index.js (Express.js server)

JavaScript
```
const express = require('express');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');
const cors = require('cors');
```

Dependencies:

- Express.js: A web application framework for Node.js.
- Body-parser: A middleware for parsing incoming JSON data.
- JSON Web Token (JWT): A standard for creating and verifying security tokens.
- CORS: A mechanism that allows requests to be made from one domain to another.

JavaScript
```
const app = express();
const secretKey = 'yourSecretKey'; // Change this to a secure key in
production
app.use(cors());
app.use(bodyParser.json());
```

Server initialization and configuration:

- The express() function creates a new Express application.
- The secretKey variable is used to sign and verify JWT tokens. It is important to keep this key secret.
- The cors() middleware enables CORS.
- The bodyParser.json() middleware parses incoming JSON data into JavaScript objects.

JavaScript
```
const authenticateJWT = (req, res, next) => {
 const token = req.body.token;

 if (!token) {
  return res.status(401).json({ message: 'Unauthorized' });
 }

 jwt.verify(token, secretKey, (err, user) => {
  if (err) return res.status(403).json({ message: 'Forbidden' });

  req.user = user;
  next();
 });
};
```

JWT authentication middleware:

- This middleware verifies the JWT token in the request body. If the token is valid, the user's data is added to the req.user object.
- If the token is invalid, a 401 Unauthorized response is sent.
- If the token is valid but the user does not have permission to access the requested route, a 403 Forbidden response is sent.

JavaScript
```
const users = [
 { id: 1, username: 'user1', password: 'password1' },
 { id: 2, username: 'user2', password: 'password2' },
];
```

Mock user data:

- This is a mock array of users. In production, user data would typically be stored in a database.

JavaScript
```
app.use(express.static(__dirname, { 'X-Content-Type-Options':
'nosniff' }));
```

Serving static files:

- This middleware serves static files (HTML, JavaScript, CSS, etc.) from the specified directory.

JavaScript
```
app.get('/', (req, res) => {
 res.sendFile(__dirname + '/index.html');
});
```

Home route:

- This route serves the index.html file.

JavaScript
```javascript
app.post('/dashboard', authenticateJWT, (req, res) => {
 // ... protected route logic ...
});
```

Dashboard route:

- This route is a protected route that can only be accessed by authenticated users.
- The authenticateJWT middleware is used to verify the JWT token in the request body.
- If the token is valid, the user's data is added to the req.user object.
- The protected route logic can then be executed.

JavaScript
```javascript
app.post('/login', (req, res) => {
 // ... login logic ...
});
```

Login route:

- This route allows users to log in and generate a JWT token.
- The login logic would typically involve verifying the user's credentials against a database.
- If the login is successful, a JWT token is generated and returned to the user.

JavaScript
```javascript
const PORT = 3000;
app.listen(PORT, () => {
 console.log(`Server is running on http://localhost:${PORT}`);
});
```

Server listening:

- The server starts listening on port 3