Team: Nuibility

Project: ClassSchedule

Member: Miao Wenting， Luo Siwei, Li Runfa

Jobs we done:

Miao Wenting: Backend, Part of the front end, VCS, CI, CD, Test, Documentation.

Luo Siwei: Part of the front end.

Li Runfa: README.md

1 VCS

Tools: Git, GitHub
link: https://github.com/WentingMiao/DesignYourClassSchedule

2 CI & Test

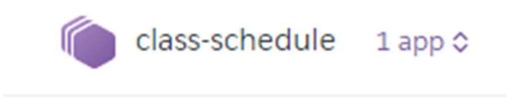
Tools: Heroku CI, Heroku CLI

link: https://id.heroku.com/login

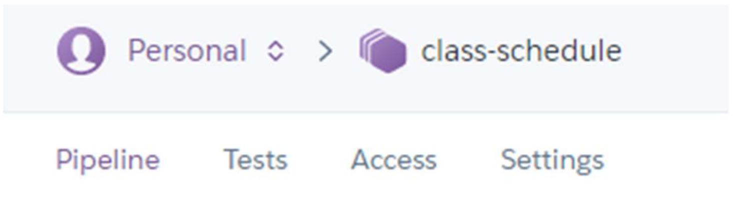
Email: mwt609042270@gmail.com

Password: tool2020@

2.1 After login there is a list of pipelines

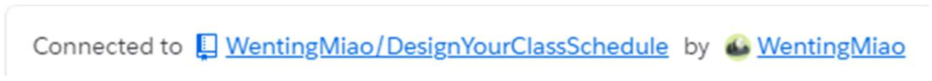


2.2 Click on the “class-schedule” will buring us to the pipeline “class-schedule”

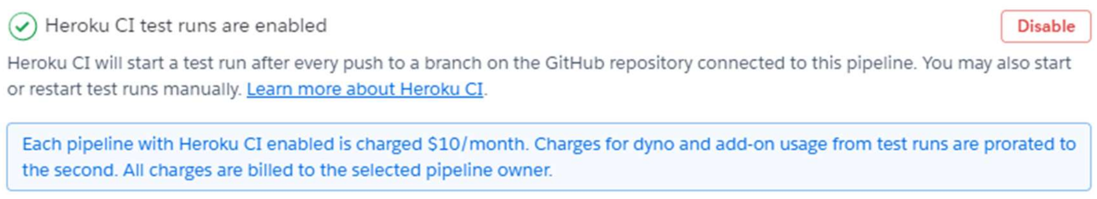


2.3 After click on the Settings.

We can see this pipeline is conncted to the github mentioned in the previous section.


Connected to WentingMiao/DesignYourClassSchedule by WentingMiao

And the Heroku CI test runs are enbled


Heroku CI test runs are enabled

Heroku CI will start a test run after every push to a branch on the GitHub repository connected to this pipeline. You may also start or restart test runs manually. Learn more about Heroku CI.

Each pipeline with Heroku CI enabled is charged $10/month. Charges for dyno and add-on usage from test runs are prorated to the second. All charges are billed to the selected pipeline owner.

Heroku CI will run all the tests after every push to a branch on the Github repo.

For more details about test configuration and the tests, please jump to the section 4.1.

3 CD

Tools: Heroku CD

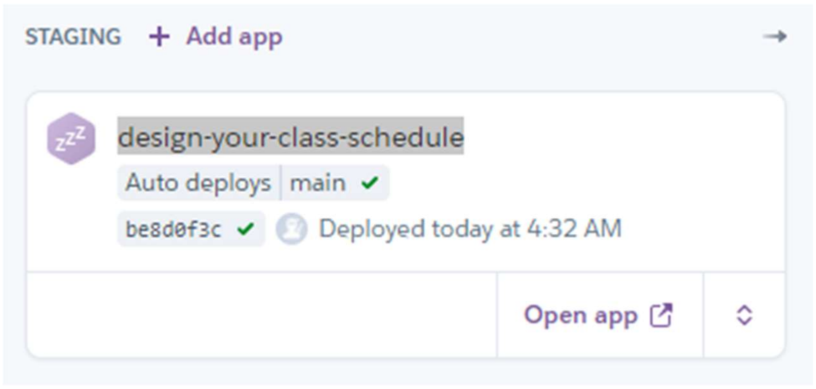
link: https://id.heroku.com/login

Email: mwt609042270@gmail.com

Password: tool2020@

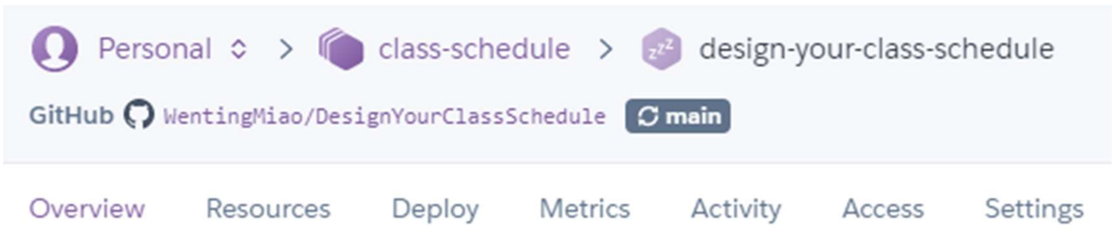
It enables a chosen branch to be automatically deployed to this app.

3.1 After go back to the pipeline "class-schedule", we can see there is a Staging app called "design-your-class-schedule" is automatically deployed.



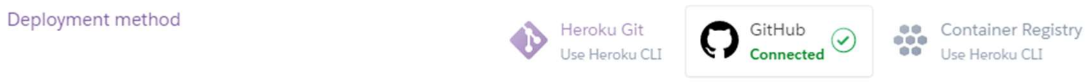
After click on the "Open app", we could find our latest version of the automatically deployed project.

3.2 After we click on the app "design-your-class-schedule", It shows the dashborad of the app.



3.3 After we click on the botton "Deploy"

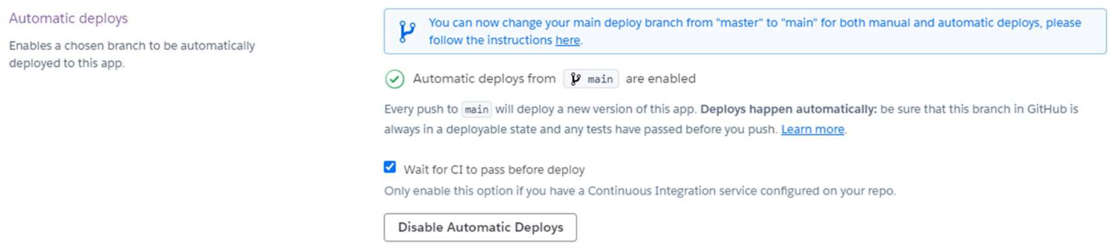
we will find the deployment method of this application is Github



As suggested by the Heroku CD, we need to create a new branch named "main" and set it to default branch first in github.

As we can see, the automatic deploys from branc "main" is enabled. Every push to branch "main" will deploy a new version of this app. Deploys happen automatically.

And the before each deployment, it will wait for CI to pass all the tests.



Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Automatic deploys from `main` are enabled

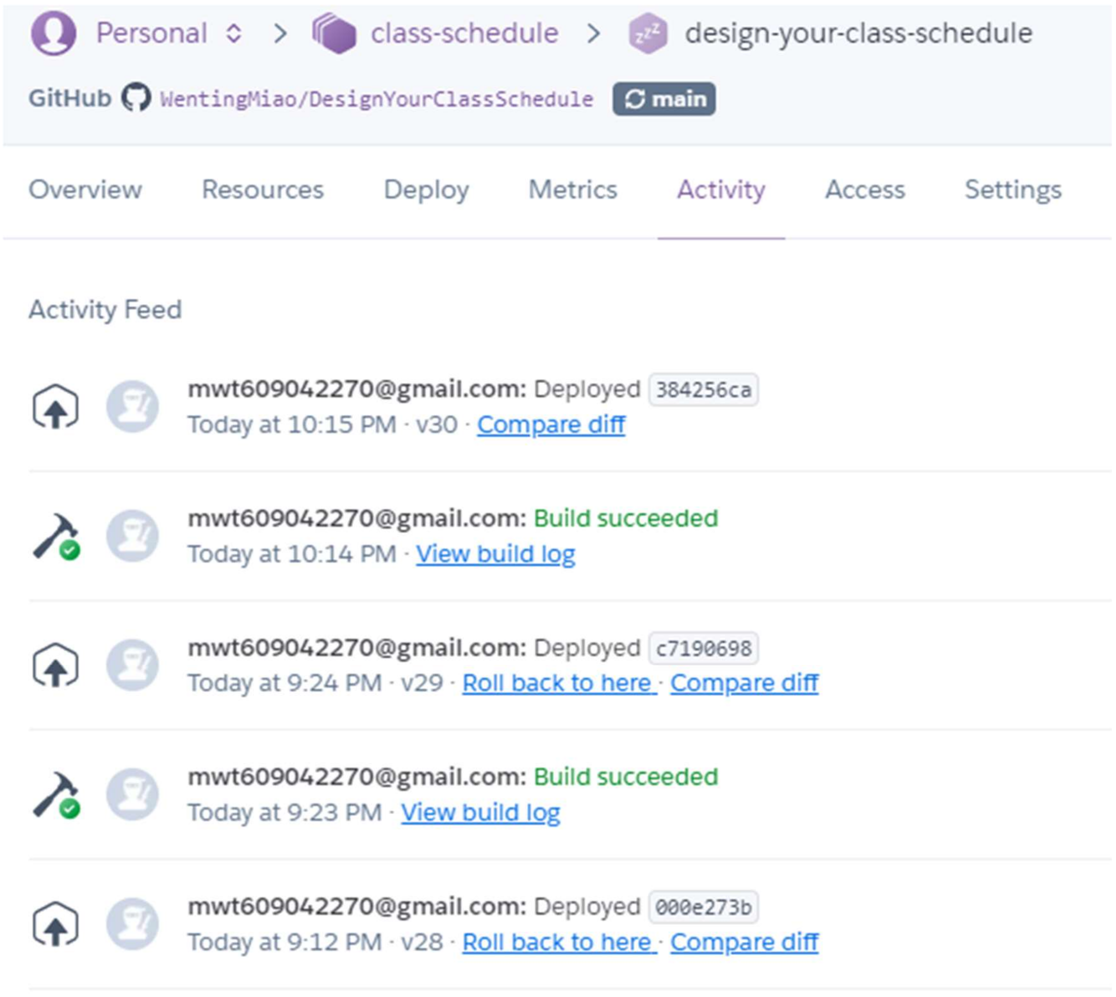
Every push to `main` will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. Learn more.

☑ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

3.4 After we click on the "Activity"

We could check all the deploy activities


Personal > class-schedule > design-your-class-schedule

GitHub WentingMiao/DesignYourClassSchedule main

Overview Resources Deploy Metrics Activity Access Settings

Activity Feed

mwt609042270@gmail.com: Deployed 384256ca
Today at 10:15 PM · v30 · Compare diff

mwt609042270@gmail.com: Build succeeded
Today at 10:14 PM · View build log

mwt609042270@gmail.com: Deployed c7190698
Today at 9:24 PM · v29 · Roll back to here · Compare diff

mwt609042270@gmail.com: Build succeeded
Today at 9:23 PM · View build log

mwt609042270@gmail.com: Deployed 000e273b
Today at 9:12 PM · v28 · Roll back to here · Compare diff

3.5 The automatical deploy flow

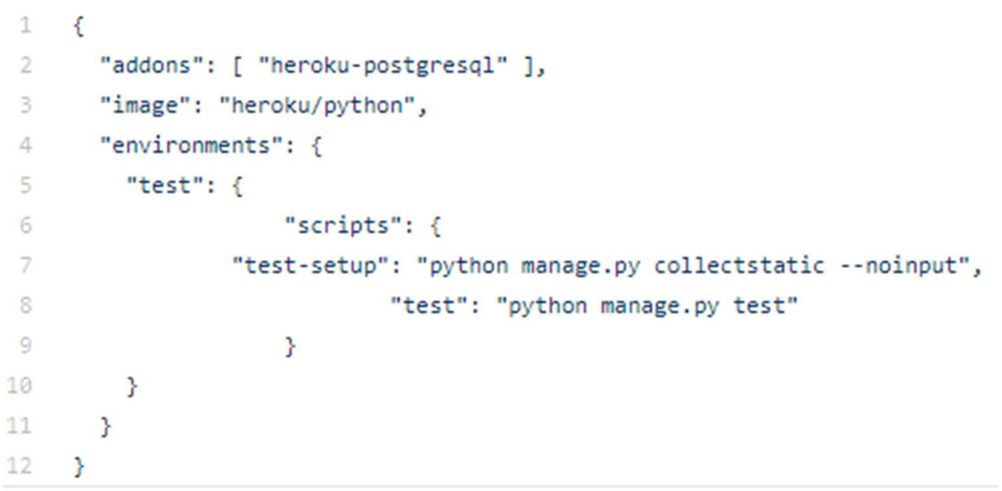
First, the dependencies in the “requirement.txt” are automatically installed.

Second, We use Django, so the collectstatic command runs automatically.

Third, run the tasks specified in the “Peocfile”. In our case, it run automatically our HTTP server. And let it run our project.

4 Configurations to enable the CI/CD

4.1 Add a file named “app.json” at the root of the github repository.

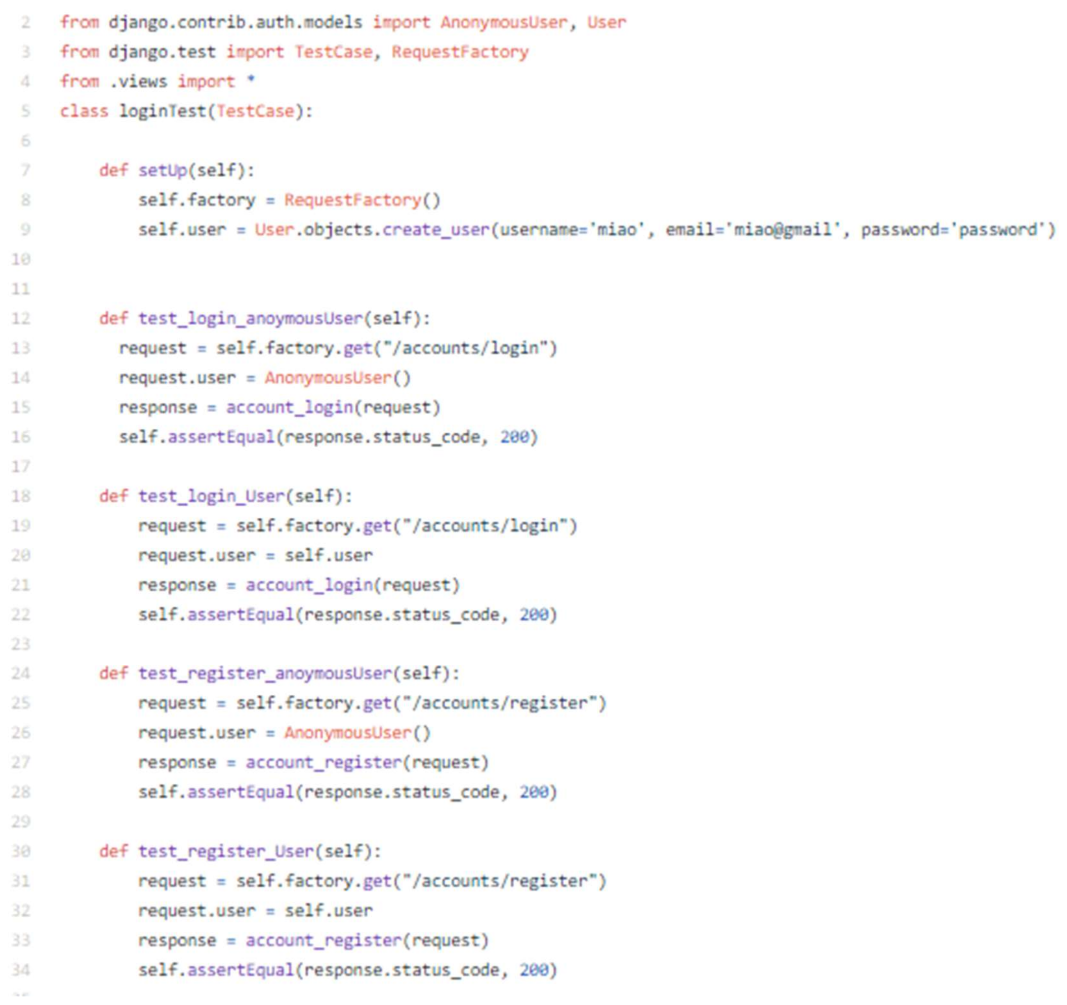


```
{
  "addons": [ "heroku-postgresql" ],
  "image": "heroku/python",
  "environments": {
    "test": {
                "scripts": {
            "test-setup": "python manage.py collectstatic --noinput",
                        "test": "python manage.py test"
                }
    }
  }
}
```

As we can see in the “environments”->”test” -> “scripts”. Here is all the test scripts it will automatically run after each push. We test the setup by run the command “python manage.py collectstatic --noinput”.

We run all the tests in the project by run the command “python manage.py test”. It will run all the tests of the “tests.py”, which is in all the modules of the django project.

“test.py” in accounts module



```python
from django.contrib.auth.models import AnonymousUser, User
from django.test import TestCase, RequestFactory
from .views import *
class loginTest(TestCase):

    def setUp(self):
        self.factory = RequestFactory()
        self.user = User.objects.create_user(username='miao', email='miao@gmail', password='password')


    def test_login_anoymousUser(self):
      request = self.factory.get("/accounts/login")
      request.user = AnonymousUser()
      response = account_login(request)
      self.assertEqual(response.status_code, 200)

    def test_login_User(self):
        request = self.factory.get("/accounts/login")
        request.user = self.user
        response = account_login(request)
        self.assertEqual(response.status_code, 200)

    def test_register_anoymousUser(self):
        request = self.factory.get("/accounts/register")
        request.user = AnonymousUser()
        response = account_register(request)
        self.assertEqual(response.status_code, 200)

    def test_register_User(self):
        request = self.factory.get("/accounts/register")
        request.user = self.user
        response = account_register(request)
        self.assertEqual(response.status_code, 200)
```

“test.py” in dashboard module

```python
from django.contrib.auth.models import AnonymousUser, User
from django.test import TestCase, RequestFactory
from .views import *


class dashboardTest(TestCase):

    def setUp(self):
        self.factory = RequestFactory()
        self.user = User.objects.create_user(
            username='miao', email='miao@gmail', password='password')


    def test_index_anoymousUser(self):
         request = self.factory.get("/")
         request.user = AnonymousUser()
         response = index(request)
         self.assertEqual(response.status_code, 200)

    def test_index_User(self):
        request = self.factory.get("/")
        request.user = self.user
        response = index(request)
        self.assertEqual(response.status_code, 200)
```

4.2 Add a file named “runtime.txt” at the root of the repository. It specify the version of the python runtime.

```
python-3.7.8
```

4.3 Add a file named “requiremetns.txt” at the root of the repository. It specify the app dependencies. When an app is deployed, Heroku reads this file and installs the appropriate Python dependencies using the pip install -r command. If this file are in the repo root, Heroku automatically identifies our app as a Python app.

```
django
gunicorn
django-heroku
whitenoise
```

“django” is our app framework. “gunicorn” is our HTTP server. “django-heroku” is the Heroku buildpack for python apps. “whitenoise” serving static files for django.

4.4 Add a file named “Procfile” in the repo root. It declare what command should be executed to start our app.

```
web: gunicorn ClassSchedule.wsgi --log-file -
```

4.5 modify DesignYourClassSchedule/ClassSchedule/settings.py

4.5.1

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

4.5.2

```
MIDDLEWARE = [
    'whitenoise.middleware.WhiteNoiseMiddleware',
```

4.5.3

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3')
    }
}
```

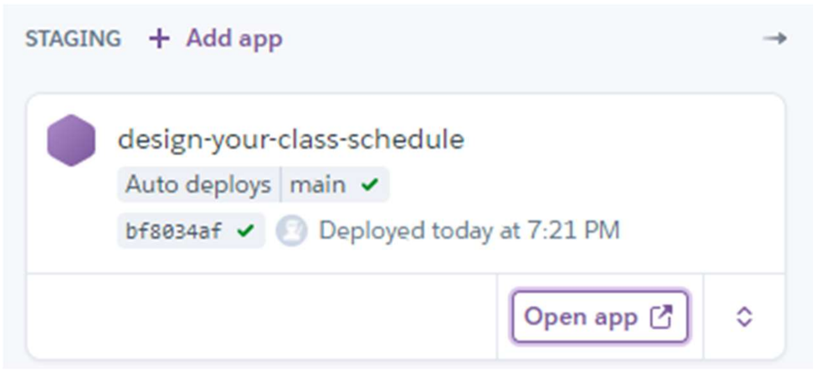
4.5.4

```
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
STATIC_URL = "/static/"
STATICFILES_DIRS = ( os.path.join(BASE_DIR, 'static'),
)
```

5 User Documentation

5.1 Dashboard

Click the “Open app” to open the our application.



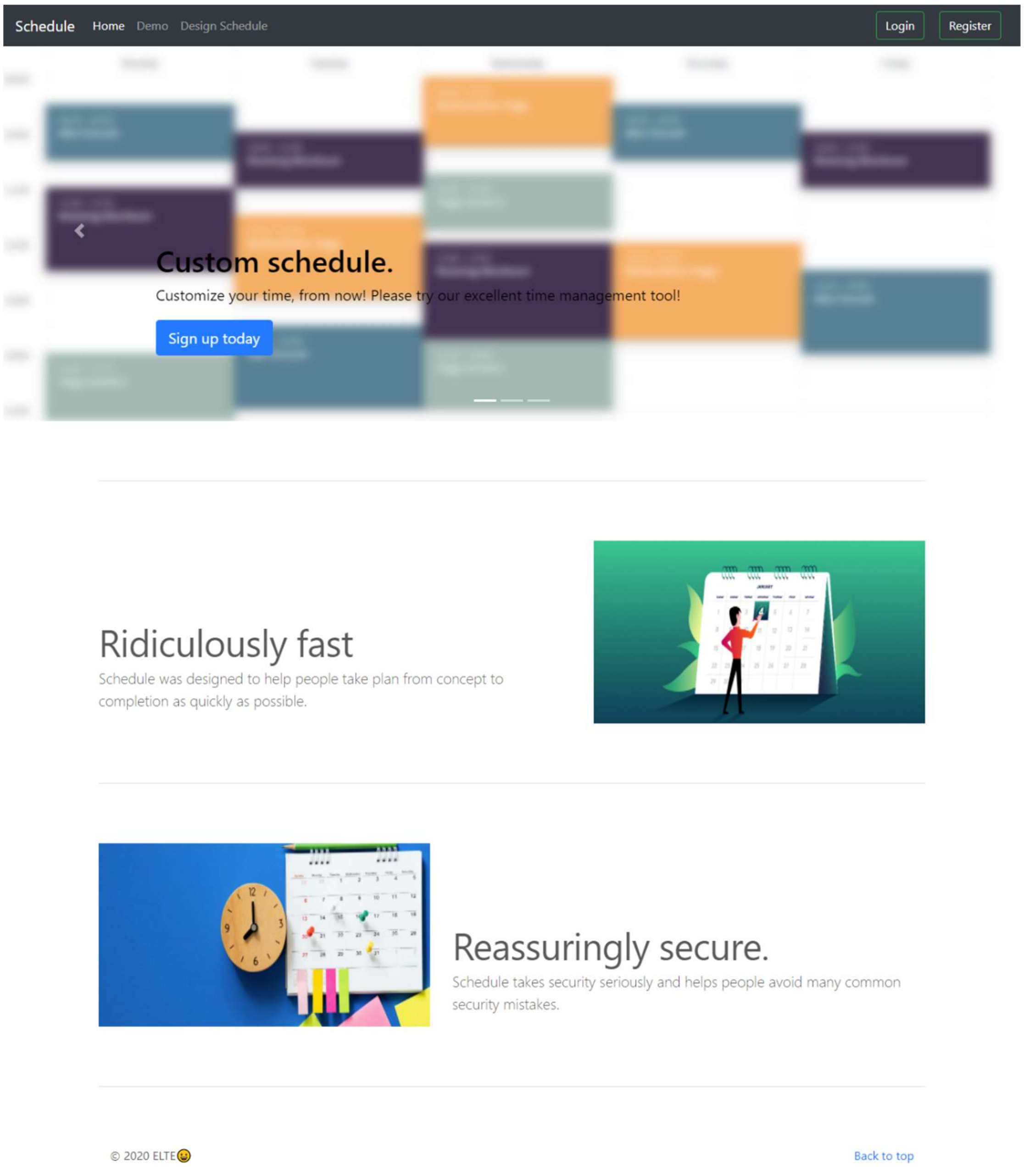
The dashborad

The top nevigation bar have: “Home”, “Demo”, “Design Schedule”

At the right uppoer corner, there are “Login“, “Register”.

At the middle there is a slideshow that contains the atractive words with the register url, the source code url and the donate url. They all use the flur demo as the backgound.

Below the slideshow is the features of our project.



Custom schedule.

Customize your time, from now! Please try our excellent time management tool!

Ridiculously fast

Schedule was designed to help people take plan from concept to completion as quickly as possible.

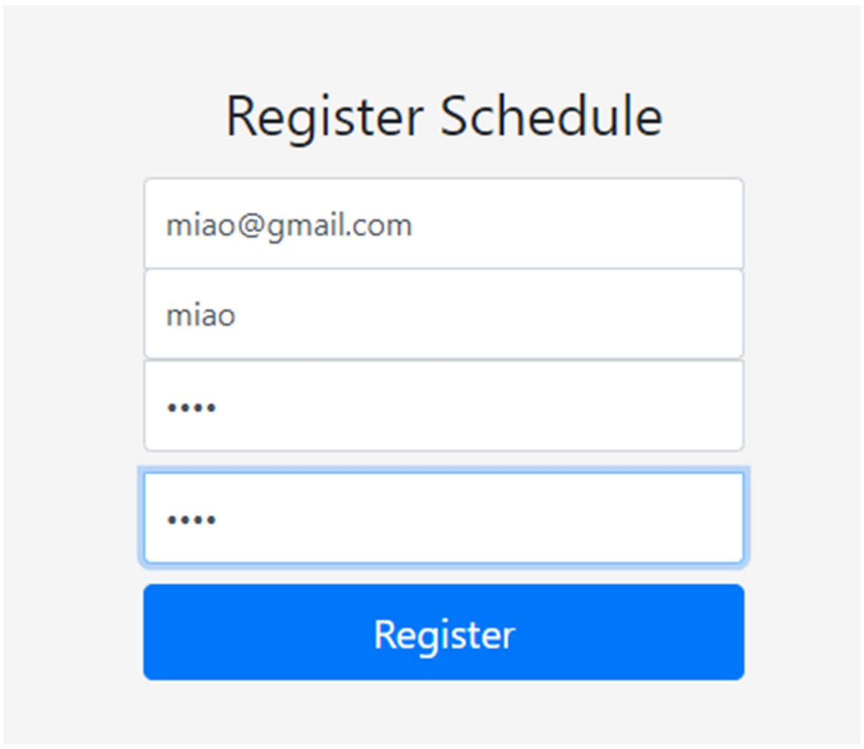
Reassuringly secure.

Schedule takes security seriously and helps people avoid many common security mistakes.

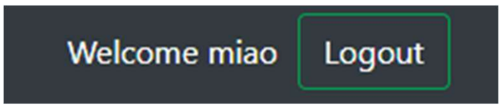
© 2020 ELTE😀

5.2 Register, Login, Logout

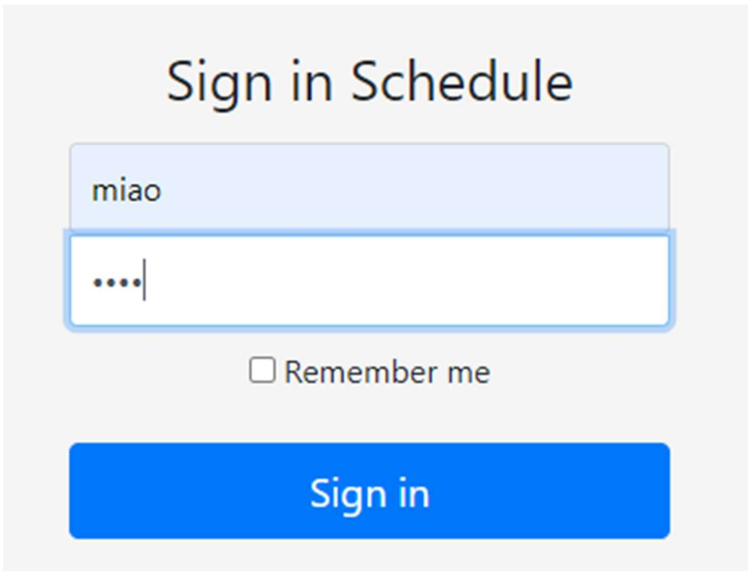
After we click on the register button, there will be a form for register.



After register we will automatically login. We will found the welcome words and our username.



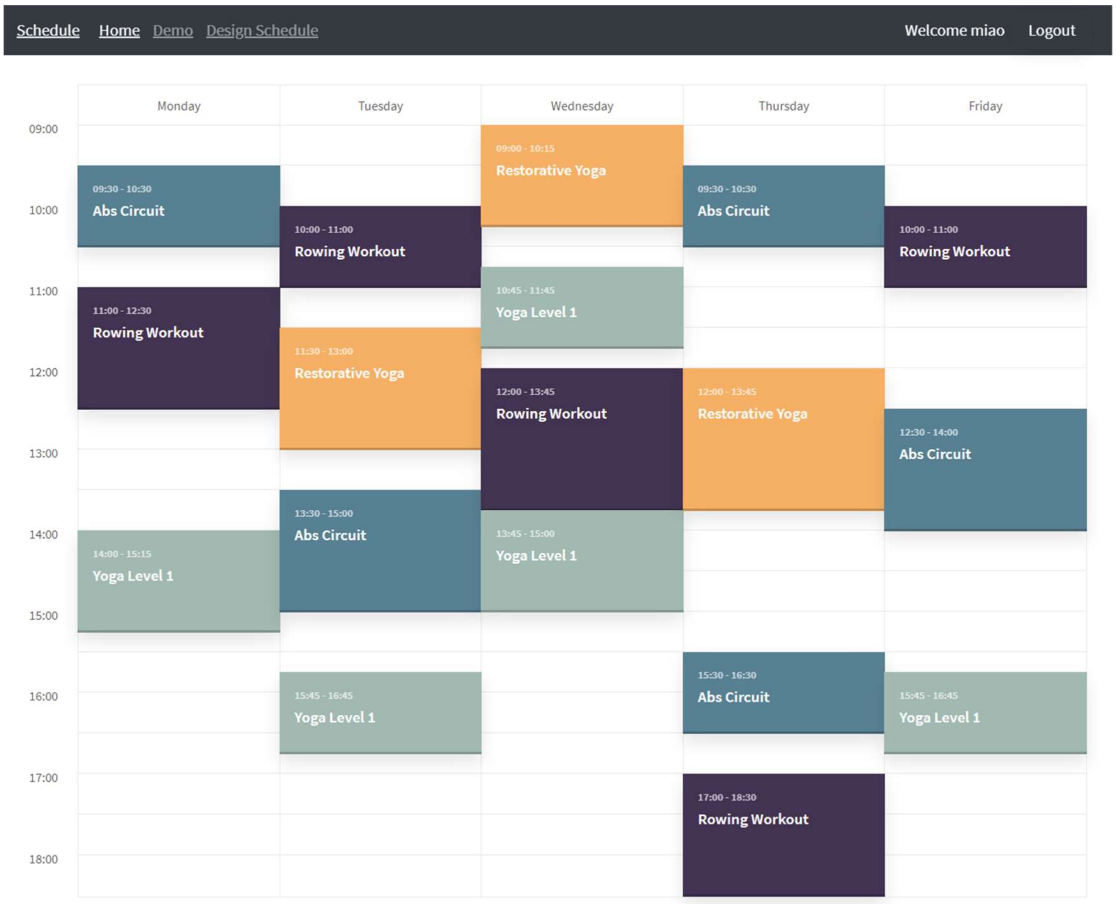
After we click on the “Logout“, We will be redirected to the login interface.



Use the account we just registerd, we can login again to the system.

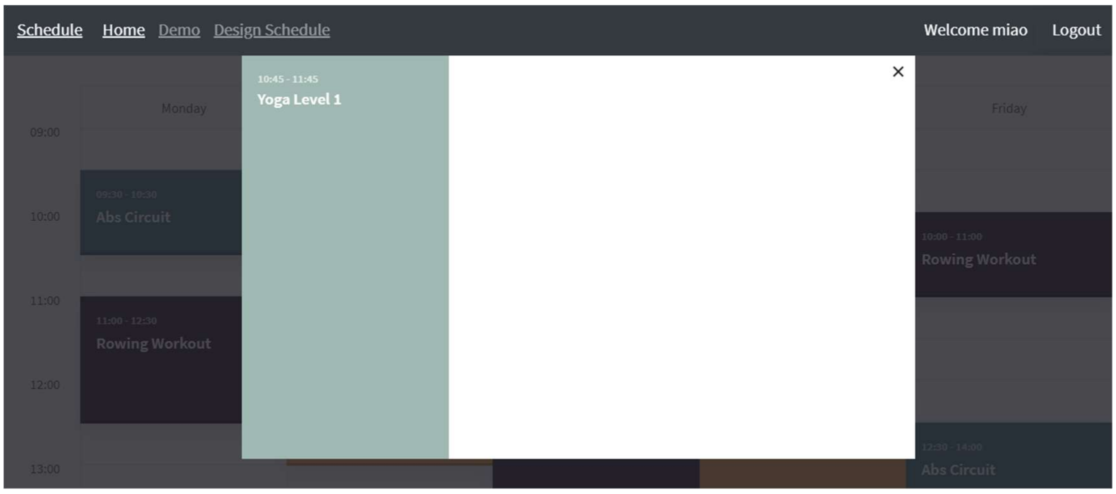
5.3 Demo

After we click on the “Demo” in the nav bar. We will found the sample of the result of the time table. It shows how the schedule table look like finally.



	Monday	Tuesday	Wednesday	Thursday	Friday
09:00			09:00 - 10:15 Restorative Yoga		
	09:30 - 10:30 Abs Circuit			09:30 - 10:30 Abs Circuit	
10:00		10:00 - 11:00 Rowing Workout			10:00 - 11:00 Rowing Workout
			10:45 - 11:45 Yoga Level 1		
11:00	11:00 - 12:30 Rowing Workout				
		11:30 - 13:00 Restorative Yoga			
12:00			12:00 - 13:45 Rowing Workout	12:00 - 13:45 Restorative Yoga	
					12:30 - 14:00 Abs Circuit
13:00					
		13:30 - 15:00 Abs Circuit	13:45 - 15:00 Yoga Level 1		
14:00	14:00 - 15:15 Yoga Level 1				
15:00					
				15:30 - 16:30 Abs Circuit	
		15:45 - 16:45 Yoga Level 1			15:45 - 16:45 Yoga Level 1
16:00					
17:00				17:00 - 18:30 Rowing Workout	
18:00					

After we click on one of the course, the details will be shown in a small window.

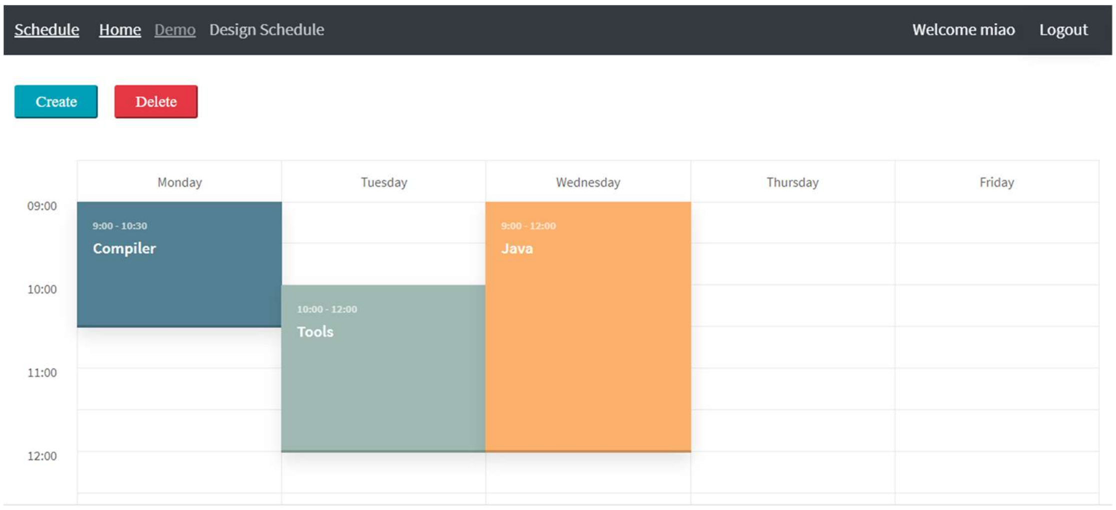


5.4 Operations on our own Course Schedules

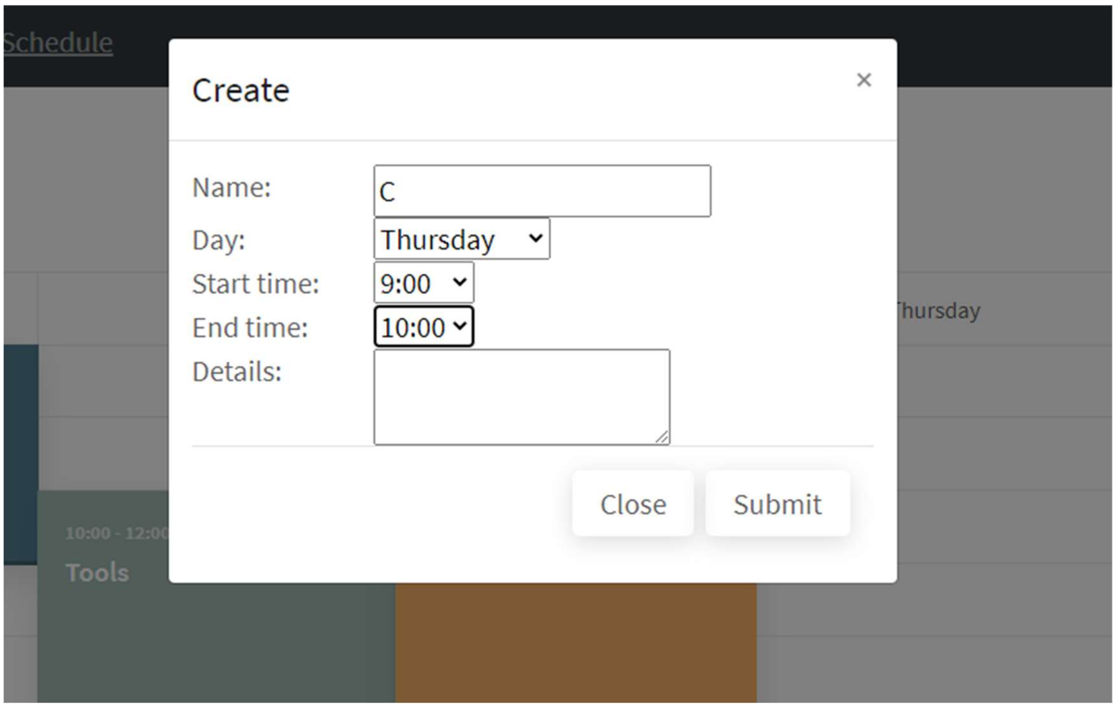
If we have not logged in, after we click on the “Design Schedule” button in the nav bar, we will be redirected to the Sign in page.(Need identification)

We can easily register one if we don’t have an account.

If we have logged in, after we click on the “Design Schedule” button in the nav bar, we will find the place to show, create, and delete our own course schedules.

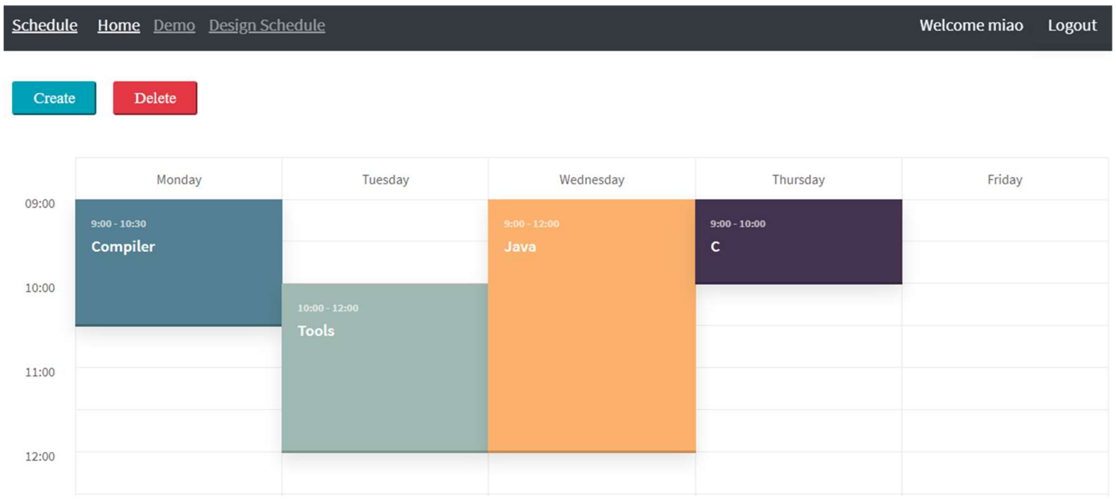


After we click on the “Create”, fill in the form and click “Submit”.

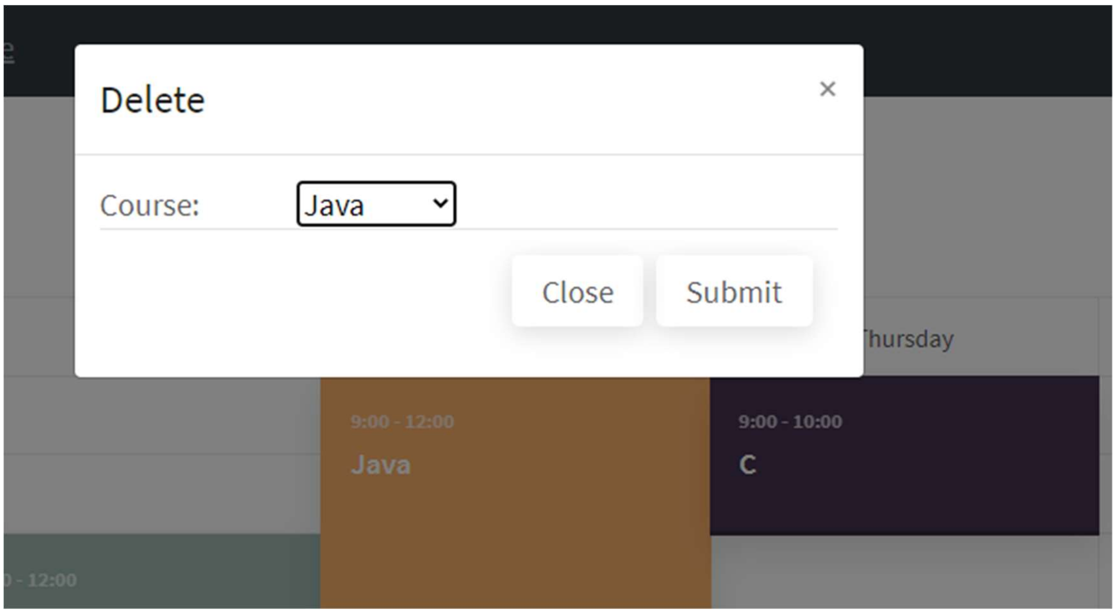


Name: C
Day: Thursday
Start time: 9:00
End time: 10:00
Details:

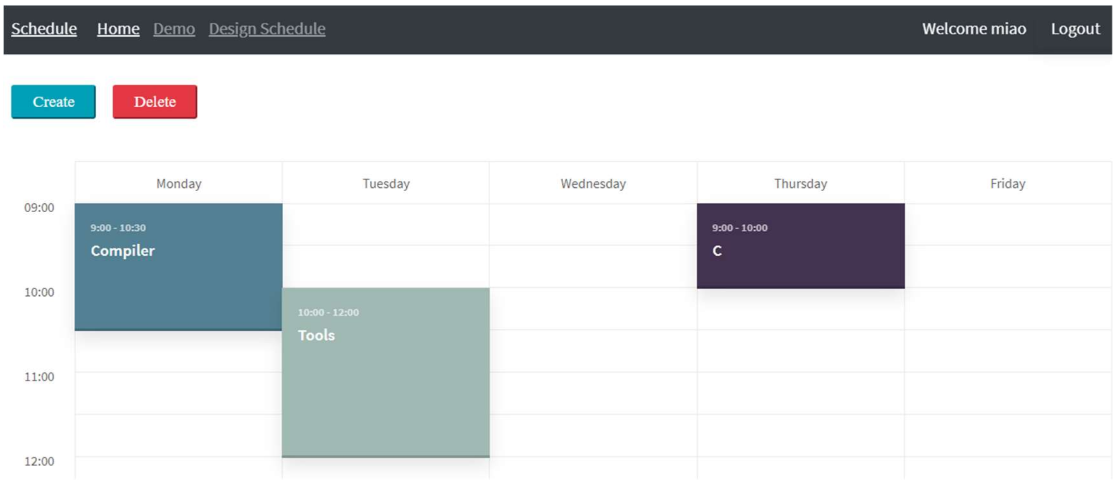
We will see find the new course schedule



After we clikc on the “Delete”, We can select the course we want to delete. And click on the “Submit”.



The selected course will be deleted



6 extra tools

Pycharm. Postman.