LaSalle College
Montréal

Course Identification

Name of programs – Codes:	COMPUTER SCIENCE TECHNOLOGY– PROGRAMMING (420.BP) INFORMATION TECHNOLOGY PROGRAMMERANALYST (LEA.3Q)
Course title:	WEB SERVER APPLICATIONS DEVELOPMENT II
Course number:	420-DW4-AS
Group:	7268
Teachers' names:	Renan Cavalcanti
Duration:	8 hours
Semester:	Fall 2022

Standard of the Evaluated Competencies

Statement of the evaluated competency – Code

1. Develop transactional Web applications - 00SU
2. Develop data exchange services - 00SV

Evaluated elements of the competencies

00SU 1. Analyze the application development project. 3. Prepare the database. 5. Program the server-side application logic.

00SV 4. Program the application logic for the service

Important Dates

1. November 7 – Project Implementation
2. November 11 – Project Implementation
3. November 14 **(30% of project grade)** – Meeting online one by one to check how is the project and some tests.
4. November 18 **(70% of project grade)** – Fix bugs and Due date to deliver the project.

Project Web Server II

Your task for the project will be implement an entire backend application by creating an .net Core API with SQL.

Project requirements

1. Models
   - Task model must have the following attributes:

     TaskUid -> string (PK)

     CreatedByUid -> string

     CreatedByName -> string

     AssignedToUid -> string

     AssignedToName -> string

     Description -> string

     Done -> bool

     The constructor of this Model must initialize the attribute “done” as false and generate a random Uid .

     Check the windows documentation in how to generate UIDS in C# **https://learn.microsoft.com/en-us/dotnet/api/system.guid.newguid?view=net-7.0**

   - User Model must have the following attributes:

     Uid -> string (PK)

     Name -> string

     Email -> string

     Password -> string

     The constructor of this Model a random Uid .

   - Session Model must have the following attributes:

     Token -> string (PK)

Email -> string

This will be used to generate an UID for the user logged, when the user uses the endpoint login, a new session object will be generated and saved inside the table. The user will use this Token for all the requests.

Token will be an uid generated as you did before.

2. Endpoints:

You will need to implement the following endpoints in the Task controller:

- Create task: “/tasks/” -> method: POST
- Get tasks created by the user: “/tasks/createdby/” -> method: GET
- Get tasks assigned to the user: “/tasks/assignedto/” -> method: GET
- Update task: “/tasks/{taskUid}” -> method: PUT
- Delete task: “/tasks/{taskUid}” -> method: DELETE
- Create user: “/users/” -> method: POST
- Login: “/users/” -> method: POST

2.1)

Create a task must receive the following elements in its request:

Ex.

```
{
    "token": "1d52b561dse3d366fb561ds",
    "description": "New Task",
    "assignedToUid": "635b561d52c61766f0874c7c"
}
```

Using the token, you can access the table Session and check what is the email of the user that is doing that request!

Using assignedToUid value that is coming into the body request you can find in the user table who is the assignedTo user.

With all those information you can create a new task inside the database.

Please, validate all the possible error that you think that could happen:
Ex.
Token does not exist inside the table session,
Assigned to user does not exist
And so on…

2.2)

Get tasks created by the user, you will return all the tasks created by the user that is doing the request.

Your endpoint must return a list of tasks that were created by the user, you can access the user from the token information.

You will need to check in the tasks table which tasks were created by the user who is doing the request.

Please, validate all the possible error that you think that could happen.

2.3)

Get tasks assigned to the user, you will return all the tasks assigned to the user that is doing the request.

Your endpoint must return a **list of tasks** that were assigned to the user, you can access the user from the token information.

Please, validate all the possible error that you think that could happen.

2.4)

Update a task will have a small restriction, only the user that is assigned to that task will be able to update the task.

You can take the user information from the token and check if that user can update the task.

The task Uid will go into the url, the token inside the body of the request.

You should return the task updated.

Please, validate all the possible error that you think that could happen.

2.5)

Delete a task will have a small restriction, only the user that created that task will be able to delete the task.

You can take the user information from the token and check if that user can delete the task.

The task Uid will go into the url, the token inside the body of the request.

You should return the task deleted.

Please, validate all the possible error that you think that could happen.

2.6)

Create user endpoint will receive in the body of the request all the user information.

Your action must check if the email for that user already exists in the database.

An Uid for that user must be generated automatically. You should return the user created as response.

Please, validate all the possible error that you think that could happen.

2.7)

Login endpoint will receive the email and password from the user in the body of the function.

If the user is able to do the login, it means, email and password matches with any record inside the database we need to create a session record.

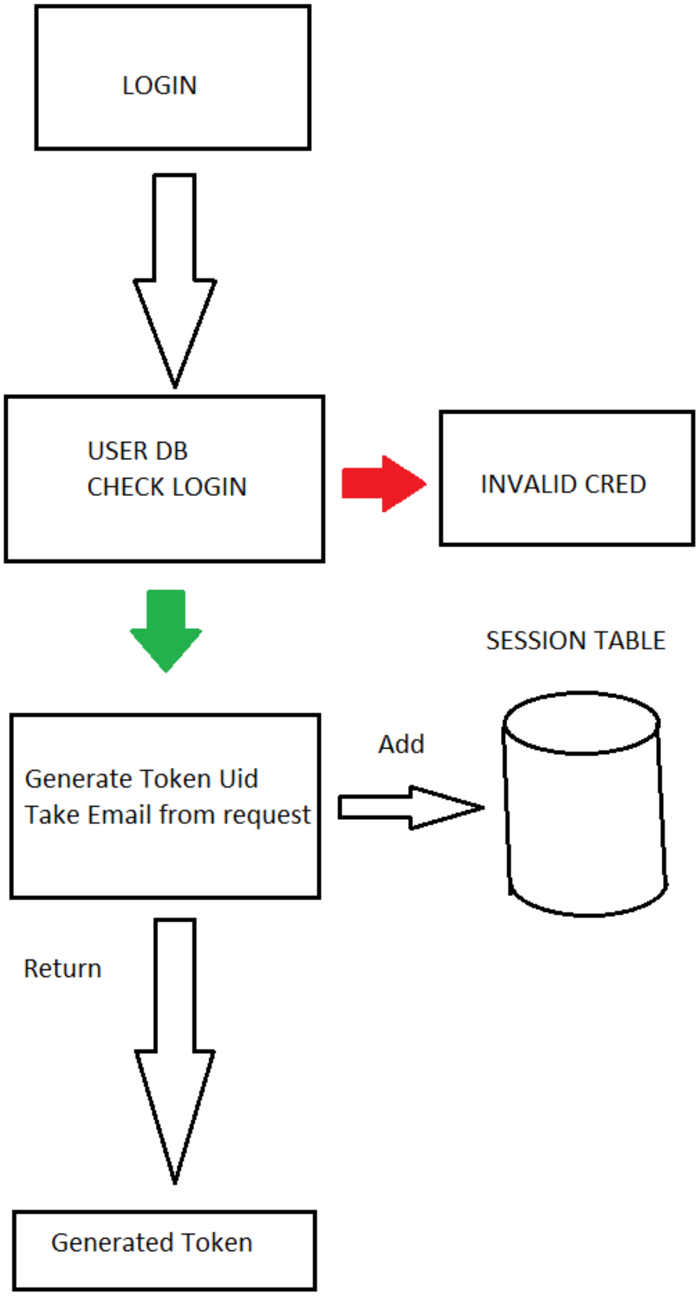
This new session will be saved inside the table Session with an Uid generated automatically and the email of the user that is doing the login.

You should return to the user the UID generated for the session.

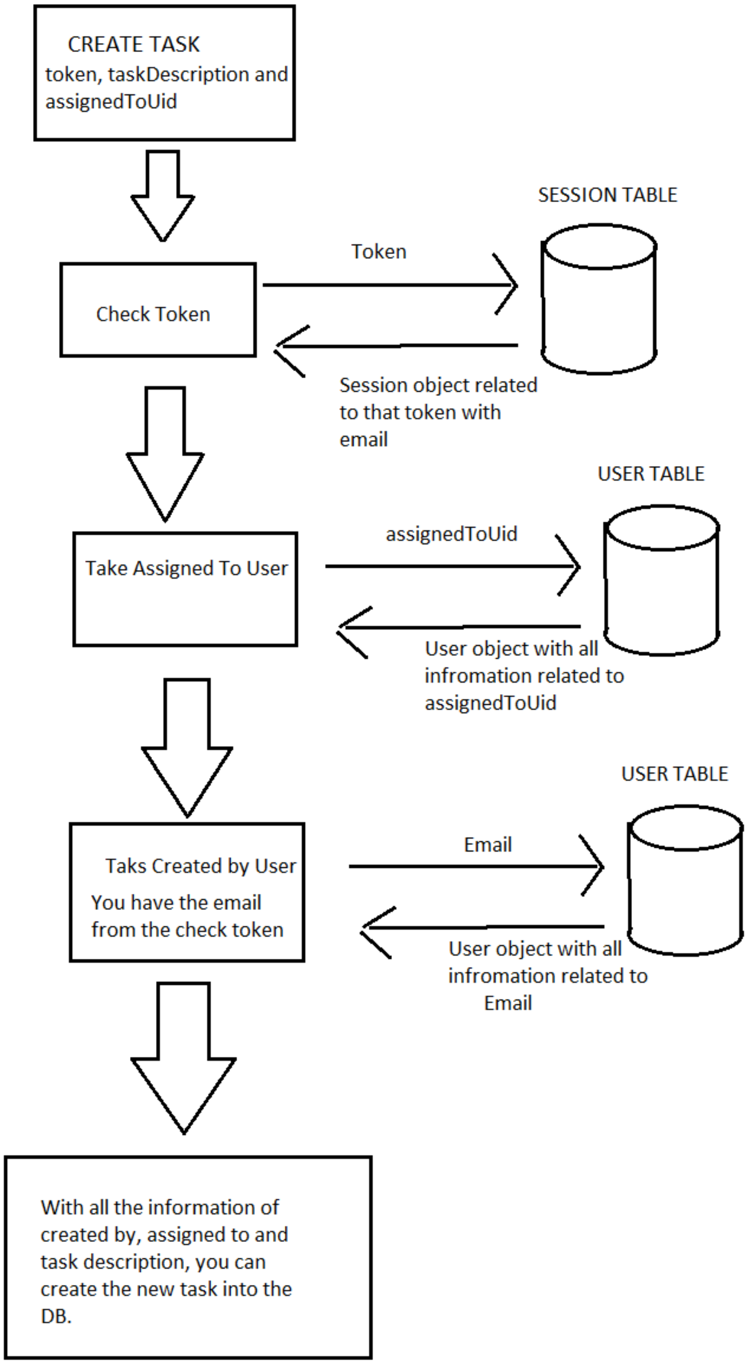
The user will use this UID as Token to make all the requests for the tasks.

Please, validate all the possible error that you think that could happen.

LOGIN FLOW:



USING TOKEN TO OPERATE TASKS:


CREATE TASK
token, taskDescription and assignedToUid
SESSION TABLE
Token
Check Token
Session object related to that token with email
USER TABLE
assignedToUid
Take Assigned To User
User object with all infromation related to assignedToUid
USER TABLE
Email
Taks Created by User
You have the email from the check token
User object with all infromation related to Email
With all the information of created by, assigned to and task description, you can create the new task into the DB.

GOOD LUCK

It always seems impossible until it's done.

Nelson Mandela

BrainyQuote