We can use React for the frontend, Node.js for the backend, and Visual Studio Code (VS Code) as your editor for both parts of your web application. Here's a simplified guide on how to set up and work on the project:

Setting Up Your Frontend with React:

1. **Install Node.js**: Ensure Node.js is installed on your system because it includes **npm**, which you need to manage your JavaScript packages.
2. **Create React App**: In your terminal or command line, use the following commands to create a new React application:
   ```
   npx create-react-app livebeat-frontend
   cd livebeat-frontend
   ```
3. **Start Development Server**: To start your app and see it in the browser, run:
   ```
   npm start
   ```

This will open up your default web browser to localhost:3000 where you can see your React application.

4. **Write Your Code**: Use VS Code to open your project folder and start writing your code. Build components in the **src** folder and manage the application state with hooks or context as needed.

Setting Up Your Backend with Node.js:

1. **Initialize Your Backend**: In a new terminal window, set up your backend in a separate directory to keep it organized:
   ```
   mkdir livebeat-backend
   cd livebeat-backend
   npm init -y
   npm install express
   ```
2. **Create Your Server File**: In your backend directory, create a file named **server.js** or **app.js** (or another name you prefer) and set up the basic Express server:
   ```
   const express = require('express');
   const app = express();
   const port = process.env.PORT || 5000;
   app.get('/', (req, res) => {
    res.send('LiveBeat Backend Running');
   });
   app.listen(port, () => {
    console.log(`Server listening on port ${port}`);
   });
   ```

3. **Run Your Backend Server**: Start your backend server by running:
   node server.js
   Developing Your App in VS Code:
   - **Frontend and Backend Together**: Open two instances of VS Code or split your VS Code workspace to have both the frontend and backend open at the same time for convenience.
   - **Use Extensions**: VS Code has a plethora of extensions to help with React and Node.js development. Some useful extensions include:
     - ESLint for linting
     - Prettier for code formatting
     - GitLens for enhanced Git capabilities
     - Debugger for Chrome for frontend debugging
     - REST Client to test API requests directly from VS Code

Tips for Working with React and Node.js in VS Code:

- **Use the Integrated Terminal**: VS Code has an integrated terminal that you can use to run your frontend and backend servers simultaneously.
- **Environment Variables**: Store sensitive information like API keys in environment variables and add **.env** to your **.gitignore** file.
- **Hot Reloading**: React's development server supports hot reloading out of the box, and for Node.js, you can use a tool like **Nodemon** for automatic server restarts during development.

Version Control with Git:

- **Initialize Git**: If you haven't already, initialize a Git repository in your project directory.
- **Commit Regularly**: Make commits after significant changes or features to ensure you have a rollback point and clear history.
- **Collaboration**: Use branches for specific features or sections of the project to avoid conflicts, and merge them into the main branch upon completion.

Using React with Node.js in VS Code provides a cohesive development experience, as you can handle both client-side and server-side code within the same environment. This approach is widely used in the industry and is great for both learning and building professional projects.