Lab L-9: Implementing User Authentication in a Blockchain Application

```
}
```

Explanation: Your project structure extends Lab L-8.5 by adding an authserver directory for the backend, making Lab L-9 the first lab with a full-stack setup. The root `package.json` includes workspaces for both `frontend` and `authserver`. The `frontend/package.json` is similar to Lab L-8.5 but adds `@mui/material` and `react-toastify` for UI and notifications. The `authserver/package.json` includes backend-specific dependencies like `express`, `body-parser`, `dotenv`, and `@bsv/auth-express-middleware`. Running `npm install` in each directory installs all dependencies.

2. Configure the Backend (Auth Server)

In the `authserver` directory, you'll implement the Express server in `authServer.ts` and create a `.env` file for secure configuration. This is the first lab requiring a backend, so using a `.env` file to store sensitive data (e.g., SERVER_PRIVATE_KEY) is mandatory for security, unlike Lab L-8.5's frontend-only setup. The backend Express server runs on **port 3000** during development and protects the /protected endpoint using @bsv/auth-express-middleware.

- Here is a `.env` file .env to be located in `authserver` directory:

```
SERVER_PRIVATE_KEY=055d459c8d7cba2f8d22155093beb97848cf6b903f3af0a3c4eb45ba
```

2. Configure the Backend (Auth Server)

- In the authserver directory, implement the Express server in authServer.ts by completing the TODO comments in the provided template. The server should:
  - Initialize a Wallet using PrivateKey.fromHex with the provided key (note: use a .env file in production).
  - Use createAuthMiddleware to protect routes, disallowing unauthenticated requests.
  - Enable CORS with manual headers (Access-Control-Allow-* and Private-Network) and handle OPTIONS requests.
  - Set up a non-protected / route returning "Hello, world!".
  - Set up a /protected route returning "Hello, authenticated peer with public key: <key>" or a 401 "Unauthorized" response.
  - Start the server on port 3000.

Create `authserver/src/authServer.ts`:

```
import express, { Request, Response, NextFunction, RequestHandler } from 'e
import bodyParser from 'body-parser'
import dotenv from 'dotenv'
import { Setup, sdk } from '@bsv/wallet-toolbox'
import { createAuthMiddleware, AuthRequest } from '@bsv/auth-express-middle
import { PubKeyHex, VerifiableCertificate } from '@bsv/sdk'

const app = express();
app.use(bodyParser.json());

// TODO: Instantiate a BSV wallet to manage transactions
// Hint: Use PrivateKey.fromHex with the key '055d459c8d7cba2f8d22155093beb
// Note: In production, load the key from a .env file using dotenv

// TODO: Configure the Auth middleware
// Hint: Use createAuthMiddleware with the wallet and set allowUnauthentica

// TODO: Enable CORS for frontend-backend communication
// Hint: Add middleware to set Access-Control-Allow-* headers (Origin, Heac

// TODO: Apply the auth middleware to all routes
// Hint: Use app.use() with the authMiddleware
```

Lab L-9: Implementing User Authentication in a...
Submit your GitHub link:
Add any comments on the lab here:

No comments yet
A question, something to add? Be the first to comment

Comment

```
// TODO: Configure a non-protected route
// Hint: Create a GET route for '/' that sends a "Hello, world!" response

// TODO: Configure a protected route
// Hint: Create a GET route for '/protected' that sends a greeting with rec

// TODO: Start the server on port 3000
// Hint: Use app.listen() and log "Server is running on port 3000"
```

3. Build the Frontend

- In the `frontend` directory, set up the React app:
  - Create a custom theme in `src/theme.ts` using Material-UI's `createTheme`.
  - Set up the entry point in `src/index.tsx` with `ThemeProvider`, `ToastContainer`, and `CssBaseline`.
  - Use the provided src/App.tsx, which sends authenticated GET requests to **http://localhost:3000/protected** and displays the backend response (e.g., public key greeting) or errors in a Material-UI Typography.
  - Configure a test wallet for WalletClient (e.g., with a valid private key and access to a wallet service for default settings).
- Here is `frontend/src/App.tsx`:

```
import React, { useState } from 'react'
import { Button, Typography, Container, CircularProgress } from '@mui/mater
import { WalletClient, AuthFetch } from '@bsv/sdk'

const App: React.FC = () => {
  const [isLoading, setIsLoading] = useState(false)
  const [response, setResponse] = useState<string | null>(null)

  const handleButtonClick = async () => {
    setIsLoading(true)
    setResponse(null)

    try {
      const wallet = new WalletClient()
      const authFetch = new AuthFetch(wallet)

      const res = await authFetch.fetch('http://localhost:3000/protected',
        method: 'GET',
        headers: { 'Content-Type': 'application/json' }
      })

      if (!res.ok) {
        throw new Error(`Server responded with status ${res.status}`)
      }

      const text = await res.text()
      setResponse(text)
    } catch (err: any) {
      setResponse(`Error: ${err.message || err}`)
    } finally {
      setIsLoading(false)
    }
  }

  return (
    <Container maxWidth="sm">
      <Typography variant="h4" component="h1" gutterBottom>
        User Authentication App
      </Typography>
      <Button
        variant="contained"
        color="primary"
        onClick={handleButtonClick}
        disabled={isLoading}
      >
        {isLoading ? <CircularProgress size={24} /> : 'Send Request to Back
      </Button>
      {response && (
        <Typography
          variant="body1"
          style={{ marginTop: '20px', whiteSpace: 'pre-wrap' }}
        >
          Response from backend: {response}
        </Typography>
      )}
    </Container>
```

```
  )
}

export default App
```

Comment

Lab L-9: Implementing User Authentication in a...

Submit your GitHub link:

Add any comments on the lab here:

4. Test the Application

Comment

- Start the development environment:
  - `npm run start` will start the Vite-powered frontend AND the Express backend via LARS.
  - Open your browser and navigate to **http://localhost:5173**.
- Click the "Send Request to Backend" button:
  - If your backend is correctly implemented, you should see a message like: Hello, authenticated peer with public key: <public-key>.
  - If unauthorized, you should see: Error: Server responded with status 401.
  - Verify CORS is working by checking that the frontend can communicate with the backend.

Comment

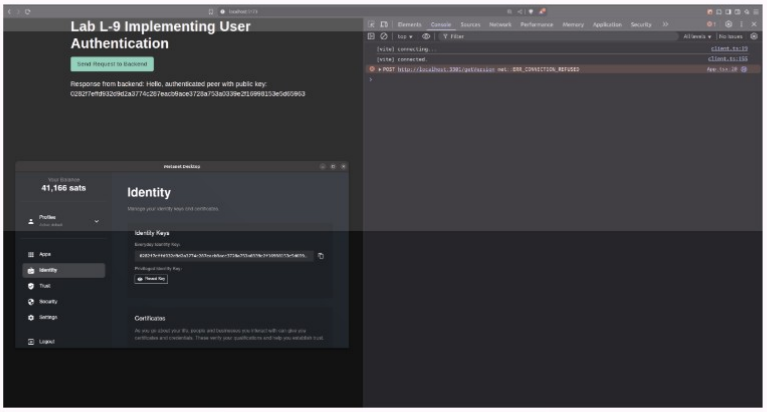
Run Deployed App (Interact with the reference working version)

Lab L-9 User Authentication App

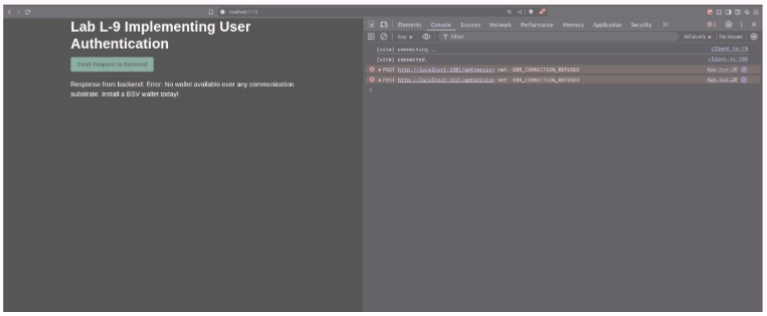
Comment

Screens

Your screens should look something like these:



Comment



Comment