

All code mentioned in this document can be found on the github at:

<https://github.com/ludan1214/java-fsd-phase3/tree/main/TaskManager>

Made according to the specifications listed here:

[Requirements \(Click Here\)](#)

1. Setup

- ① A MySQL server was started on localhost port 3308 using docker desktop, a user table was created in phase3db with the following command:

```
➤ use phase3db;
➤ CREATE TABLE IF NOT EXISTS `phase3db`.`usertable` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `name_UNIQUE` (`username` ASC) VISIBLE)
ENGINE = InnoDB;
```

- ② And a task table was made with the following command:

```
➤ CREATE TABLE IF NOT EXISTS `phase3db`.`task_tbl` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  `start_date` DATETIME NULL,
  `end_date` DATETIME NULL,
  `description` VARCHAR(250) NULL,
  `severity` VARCHAR(10) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

- ③ Spring Setup

The spring initializr was used to setup the project and the following Maven dependencies were included in the POM either manually or through the initializer:

1. Spring Web
2. Spring Data JPA
3. Spring Security
4. MySQL connector
5. Lombok
6. Springfox Swagger 2
7. JSTL
8. Jasper

2. Project Overview

A simple task manager application that supports the creation, deletion, updating, and displaying of tasks as well as user registration and login was implemented.

Two entities were implemented (Task and TaskUser) as well as two repositories (UserRepository and TaskRepository) and two services (UserService and TaskService).

Task was mapped to TaskUser with a @ManyToOne annotation with “user_id” as the foreign key. This allows us to retrieve tasks specific to certain users and display only tasks that were created by that user.

Spring Security was used to handle the login verification and sessions. It was also used to restrict endpoints until the user is logged in. The configuration for it is located in the WebSecurityConfig class. Additionally a service called TaskUserDetailsService implemented a function called loadUserByUsername defines the verification function that extracts the username and password information from the TaskUser object.

Custom exceptions:

TaskAlreadyExistsException,

TaskNotFoundException,


UserAlreadyExistsException,

UserNotFoundException

The exceptions above were caught and handled by the controllers. Information about the errors are displayed in the console using log4j as well as on the webpage itself via an error notification message which is demonstrated in the demo section of this document below.

Swagger was used to generate documentation for the various POST and GET endpoints. The configuration for it is located in the SpringFoxConfig class. This can be visited by directly visiting <http://localhost:8080/swagger-ui.html#/> after starting up the application (assuming default spring settings).

Note: The configured endpoints will be hidden by spring security, so for the documentation to show, the user must first register and login.

 **swagger**

public-api (/v2/api-docs?group=public-api) Explore

JavaInUse API

JavaInUse API reference for developers

Created by javainuse@gmail.com
[JavaInUse License](#)

basic-error-controller : Basic Error Controller

Show/Hide | List Operations | Expand Operations

DELETE	/error	error
GET	/error	error
HEAD	/error	error
OPTIONS	/error	error
PATCH	/error	error
POST	/error	error
PUT	/error	error

login-registration-controller : Login Registration Controller

Show/Hide | List Operations | Expand Operations

GET	/loginError	showLoginError
POST	/register	submitRegister

main-controller : Main Controller

Show/Hide | List Operations | Expand Operations

GET	/	showIndex
GET	/createTask	showPostTask
GET	/dashboard	showDashboard
GET	/deleteTask	showDeleteTask
GET	/login	showLogin
GET	/register	showRegister
GET	/updateTask	showUpdateTask

task-controller : Task Controller

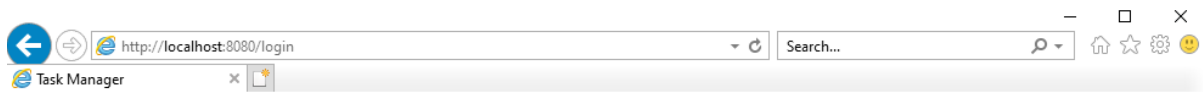
Show/Hide | List Operations | Expand Operations

POST	/addTask	submitTask
POST	/deleteTask	deleteTask
GET	/displayTasks	displayTasks
POST	/updateTask	updateTask

[BASE URL: / , API VERSION: 1.0]

3. Project Demo

Visiting the site we are immediately brought to the login page, we can choose to register or login with credentials.



Sign in

Login

Register

Inputting invalid information will bring up an error.



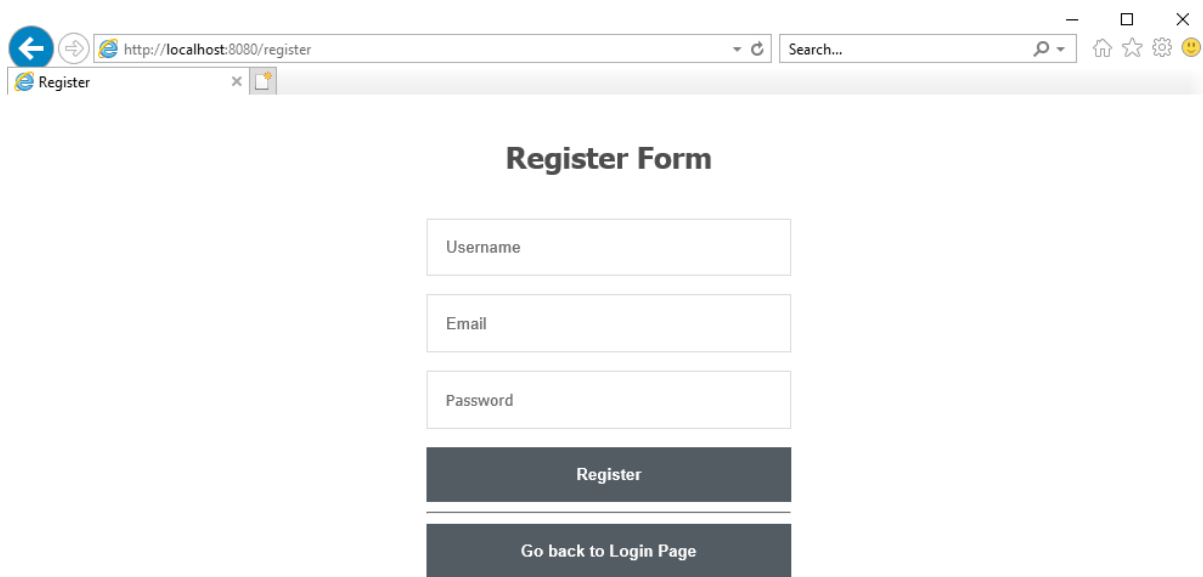
Sign in

Username/Password combination not found!

Login

Register

Visiting the register page, we can see a form that lets us input new information



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/register`. The page title is "Register". The main content area features a "Register Form" with three input fields: "Username", "Email", and "Password". Below these fields are two buttons: "Register" and "Go back to Login Page".

Register Form

Username

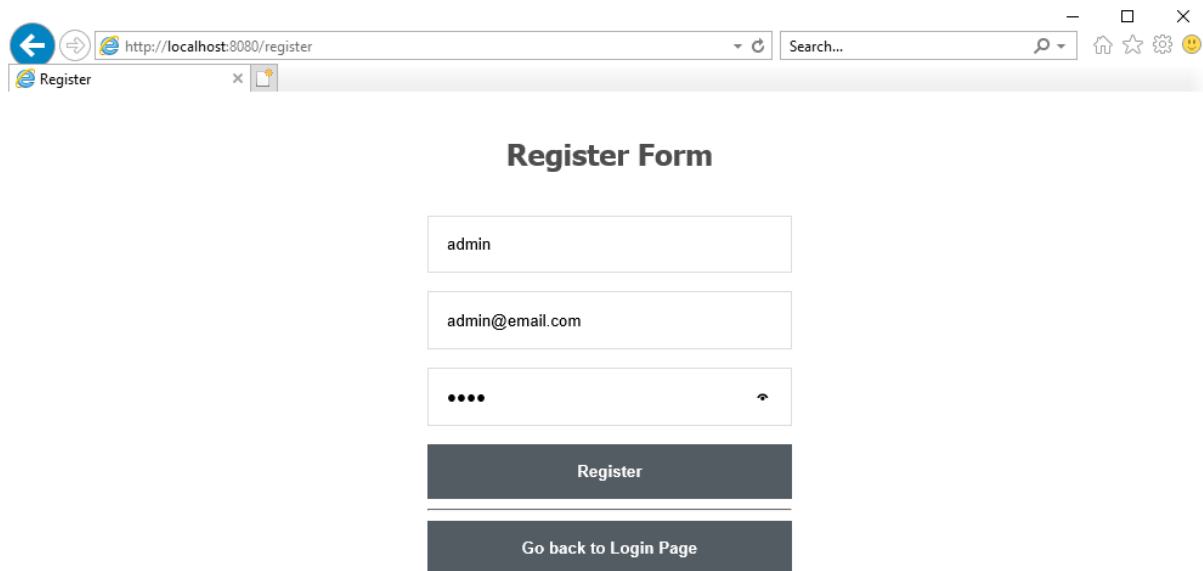
Email

Password

Register

Go back to Login Page

We create a dummy account with the username “admin” and register.



The screenshot shows the same "Register Form" page, but with the "Username" field containing the text "admin" and the "Email" field containing the text "admin@email.com". The "Password" field is masked with four dots. The "Register" and "Go back to Login Page" buttons are still visible.

Register Form

admin

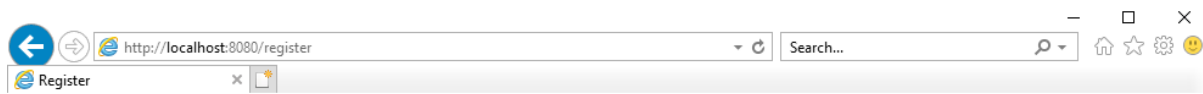
admin@email.com

....

Register

Go back to Login Page

Pressing the “Register” button will bring the user to the index page to login again. Registering with a pre-existing username will cause an error.



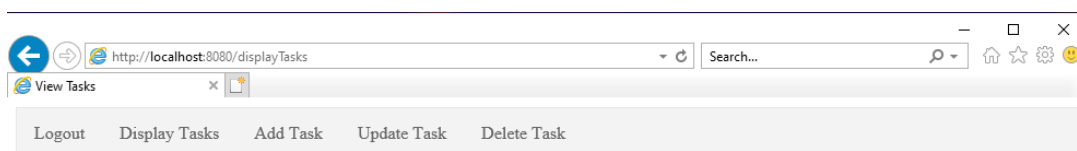
Register Form

Register Failed: User already exists!

Register

Go back to Login Page

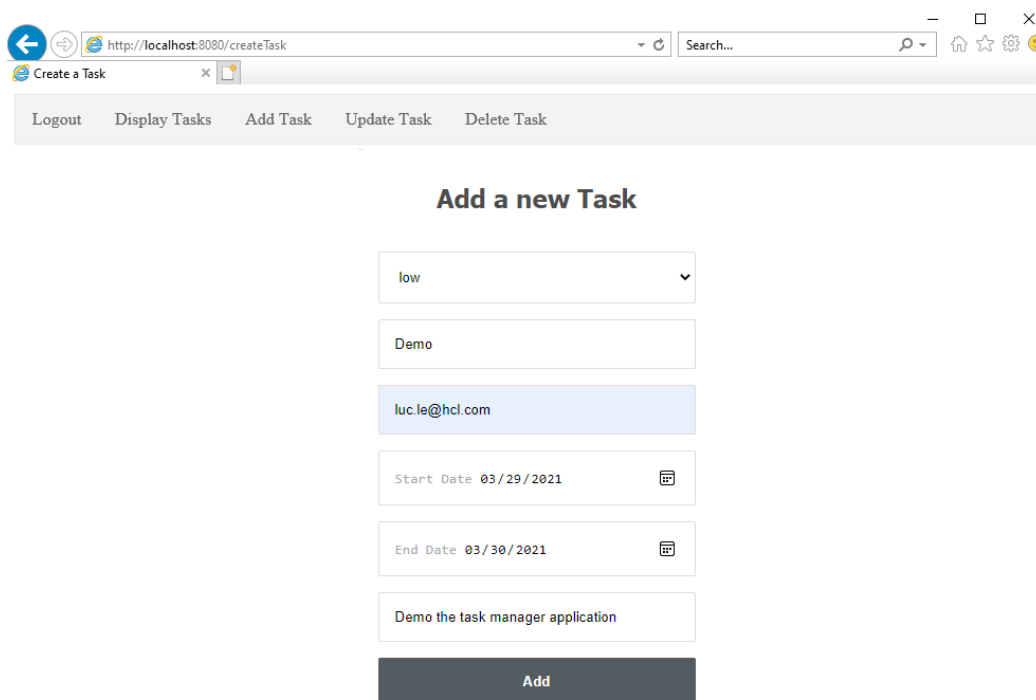
After returning to the login page and logging in with the details we registered with, we will arrive at the landing page which will display the user's tasks. There currently are no tasks as we haven't added any yet.



List of Tasks

Id	Name	Email	Severity	Start Date	End Date	Description
----	------	-------	----------	------------	----------	-------------

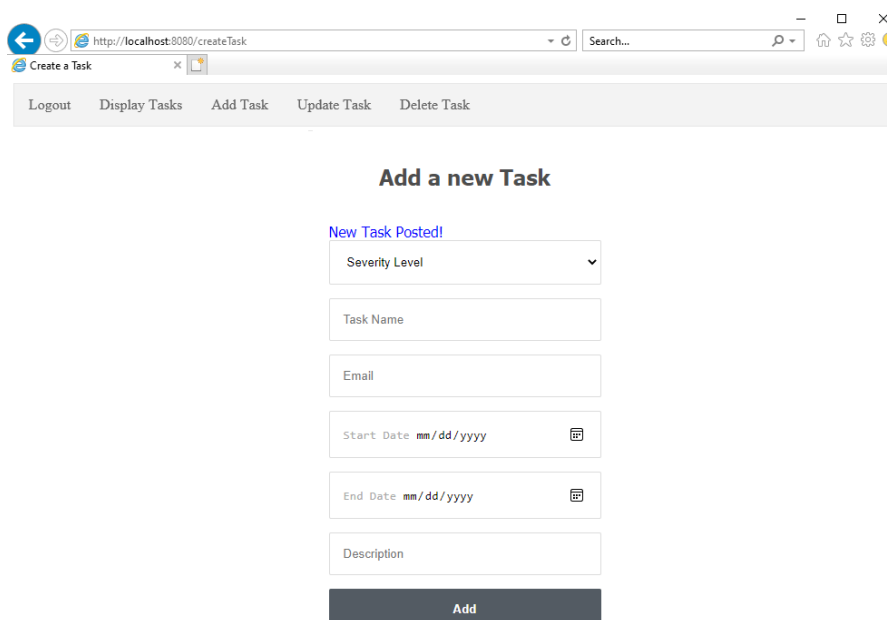
Clicking the “Add Task” tab will bring us to a form to add a new task



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/createTask`. The browser's tab is labeled "Create a Task". Below the browser window is a navigation bar with five tabs: "Logout", "Display Tasks", "Add Task", "Update Task", and "Delete Task". The "Add Task" tab is currently selected. The main content area is titled "Add a new Task" and contains a form with the following fields:

- A dropdown menu for "Severity Level" with "low" selected.
- A text input field for "Task Name" containing the value "Demo".
- A text input field for "Email" containing the value "luc.le@hcl.com".
- A date input field for "Start Date" showing "03/29/2021".
- A date input field for "End Date" showing "03/30/2021".
- A text input field for "Description" containing the value "Demo the task manager application".
- A dark grey "Add" button at the bottom of the form.

Pressing “Add” will send the form to the LoginRegistration Controller to process. If the task successfully posted, a message will appear.



This screenshot shows the same "Add a new Task" form after a successful submission. A blue message "New Task Posted!" is displayed at the top of the form area. The form fields are now empty, with the following labels:

- "Severity Level" dropdown menu.
- "Task Name" text input field.
- "Email" text input field.
- "Start Date" date input field with the placeholder "mm/dd/yyyy".
- "End Date" date input field with the placeholder "mm/dd/yyyy".
- "Description" text input field.
- A dark grey "Add" button at the bottom.

If the user tries to create a task with the same name, an error message will appear.

Task name already exists, please enter a different name

Severity Level

Task Name

Email

Start Date mm/dd/yyyy

End Date mm/dd/yyyy

Description

Add

Now visiting the display page again, we can verify that the task has been successfully posted.

Logout Display Tasks Add Task Update Task Delete Task

List of Tasks

Id	Name	Email	Severity	Start Date	End Date	Description
3	Demo	luc.le@hcl.com	low	2021-03-29 00:00:00.0	2021-03-30 00:00:00.0	Demo the task manager application

Now suppose we wanted to update that task, we can click the “Update Task” tab which will bring us to a new form.

http://localhost:8080/updateTask

Search...

Update a Task

Logout Display Tasks Add Task Update Task Delete Task

Update a Task

Severity Level ▼

This task isn't real

example@email.com

Start Date 03/30/2021

End Date 03/30/2021

asdf

Update

Suppose we tried to update a task that doesn't exist, an error will appear.

http://localhost:8080/updateTask

Search...

Update a Task

Logout Display Tasks Add Task Update Task Delete Task

Update a Task

Task not found!

Severity Level ▼

Task Name

Email

Start Date mm/dd/yyyy

End Date mm/dd/yyyy

Description

Update

Now if we properly use the correct task name we used when we added a task, the task will successfully update and a message will appear.

The screenshot shows a web browser window with the address bar at `http://localhost:8080/updateTask`. The page has a navigation bar with links: Logout, Display Tasks, Add Task, Update Task, and Delete Task. The main heading is "Update a Task". Below it, a blue message says "Task Successfully updated!!". The form contains the following fields: a dropdown menu for "Severity Level", text input for "Task Name", text input for "Email", date input for "Start Date" (format mm/dd/yyyy), date input for "End Date" (format mm/dd/yyyy), and a text area for "Description". At the bottom is a dark "Update" button.

We can verify this by visiting the display page again.

The screenshot shows a web browser window with the address bar at `http://localhost:8080/displayTasks`. The page has a navigation bar with links: Logout, Display Tasks, Add Task, Update Task, and Delete Task. The main heading is "List of Tasks". Below it is a table with one row of task data.

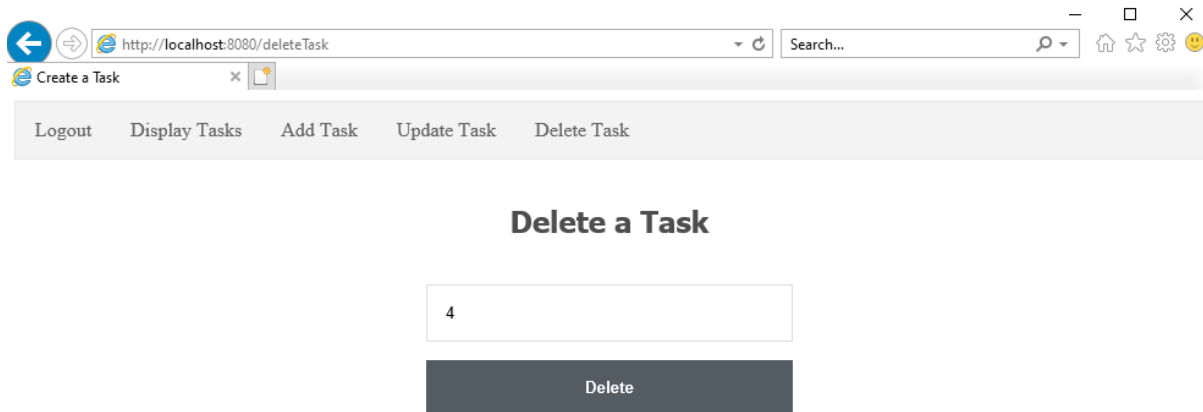
List of Tasks

Id	Name	Email	Severity	Start Date	End Date	Description
3	Demo	luc.le@hcl.com	low	2021-03-29 00:00:00.0	2021-03-31 00:00:00.0	Updated the task!

We can see an updated description after filling out the form and pressing update.

The last feature with tasks is the delete feature, in which we can delete a task by its Id listed in the leftmost column.

From above, we know that the task id is 3, but suppose we tried to delete a task of id 4 which shouldn't exist.



http://localhost:8080/deleteTask

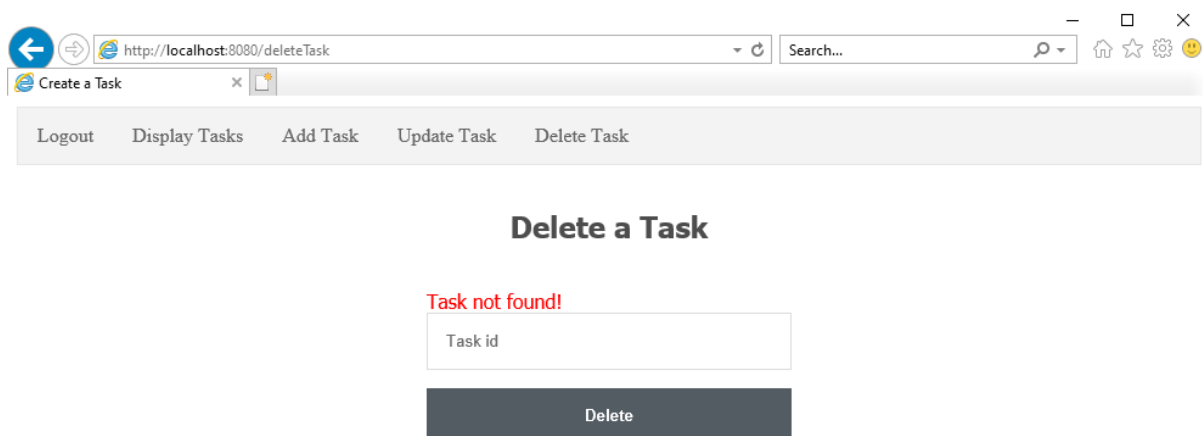
Create a Task

Logout Display Tasks Add Task Update Task Delete Task

Delete a Task

Delete

An error will appear. Additionally, if a task that exists with that ID is found under another user, the controller will check to see if the logged in user's username matches the user associated with the task. If the names do not match, the delete request will be rejected with the same error below. So users cannot delete or modify each other's tasks.



http://localhost:8080/deleteTask

Create a Task

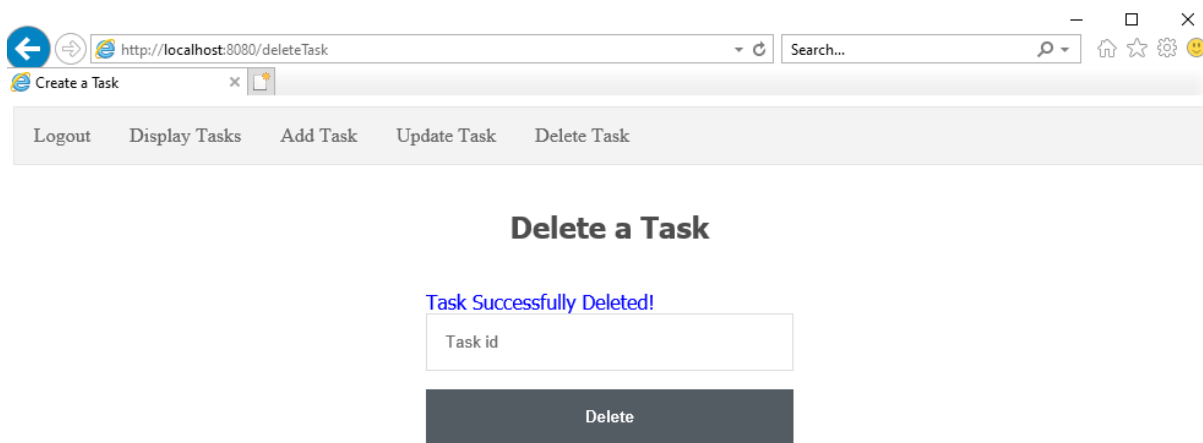
Logout Display Tasks Add Task Update Task Delete Task

Delete a Task

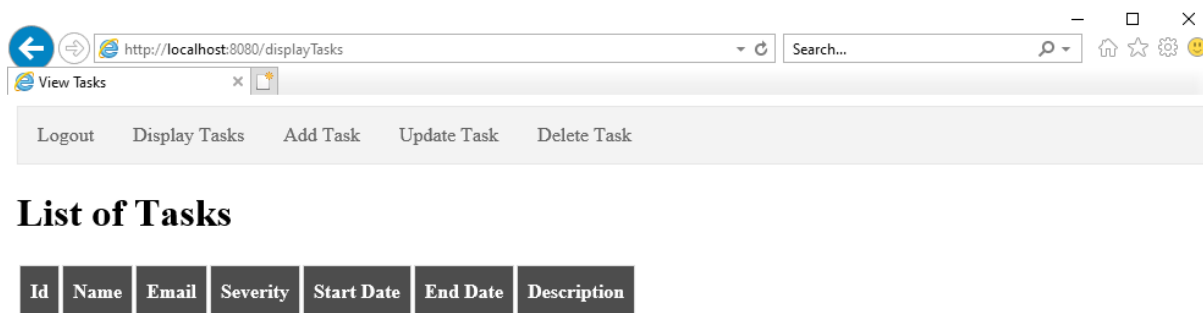
Task not found!

Delete

On a successful delete, a success message will appear.



We can verify the task has been deleted by visiting the display page once again.



Finally since we are done demonstrating all of the functionality of the application, we can logout of the session by pressing the logout button in the top left corner.

This will bring us to the login page where can choose to register a new user or login once again.



Sign in

Username

Password

Login

Register

This concludes all of the basic functionality of the Task Manager application.