COSMOS BUSINESS SYSTEMS AEBE

TURN Key SOLUTIONS

Newcomer Developer

Todo List with User Authentication

v.1.0

ATHENS
44, P. Bakogianni Str.
Metamorfosi GR - 14452, Athens, Greece
Tel.: +30 210 6492800 Fax: +30 210 6464069
email: cosmos@cbs.gr
www.cbs.gr

THESSALONIKI
9th km Thessaloniki - Thermi
Pylaia Thessaloniki - 57001, Greece
Tel.: +30 2310 477670 Fax: +30 2310 477672
email: cosmos.thess@cbs.gr
www.cbs.gr

CRETE
36, Sfakion Str.
Chania - 73134 Crete, Greece
Tel.: +30 28210 46857
email: gekosp@cbs.gr
www.cbs.gr

Περιεχόμενα

Περιγραφή

Δημιουργία Todo List web application με δυνατότητα αυθεντικοποίησης των χρηστών. Η εφαρμογή θα επιτρέπει στους χρήστες να εγγράφονται, να συνδέονται και να διαχειρίζονται τις εργασίες τους. Οι εργασίες θα αποθηκεύονται τοπικά χρησιμοποιώντας local storage και μόνο οι αυθεντικοποιημένοι χρήστες θα έχουν πρόσβαση στις λίστες.

User story 1: User Authentication System

As a user,
I want to be able to securely log in and log out,
so that I can access my Todo lists.

Acceptance Criteria:

- Login Screen:

User should be able to navigate to a login screen.
The login screen should have fields for entering a email and password.
The login screen should have a "Login" button.
There should be a link to navigate to the registration screen if the user does not have an account.

- Registration Screen:

User should be able to navigate to a registration screen from the login screen.
The registration screen should have fields for entering a username, email, and password.
The registration screen should have a "Register" button.
After successful registration, the user should be redirected to the login screen.

- Form Validation:

All fields should have client-side validation.
The email should be in a valid email format.
The password should be at least 8 characters long and include a mix of letters and numbers.
Display appropriate error messages for invalid inputs.

- Authentication:

Use a simple mock authentication service (local storage or a mock API) to validate user credentials.
On successful login, store user information in local storage or session storage.
On successful login, redirect the user to a protected route (e.g., a dashboard).
Display an error message for invalid login attempts.

- Logout Functionality:

User should be able to log out from any protected page.
Logging out should clear user information from local storage or session storage.
After logging out, redirect the user to the login screen.

- Protected Routes:

Implement protected routes that can only be accessed by authenticated users.
Redirect unauthenticated users attempting to access protected routes to the login screen.

User Story 2: Todo List Management

As a user,
I want to manage a list of tasks,
so that I can keep track of what needs to be done.

Acceptance Criteria:

- Task Creation:

User should be able to add a new task to the list.
Each task should have a title and an optional description.
Task titles are required and cannot be empty.

- Task Display:

Display the list of tasks with their titles and descriptions.
Each task should have a checkbox to mark it as completed.
Sort tasks by their creation date, with the most recent tasks at the top.
Display tasks in different sections: "Pending" and "Completed."

- Task Completion:

User should be able to mark a task as completed by clicking on the checkbox.
Completed tasks should be visually distinct from incomplete tasks.
Completed tasks should move to the "Completed" section.

- Task Deletion:

User should be able to delete a task from the list.

- Data Persistence:

The list of tasks should persist across page reloads using local storage.

Technical Requirements

- Use React, Angular, .NET or HTML with JS.
- if preferred use Bootstrap, Tailwind CSS.
- Use vanilla CSS or a CSS preprocessor like SASS or SCSS.

Implementation Details

Project Structure

- Login Screen
  Fields: email, password
  Button: Login
  Link: Register

- Registration Screen
  Fields: email, password
  Button: Register
  Link: Back to Login

- Todo List Screen
  Sections: Pending Tasks, Completed Tasks
  Fields: Task title, optional description
  Buttons: Add Task, Delete Task, Mark as Completed
  Checkbox: To mark tasks as completed

- Mock Authentication Service
  Use local storage to store user credentials.
  Implement basic form validation and error handling.
  Redirect to the appropriate screens based on authentication status.
  Components

- Login Component:
  Handles user login.
  Validates inputs and displays error messages.
  Redirects to the Todo List screen on successful login.

- Registration Component:
  Handles user registration.
  Validates inputs and displays error messages.
  Redirects to the Login screen on successful registration.

- TodoList Component:
  Displays tasks in Pending and Completed sections.
  Allows task creation, completion, and deletion.
  Persists tasks in local storage.

- ProtectedRoute Component:
  Wraps around protected routes to ensure only authenticated users can access them.
  Example Flow
  User navigates to the application and sees the login screen.
  User clicks on the "Register" link and is taken to the registration screen.
  User enters a username, email, and password, and clicks "Register."
  After successful registration, the user is redirected to the login screen.
  User enters their credentials and clicks "Login."
  On successful login, the user is redirected to the protected Todo List screen.
  User creates, completes, and deletes tasks as needed.
  User clicks the "Logout" button and is redirected to the login screen.
  User attempts to access the Todo List screen without logging in and is redirected to the login screen.

Submission

- Include a README file with instructions on how to run the project.
- Ensure your code is well-documented and easy to understand.