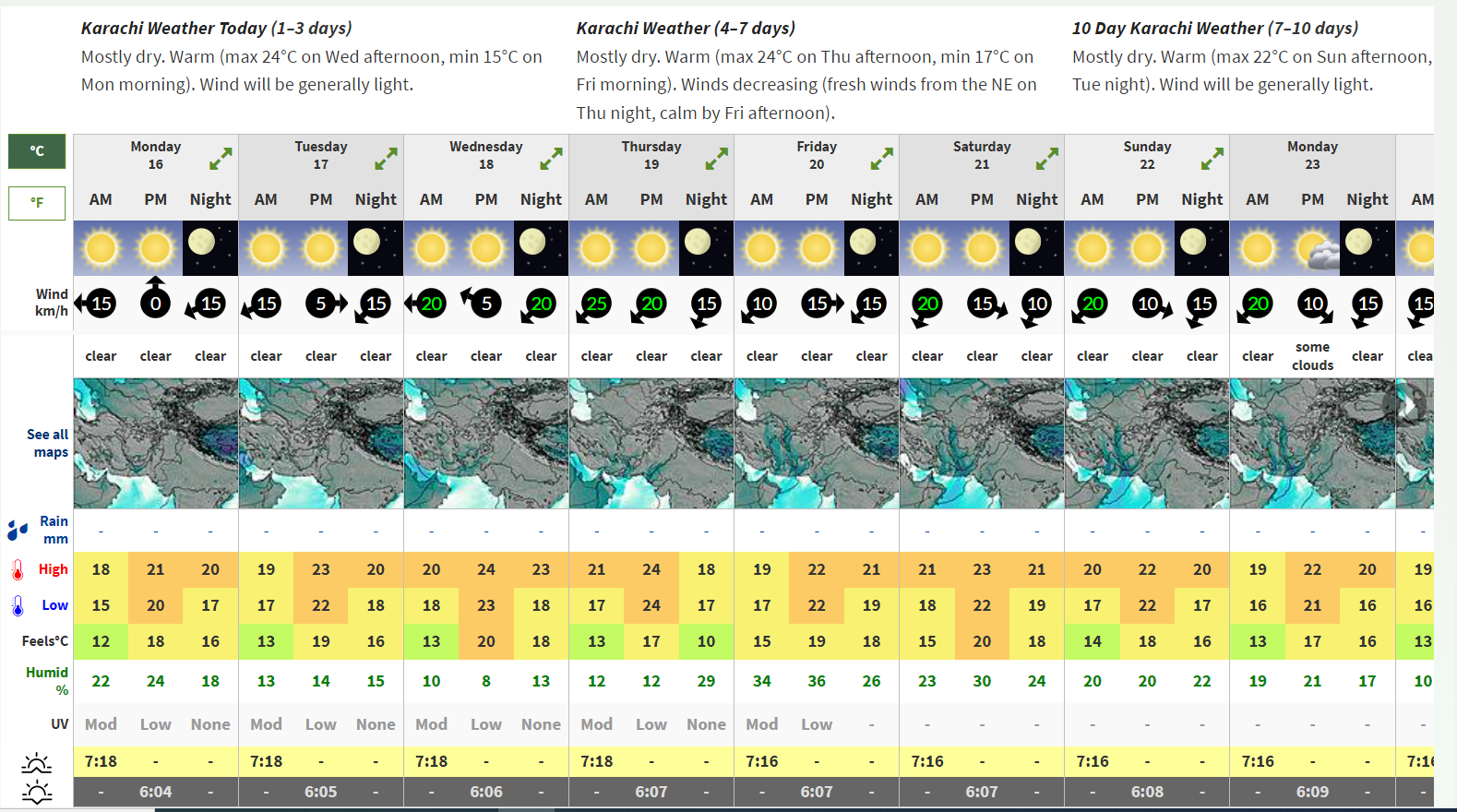
|  |  |
| --- | --- |
| **MERN – PAK Weather Forecasting** | |
| **Project Name** | *PAK Weather Monitoring* |
| **Duration** | * You will have **24** in total to complete this project. |
| **General Objectives** | * **Create** a Weather Monitoring web app from scratch. * **Demonstrate** the technical and non-technical skills developed during the Junior Full Stack Developer program. |
| **Details** | * You will work in **groups of 2-3 people** as assigned by the instructor. * This project has **10 tasks** (~2 hours each) divided into **3 Sprints**. * Each Sprint will have a **demo** and a **retrospective** at the end following the Scrum methodology. * At the end of the project, you and your group will do a **final project presentation** to the entire class and potentially to a group of employers. |
| **Assessment** | * In each task you will be assessed by the instructors on the “Assessment Criteria” in [this rubric](https://docs.google.com/spreadsheets/d/1X-LhsK5TaDvQZl-YS6XFxemVx3UhHdAY-vRcdR-rt9Q/edit#gid=1967728531). |
| **Materials** | * All participant guides, assets, and possible solutions can be found here -> [GitHub practice](https://github.com/generation-org/jwd-final-project). |
| **RR-FP - Sprint 1 - Task 1: Design your App Wireframes** | |
| **Session Objectives** | * The objective of this task is to create the *PAK Weather Monitoring* App wireframes to understand how it will work and look. |
| **Assessment Criteria** | 1. Contains a city wise detailed view of the weather 2. Contains a List All Cities weather forecasting dashboard 3. Wireframe solves all the UI challenges to represent the PAK Weather Monitoring App and a clear view of a city detailed weather info with the required fields (image is attached with this doc) |
| **Sequencing** | After WEB - 4 - Learn to Build Websites in Your Own Computer |
| **RR-FP - Sprint 1 - Task 2: Implement your Wireframes using Material UI** | |
| **Session Objectives** | * Implement the basic HTML structure of your Wireframes design. * Create a private Github repository for your project that is shared with your instructor. |
| **Assessment Criteria** | * There should be a separate input form asking for adding new city and the temperature unit to view the weather info just like other existing cities (showing 5-10 cities info by default. Also make sure to NOT accept the already existing cities) * City weather info fields are displayed in an organized way and with the proper label and input types. * All input fields are mapped in the city weather info schema (city name, temperature unit etc and whatever is needed to maintain as per the image. Define all the info fields in your schema) |
| **Sequencing** | After WEB - 4 - Learn to Build Websites on Your Own Computer |
| **RR-FP - Sprint 1 - Task 3: Create an City Weather Card layout and an Cities Weather List component** | |
| **Session Objectives** | * Define the City Weather Info component with the required fields displayed (refer the image attached) |
| **Assessment Criteria** | * The List organizes cities weather Layouts accordingly. * The List group contains at least 5 cities weather sample info. |
| **Sequencing** | After WEB - 4 - Learn to Build Websites on Your Own Computer |
| **RR-FP - Sprint 2 - Task 4: Migrating to React Components: city weather info** | |
| **Session Objectives** | For this task, we'll be migrating the code you created with Material UI to create a new city weather info to become a React component using the state, style and properties as needed. |
| **Assessment Criteria** | * Classes and styling are migrated correctly using React. * React component stores all weather info properties:   + City Name,   + Temperature Unit,   + etc * JSX is used correctly to map the structure created before with Material UI. |
| **Sequencing** | After React - 5 - Advanced React |
| **RR-FP - Sprint 2 - Task 5: Migrating to React Components: Cities Weathers Info list** | |
| **Session Objectives** | For this task, we'll be migrating the code you created with Material UI to display the list of cities weather list to become a React component using the state, style and properties as needed. |
| **Assessment Criteria** | * You must implement a React component to represent a city weather info. * You must implement a React component to represent the list of cities weather info (refer the image below for the listing). * JSX is used correctly to map the structure created before with Material UI. |
| **Sequencing** | After React - 5 - Advanced React |
| **RR-FP - Sprint 2 - Task 6: Implementing the PAK Weather Monitoring API with Express + Socket.io** | |
| **Session Objectives** | For this task, we'll write the code to create a new Express project to implement the REST API + Socket for the PAK Weather Monitoring App. |
| **Assessment Criteria** | * A new express project must be created and committed to a separate Github repository. * Implement the different Routing to support the CRUD operations for the PAK Weather Monitoring App:   + Read the cities weather info for a given user (show only permissible cities for a user).   + Read the information of a given city weather info using its ID.   + Create/Add a new City Weather Info for a user.   + Update a City Weather info (for example temperature unit) using its ID and the new information to be updated.   + Delete a City Weather info for a user using its ID. * After a pre-defined interval, update the UI (for a particular user) with Weather info   + Update the DB first   + Send the updated/latest weather info to the user using Socket.io   + Generate a notification on the UI with the message that “Last Updated: <Date +Time> * The server can be started with no errors displayed. |
| **Sequencing** | After NEM - 2 - Express Routes |
| **RR-FP - Sprint 2 - Task 7: Connecting with MongoDB database** | |
| **Session Objectives** | For this task, we'll write the code to connect your REST API created with Express with a MongoDB database instance using Mongoose. |
| **Assessment Criteria** | * All the CRUD operations must interact with the MongoDB database. * Mongoose is installed and included as dependency on the application using NPM. * Mongoose and the database setup is done properly and no errors occur when the application server is started. |
| **Sequencing** | After NEM - 4 |
| **RR-FP - Sprint 3 - Task 8: Connecting your Frontend with your Backend** | |
| **Session Objectives** | In this task, we'll connect the two projects created so the React App can consume and use the Express API + Socket and the data is persisted on the MongoDB Database. |
| **Assessment Criteria** | * The React application consumes the REST API created with express using the fetch function to:   + Read the cities weather info of a given user.   + Read the information of a given city weather using its ID.   + Create/Add a new city weather info.   + Update a city info using its ID and the new information to be updated.   + Delete a city info using its ID for a particular user. * A **socket endpoint** is defined for sending the latest weather info to the UI for a particular user * The application works as expected and no error happens when you perform the CRUD operations. |
| **Sequencing** | After NEM - 4 - Introduction to MongoDB. |
| **RR-FP - Sprint 3 - Task 9: Deploying and testing your application** | |
| **Session Objectives** | In this task, we'll deploy the final solution to the Cloud so you can share your project URL with the instructor and other classmates. |
| **Assessment Criteria** | * The React application should be deployed and accessible via a public URL over the internet. * The Express API should be deployed and accessible via a public URL over the internet. * The React application and express + **socket** application must communicate and work as expected. |
| **Sequencing** | After NEM - 6 - Deploy to Cloud Infrastructure. |
| **RR-FP - Sprint 3 - Task 10: Final Presentation** | |
| **Session Objectives** | Present your final project. |
| **Assessment Criteria** | * The City Weather Info fields use the correct input type * The City Weather info input uses the select input type to prevent the user from entering the wrong data type. * All form fields are validated (name, temperature unit etc). * A meaningful error message is displayed when a form field is invalid. * A React component is used to represent a city weather info. * A React component is used to represent a list of cities weather info. * The delete and update feature is consistent with the city weather info and delete the data stored on the MongoDB database. * The update feature updates both the city info and the data stored on the MongoDB database. * The Cities Weather data persists using the MongoDB database via the Express API. * The express API support all the CRUD operations:   + Read the cities weather info of a given user.   + Read the information of a given city weather using its ID.   + Create a new city weather info.   + Update a city weather info using its ID and the new information to be updated.   + Delete a city weather info using its ID. * On page refresh, the last timestamp + msg of the weather forecasting update should remain intact (i.e., you need to save the last time stamp info in the DB as well) |
| **Sequencing** | After NEM - 6 - Deploy to Cloud Infrastructure. |

**CITY DETAILED WEATHER INFO (new page):**

<This image is just to give an idea of what kind of detailed weather info of a city is required>. Ignore the map row in the image given below



**CITIES WEATHER LIST INFO (new page):**

PUSH NOTIFICATION using socket.io

<This image is just to give an idea of what kind of listing of cities weather info is required>

