typicode.com/posts/1

Body: raw JSON

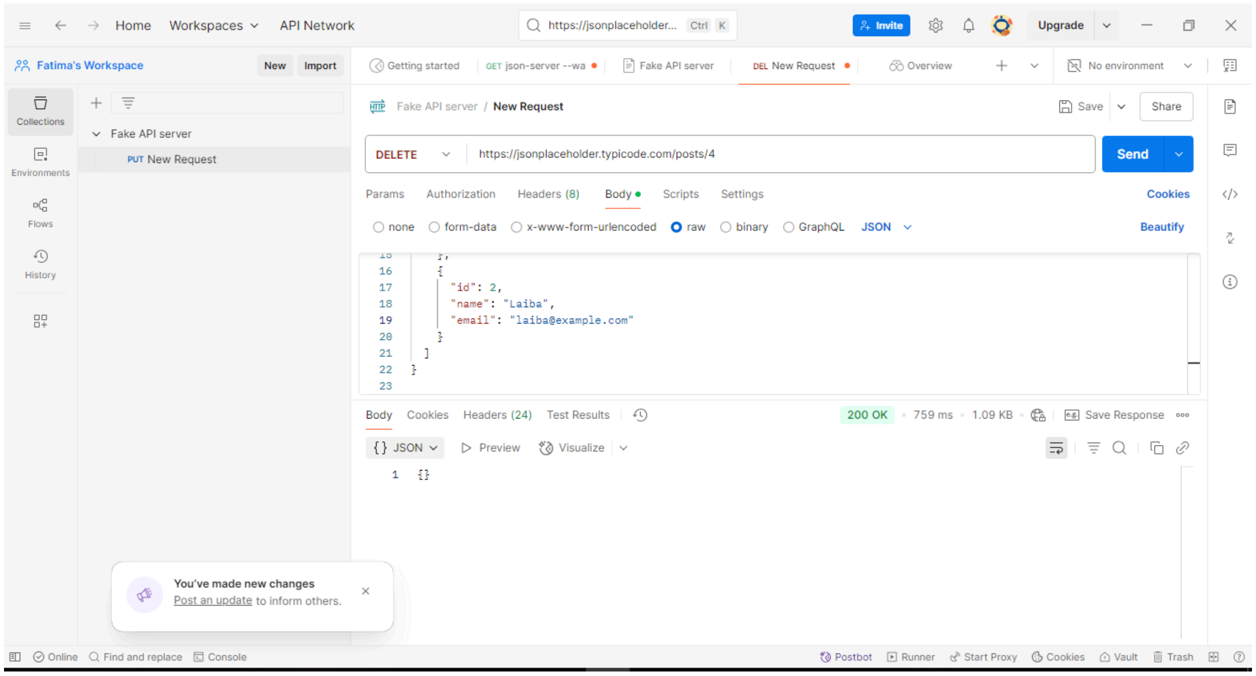
```
    },
    {
      "id": 2,
      "name": "Laiba",
      "email": "laiba@example.com"
    }
  ]
}
```

200 OK · 793 ms · 1.25 KB

```
            "id": 1,
            "name": "Fatima",
            "email": "fatima@example.com"
        },
        {
            "id": 2,
            "name": "Laiba",
            "email": "laiba@example.com"
        }
    ],
    "id": 1
```

You've made new changes
Post an update to inform others.

1.4.DELETE:-



Fake API server / New Request

DELETE https://jsonplaceholder.typicode.com/posts/4

none ○ form-data ○ x-www-form-urlencoded ○ [✓ raw] ○ binary ○ GraphQL JSON

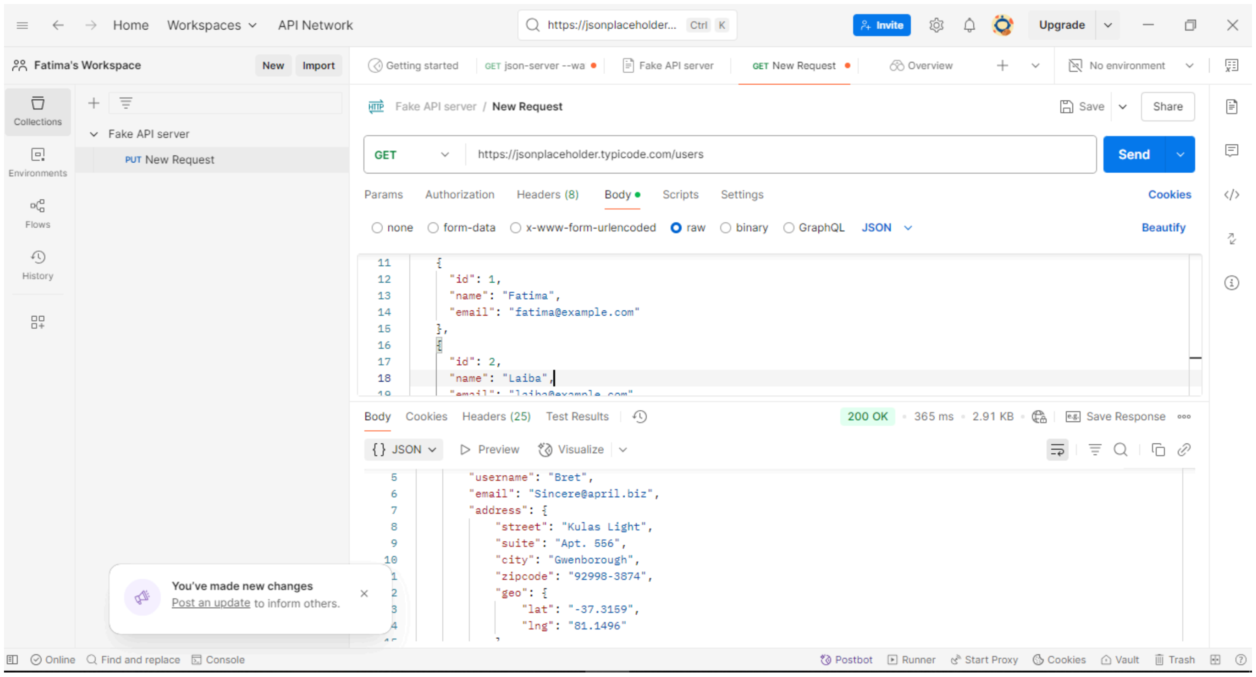
```
    },
    {
      "id": 2,
      "name": "Laiba",
      "email": "laiba@example.com"
    }
  ]
}
```

Body Cookies Headers (24) Test Results — 200 OK • 759 ms • 1.09 KB

```
{}
```

1.5. Similarly, for users:

- **GET** `/users` – fetches a list of users.



Fake API server / New Request

GET https://jsonplaceholder.typicode.com/users

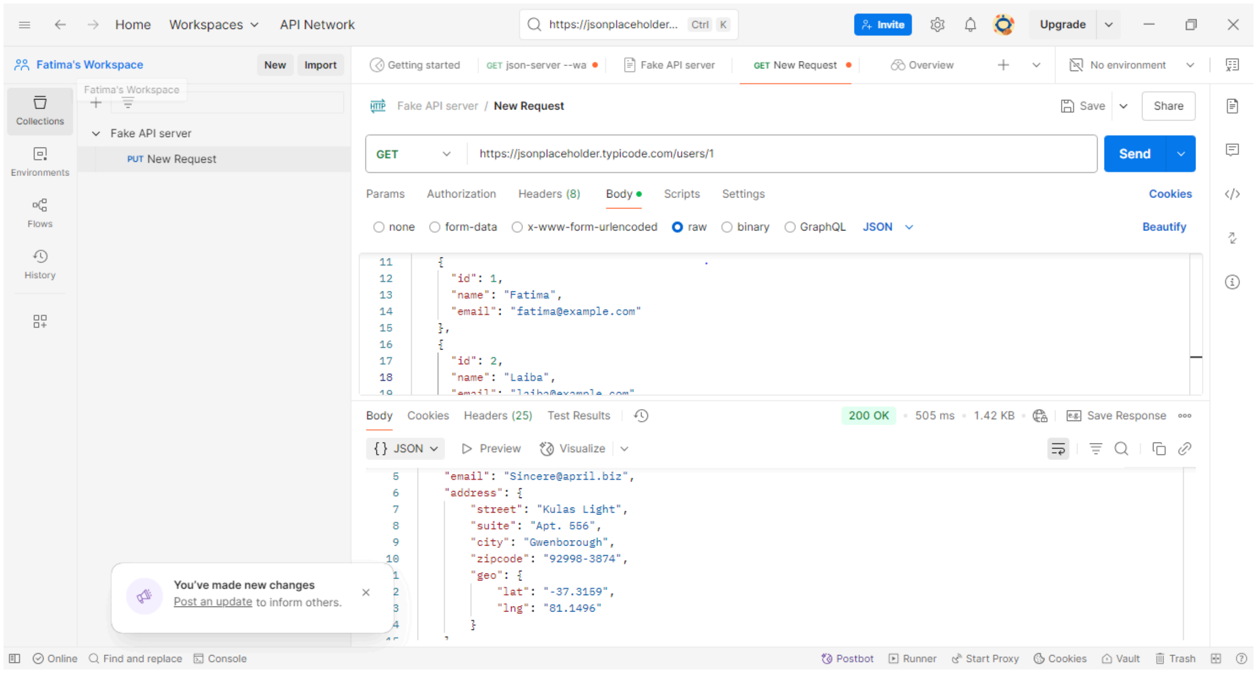
none ○ form-data ○ x-www-form-urlencoded ○ [✓ raw] ○ binary ○ GraphQL JSON

```
    {
      "id": 1,
      "name": "Fatima",
      "email": "fatima@example.com"
    },
    {
      "id": 2,
      "name": "Laiba",
```

Body Cookies Headers (25) Test Results — 200 OK • 365 ms • 2.91 KB

```
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
```

- **GET** `/users/{id}` – fetches a single user.



GET https://jsonplaceholder.typicode.com/users/1

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
    {
      "id": 1,
      "name": "Fatima",
      "email": "fatima@example.com"
    },
    {
      "id": 2,
      "name": "Laiba",
      "email": "laiba@example.com"
```

```
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
```

2. CODE:-

1. Importing Libraries

```
import React, { useEffect, useState } from 'react';
import { View, FlatList, Alert, StyleSheet, ScrollView,
KeyboardAvoidingView } from 'react-native';
import { Provider as PaperProvider, Card, Text, TextInput, Button,
ActivityIndicator, Modal, Portal, Avatar, Title, IconButton } from
'react-native-paper';
import axios from 'axios';
```

Explanation:
This imports all the tools needed: basic React Native components, UI from React Native Paper, and axios for API requests.

2. API Base URL

```
const API_URL = 'https://jsonplaceholder.typicode.com';
```

Explanation:
Base URL for the fake REST API where we get posts and users.

3. State Variables

```
const [posts, setPosts] = useState([]);
const [users, setUsers] = useState([]);
const [loadingPosts, setLoadingPosts] = useState(false);
const [loadingUsers, setLoadingUsers] = useState(false);
const [title, setTitle] = useState('');
const [body, setBody] = useState('');
const [editingPostId, setEditingPostId] = useState(null);
const [visible, setVisible] = useState(false);
```

Explanation:
These variables store data and control UI behavior like loading, form inputs, and modal visibility.

4. Fetch Posts and Users

```
useEffect(() => {
  fetchPosts();
  fetchUsers();
}, []);
```

Explanation:
Runs only once when the app loads to get the posts and users from the API.

5. Fetch Functions

```
const fetchPosts = async () => {
  setLoadingPosts(true);
  const response = await axios.get(`${API_URL}/posts`);
  setPosts(response.data.slice(0, 10));
  setLoadingPosts(false);
};

const fetchUsers = async () => {
  setLoadingUsers(true);
  const response = await axios.get(`${API_URL}/users`);
  setUsers(response.data.slice(0, 5));
  setLoadingUsers(false);
};
```

Explanation:
Get posts and users from the API and store them in the state.

6. Add or Edit Post

```
const handleSavePost = async () => {
  if (editingPostId) {
    await axios.put(`${API_URL}/posts/${editingPostId}`, { title,
body, userId: 1 });
    setPosts(posts.map(p => p.id === editingPostId ? { ...p, title,
body } : p));
  } else {
    const response = await axios.post(`${API_URL}/posts`, { title,
body, userId: 1 });
```

```
    setPosts([response.data, ...posts]);
  }
  setTitle('');
  setBody('');
  setEditingPostId(null);
  setVisible(false);
};
```

Explanation:
If editing, update the post. Otherwise, create a new one. Then reset the form.

7. Edit and Delete Post

```
const handleEditPost = (post) => {
  setTitle(post.title);
  setBody(post.body);
  setEditingPostId(post.id);
  setVisible(true);
};

const handleDeletePost = async (id) => {
  await axios.delete(`${API_URL}/posts/${id}`);
  setPosts(posts.filter(p => p.id !== id));
};
```

Explanation:
Edit fills the form with post data. Delete removes the post from the list.

8. Post Item UI

```
const renderPostItem = ({ item }) => (
  <Card>
    <Card.Title title={item.title} subtitle={`Post ID: ${item.id}`}
/>
    <Card.Content><Text>{item.body}</Text></Card.Content>
    <Card.Actions>
      <IconButton icon="pencil" onPress={() => handleEditPost(item)}
/>
      <IconButton icon="delete" onPress={() =>
handleDeletePost(item.id)} />
    </Card.Actions>
  </Card>
);
```

Explanation:
Displays each post with edit and delete buttons.

9. User Item UI

```
const renderUserItem = ({ item }) => (
  <Card>
    <Card.Title title={item.name} subtitle={item.email} />
  </Card>
);
```

Explanation:
Displays basic user info (name and email).

10. Main UI Return

```
return (
  <PaperProvider>
    <ScrollView>
      {loadingUsers ? <ActivityIndicator /> : <FlatList data={users}
renderItem={renderUserItem} horizontal />}
      <Button onPress={() => { setVisible(true); }}>Add New
Post</Button>
      {loadingPosts ? <ActivityIndicator /> : <FlatList data={posts}
renderItem={renderPostItem} />}
    </ScrollView>

    <Portal>
      <Modal visible={visible} onDismiss={() => setVisible(false)}>
        <TextInput label="Title" value={title}
onChangeText={setTitle} />
        <TextInput label="Body" value={body} onChangeText={setBody}
multiline />
        <Button onPress={handleSavePost}>{editingPostId ? 'Update' :
'Add'} Post</Button>
      </Modal>
    </Portal>
  </PaperProvider>
);
```

Explanation:
Main layout showing users and posts. Includes loading state, add/edit form, and modal for post input.

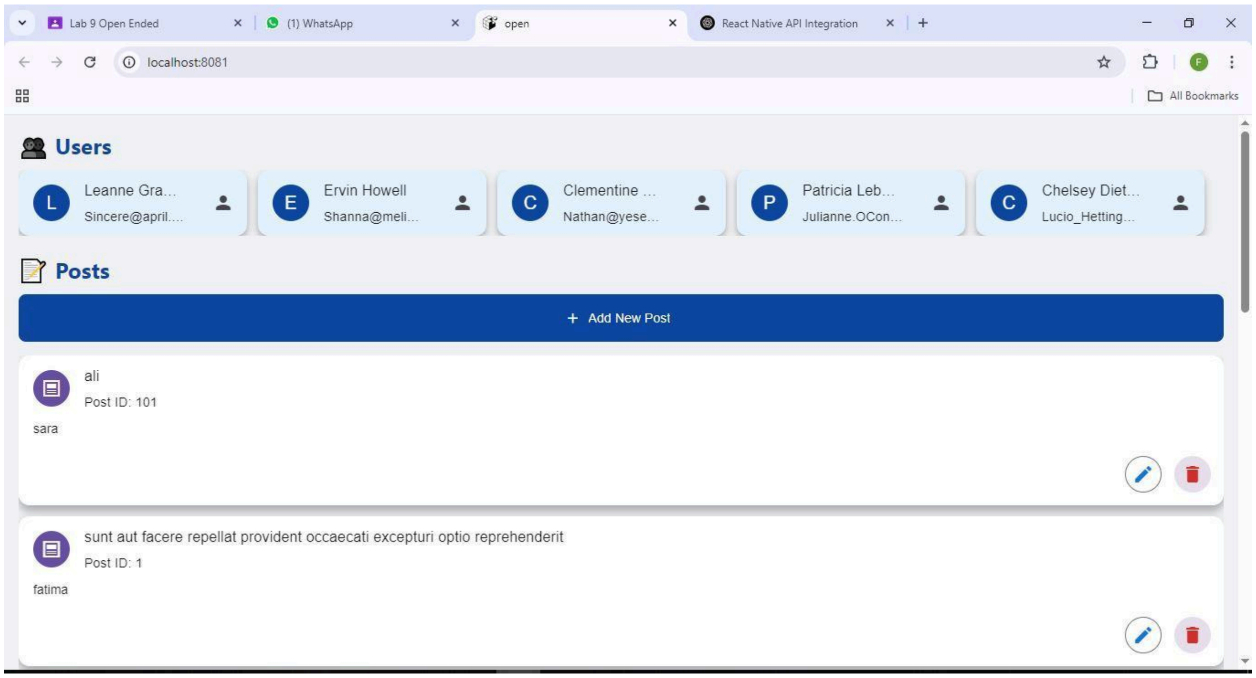
11. Basic Styles

```
const styles = StyleSheet.create({
  container: { padding: 16, backgroundColor: '#f0f2f5' },
  card: { marginBottom: 12 },
});
```

Explanation:
Adds padding and spacing to make the UI look cleaner.

- **Output:-**

✔ **Fetching users and posts**



Users

Leanne Gra...	Sincere@april....
Ervin Howell	Shanna@meli...
Clementine ...	Nathan@yese...
Patricia Leb...	Julianne.OCon...
Chelsey Diet...	Lucio_Hetting...

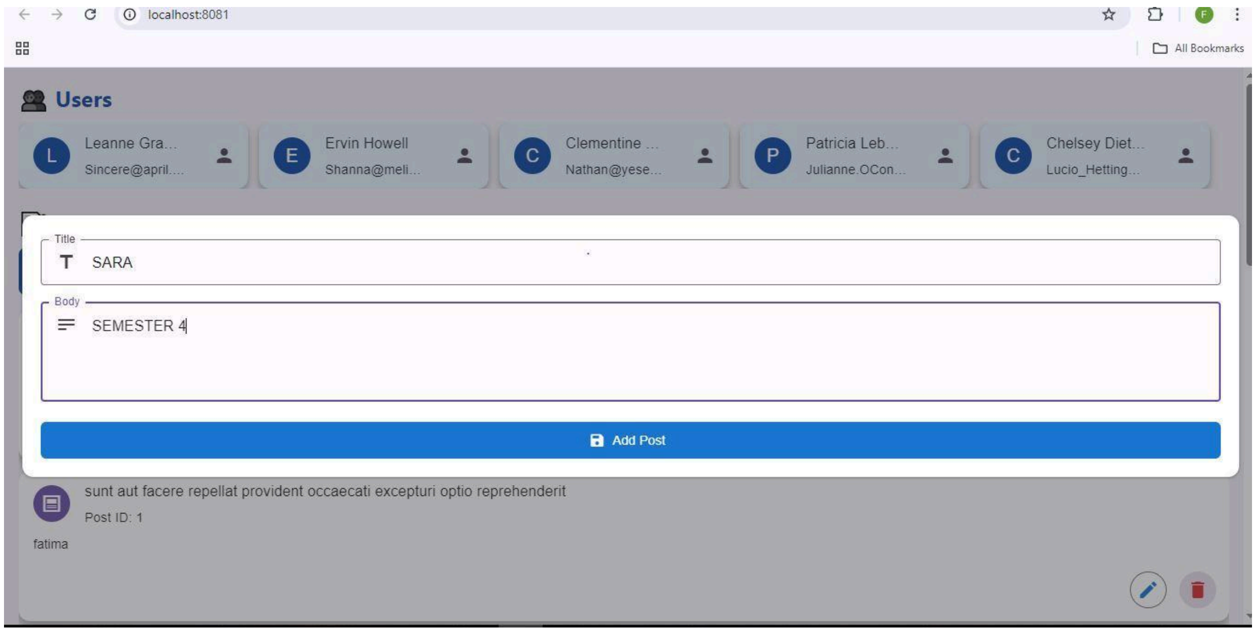
Posts

+ Add New Post

ali
Post ID: 101
sara

sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Post ID: 1
fatima

✔ **Allow to post users**



Users

Leanne Gra...	Sincere@april....
Ervin Howell	Shanna@meli...
Clementine ...	Nathan@yese...
Patricia Leb...	Julianne.OCon...
Chelsey Diet...	Lucio_Hetting...

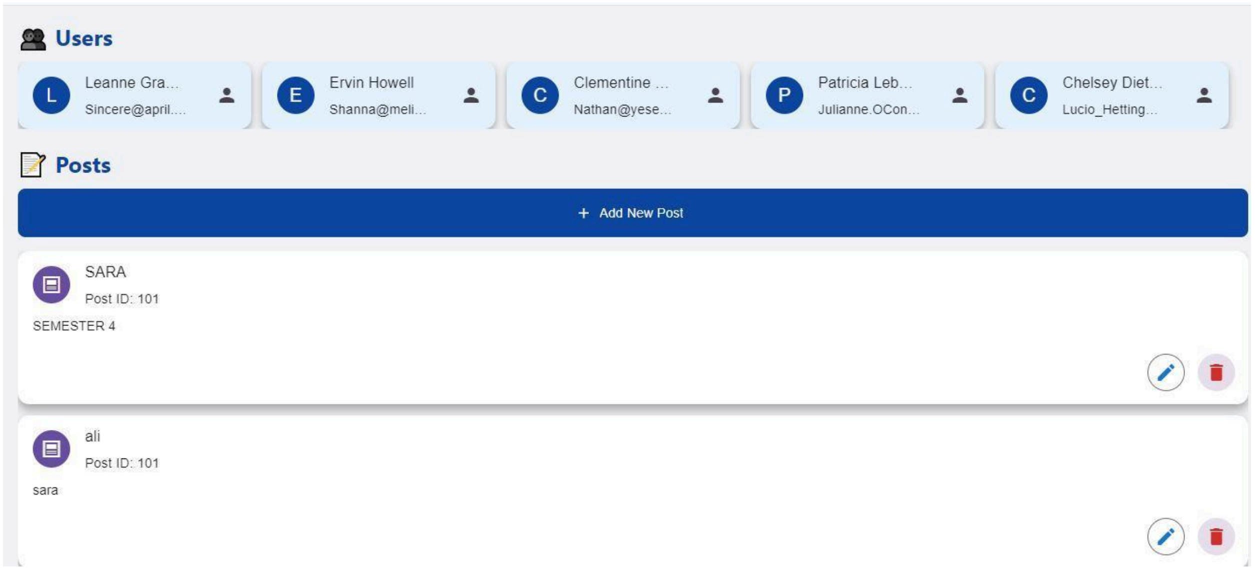
Title: SARA

Body: SEMESTER 4

Add Post

sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Post ID: 1
fatima

✔ **New post added with name Sara**


Users

Leanne Gra...
Sincere@april....

Ervin Howell
Shanna@meli...

Clementine ...
Nathan@yese...

Patricia Leb...
Julianne.OCon...

Chelsey Diet...
Lucio_Hetting...

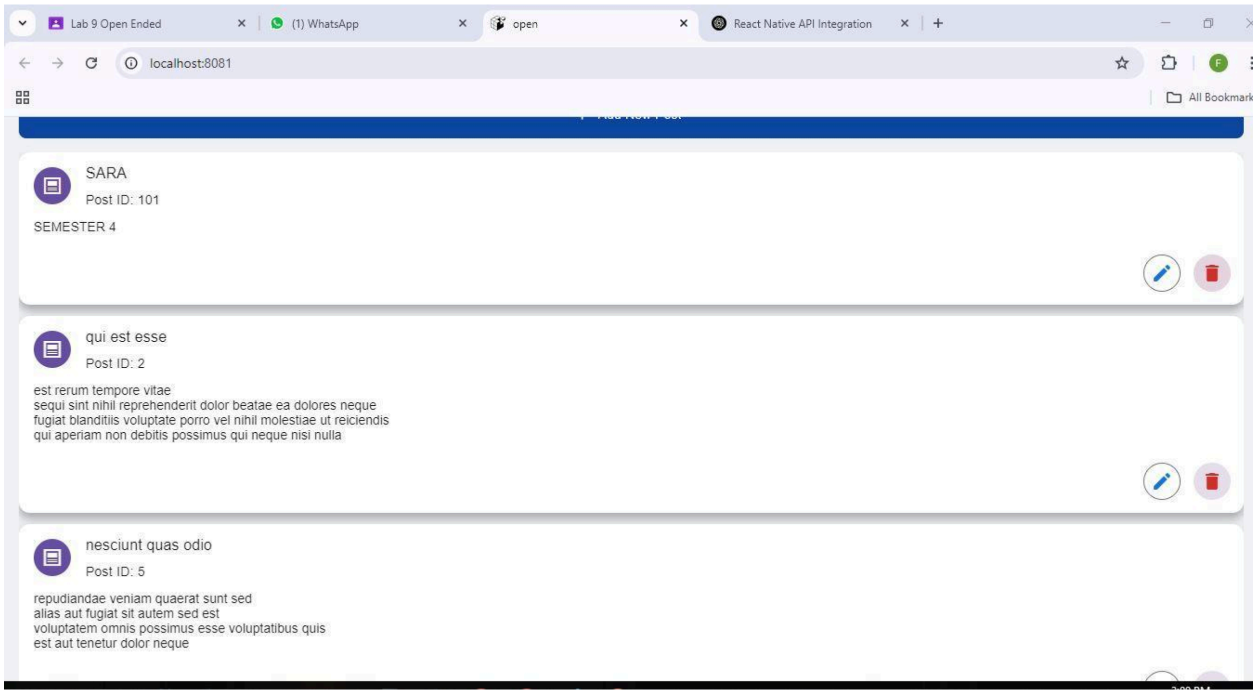
Posts

+ Add New Post

SARA
Post ID: 101
SEMESTER 4

ali
Post ID: 101
sara

✔ **Allow to delete and update posts**


Lab 9 Open Ended | (1) WhatsApp | open | React Native API Integration

localhost:8081

SARA
Post ID: 101
SEMESTER 4

qui est esse
Post ID: 2
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla

nesciunt quas odio
Post ID: 5
repudiandae veniam quaerat sunt sed
alias aut fugiat sit autem sed est
voluptatem omnis possimus esse voluptatibus quis
est aut tenetur dolor neque