**Project Online TODO List:**

I have built a simple TODO list, using **ASP.Net Core Razor Pages, an in-memory database SQLite** is used for data storage and used **Docker container** for deploying the application on **Azure.**

I have created a user registration page and login using **cookie authentication and authorization**, to build a TODO list user specific. User can register and login to add, view, delete and edit his/her tasks (One demo user credential provided).

**Instructions to run the application:**

* Application is deployed on azure.
* Link to access the application is [**https://todolistey.azurewebsites.net**](https://todolistey.azurewebsites.netges)
* Click on “Click here” button.
* Credential of Demo user login is Username: [**bob@og.com**](mailto:bob@og.com), Password: **12345**
* A new user can also be created using Register button on login page**.**

**GitHub Link for code:** [https://github.com/RahulKJoshi11/TodoEY.git](https://www.google.com/url?q=https://github.com/RahulKJoshi11/TodoEY.git&sa=D&source=hangouts&ust=1634399273650000&usg=AOvVaw2-Re_3TxlJq7t-DklKoTtK)

**Functionality Implemented:**

When the application is run a landing page opens to create a TODO list. Below are few functionalities I have implemented,

1. User can Register
2. User can Login
3. User can view his/her task list.
4. User can add and remove his/her task list.
5. User can edit his/her task list.
6. All changes are persistent within an application run.
7. Each task is displaying the date of Last Updates and description
8. All changes can be persistent to allow to view them with page refresh.
9. User can Logout.

**Approaches and architecture Used:**

Used **ASP.net Core 3.1** as the framework**, C#** as coding language, **SQLite** as database( In-memory Database), for developing the app. I have used **razor pages** to create the app and **cookie -based authentication** in Login. I have **used entity framework code first approach** to communicate with the database, and the database is in memory database, so it is created dynamically once the connection is open that is when application starts, and dropped once it is closed.

I have used Razor pages instead of MVC because of its simpler implementation which makes it ideal for smaller applications.

**Testing:** For testing I have used **MSTest for unit testing**.

**If given more time would have worked:** on testing, I would have covered more scenarios, also on improving code structure and application user interface using Java Script and would have implemented firebase authentication.