Pau Casesnoves
Centre Integrat de Formació Professional

C/Joan Miró, 22 07300 Inca
Tel. 971 881711
secretaria@paucasesnovescifp.cat

Desktop chat application

Course	23/24	**Group**	**S2P**	**Delivery date**	28/1 - 23:55
Module	Interfaces Development				
Title	Developing Visual Components - A Chat Application				

Type	Individual
Instructions	

Using components in Electron

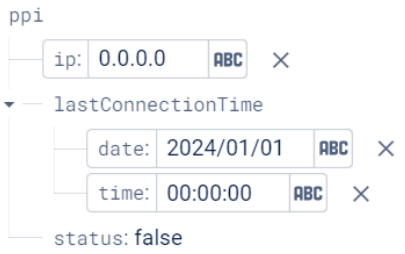
Objective

Develop a desktop chat application that allows users to register their data, view registered users, initiate chat sessions with others users and persist chat messages using Electron, React and Firebase.

Requirements

1. Technology Stack
   - Electron framework: for building cross-platform desktop applications.
   - React: for building the user interface.
   - Firebase Real time database: NoSql cloud database to store and sync data across all clients in real time.
   - Express framework: to enable an endpoint to which other users will send messages.

2. User data registration
   - Users must register their name (userName), IP address (ip), last connection time (lastConnectionTime) and online status (status) from their application into the Firebase database.
   - Create an application component from where you can register your data, for example using a form. Place it on a separate section of your app.
   - The structure of the information sent to the DB must follow this example:



Pau Casesnoves
Centre Integrat de Formació Professional
C/Joan Miró, 22 07300 Inca
Tel. 971 881711
secretaria@paucasesnovescifp.cat

3. Display Registered Users
   - Fetch and display registered users' data from the Firebase database.
   - Using a React component, display a list of registered users, name, last connection time and status.
   - All the contact information will be inside the database value **chatUsers/{contact_name}**

4. Initiating or continuing a Chat
   - The user can select an online registered contact from the list to chat with.
   - The chat window will show the conversation history with that user.
   - The messages will be sent to the registered IP of the chat contact using the **/chat** endpoint and the 4000 port.
   - Send a POST message to the contact using this format:

```
{
  userName: "YourUserName",
  message: "Your Message"
}
```

5. Message Persistence
   - The application will use a storage system (local storage, Firebase, or whatever) to preserve chat messages between you and other users.
   - Ensure that chat messages are retrieved and displayed even after application restart.

Firebase connection info:
https://spdvi-chat-default-rtdb.europe-west1.firebasedatabase.app/

Firebase realtime DB documentation:
https://firebase.google.com/docs/database

Deliverables
- Your Electron project folder that contains the necessary code to run it.
- Use comments in your application code for clarity and organize your files in folders.
- A document in PDF format where you have to explain how you solved the 5 requirements. For the 1st requirement, also explain how React can be integrated in an Electron application and how React can generate the final HTML code.

Qualification criteria

This activity corresponds to 25% of the practical part for the competence in Creating custom visual components following usability guidelines.

Pau Casesnoves
Centre Integrat de Formació Professional
C/Joan Miró, 22 07300 Inca
Tel. 971 881711
secretaria@paucasesnovescifp.cat

Students will be assessed based on the following criterions, each carrying an specific weight:

1. Functionality (40%): Based on the completeness of the application's features (user data registration, display of registered users, chat communication and message persistence.

2. User Interface (25%): Evaluates the design and usability of the application's interface, including the registration form, user list component and chat window.

3. Code Quality (15%): Assess the cleanliness, organization, and adherence to best practices in the codebase. Proper use of React components and Electron integration will be considered.

4. Firebase Integration (10%): Evaluates how well Firebase realtime database is integrated into the application, specifically focusing on user data registration, real-time updates, and message storage/retrieval.

5. Error Handling and Edge Cases (10%): Checks for proper error handling and considers edge cases. For example, how does the application handle scenarios like a user trying to initiate a chat with an offline contact or an error when sending a message

Each criterion is valuated using 1 (bad), 2 (good), 3 (excellent). To achieve an excellent, you should expand what the statement asks for and justify it in the documentation.

Submission
Submit your project source code and the documentation file.