## Approaches used

### Three tier architecture –

Todo list uses the three tier architecture, to make all the components loosely coupled,

1. The **DataModels** component will have the POCO/plain classes, this will be shared with the BAL and Presentation layer/s.
2. DataModels are the replica of the database tables.
3. The BAL (Business application layer) will be shared with the presentation layer/s and holds the reference of **DataModels** layer.
4. The presentation layer holds the reference of DataModels and BAL both.
5. Any customization to DataModels will handle at presentation layer inside **Models** folder.
6. The customization’s like data annotations, change in fields as per view, will placed in Models folder of presentation layer.

### Redis

### 

1. Redis is an open source BSD licensed advanced key-value cache store and persistent storage, which also referred to as data structure server as keys can contain a string, hashes, lists, sets, bitmap, sorted lists, and hyperloglogs
2. Fastest performance among other cache storage techniques. As it applies indexes on columns.
3. Support for SignalR, which enables the live processing and transformation of contents. This feature will be in high demand in the future so that it will satisfy future demands.
4. Data persist until the business specified time, in other caches data will vanish after some time.
5. Other in-memory caching techniques follows limited data structure, hence the performance will impact.
6. Redis allows storing key and value pairs as large as 512 MB.
7. Redis supports multiple programming languages, and capable to communicate regardless of which language does the last changes.
8. Redis offers data replication.
9. Supports big data platform.

Note – To install redis cache please find the following URL,

<https://www.c-sharpcorner.com/article/installing-redis-cache-on-windows/>

## Any design patterns used

### Strategy Design Pattern

1. Here project task belong to two types of project types, first is agile and other is waterfall project’s task.
2. So here we used two Strategy first agile strategy and other is a waterfall or normal project strategy.
3. Each type of projects can have different terms like agile has burned hours, story points whereas waterfall has estimated time, the severity of the task.
4. The implemented pattern capable to handle present changes and future changes.
5. The new type of project can also be created by doing changes at three places in the scope of C#.

Observer Design Pattern

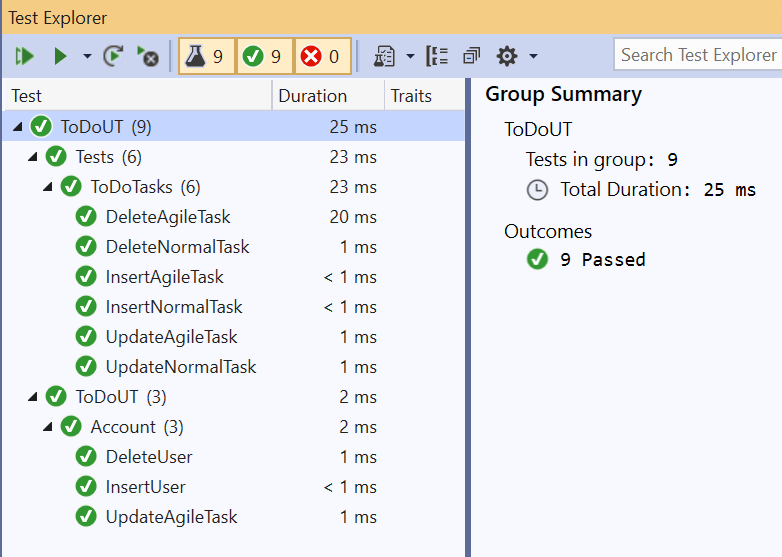
1. This is like triggers in SQL.
2. Notification sent after the execution of specific action like insert, update, delete and logged into the file.
3. So the developer has to concentrate on the main functionality, not on the other activity like logging the information.
4. Maintainance of logs will becomes smooth using the observer design pattern.

### Anything extra you would have done given more time

1. Atomic structure
2. Modular design pattern

## Unit testing

Following unit test cases are written to ensure the functionality is working correctly. It would be helpful to keep the system’s health for future changes. The below image shows the result of the unit test cases,



## Security features

### Forms authentication

* Token is issued at the time of login, which is valid for 20 miniutes
* With the reference of this token all the activities like, task creation, updation and deletion etc. are performed.
* Without authentication nobody will access the task functionality

### Antiforgery token

Antiforgery token is checks the authenticity of the person who is posting back the form.

Steps,

* In Antiforgery token is created at page load
* User is posting back the form with the anti forgery key token
* The token is validated using [ValidateAntiForgeryToken] filter

*What we achieved?*

When the un-authroized user calls the action method, then

1. Encoded antiforgery token will not exists and system declines the access to the action
2. Even user got the antiforgery key from somewhere, still its invalid token as will contains the user specific contents.

### Security Headers

Application is secured with the several security headers,

1. X-Permitted-Cross-Domain-Policies (sets to “none”), By using this header the application is not accessible from outside of domain.

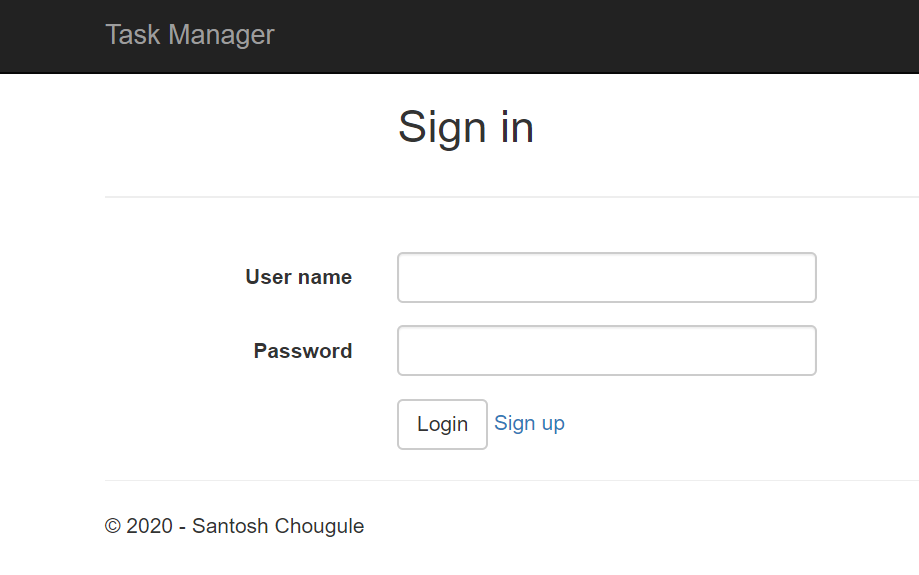
For example, [www.hack.com](http://www.hack.com) will not able to communicate (like via ajax call) with Todo list application.

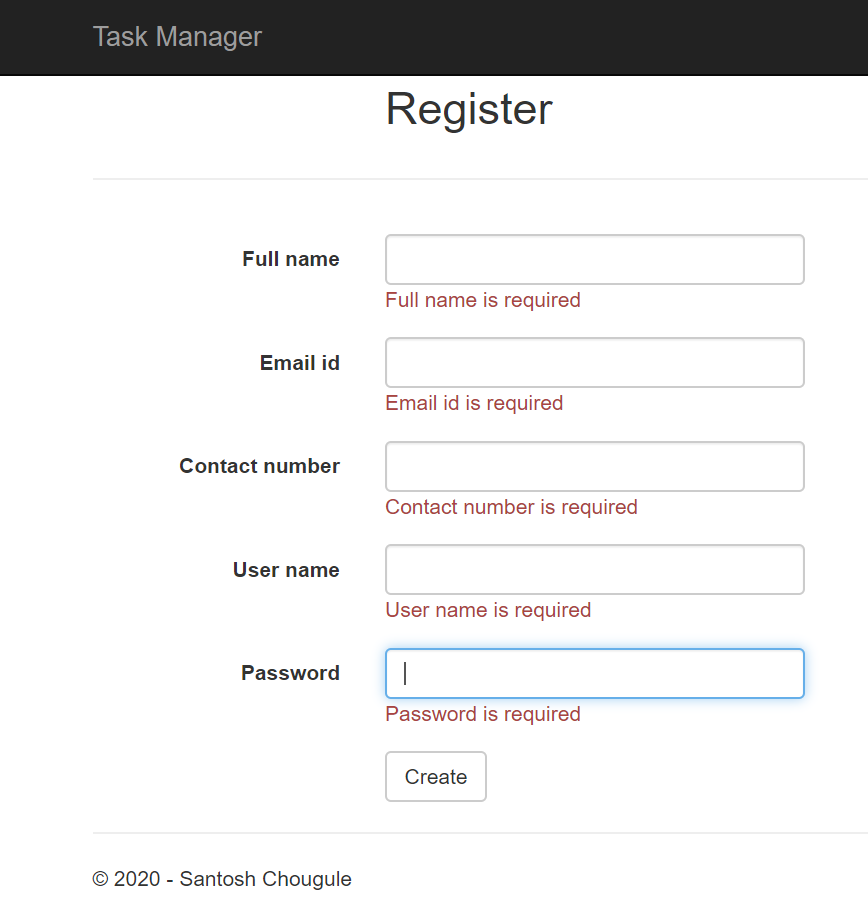
1. X-Frame-Options (sets to “DENY”), by using this header Todo list application will not be accessible through “iframes”.
2. Contene security policy, by using this header Todo list will fetch the contents from the specified sources.

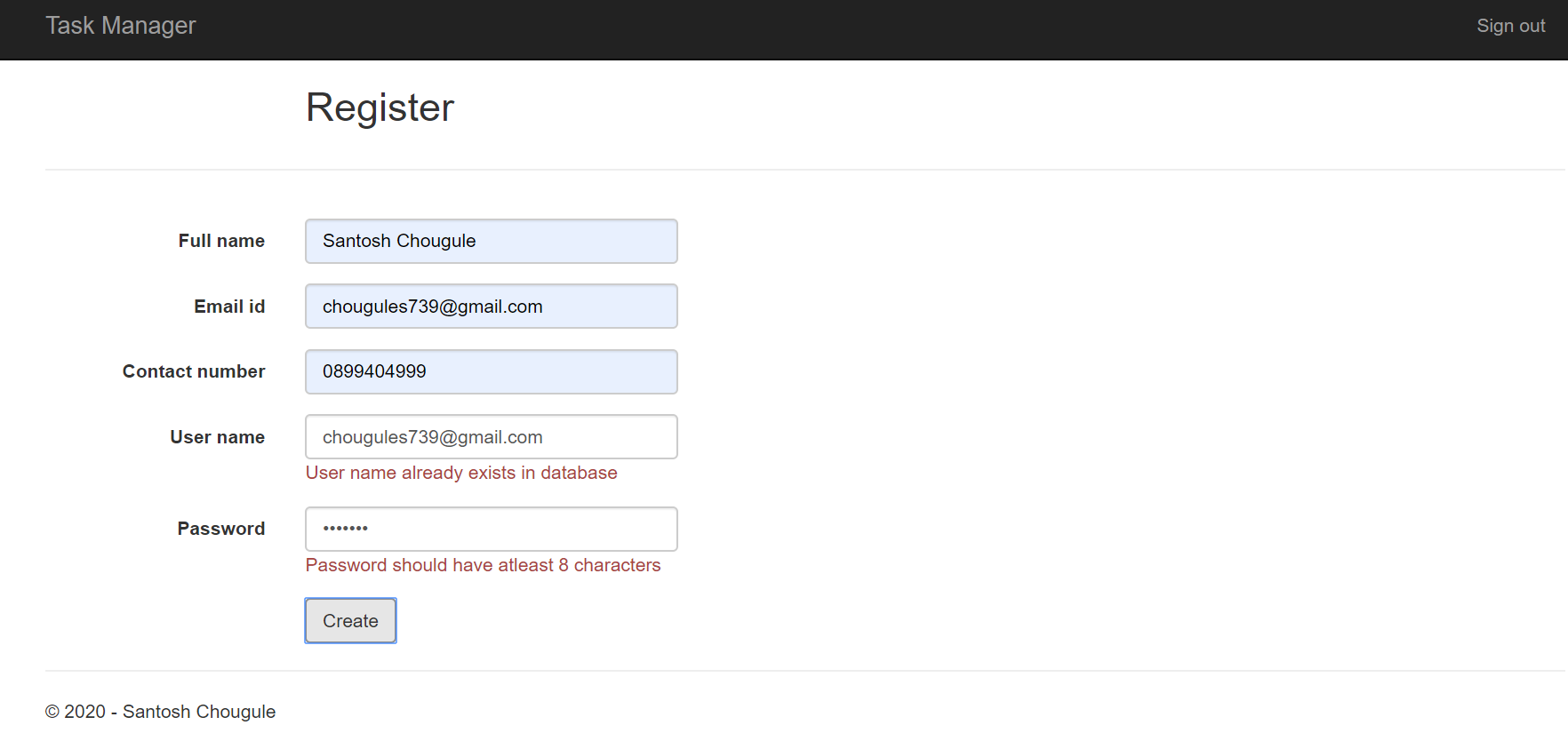
We also introduce other headers like, X-Content-Type-Options, Referrer-Policy, X-XSS-Protection, Strict-Transport-Security.

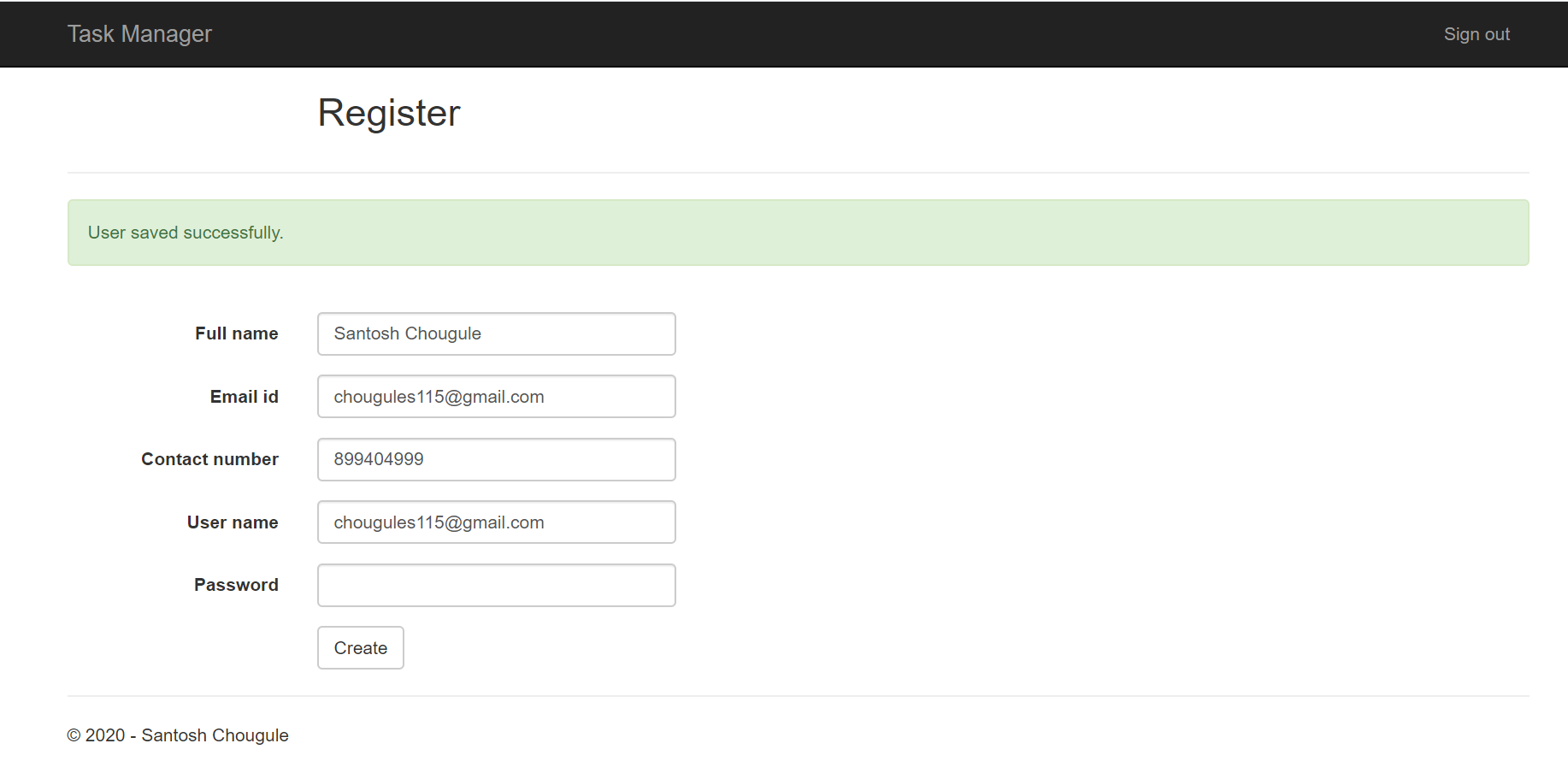
Note - The projects are configured statically due to the time constrain. System will by default adds the project types i.e. Waterfall and Agile.

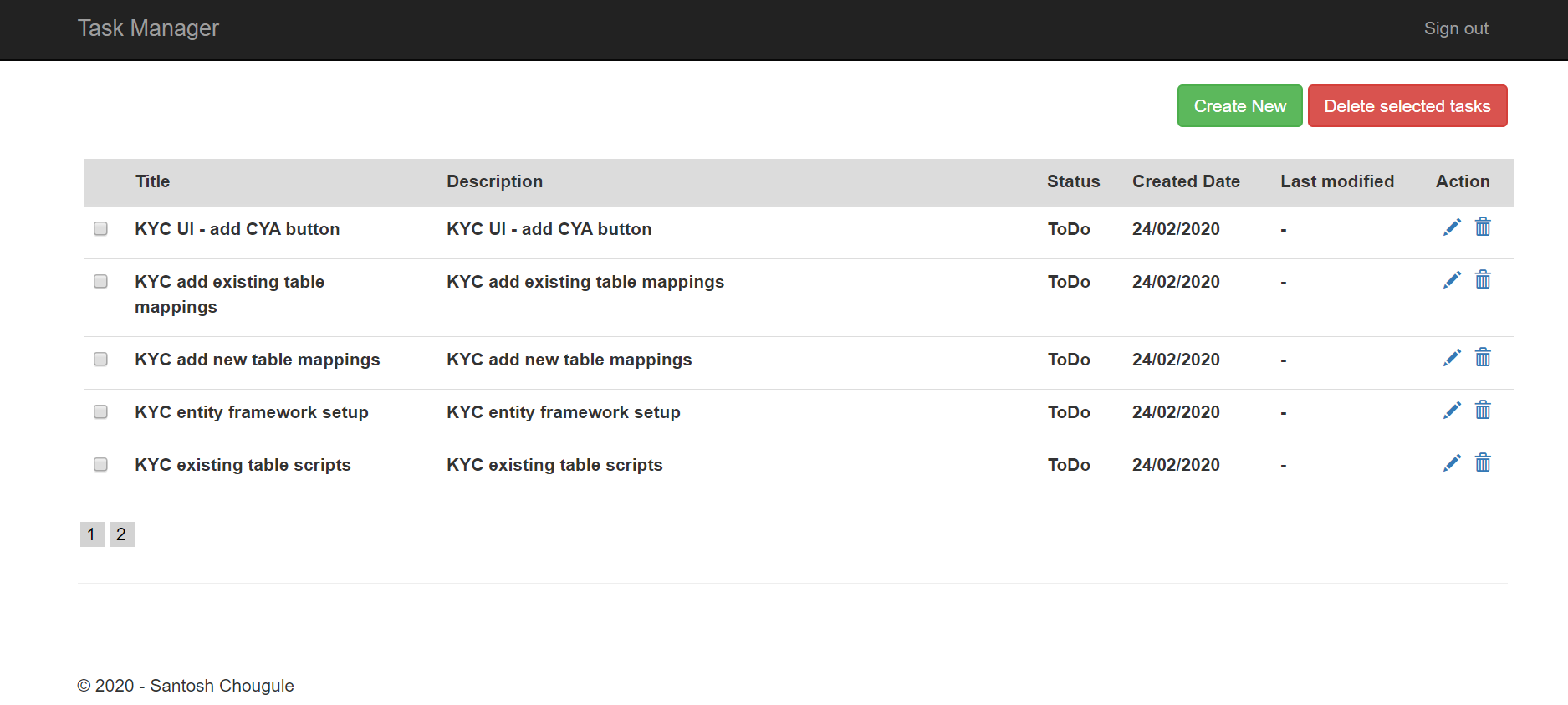
## Testing screen











Mobile screen,

