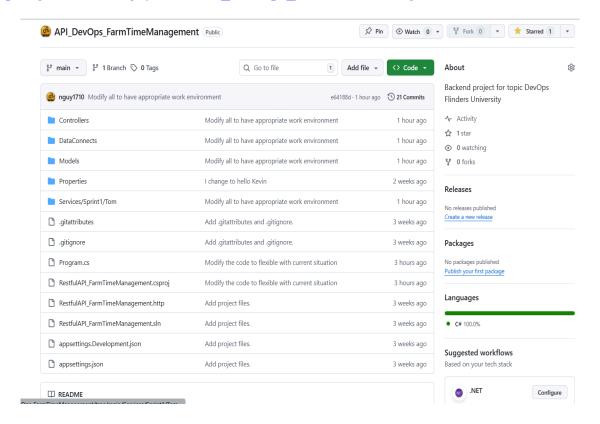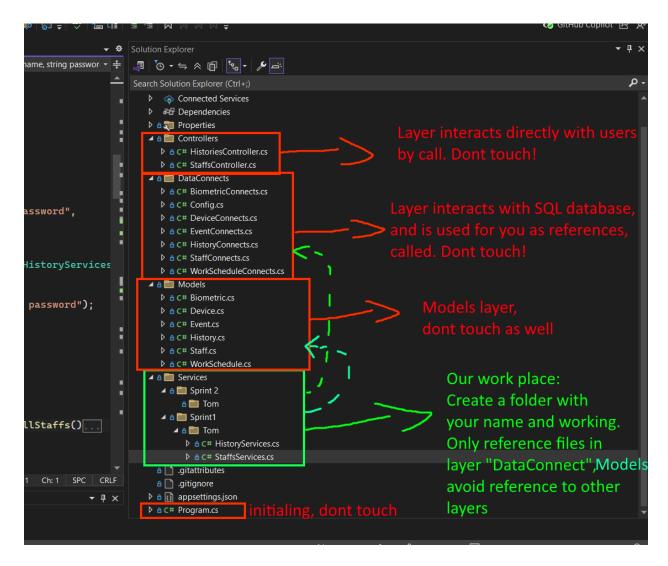# Instruction for developers working.

API BACK-END .NET API

Please clone/pull project into your PC:
https://github.com/nguy1710/API_DevOps_FarmTimeManagement

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Connected Services
- Dependencies
- Properties
- Controllers — **Layer interacts directly with users by call. Dont touch!**
  - HistoriesController.cs
  - StaffsController.cs
- DataConnects — **Layer interacts with SQL database, and is used for you as references, called. Dont touch!**
  - BiometricConnects.cs
  - Config.cs
  - DeviceConnects.cs
  - EventConnects.cs
  - HistoryConnects.cs
  - StaffConnects.cs
  - WorkScheduleConnects.cs
- Models — **Models layer, dont touch as well**
  - Biometric.cs
  - Device.cs
  - Event.cs
  - History.cs
  - Staff.cs
  - WorkSchedule.cs
- Services — **Our work place: Create a folder with your name and working. Only reference files in layer "DataConnect",Models avoid reference to other layers**
  - Sprint 2
    - Tom
  - Sprint1
    - Tom
      - HistoryServices.cs
      - StaffsServices.cs
- .gitattributes
- .gitignore
- appsettings.json
- Program.cs — **initialing, dont touch**

- Project's structure mainly separates into layers as picture above.
- I have also made layer which connect directly to database for use. It named **DataConnects**, you we don't need to try hard to interact with database anymore.
- Basically, we will do our work in Folder **Services**, you guy just creates your own folder with your name and do your task as create any file logical functions you are responsible for.
- To make clean code, clear structure, everything to be consistent, I strongly recommend we follow the rule that we only reference and call to source code in folders **Models** and **DataConnects .** I believe that with these files I made, we can do all logical to meet the requirements in our project.
- I will also create controllers in Controllers layer for sprint 2 as **RosterController**, **ScheduleController** for frontend to call as long as we make the code for them in Services layer.

- If you still want to make new controllers for calling, please make sure they are the same structure as **Histories controller** and **Staff controller**
- Once we done, just create your new branch in Github and push them to. After we review code together, if they are appropriate. I will merge to main branch.

FRONT-END WEBAPP

- **The API Endpoind I provide to you to implement into your project:**

  https://flindersdevops.azurewebsites.net

- **You can have a look at the Swagger as visualiation for API and call:**

  https://flindersdevops.azurewebsites.net/swagger/index.html

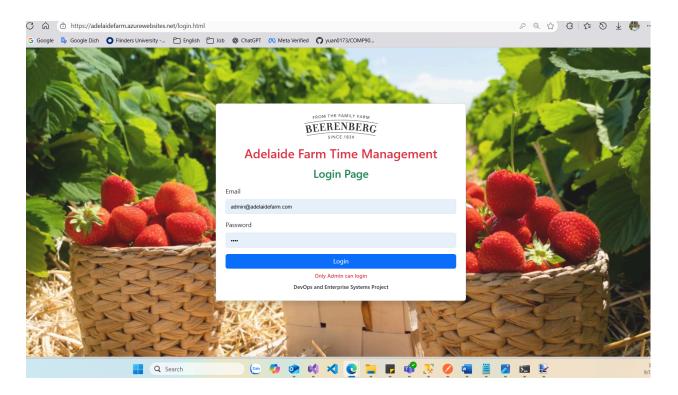# Demonstration last sprint:

- **The draft website in sprint 1 for you guy to have a look and reference:**

  https://adelaidefarm.azurewebsites.net/

  Username: admin@adelaidefarm.com

  Password: admin

- In last sprint, we have a form to login:

URL: https://flindersdevops.azurewebsites.net/api/staff/login
Method: POST
Body Json string:
{
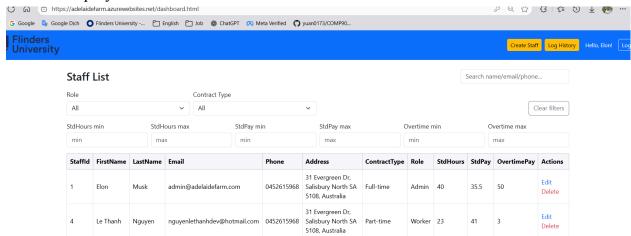"Email" : "admin@adelaidefarm.com",
"Password" : "admin"
}
Response:
{"StaffId":1,"FirstName":"Elon","LastName":"Musk","Email":"admin@adelaidef
arm.com","Phone":"0452615968","Password":"8c6976e5b5410415bde908bd4dee
15dfb167a9c873fc4bb8a81f6f2ab448a918","Address":"31 Evergreen Dr,
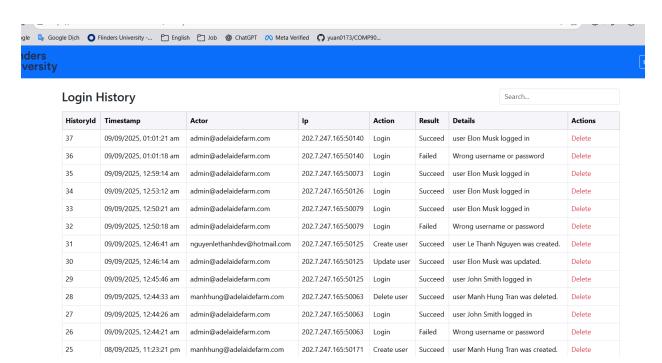Salisbury North SA 5108,
Australia","ContractType":"Full-time","Role":"Admin","StandardHoursPerWeek":
40.00,"StandardPayRate":35.50,"OvertimePayRate":50.00}

(Demonstration of login process)

- Form to display staff

- using GET: https://flindersdevops.azurewebsites.net/api/Staffs

- using GET: https://flindersdevops.azurewebsites.net/api/Histories
- You can see references in either Swagger above to know more about our API

For front-end, we can do or prompt AI anything whatever you want as long as it display the list of staff, list of history in api, create staff, edit, delete staff (I also provided full in API)

- Once done your project and run smoothly, just deploy into your github and I will fork it to my github and deploy into website internet for you automatically.