**MVC Task 1: Conception & Structure**

**Online Game Store**

For this task, no UI required. For communication with server use JSON, (use Fiddler for testing). SOLID is must.

1. **General Setup:**
   1. **Domain model:**

Create model in separated project(s) which will contains services, business objects and repositories interfaces. There is a game entity.

*Game:*

String Key – unique;

String Name;

String Description;

*Comment:*

String Name;

String Body;

Comment has 1 – to – many relation with game (Game can has 0 or many comments; comment always related to 1 game).

Comment could be answer to another comment. Answer should start from: “[Author], “.Where [Author] is link to parent comment. [Author] is parent comments author name.

*Genre:*

*String Name -* unique*;*

Genres could be nested. Store has some basic genres: Strategy, RPG, Sports, Races, Action, Adventure, Puzzle&Skill, Misc. Strategy genre has sub-genres: RTS, TBS. Races has sub-genres: rally, arcade, formula, off-road. Action has sub-genres: FPS, TPS, Misc.

Game has many – to – many relation with game.

*PlatformType:*

*String Type -* unique*;*

Store has several types of platform: mobile, browser, desktop, console.

Game has many – to – many relation with platform type.

* 1. **Services:**

Model provides services for:

create new game/edit game /delete game /get game by key/get all games

add comment to game/ get all comments by game key

get games by genre/get games by platformTypes.

* 1. **DAL:**

Use Entity framework 5.0.

Use MSSQL Express (don’t use mssql Compact).

Use repository pattern.

Implement Unit of Work pattern for working with repositories.

1. **Admin Site:**
   1. **Logic**

Implement following logic using ASP.NET MVC

User can create game (POST URL: /games/new). User can edit game (POST URL: /games/update). User can get game details by key (GET URL: /game/{key}). User can get all games (GET URL: /games). User can delete game (POST URL: /games/remove). User can leave comment for game (POST URL: /game/{gamekey}/newcomment). User can leave comment for another comment (POST URL: /game/{gamekey}/newcomment) User can get all comments by game key (POST URL: /game/{gamekey}/comments). User can download game (jut return any binary file as response) (GET URL: /game/{gamekey}/download)

* 1. **General**

Use **empty**  ASP.NET MVC 4 template.

Use Output Cache filter to cache get post and get all post response for 1 minute.

Implement error and events logging.

Use global filter to log IP of requests in txt file.

Use filters for logging performance of services working.

**MVC Task1.1: DI & unit-testing**

Online Game Store

**1. DI:**

Implement DI and Ninject as IoC-conteiner.

**2. Unit-testing:**

Cover 40% services, controllers and routes with unit-tests. Use Mock.

**MVC Task1.2: Logs**

Online Game Store

**1. Logger:**

Implement logging in application by using NLog.

**2. DI:**

Work with logger through interface.

**3. Log format:**

You should log name of class at least for error and debug level.