GrocerEase Application Documentation

Authentication and Role-Based Access Control Implementation

Overview

This document outlines the implementation of authentication and role-based access control (RBAC) in the GrocerEase application. The system supports two types of users:

1. Admin users with full access to all features

2. Regular users with access to the chat assistant only

User Types and Credentials

1. Admin User:

- Username: `admin`
- Password: `admin`
- Role: `admin`
- Access: Full access to all features

2. Regular User:

- Username: `user1`
- Password: `user1`
- Role: `user`
- Access: Chat assistant only

Implementation Details

1. Authentication Context (AuthContext.js)

```
javascript
```

```
const AuthContext = createContext(null);

export const AuthProvider = ({ children }) => {

const [user, setUser] = useState(null);

const [loading, setLoading] = useState(true);

// Authentication Functions

const login = async (username, password) => {

// Handles user login and token storage

};

const logout = () => {

// Handles user logout and token removal

};

const isAuthenticated = () => {

return !!user;  // Checks if user is logged in

};

const isAdmin = () => {

return !!user && user.role === 'admin';  // Checks if user is admin

};

};
```

2. Protected Routes (App.js)

Two types of protected routes were implemented:

1. Admin Routes:

```
javascript

const AdminRoute = ({ children }) => {
```

```
const { isAuthenticated, loading, isAdmin } = useAuth();

if (loading) return

Loading...
;

if (!isAuthenticated()


 !isAdmin()) return ;

return children;

};
```

2. Protected Routes (for authenticated users):

```
 javascript

const ProtectedRoute = ({ children }) => {

const { isAuthenticated, loading } = useAuth();

if (loading) return

Loading...
;

if (!isAuthenticated()) return ;

return children;

};
```

3. Login Component (Login.js)

```
 javascript
```

```
const Login = () => {

const handleSubmit = async (e) => {

e.preventDefault();

try {

await login(username, password);

// Role-based redirection

if (user?.role === 'admin') {

navigate('/users');

} else {

navigate('/chat');

}

} catch (error) {

setError(error.message

 'Login failed');

}

};

};
```

Key Changes and Fixes

1. Authentication Flow:

- Implemented JWT-based authentication
- Added token storage in localStorage
- Created role-based access control

2. Route Protection:

- Added AdminRoute for admin-only pages
- Added ProtectedRoute for authenticated users
- Implemented proper redirection based on user role

3. User Interface:

- Updated menu items visibility based on user role
- Added user role display in profile menu
- Improved mobile responsiveness

4. Security Enhancements:

- Implemented proper password hashing
- Added token-based authentication
- Created role-based access restrictions

Access Control Matrix

Feature	Admin	Regular User
Chat Assistant	✓	✓
User Management	✓	✗
Shop Management	✓	✗
Aisle Management	✓	✗
Item Management	✓	✗

Testing Instructions

1. Admin Login:

- Navigate to login page

- Enter admin credentials
- Should be redirected to /users
- Should see all menu items

2. Regular User Login:

- Navigate to login page
- Enter user1 credentials
- Should be redirected to /chat
- Should only see chat assistant in menu

Troubleshooting

Common issues and solutions:

1. Login Not Working:

- Clear browser localStorage
- Restart backend server
- Verify credentials

2. Access Issues:

- Check user role in localStorage
- Verify token validity
- Ensure proper redirection

3. Menu Items Not Showing:

- Verify user role
- Check authentication status
- Refresh page

Future Improvements

1. Security:

- Implement token refresh mechanism

- Add password complexity requirements
- Implement session timeout

2. User Experience:

- Add remember me functionality
- Implement password reset
- Add user profile management

3. Features:

- Add more user roles
- Implement role-based permissions
- Add audit logging