# C# Web Development Basics – Retake Exam

# TORSHIA

**TORSHIA** (**T**ask **O**rganization and **R**esponsive **S**ervice **H**istory **I**ntegration **A**pplication) is a web application for task management. You have been tasked to implement this application by the BG Corporation (**B**ai **G**osho). There are several requirements you must follow in the implementation.

## Technological Requirements

* Use the Web Server
* Use the MVC Framework
* Use Entity Framework Core

The Technological Requirements are **ABSOLUTE**. If you **do not follow** them, you will **NOT** be scored for other Requirements.

Now that you know the Technological Requirements, let us see what are the Functional Requirements.

## Database Requirements

The **Database** of the TORSHIA application needs to support **3 entities**:

### User

* Has an Id – a GUID String or an **Integer**.
* Has an Username
* Has a Password
* Has an Email
* Has an Role – can be one of the following values (“User”, “Admin”)

### Task

* Has an Id – a GUID String or an **Integer**.
* Has a Title
* Has a Due Date – a Date object (by **default** – false).
* Has a IsReported – a boolean.
* Has a Description
* Has Participants – (entered as comma separated string values... You can store them however you want)
* Has Affected Sectors – a collection which **may** contain **1** or **more** of the following values (“Customers”, “Marketing”, “Finances”, “Internal”, “Management”)

### Report

* Has an Id – a GUID String or an **Integer**.
* Has a Status – can be **one** of the following values (“Completed”, “Archived”).
* Has a Reported On – a Date object.
* Has a Task – a Task object.
* Has a Reporter – an User object.

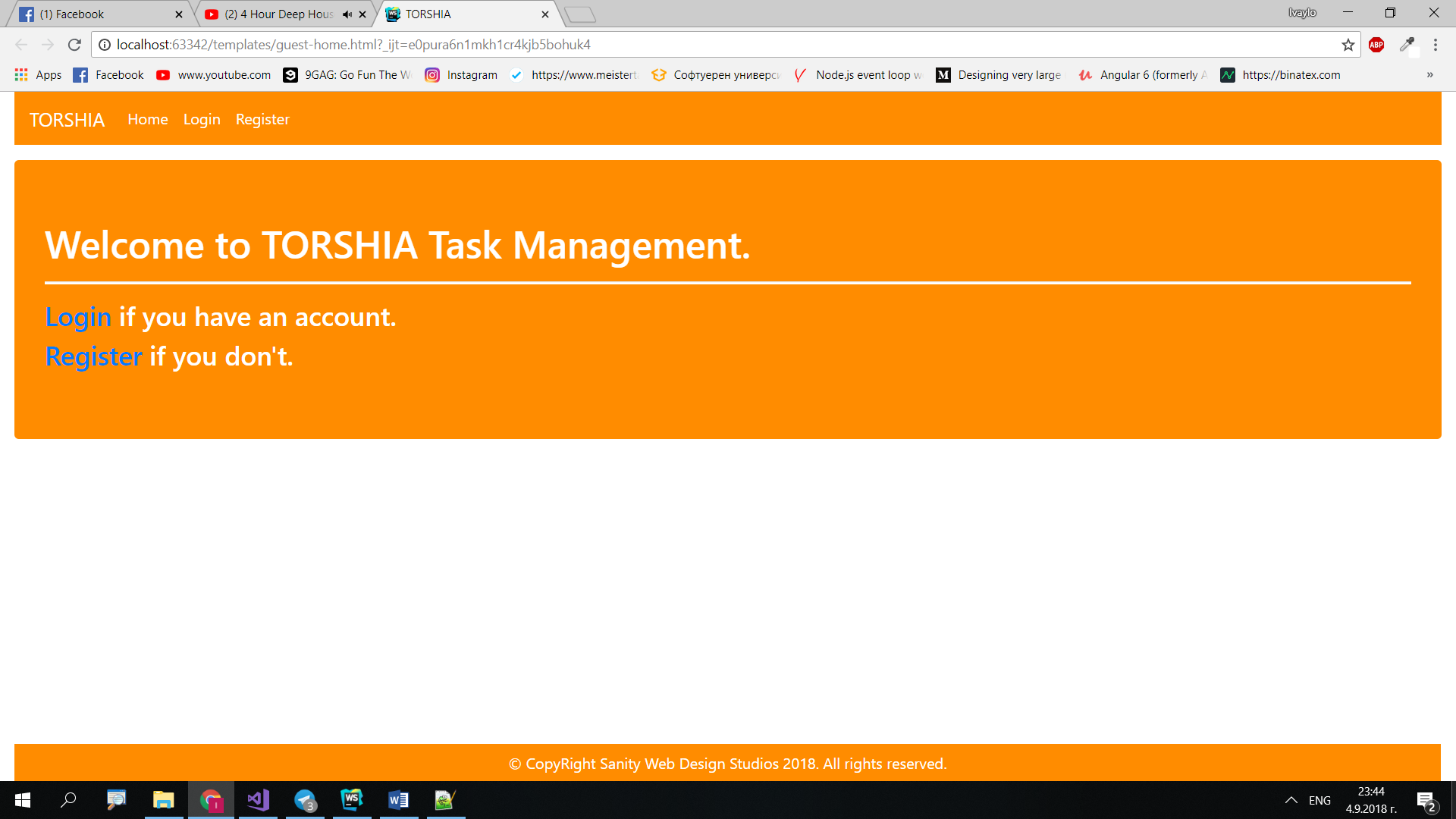
Implement the entities with the **correct datatypes**.

## Template Requirements

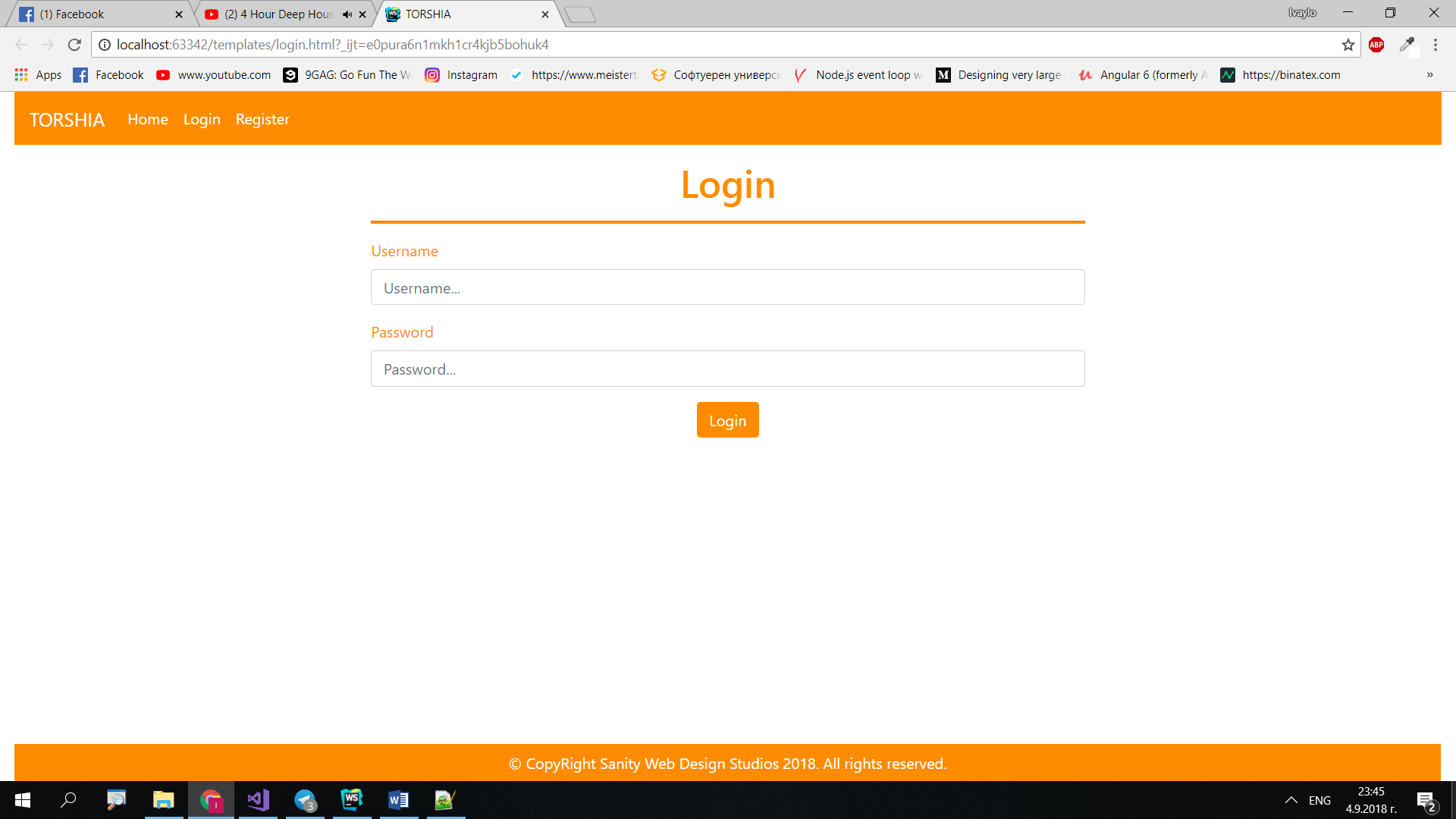
### Guest Templates

These are the **templates** and **functionalities**, accessible by Guests (**logged out** users).

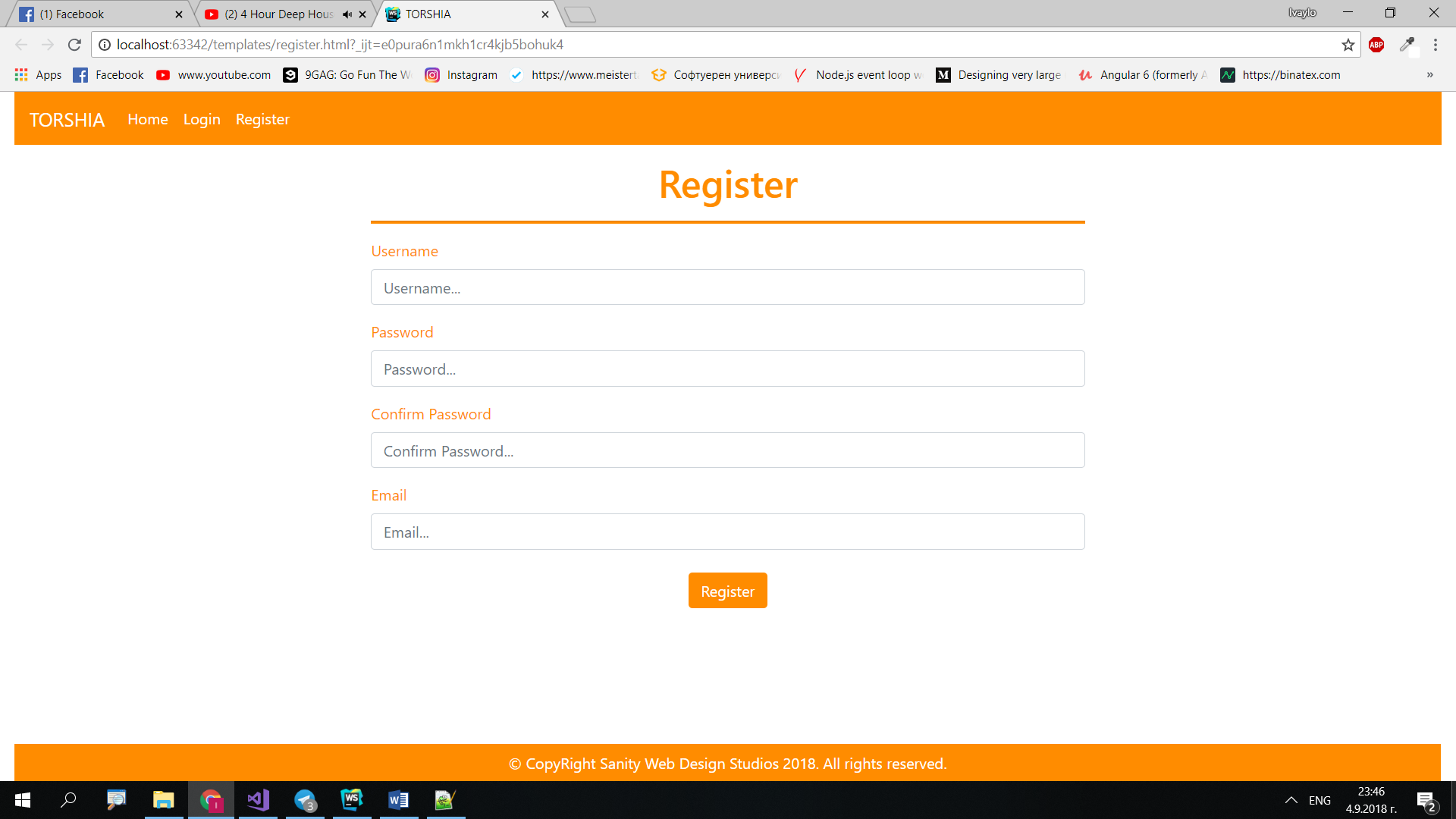
#### Index Template (route = “/Home/Index”) (logged out user)



#### Login Template (route = “/Users/Login”) (logged out user)



#### Register Template (route = “/Users/Register”) (logged out user)

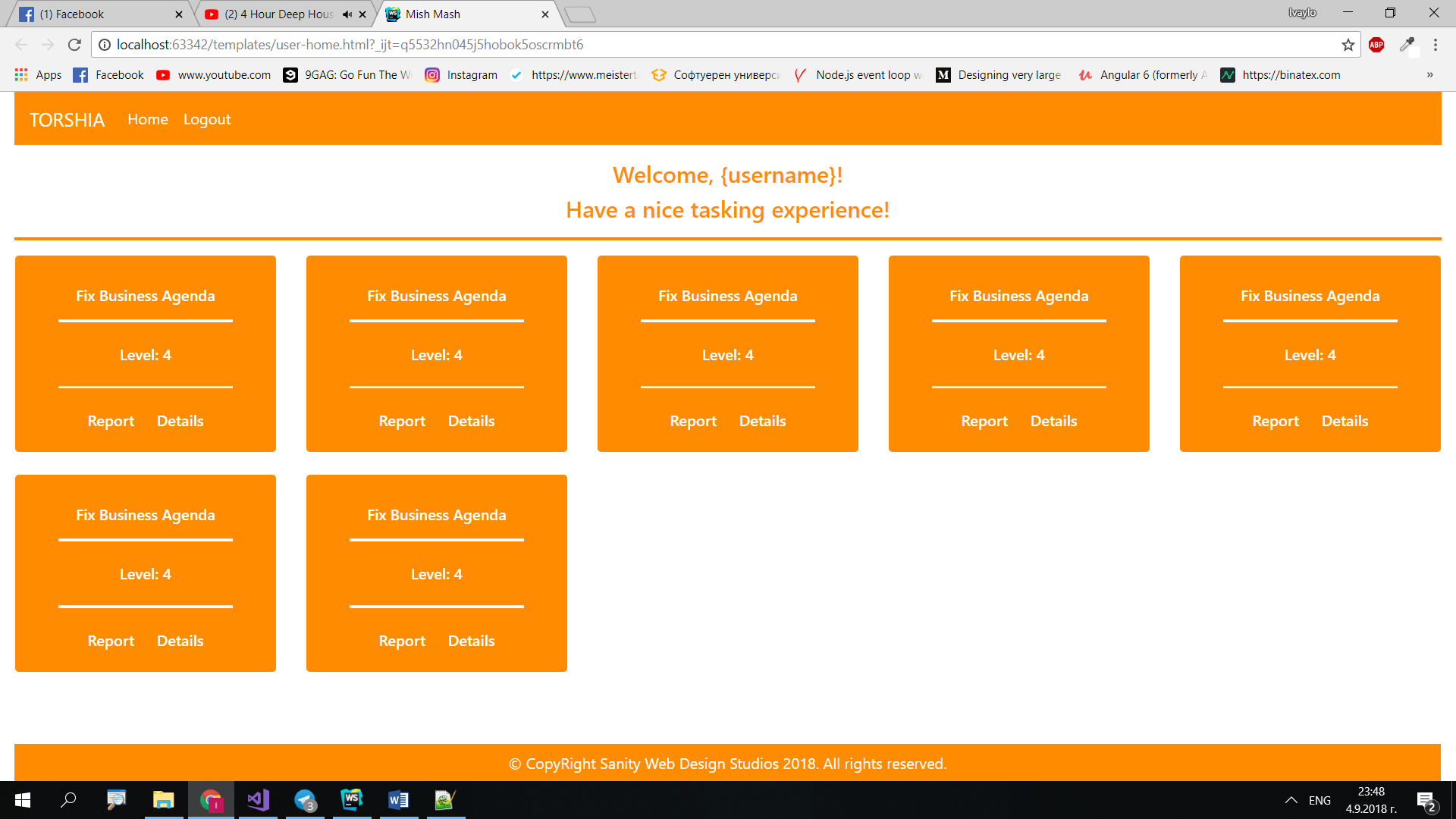


### User Templates

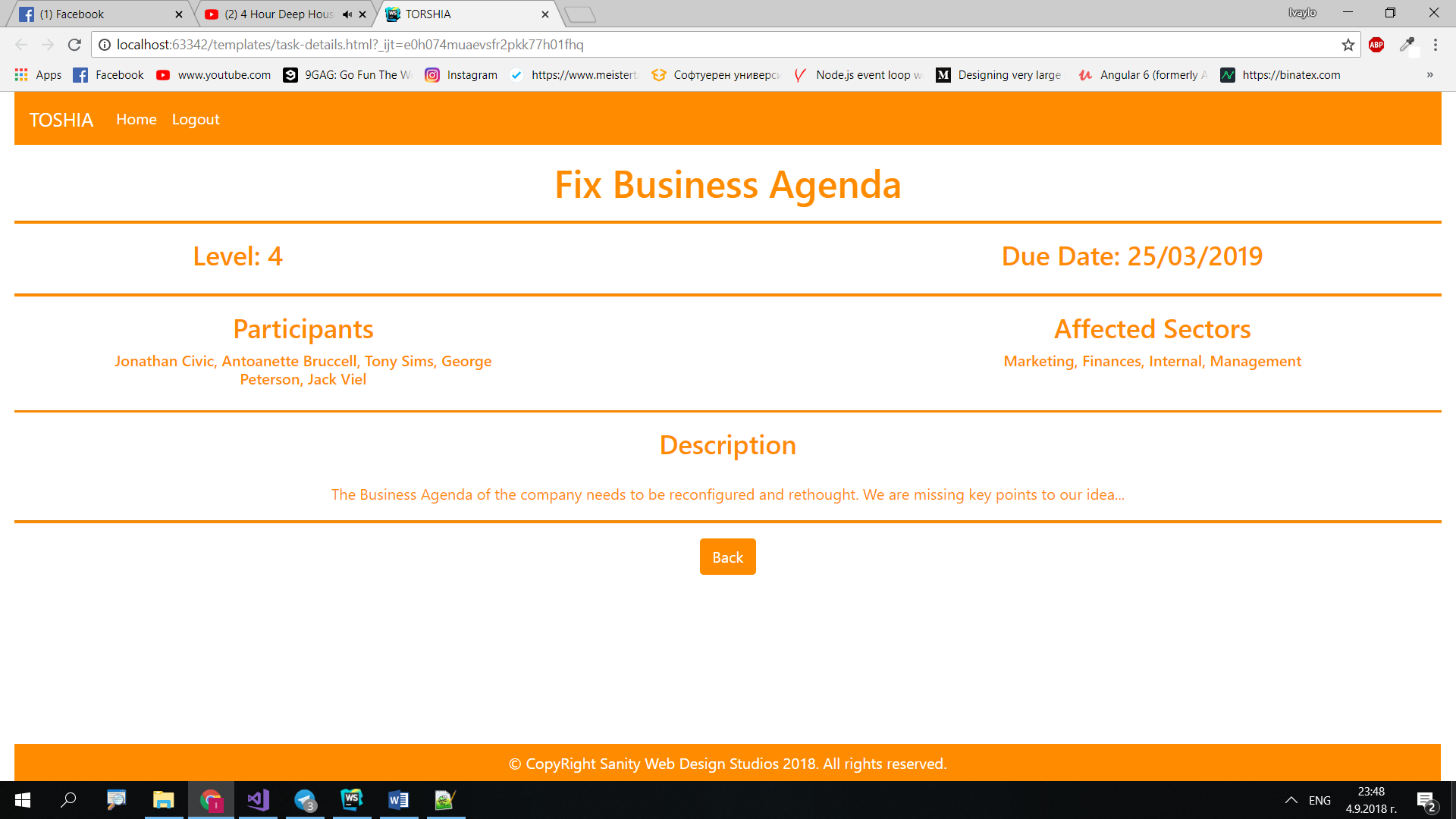
These are the **templates** and **functionalities**, accessible by Users (**logged in** users with Role - User).

#### LoggedIn Index Template (route=”/Home/Index”) (logged in user)

**NOTE**: As you can see the **elements** are **aligned** to the **left**, **regardless of their count**. The **maximum count** is **5 per** **row**, however they are **always aligned** to the **left**.



#### Task Details Template (route=”/Tasks/Details/{id}”) (logged in user)

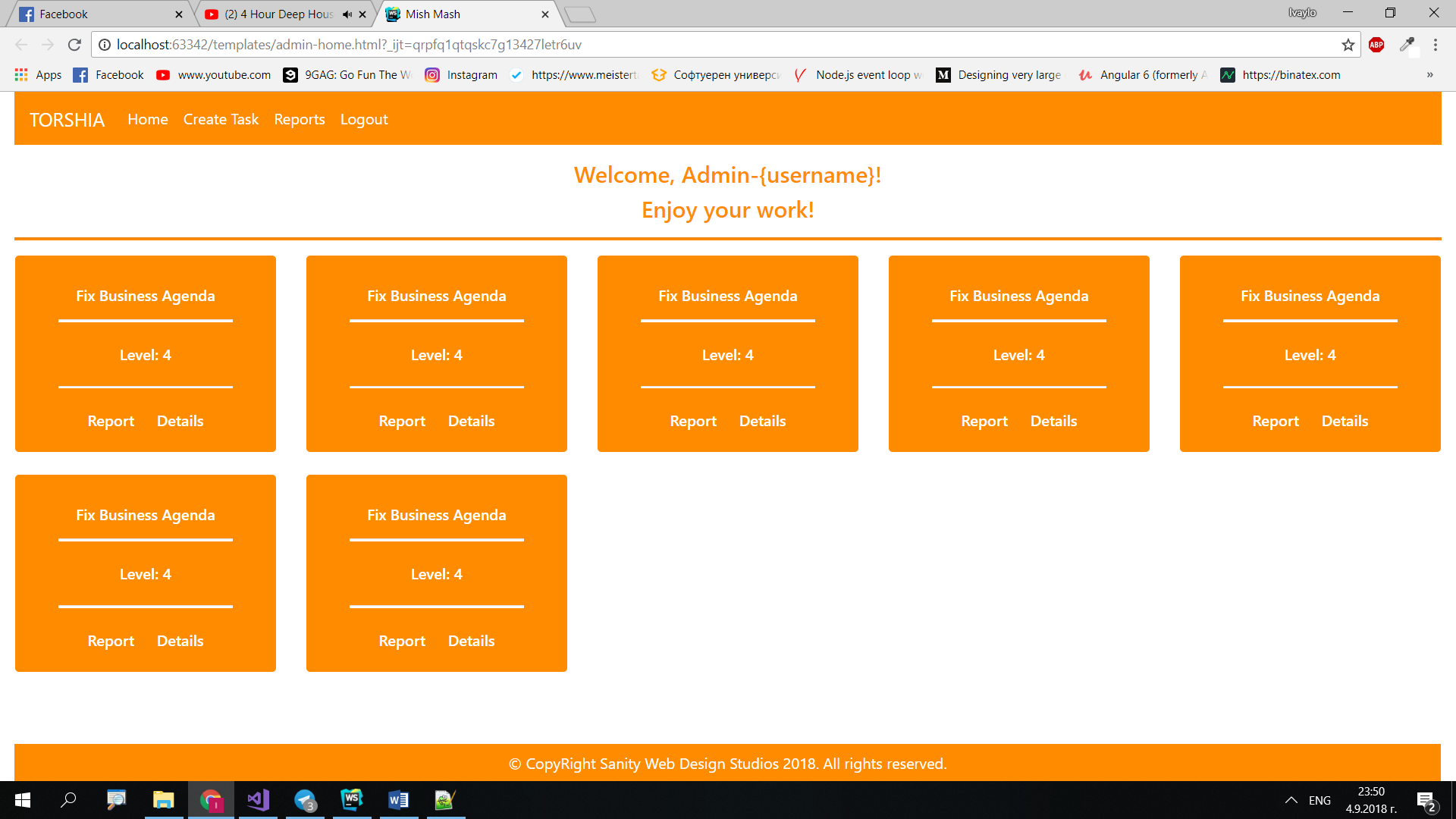


### Admin Templates

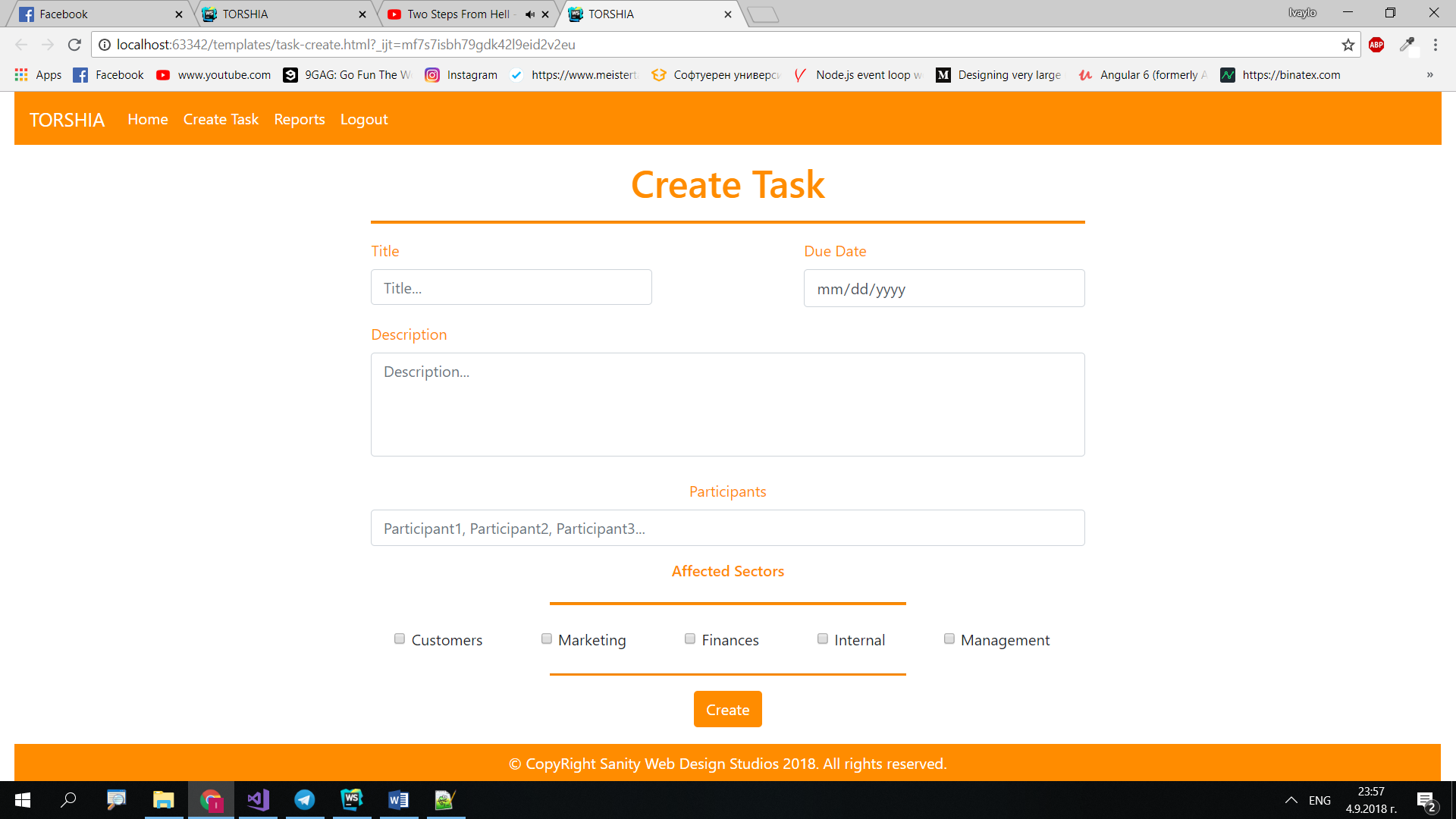
These are the **templates** and **functionalities**, accessible by Admins (**logged in** users with Role - Admin).

#### Admin Index Template (route=”/Home/Index”) (logged in admin)

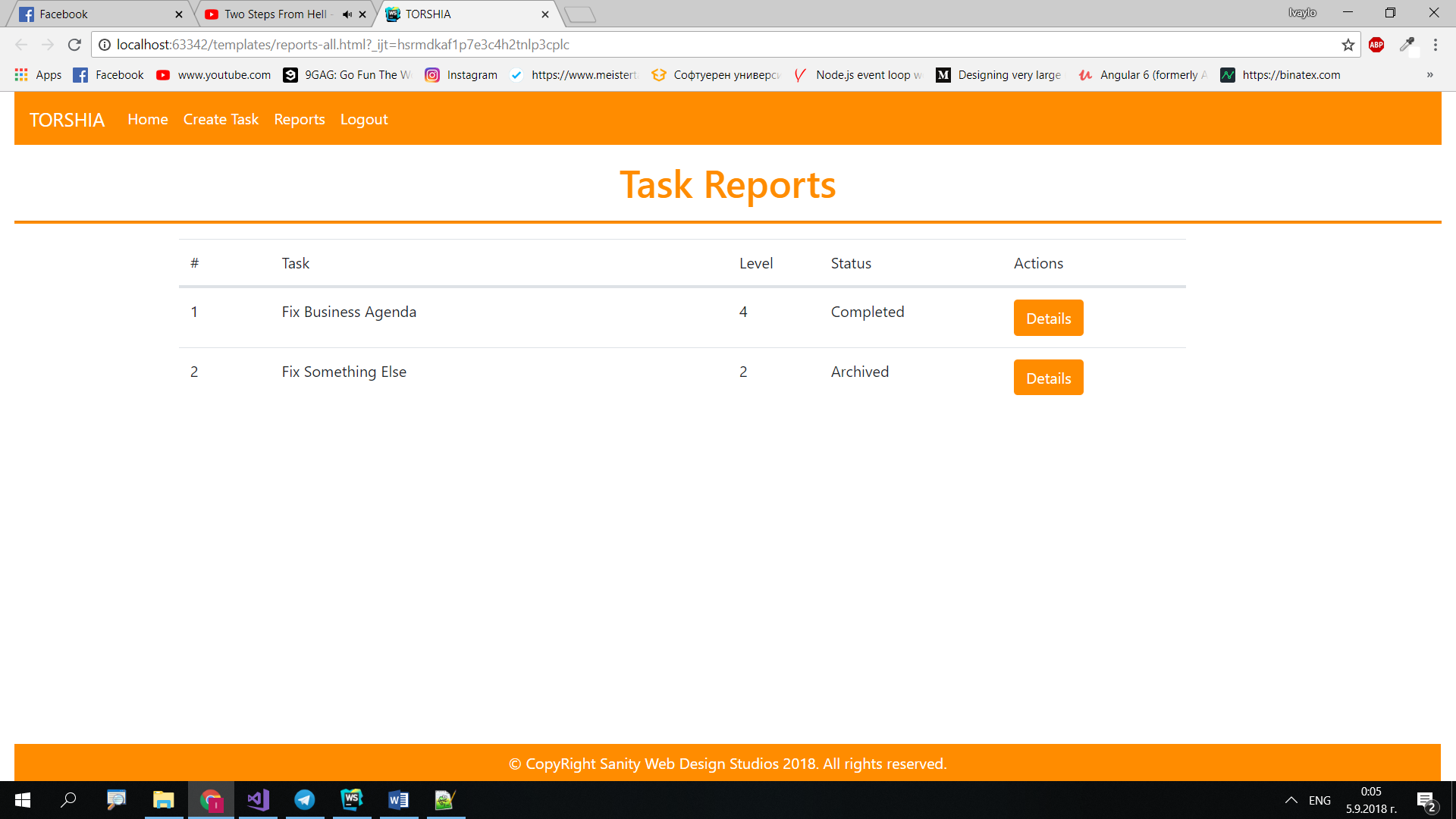
**NOTE**: As you can see the **elements** are **aligned** to the **left**, **regardless of their count**. The **maximum count** is **5 per** **row**, however they are **always aligned** to the **left**.



#### Task Create Admin Template (route=”/Tasks/Create”) (logged in admin)

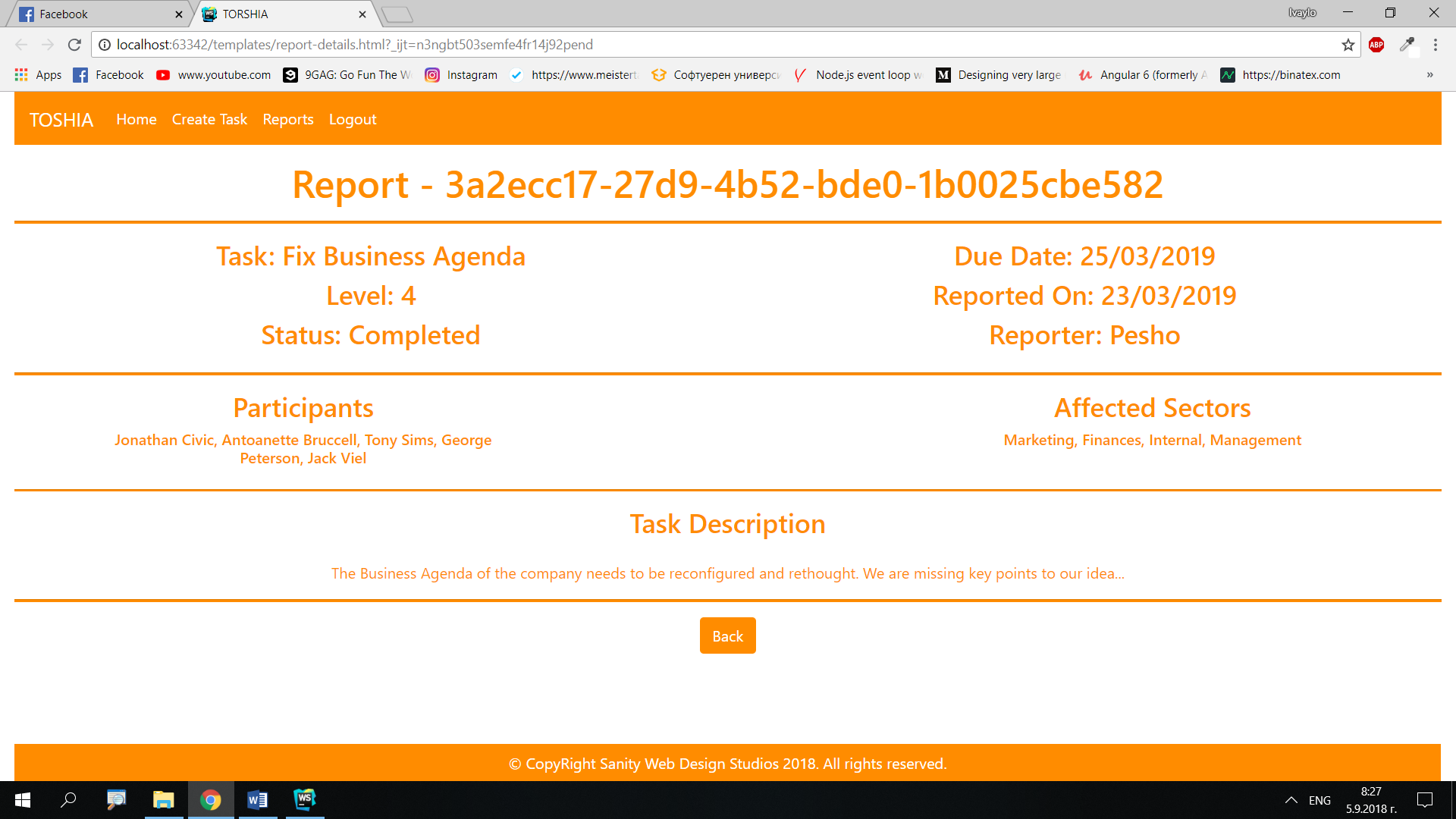


#### Reports Admin Template (route=”/Reports/All”) (logged in admin)



#### Task Create Admin Template (route=”/Reports/Details/{id}”) (logged in admin)

**NOTE**: If you are using **integer Id**’s just output the **integer id** instead.



Some of the templates have been given to you in the application skeleton, but the others will be for you to implement, so make sure you implement them correctly. You can use the given ones as helpers.

**NOTE**: The templates should look **EXACTLY** as shown above.

**NOTE**: The templates do **NOT** **require** **additional** **CSS**. Only **bootstrap** and the given **style.css** are enough.

## Functional Requirements

The Functionality Requirements describe the functionality that the **Application** must support.

The **application** should provide Guest (not logged in) users with the functionality to login, register and **view** the Index page.

The **application** should provide Users (logged in) with the functionality to logout, and view all tasks, and **report** a Task, and **view details** about a Task.

The **application** should provide Admins (logged in, with role - Admin) with the functionality to logout, and view all tasks, and **report** a Tasks, and **view details** about a Task, **create** a Task, view all Reports, **view details** about a Report.

When you register a new User, it should be assigned with a **role** – User.

### Users

Users can **report** or **Tasks** or **view details** aboutthem. If a Task is **NOT reported**, it should be represented on the Index page as a rectangular element, which holds **2 buttons** at its bottom – [Report] and [Details].

* Upon clicking the [Report] button, the Task should be **reported** (set isReported to true) and a Report should be created with the **given Task** and the **current Date** as ReportedOn, and the **current user** as Reporter.
* Upon clicking the [Details] button, the **current user** should be **redirected** to the Task’s Details Page.

If a Task is **reported**, it should **NOT be represented** on the Index page.

### Tasks

Tasks are naturally just data entities. They don’t have much business logic around themselves. There is however one property called Level which is **represented** here and there. The Level is defined by the **count** of **sectors** the **Task** **affects**. It can be a minimum of Level 0 (no affected sectors chosen upon creation) and a **maximum** of Level 5 (all affected sectors chosen upon creation). The Level should **NOT** be **stored** in the **Database**.

### Reports

Reports are also just data entities. They don’t have much business logic around themselves. When **Reports** are **created**, though, they are **initialized** with a **Reported On** (which is the **date of creation**), a Task (which is the **reported Task**) and a Reporter (which is the **currently logged in user**). Reports are also initialized with a Status, which should be based on a **random chance**:

* **75%** chance - Completed
* **25%** chance – Archived

This can be done with a simple Random.

## Security Requirements

The Security Requirements are mainly access requirements. Configurations about which users can access specific functionalities and pages.

* Guest (not logged in) users can access Index page and functionality.
* Guest (not logged in) users can access Login page and functionality.
* Guest (not logged in) users can access Register page and functionality.
* Users (logged in) can access User LoggedIn Index page and functionality.
* Users (logged in) can access User Task Details page and functionality.
* Users (logged in) can access the Task Report functionality.
* Users (logged in) can access Logout functionality.
* Admins (logged in) can access **every functionality** a **normal** logged in User can.
* Admins (logged in) can access Admin LoggedIn Index page and functionality.
* Admins (logged in) can access the Task Create page and functionality.
* Admins (logged in) can access the All Reports page and functionality.
* Admins (logged in) can access the Report Details page and functionality.

## Scoring

This section describes how the scoring of the Exam will be made.

### Database Requirements – 15 points.

### Template Requirements – 35 points.

### Functional Requirements – 40 points.

### Security Requirements – 10 points.