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

Yuan Li & Edward Halper

**Introduction**

For this deliverable there will be evidence of created models for the final project, configured databases, an app called backend, different necessary models and or tables, CRUD tests, implementation of a registration form, and the ability to display user information once a user has registered and logged into the site. This was accomplished using Django, python, jquery, and other software. Using prebuilt Django functions, we will be able to authenticate users and create login abilities into the site. Along with this it will also be seen the ability to create and edit events and officers as described in previous deliverables building up to this project deliverable.

**Description of your work**

|  |  |  |
| --- | --- | --- |
| Name | Item | description |
| Yuan Li | Models:  Address  Officer  Event  JobTitle  Office  Volunteer  Volunteerevent  RegisterInfo | 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. |
| Edward Halper | User database connection | Used built in Django to connect login and register pages in order to register User as you would in an admin setting using built in form page from Django. User can register all pertinent information as seen below in screenshot and use it in login page which also displays in the top right the login name and logout information and pages |
| Edward Halper | Logout function | As the user logs in I built a button that will display in the upper right corner the username and will also display the logout button to allow the user to logout when ready to and be redirected to the home page not logged into the site. |

**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**

**Please make migration first**

**Step1: Click login button to login or register to register a new user**

Graphical user interface, text

Description automatically generated

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

Graphical user interface, application

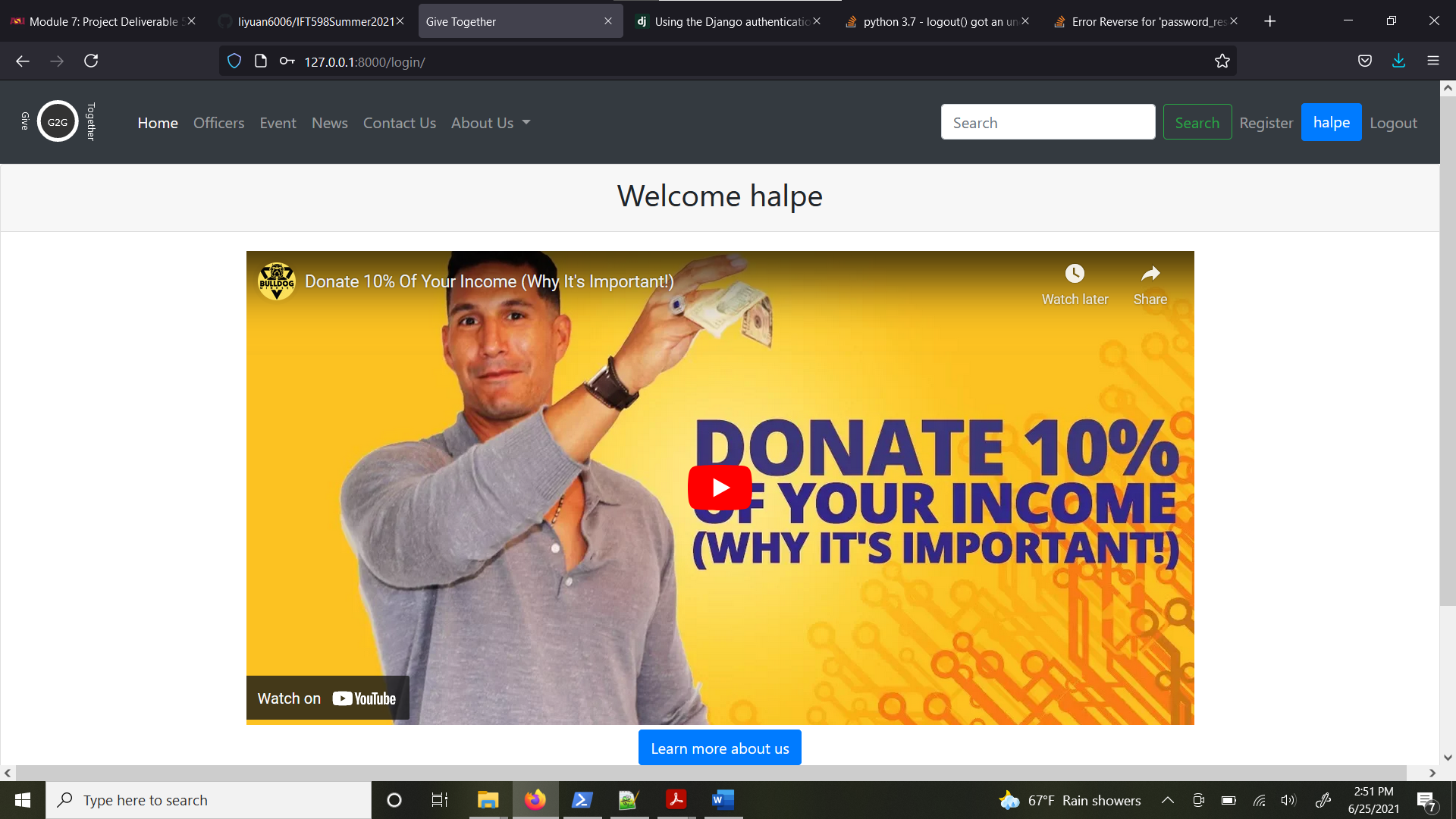
Description automatically generated

Graphical user interface, text, application

Description automatically generated

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

**Step2: After you login or register, page will be redirected to home page with the customized welcome message to the user and button in the upper right that will stay throughout all the pages with the logout button now appearing instead of the login that was there before user login.**

****

**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 digits, 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

**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 must 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.

**Edward’s Conclusion:**

During this entire project I have learned a great deal more about MySQL, Python, JQuery, and Django. With this new appreciation I have been able to create a frontend and backend and connect the 2 together in an environment where I was able to troubleshoot and get the product that the user sees before them. With this shell I believe someone would be able to take it and go further in their endeavor to create a site with a database system for volunteers and events. My favorite part of this last deliverable was the ability to really dive into the functionality with Django. I believe that I was caught digging too deep in order to do something “fancy” with the program. This caused me to lose some sleep and must reach out to my partner for some assistance and together we were able to build it all up and turn it into what we have turned in for this deliverable. I was presently surprised at how quickly everything came together with the models but was also caught in the weeds trying to do to much for a beginner with the program, which was a warning in the deliverable itself. If I were to go back and redo anything I believe that instead of building a database with MySQL I would have focused on a more JSON friendly type of arguments in order to have it setup correctly for the result. All in all, I believe we accomplished all that was asked of us and I am happy of the experience and to work with a great partner.