**1. Frontend Setup (React Native)**

**1.1. Prerequisites**

Ensure you have the following installed:

* Node.js and npm (or yarn)
* React Native CLI
* Android Studio or Xcode (for mobile device emulation)

**1.2. Creating a New React Native Project**

Create a new React Native project using the React Native CLI:

bash

npx react-native init ToDoListApp --template react-native-template-typescript

**1.3. Folder Structure**

Organize your project structure as follows:

css

ToDoListApp/

├── src/

│ ├── components/

│ ├── screens/

│ ├── utils/

│ └── App.jsx

├── .gitignore

├── package.json

└── tsconfig.json

**1.4. Install Required Dependencies**

Install necessary libraries:

bash

npm install @react-navigation/native @react-navigation/stack axios

npm install react-native-screens react-native-safe-area-context

npx pod-install ios

**1.5. API Utility**

Create an API utility file to handle HTTP requests.

**File: src/utils/api.ts**

typescript

import axios from 'axios';

const API\_URL = 'http://127.0.0.1:5000'; // Adjust based on your backend URL

export const fetchTasks = async (token: string) => {

const response = await axios.get(`${API\_URL}/tasks`, {

headers: { Authorization: `Bearer ${token}` }

});

return response.data;

};

export const addTask = async (task: { text: string }, token: string) => {

const response = await axios.post(`${API\_URL}/tasks`, task, {

headers: { Authorization: `Bearer ${token}` }

});

return response.data;

};

export const updateTask = async (taskId: number, updates: { text?: string; completed?: boolean }, token: string) => {

const response = await axios.put(`${API\_URL}/tasks/${taskId}`, updates, {

headers: { Authorization: `Bearer ${token}` }

});

return response.data;

};

export const deleteTask = async (taskId: number, token: string) => {

await axios.delete(`${API\_URL}/tasks/${taskId}`, {

headers: { Authorization: `Bearer ${token}` }

});

};

**1.6. UI Components**

Create the following screens:

* **LoginScreen.jsx**
* **SignupScreen.jsx**
* **HomeScreen.jsx**
* **ViewTaskScreen.jsx**

**File: src/screens/HomeScreen.jsx**

typescript

import React, { useState, useEffect } from 'react';

import { View, Text, Button, FlatList, TouchableOpacity } from 'react-native';

import { fetchTasks, deleteTask } from '../utils/api';

const HomeScreen = ({ navigation, route }: any) => {

const [tasks, setTasks] = useState<any[]>([]);

const [loading, setLoading] = useState<boolean>(true);

const { token } = route.params; // Get token from route params

useEffect(() => {

const loadTasks = async () => {

try {

const tasksData = await fetchTasks(token);

setTasks(tasksData);

} catch (error) {

console.error(error);

} finally {

setLoading(false);

}

};

loadTasks();

}, [token]);

const handleDelete = async (taskId: number) => {

try {

await deleteTask(taskId, token);

setTasks(tasks.filter(task => task.id !== taskId));

} catch (error) {

console.error(error);

}

};

if (loading) {

return <Text>Loading...</Text>;

}

return (

<View>

<FlatList

data={tasks}

renderItem={({ item }) => (

<View>

<Text>{item.text}</Text>

<Button title="Edit" onPress={() => navigation.navigate('ViewTask', { task: item, token })} />

<Button title="Delete" onPress={() => handleDelete(item.id)} />

</View>

)}

keyExtractor={(item) => item.id.toString()}

/>

<Button title="Add Task" onPress={() => navigation.navigate('AddTask', { token })} />

</View>

);

};

export default HomeScreen;

**1.7. Navigation**

Setup navigation between screens.

**File: src/App.jsx**

typescript

import React from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/stack';

import HomeScreen from './screens/HomeScreen';

import LoginScreen from './screens/LoginScreen';

import SignupScreen from './screens/SignupScreen';

import ViewTaskScreen from './screens/ViewTaskScreen';

const Stack = createStackNavigator();

const App = () => {

return (

<NavigationContainer>

<Stack.Navigator>

<Stack.Screen name="Login" component={LoginScreen} />

<Stack.Screen name="Signup" component={SignupScreen} />

<Stack.Screen name="Home" component={HomeScreen} />

<Stack.Screen name="ViewTask" component={ViewTaskScreen} />

</Stack.Navigator>

</NavigationContainer>

);

};

export default App;

**1.8. Running the React Native App**

Run the React Native app using:

bash

npx react-native run-android

# or

npx react-native run-ios

**2. Backend Setup (Flask)**

**2.1. Prerequisites**

Ensure you have Python and pip installed. Install the necessary Python packages:

bash

pip install Flask Flask-SQLAlchemy Flask-Cors

**2.2. Flask Application Code**

**File: app.py**

python

from flask import Flask, jsonify, request

from flask\_sqlalchemy import SQLAlchemy

from flask\_cors import CORS

app = Flask(\_\_name\_\_)

CORS(app)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///tasks.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

class Task(db.Model):

id = db.Column(db.Integer, primary\_key=True)

text = db.Column(db.String(200), nullable=False)

completed = db.Column(db.Boolean, default=False)

def to\_dict(self):

return {

'id': self.id,

'text': self.text,

'completed': self.completed

}

@app.route('/tasks', methods=['GET'])

def get\_tasks():

tasks = Task.query.all()

return jsonify([task.to\_dict() for task in tasks])

@app.route('/tasks', methods=['POST'])

def add\_task():

data = request.json

if 'text' not in data:

return jsonify({'error': 'Text field is required'}), 400

new\_task = Task(text=data['text'])

db.session.add(new\_task)

db.session.commit()

return jsonify(new\_task.to\_dict()), 201

@app.route('/tasks/<int:task\_id>', methods=['PUT'])

def update\_task(task\_id):

data = request.json

task = Task.query.get\_or\_404(task\_id)

if 'text' in data:

task.text = data['text']

if 'completed' in data:

task.completed = data['completed']

db.session.commit()

return jsonify(task.to\_dict())

@app.route('/tasks/<int:task\_id>', methods=['DELETE'])

def delete\_task(task\_id):

task = Task.query.get\_or\_404(task\_id)

db.session.delete(task)

db.session.commit()

return '', 204

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**2.3. Database Initialization**

Initialize the SQLite database:

bash

python

Inside the Python shell:

python

from app import db

db.create\_all()

exit()

**2.4. Running the Flask Server**

Start the Flask server:

bash

python app.py

Your backend will be running on http://127.0.0.1:5000.

**3. ER Diagram**

**ER Diagram for Task Management**

plaintext

+----------------+

| Task |

+----------------+

| id (PK) |

| text |

| completed |

+----------------+

* **Task**: Represents a task with attributes id, text, and completed.

**4. Commands and Codes**

**4.1. React Native Commands**

* **Install Dependencies**:

bash

npm install

* **Run React Native App**:

bash

npx react-native run-android

# or

npx react-native run-ios

**4.2. Flask Commands**

* **Initialize Database**:

bash

python

>>> from app import db

>>> db.create\_all()

>>> exit()

* **Run Flask Server**:

bash

python app.py