# Exam Preparation - TODO List

You have been tasked to create a simple **TODO List** application. The application should hold tasks, which are the main **entities**.

The functionality of the application should support creating, listing and deleting tasks.

The application should persist the data into a database.

## Overview

Your application should be built on **each one** of the **following technologies**:

### PHP

* **Symfony** as the **Framework**
* **Twig** as the **View engine**
* **MySQL** as the **Database**

### JavaScript

* **NodeJS** + **ExpressJS** as **Frameworks**
* **Handlebars.js** as the **View engine**
* **MongoDB** as the **Database**

### Java

* **Spring** as the **Framework**
* **Thymeleaf** as the **View engine**
* **MySQL** as the **Database**

### C#

* **ASP.NET** as the **Framework**
* **Razor** as the **View Engine**
* **MSSQL Server** as the **Database**

## Model

The Task entity holds **3 properties**:

* id (technology-dependent (ObjectID for JavaScript, int for all other technologies))
* title **(string)**
* comments **(string)**, which can hold **any ASCII character**

## Skeleton

You will be given the application’s Skeleton, which holds about **90%** of the logic. You’ll be given some **files** (**controllers**, **models** etc.). The files will have **partially implemented logic**, so you’ll need to write some code in order for the application to **function properly**.

The application’s views will be given to you fully implemented. You only need to include them in your business logic.

**Each technology** will have its **own skeleton**, and the **different** **skeletons** may **differ** in **terms** of **what is given to you** and **what is to be implemented**.

Everything that has been given to you inside the skeleton is **correctly implemented** and if you write your code **correctly**, the application should work just fine. You are free to change anything in the Skeleton on your account.

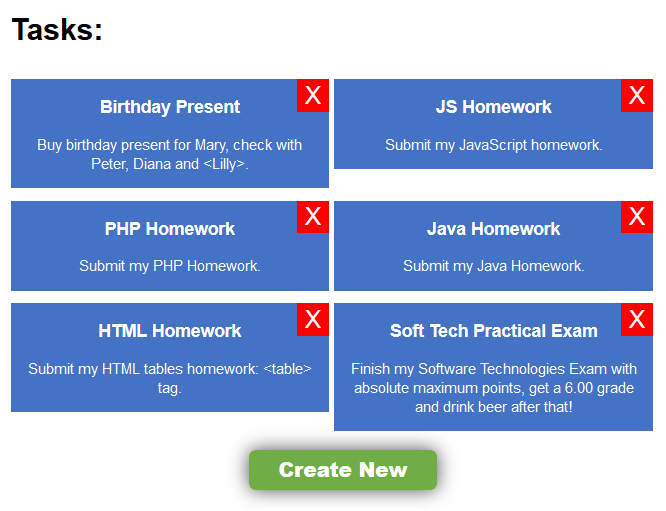
## User Interface

This is the user interface or how the application’s pages should look in their final form (fully implemented).

You have 3 pages:

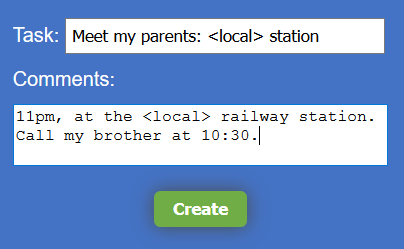
### Index page

#### Route: **“/”**



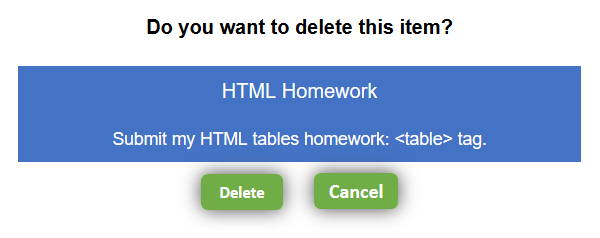
### Create Page

#### Route: **“/create”**



### Delete Page

#### Route: **“/delete/{id}”**



## Problem

As you can see the different pages are on different routes. Most of the routing logic will be given to you in the **Skeleton**, but you should make sure that the application **works properly**.

Implement the TODO List application on all 4 technologies.

## Setup

Before you start working, make sure you **download all the dependencies** (packages) required for each technology and **set up** the **databases**! Below are instructions on how to do this:

### PHP

1. Go into the **root directory** of the project (where the bin folder resides)
2. Make sure you’ve started your **MySQL server** (either from **XAMPP** or standalone)
3. Open a **command prompt/PowerShell** window in that directory (shift + right click 🡺 open command window here)
4. Enter the “php composer.phar install” command
5. Enter the “php bin/console doctrine:database:create --if-not-exists” command
6. Done!

### JavaScript

1. Go into the **root directory** of the project (where the bin folder resides)
2. Make sure you’ve started your **MongoDB server** (mongod.exe --dbpath path/to/db/directory)
3. Open a **command prompt/PowerShell** window in the **root directory** (shift + right click 🡺 open command window here)
4. Enter the “npm install” command
5. Done!

### C# and Java

C# and Java automatically resolve their dependencies when project is built.