**IFT458/598 – Project 1 Deliverable 5**

Yuan Li & Edward Halper

**Introduction**

**Description of your work**

|  |  |  |
| --- | --- | --- |
| Name | Item | description |
| Yuan Li | Models:  Address  Officer  Event  JobTitle  Office  Volunteer  Volunteerevent | Created the models. Defined schemas for each entity including type, length, and foreign keys.  Each model also has a default \_\_str\_\_ function to return the meaningful information of the model object. |
| Yuan Li | CRUD operations on officer, event entities | Use project urls.py to map url to templates.  Below are the CRUD urls.  Read <http://127.0.0.1:8000/api/officers/>  Create <http://127.0.0.1:8000/createnewofficer/>  Update <http://127.0.0.1:8000/officer/update/3>  Delete <http://127.0.0.1:8000/officer/delete/3> |
| Yuan Li | Login | Authenticate user by using Django built in module auth. It compares email and password that user entered with the stored values in the database. After user login successfully, the home page shows welcome message with the logged in user name. |
| Yuan Li | Rest framework | Use Django REST framework to expose APIs for other programs to use. I created a api folder to organize the api related files.  The serializers includes Officer, Event and User.  You can use the api to retrieve these entities information. Below are the API urls.  <http://127.0.0.1:8000/api/officerlist/>  <http://127.0.0.1:8000/api/officerlist/2/>  <http://127.0.0.1:8000/api/eventlist/>  <http://127.0.0.1:8000/api/eventlist/1/>  <http://127.0.0.1:8000/api/userlist/> |
| Yuan Li/ Edward Halper | Register page | Register page will make sure the user registered information populate into user and registerinfo tables. Registerinfo table has foreign key userid which is linked to userid in user table. It also has a foreign key addressid which is the primary key in address table. After login, the page will redirect to home page with the registered user name. You can use the user name and password to login too. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Screenshots of code narrative from above**

**Models:**

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

**Project Urls:**

Text

Description automatically generated

**Views:**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

**APIs:serializers**

Text

Description automatically generated

**API:urls**

Text

Description automatically generated

**API:views**

Text

Description automatically generated

**User manual**

**Step1: Click login button to login**

Graphical user interface, text

Description automatically generated

**If you don’t have a user name and password, please click Register button to register**

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**You can use the default value to register or change them.**

**Step2: Direct to home page with the customized welcome message to the user**

Graphical user interface, website

Description automatically generated

**Step3: Click Officers menu. You will see a list of officers. If you don’t have any officers, you can click Add new Officer button to create one**

Graphical user interface, text, application

Description automatically generated

**Step4: Create new officer by clicking Create a Officer button**

Graphical user interface, application, Teams

Description automatically generated

**Step5: Making sure you enter the valid string for each input.**

**You can use any string for officer name, like “officer1”**

**Age has to be 2 digitals. Like 32**

**Resp. Description can be any string, like “hello, I am responsible for parking”**

**Street can be any string, like “432 W Lindr Ave”**

**City could be any string, like “Mesa”**

**State is a dropdown list, like “AZ”**

**Zip has to be 5 digitals, like 56754**

Graphical user interface, text, application, email

Description automatically generated

**Step6: Delete an officer by clicking delete button next to each officer**

Graphical user interface, text, application

Description automatically generated

**Step7: Update officer by click Edit button next to each officer**

Graphical user interface, text, application

Description automatically generated

**Step8: Edit detail page, the values will be prefilled depending on which officer you clicked from previous page**

Graphical user interface, text, application

Description automatically generated

**Step 9: click Update a Officer button, the new updated information will be stored into database and page will be redirected to the officer list page**

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

**Step10: Click the Event menu to see the list of event.**

A picture containing graphical user interface

Description automatically generated

**Step11: Fill out all the information to create a new event**

A computer screen capture

Description automatically generated with medium confidence

**Step12: After create a event successfully, page will redirect to event list page**

Graphical user interface, application, Teams

Description automatically generated

**Step 13: You can see the json objects returned by API too**

[**http://127.0.0.1:8000/api/officerlist/**](http://127.0.0.1:8000/api/officerlist/)

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Screenshots of code narrative from above**

**Yuan’s Conclusion:**

I have learned how to utilize Python and Django framework to create a fully functional application, including basic reading, creating, updating, and deleting entities. The built-in CRUD APIs are very easy to use, and they are mostly self-explanatory.

I really like Django built-in admin page, because you can directly interact with Django database and see the data and manipulate the data. Also, the controls are created dynamically based on your data type. For example, if you have a date time type, the controls will be rendered as calendar.

I am surprised how easy and fast you can build the database based on your model. You just need to define your model and call the migration command, then all the data tables will be created for you. However, this is code-first approach. There are many existing applications with database already. From what I researched, the database-first approach is not very convenient to use with Django framework. Although there are many tools available (e.g. python manage.py inspectdb) to help you generate your model based on your existing database, the generated models are not 100% accurate, and you still have to modify the model. I just wish there were more good tools to support this. Another thing I want to improve is to find a better debugging tool. Right now, I just rely on the runtime error message to shoot the problem, but when the project gets bigger we will need a more sophisticated debugging tool to shoot the problem. Overall, I have learned a lot of new things from this course, including MySql, Python, and Django.