Sprint Three Report

12.20.2019

Sriram Annamalai, Joe Calamia ,Hal Crossno, Michael Steinke, Robert Tyndall

Features & Function

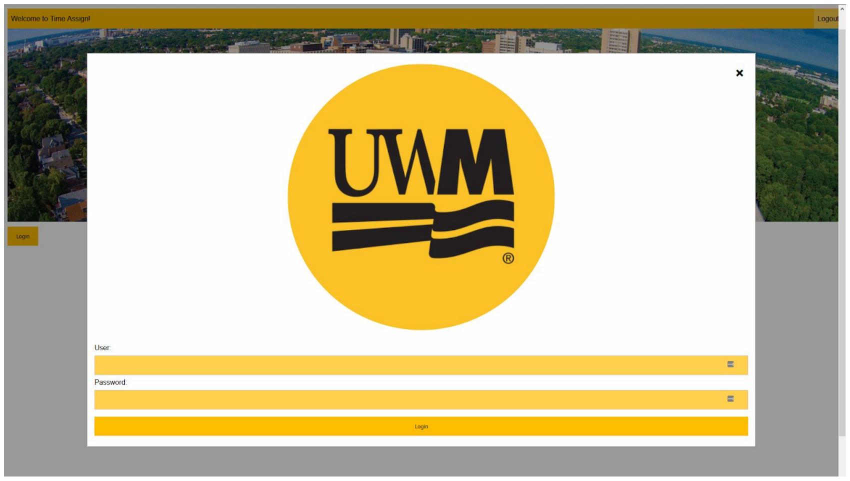
The purpose of this software is to allow teaching assistants, instructors, and faculty members at the University of Wisconsin Milwaukee to organize and schedule course meeting times without conflicts. To accomplish this the software houses a database of users and courses, and allows users to interact with the database through site pages and validate schedules.

Users in the database have different responsibilities and permissions based on UWM policy, and this is reflected in our software. The most important user, who has the most responsibilities is the chair. The chair may create and delete users and courses, as well as assign users of any responsibility level to courses. Instructor users may assign TA users assigned to them by the chair to course sections. TA users may enter times that they are unavailable to take lab sections via the add break feature. These functions are described in more detail in the next section of this report. These functions serve mainly as a means for users to seamlessly and conveniently interact with the underlying database without operating outside of their professional responsibilities.

The validate function ensures that TA users are not assigned to course sections that interfere with their defined breaks. The validate function also ensures that TAs and instructors cannot be assigned to course sections that overlap. This information is displayed to the chair, upon request through the validate page.

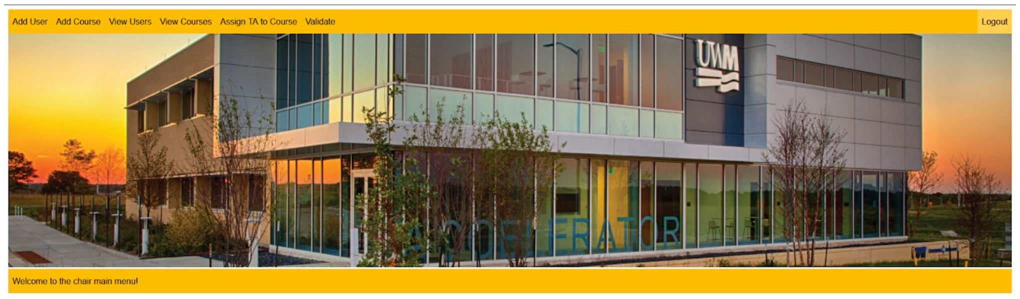
Site pages

Login



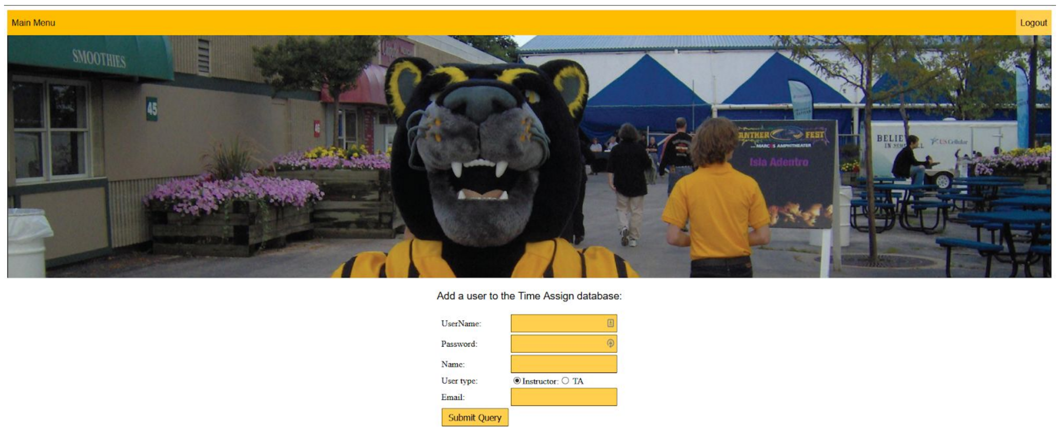
The login page allows users of any type to log in, and simply displays “login failed” in the case of an incorrect username or password. After a successful login this page redirects the user to the menu appropriate for their user type.

Chair menu



The chair menu is available to users of type chair only and links to chair specific pages and features.

Add User



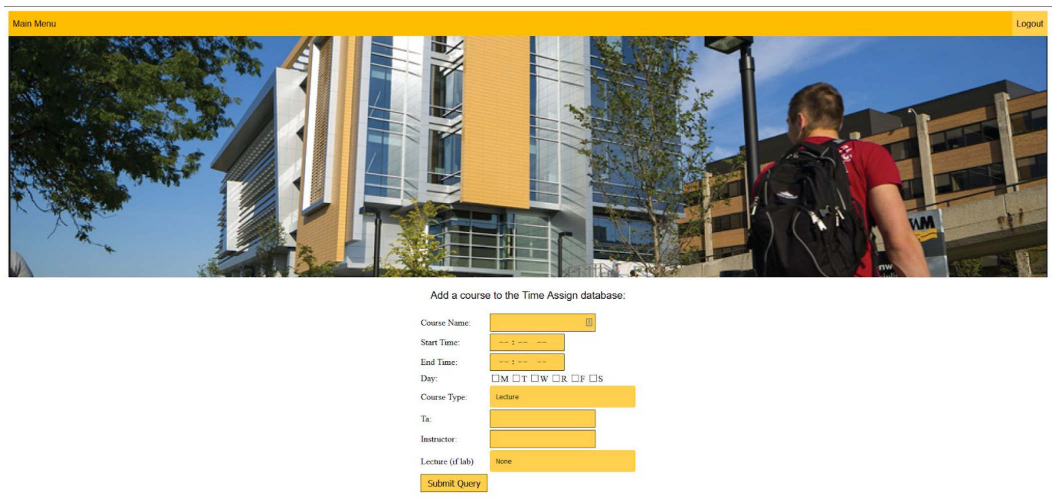
Add a user to the Time Assign database:

UserName:	
Password:	
Name:	
User type:	[✓ Instructor:] TA
Email:	

Submit Query

Adduser is also only available to the chair and allows for users of any type to be created.

Add Course



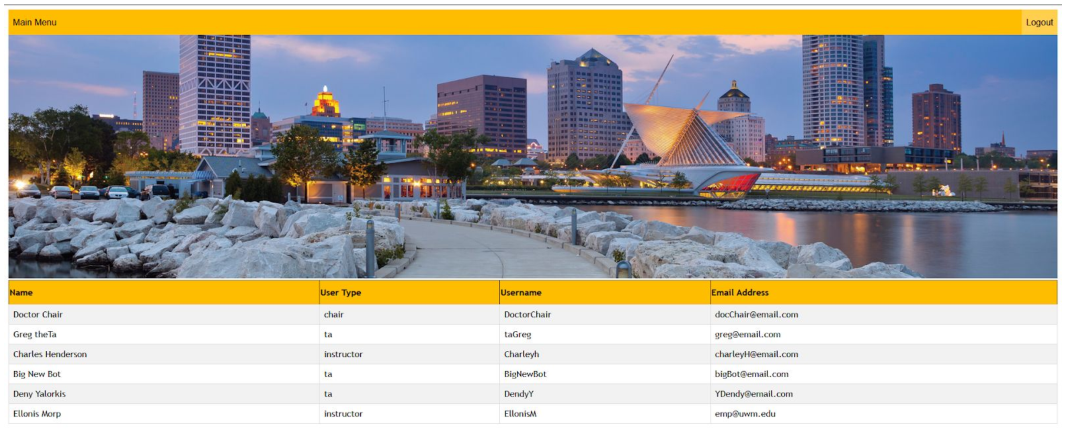
Add a course to the Time Assign database:

Course Name:	
Start Time:	-- : -- --
End Time:	-- : -- --
Day:	☐M ☐T ☐W ☐R ☐F ☐S
Course Type:	Lecture
Ta:	
Instructor:	
Lecture (if lab)	None

Submit Query

AddCourse is only available to the chair and allows for courses of any type to be created, this includes grading sections, labs, lectures, and discussions. Options for different types of sections are all available to make labs attached to existing lectures.

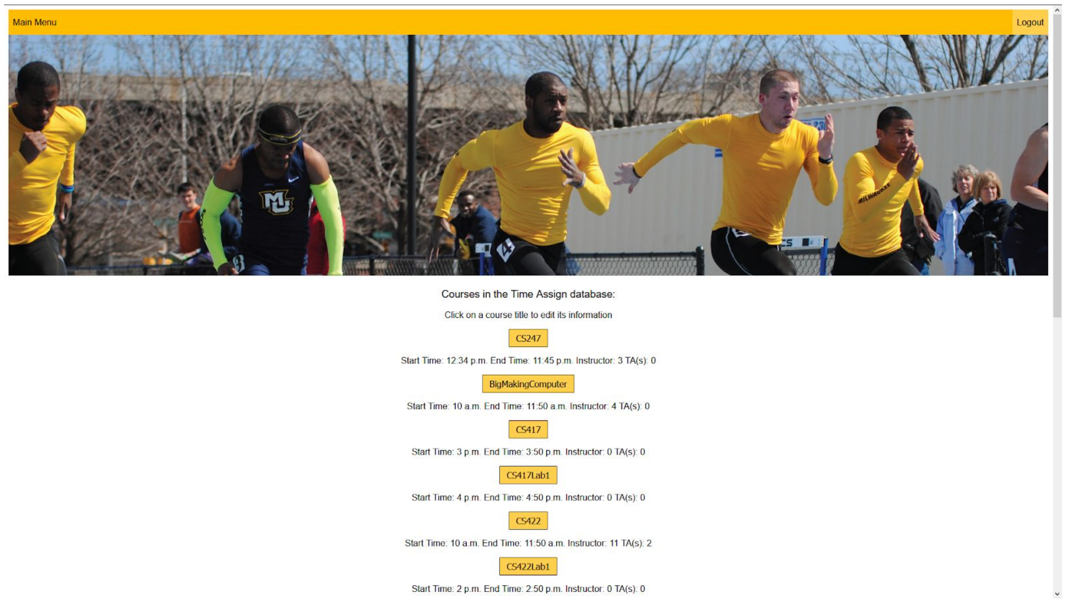
View Users



Name	User Type	Username	Email Address
Doctor Chair	chair	DoctorChair	docChair@email.com
Greg theTa	ta	taGreg	greg@email.com
Charles Henderson	instructor	Charleyh	charleyH@email.com
Big New Bot	ta	BigNewBot	bigBot@email.com
Deny Yalorkis	ta	DendyY	YDendy@email.com
Ellonis Morp	instructor	EllonisM	emp@uwm.edu

ViewUsers is only available to the chair and provides a list of all users, type, username, and email.

View Courses



Courses in the Time Assign database:

Click on a course title to edit its information

CS247

Start Time: 12:34 p.m. End Time: 11:45 p.m. Instructor: 3 TA(s): 0

BigMakingComputer

Start Time: 10 a.m. End Time: 11:50 a.m. Instructor: 4 TA(s): 0

CS417

Start Time: 3 p.m. End Time: 3:50 p.m. Instructor: 0 TA(s): 0

CS417Lab1

Start Time: 4 p.m. End Time: 4:50 p.m. Instructor: 0 TA(s): 0

CS422

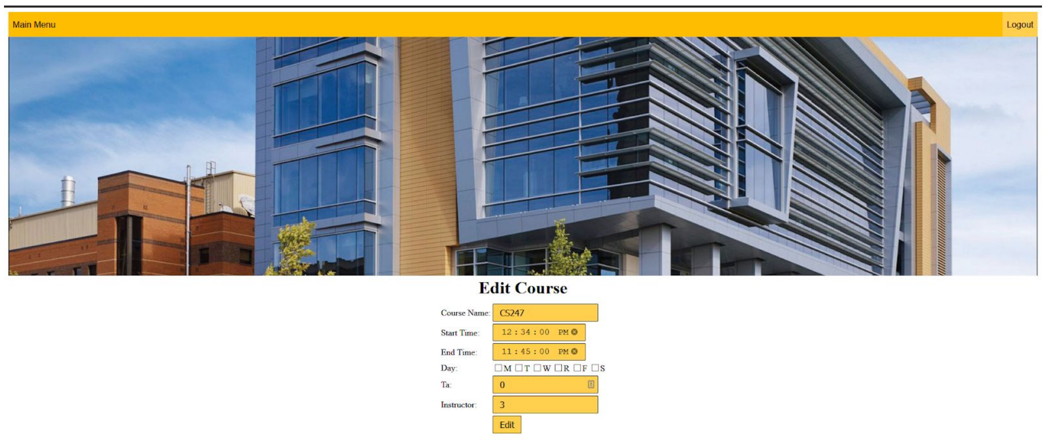
Start Time: 10 a.m. End Time: 11:50 a.m. Instructor: 11 TA(s): 2

CS422Lab1

Start Time: 2 p.m. End Time: 2:50 p.m. Instructor: 0 TA(s): 0

View courses is again only available to the chair, and lists courses as well as offering links to edit each course.

Edit Course

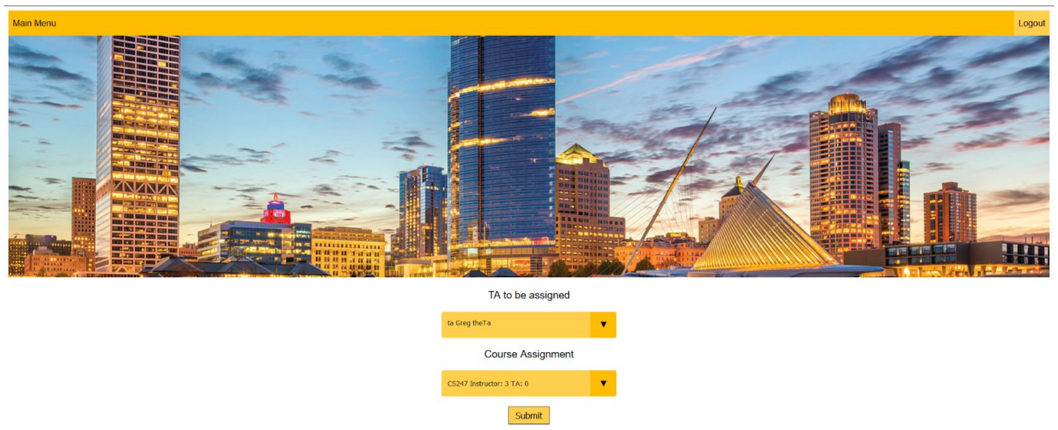


Edit Course

Course Name:	CS247
Start Time:	12 : 34 : 00 PM
End Time:	11 : 45 : 00 PM
Day:	☐M ☐T ☐W ☐R ☐F ☐S
Ta:	0
Instructor:	3

Edit courses is a page generated for each course linked from viewCourse and allows any value for each course to be changed.

Assign To Course



TA to be assigned

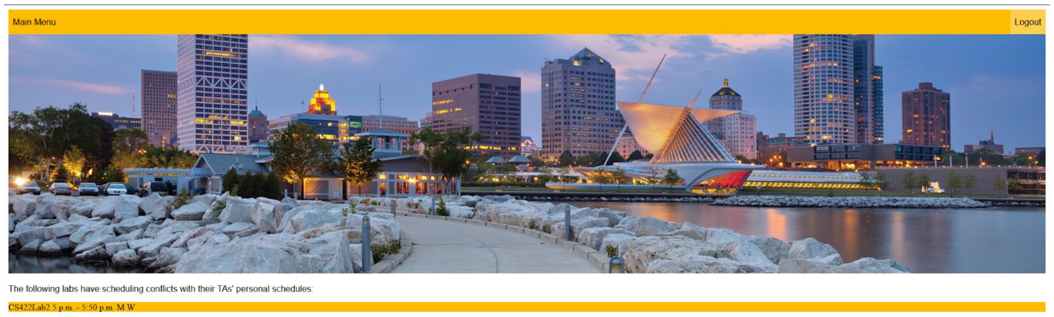
ta Greg theTa

Course Assignment

CS247 Instructor: 3 TA: 0

Assign to course allows the chair to assign instructors and TAs to courses in the database through convenient dropdowns!

Validate

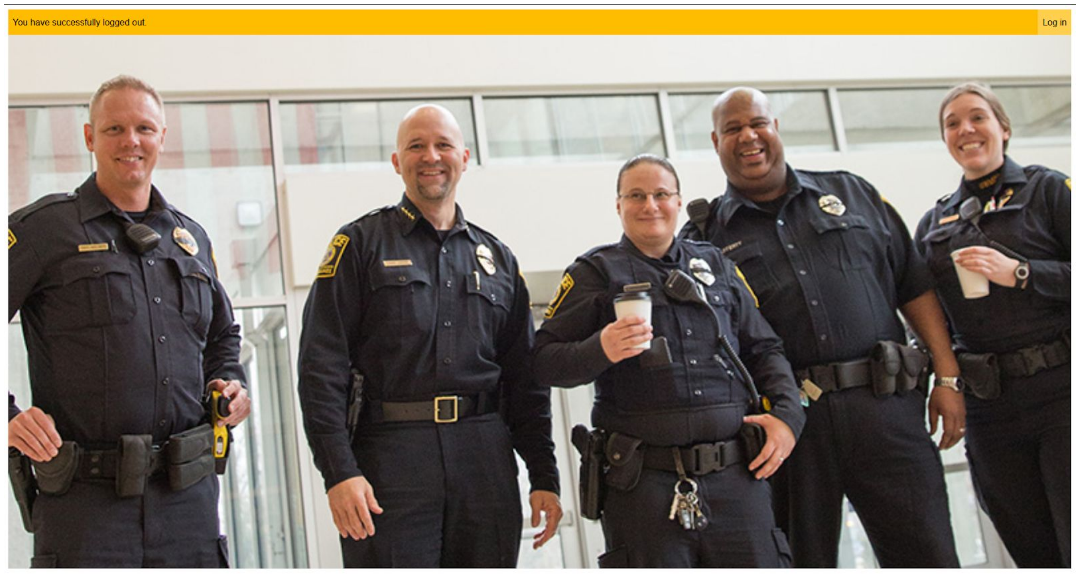


The following labs have scheduling conflicts with their TAs' personal schedules:

CS422Lab2 5 p.m. - 5:50 p.m. M W

The validate function ensures that TA users are not assigned to course sections that interfere with their defined breaks. The validate function also ensures that TAs and instructors cannot be assigned to course sections that overlap. This information is displayed to the chair, upon request through the validate page.

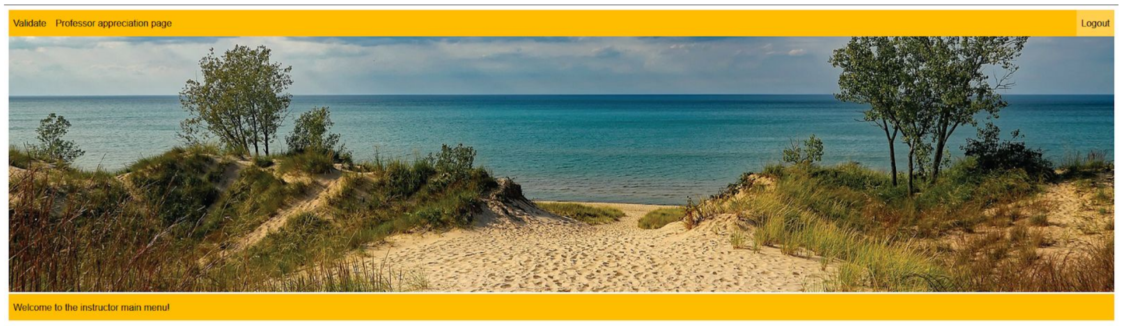
Logout



You have successfully logged out.

The logout page confirms that there is no user logged in and also scares users who are hiding something illegal.

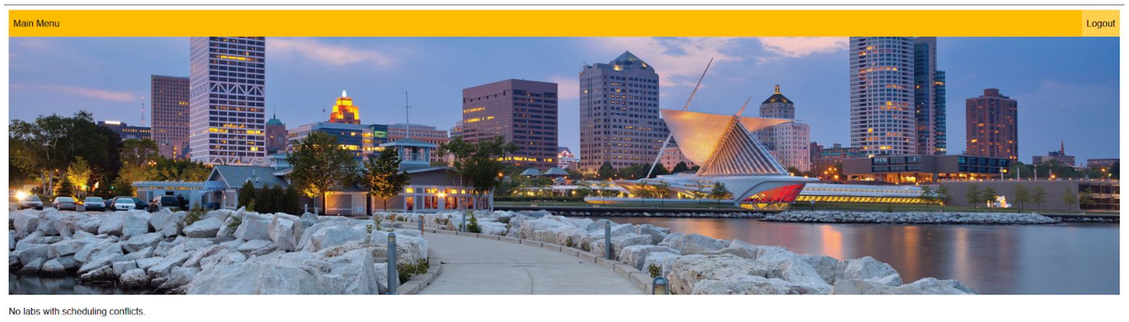
Instructor menu



Welcome to the instructor main menu!

The instructor menu allows for access to instructor features.

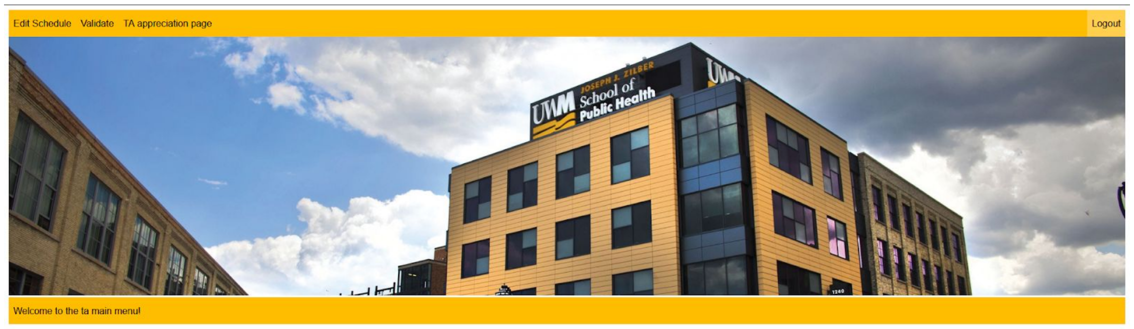
Validate



No labs with scheduling conflicts.

The instructor validate page serves the same function as the chair validate page but only shows courses relevant to the currently logged in instructor user.

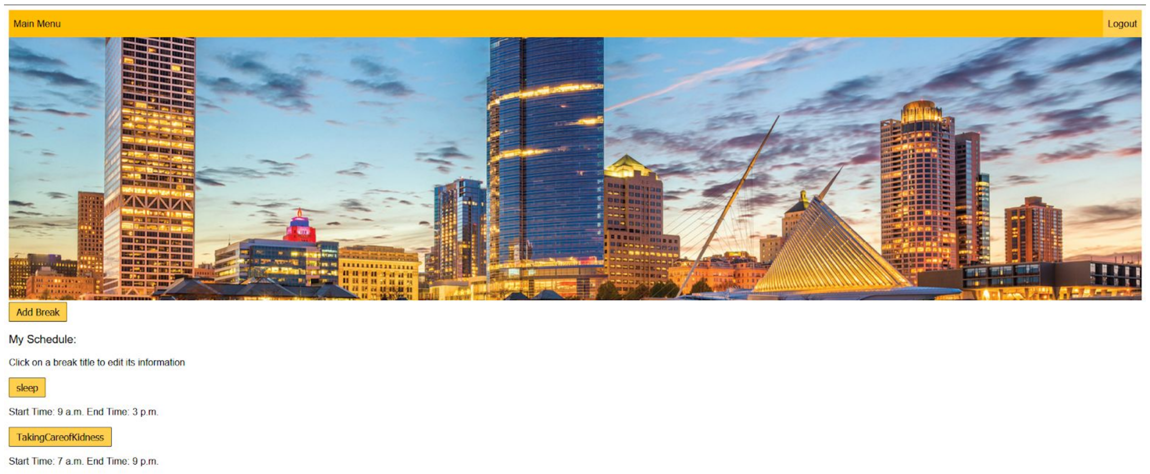
TA menu



Welcome to the ta main menu!

The TA menu allows access to TA user features.

View Breaks



My Schedule:

Click on a break title to edit its information

sleep

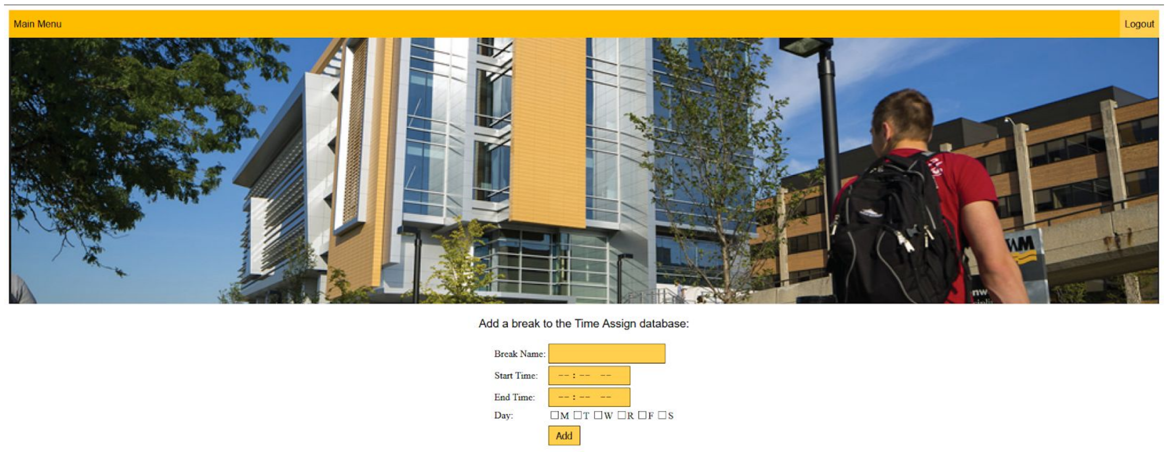
Start Time: 9 a.m. End Time: 3 p.m.

TakingCareofKidness

Start Time: 7 a.m. End Time: 9 p.m.

View breaks is similar to view courses, but for breaks in TA schedules. It also links to the add break page.

Add Break

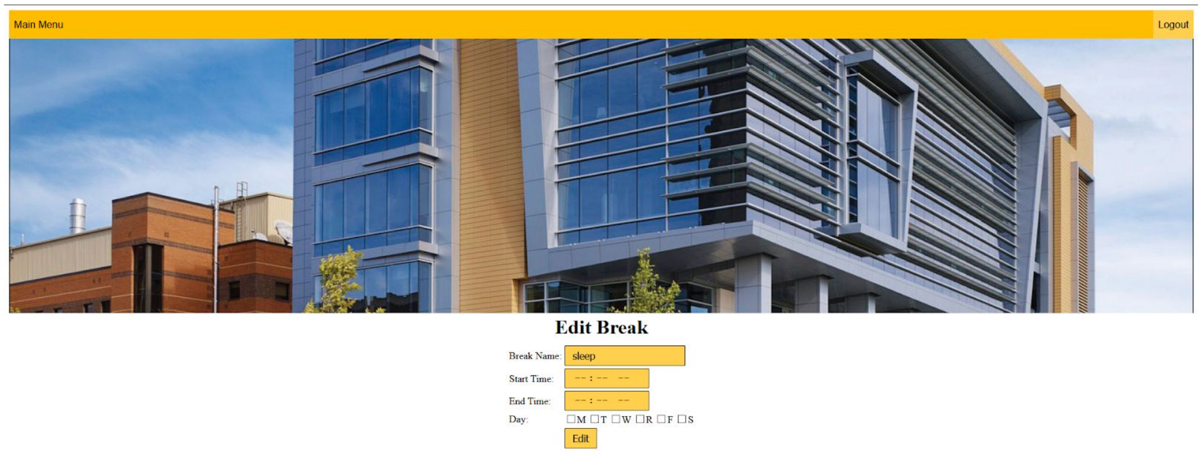


Add a break to the Time Assign database:

Break Name:	
Start Time:	-- : -- --
End Time:	-- : -- --
Day:	☐M ☐T ☐W ☐R ☐F ☐S

Add times when the TA user is unavailable to teach sections.

Edit Break

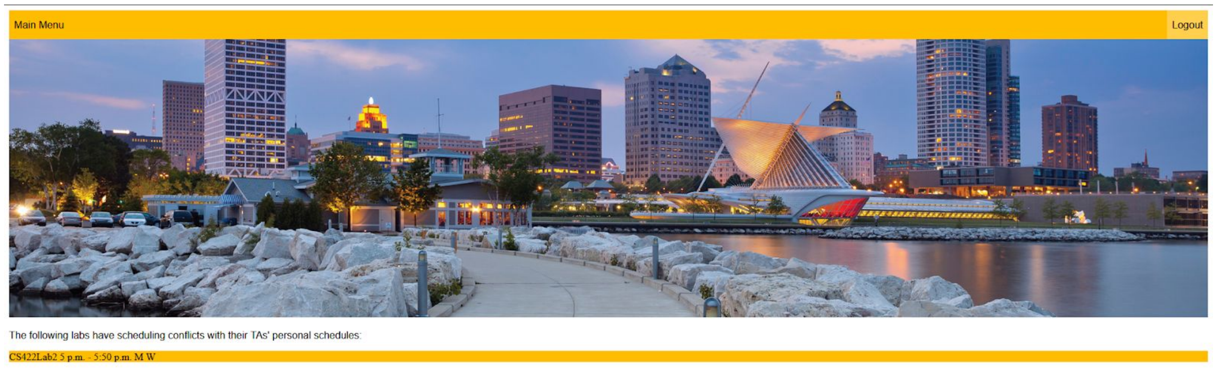


Edit Break

Break Name:	sleep
Start Time:	-- : -- --
End Time:	-- : -- --
Day:	☐M ☐T ☐W ☐R ☐F ☐S

Edit times when the TA user is unavailable to teach sections.

Validate



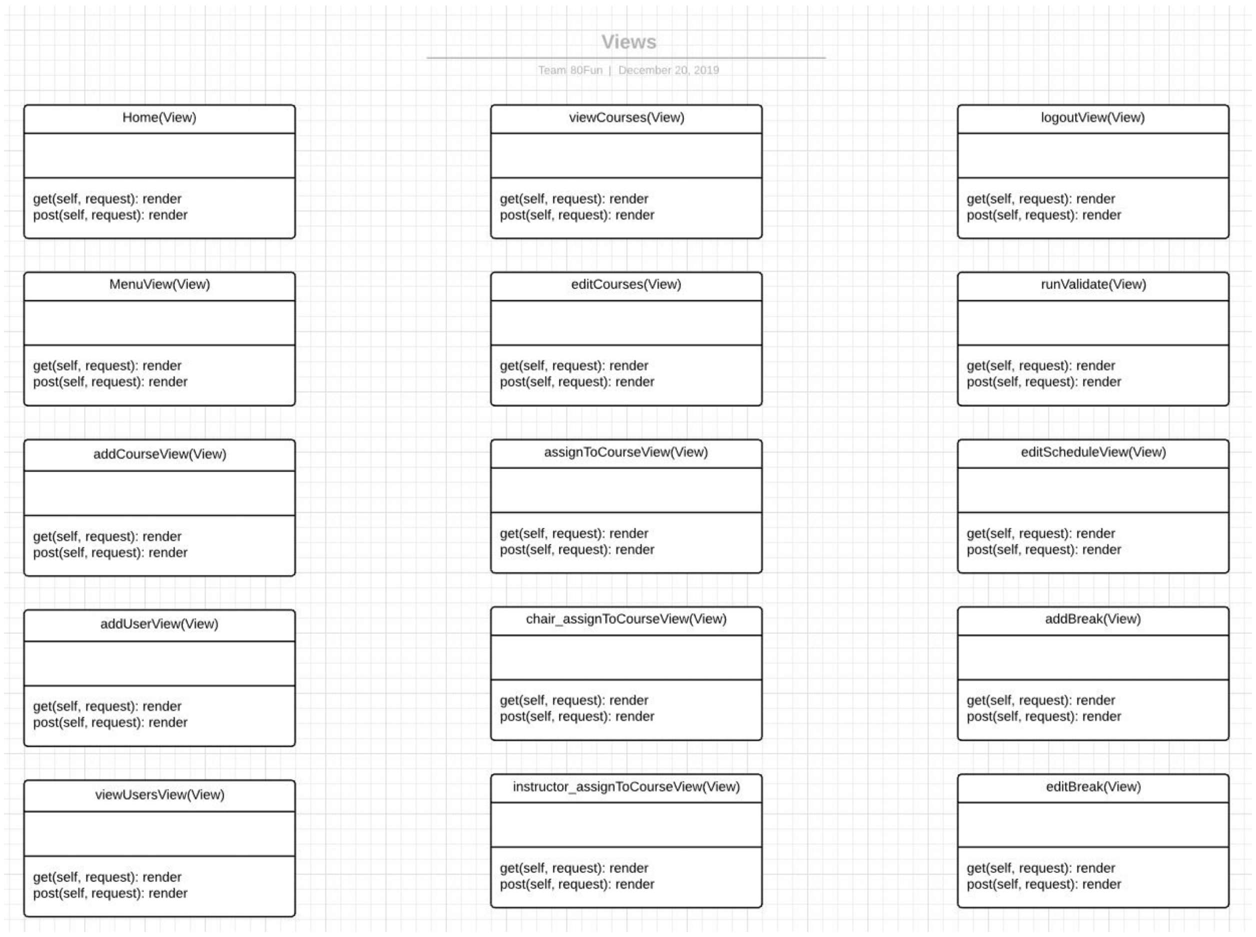
The following labs have scheduling conflicts with their TAs' personal schedules:

CS422Lab2 5 p.m. - 5:50 p.m. M W

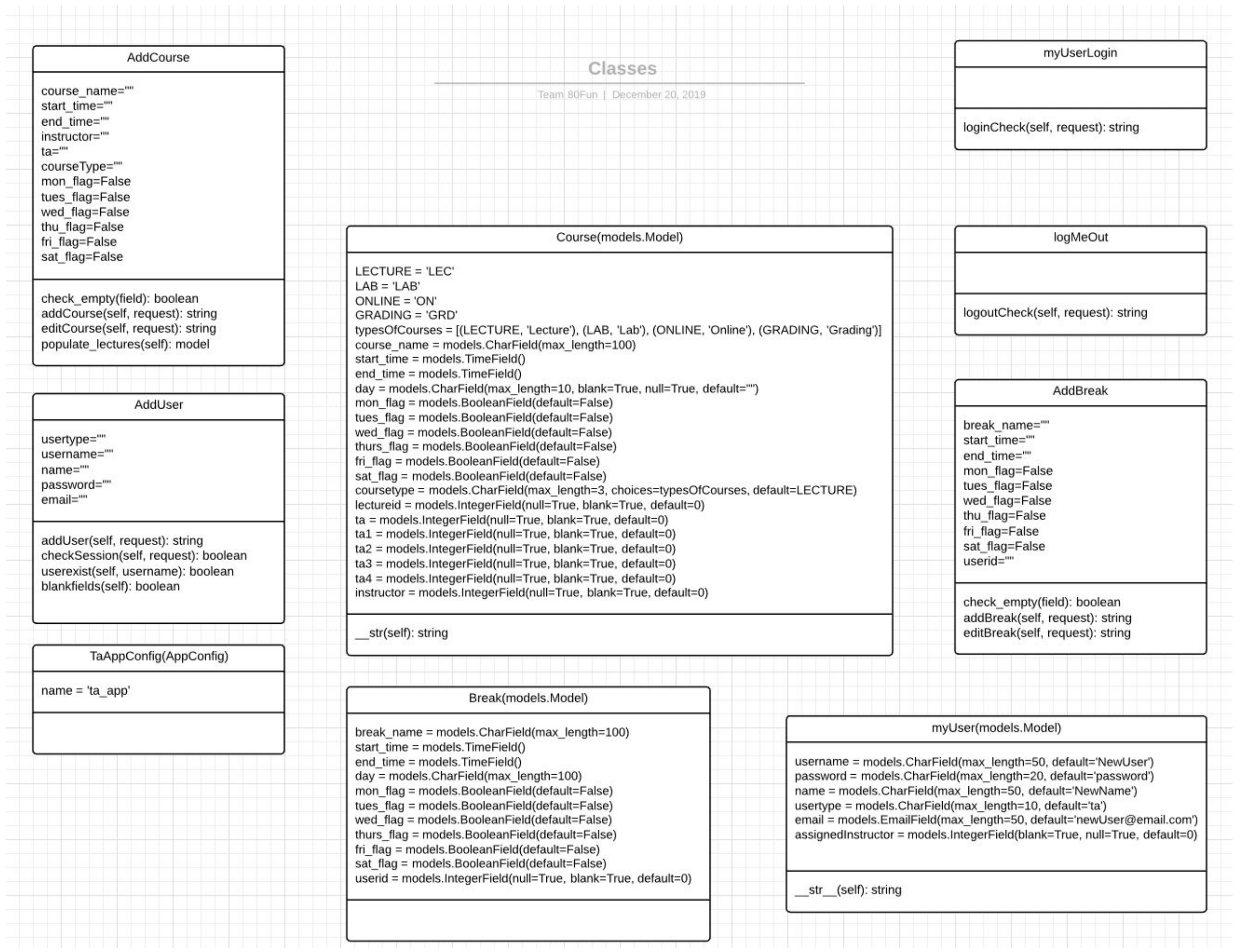
The instructor validate page serves the same function as the chair validate page but only shows courses relevant to the currently logged in TA user.

UML Diagrams

Views


Views
Team 80Fun | December 20, 2019
Home(View)
get(self, request): render
post(self, request): render
viewCourses(View)
get(self, request): render
post(self, request): render
logoutView(View)
get(self, request): render
post(self, request): render
MenuView(View)
get(self, request): render
post(self, request): render
editCourses(View)
get(self, request): render
post(self, request): render
runValidate(View)
get(self, request): render
post(self, request): render
addCourseView(View)
get(self, request): render
post(self, request): render
assignToCourseView(View)
get(self, request): render
post(self, request): render
editScheduleView(View)
get(self, request): render
post(self, request): render
addUserView(View)
get(self, request): render
post(self, request): render
chair_assignToCourseView(View)
get(self, request): render
post(self, request): render
addBreak(View)
get(self, request): render
post(self, request): render
viewUsersView(View)
get(self, request): render
post(self, request): render
instructor_assignToCourseView(View)
get(self, request): render
post(self, request): render
editBreak(View)
get(self, request): render
post(self, request): render

Classes



```
AddCourse
---
course_name=""
start_time=""
end_time=""
instructor=""
ta=""
courseType=""
mon_flag=False
tues_flag=False
wed_flag=False
thu_flag=False
fri_flag=False
sat_flag=False
---
check_empty(field): boolean
addCourse(self, request): string
editCourse(self, request): string
populate_lectures(self): model
```

```
AddUser
---
usertype=""
username=""
name=""
password=""
email=""
---
addUser(self, request): string
checkSession(self, request): boolean
userexist(self, username): boolean
blankfields(self): boolean
```

```
TaAppConfig(AppConfig)
---
name = 'ta_app'
---
```

```
Course(models.Model)
---
LECTURE = 'LEC'
LAB = 'LAB'
ONLINE = 'ON'
GRADING = 'GRD'
typesOfCourses = [(LECTURE, 'Lecture'), (LAB, 'Lab'), (ONLINE, 'Online'), (GRADING, 'Grading')]
course_name = models.CharField(max_length=100)
start_time = models.TimeField()
end_time = models.TimeField()
day = models.CharField(max_length=10, blank=True, null=True, default="")
mon_flag = models.BooleanField(default=False)
tues_flag = models.BooleanField(default=False)
wed_flag = models.BooleanField(default=False)
thurs_flag = models.BooleanField(default=False)
fri_flag = models.BooleanField(default=False)
sat_flag = models.BooleanField(default=False)
coursetype = models.CharField(max_length=3, choices=typesOfCourses, default=LECTURE)
lectureid = models.IntegerField(null=True, blank=True, default=0)
ta = models.IntegerField(null=True, blank=True, default=0)
ta1 = models.IntegerField(null=True, blank=True, default=0)
ta2 = models.IntegerField(null=True, blank=True, default=0)
ta3 = models.IntegerField(null=True, blank=True, default=0)
ta4 = models.IntegerField(null=True, blank=True, default=0)
instructor = models.IntegerField(null=True, blank=True, default=0)
---
__str(self): string
```

```
Break(models.Model)
---
break_name = models.CharField(max_length=100)
start_time = models.TimeField()
end_time = models.TimeField()
day = models.CharField(max_length=100)
mon_flag = models.BooleanField(default=False)
tues_flag = models.BooleanField(default=False)
wed_flag = models.BooleanField(default=False)
thurs_flag = models.BooleanField(default=False)
fri_flag = models.BooleanField(default=False)
sat_flag = models.BooleanField(default=False)
userid = models.IntegerField(null=True, blank=True, default=0)
---
```

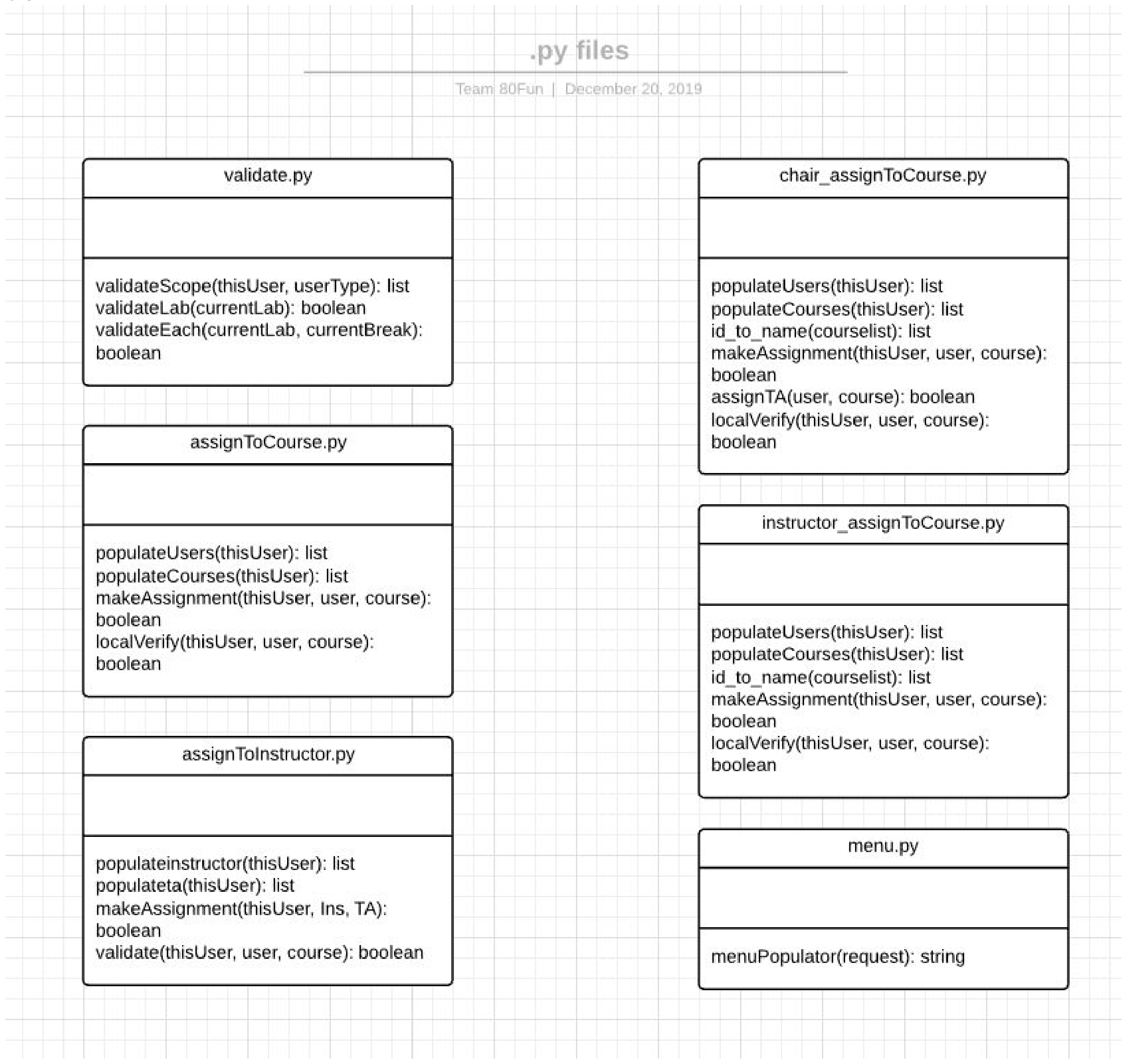
```
myUserLogin
---
---
loginCheck(self, request): string
```

```
logMeOut
---
---
logoutCheck(self, request): string
```

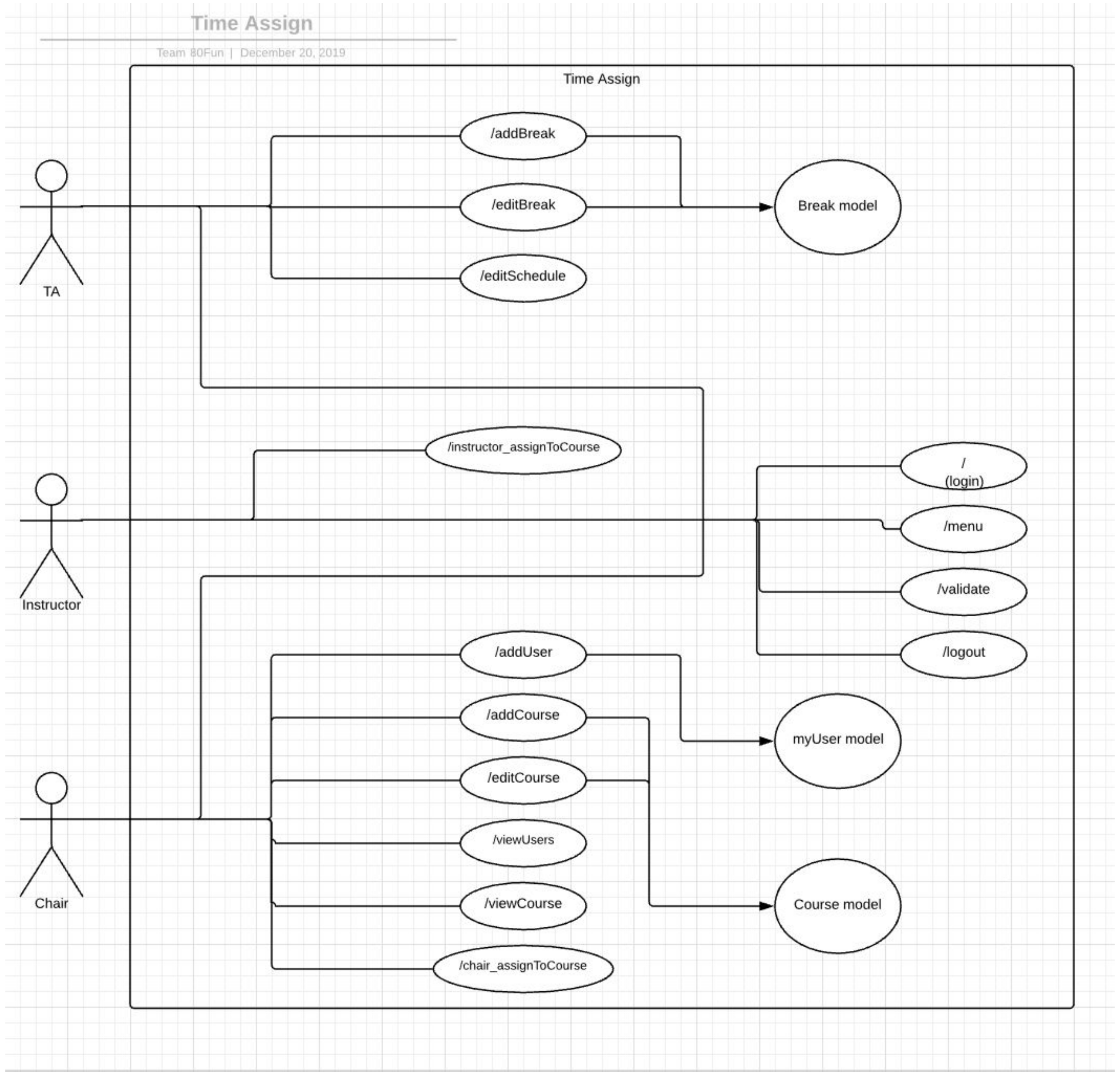
```
AddBreak
---
break_name=""
start_time=""
end_time=""
mon_flag=False
tues_flag=False
wed_flag=False
thu_flag=False
fri_flag=False
sat_flag=False
userid=""
---
check_empty(field): boolean
addBreak(self, request): string
editBreak(self, request): string
```

```
myUser(models.Model)
---
username = models.CharField(max_length=50, default='NewUser')
password = models.CharField(max_length=20, default='password')
name = models.CharField(max_length=50, default='NewName')
usertype = models.CharField(max_length=10, default='ta')
email = models.EmailField(max_length=50, default='newUser@email.com')
assignedInstructor = models.IntegerField(blank=True, null=True, default=0)
---
__str__(self): string
```

.py files


.py files
Team 80Fun | December 20, 2019
validate.py
validateScope(thisUser, userType): list
validateLab(currentLab): boolean
validateEach(currentLab, currentBreak): boolean
assignToCourse.py
populateUsers(thisUser): list
populateCourses(thisUser): list
makeAssignment(thisUser, user, course): boolean
localVerify(thisUser, user, course): boolean
assignToInstructor.py
populateinstructor(thisUser): list
populateta(thisUser): list
makeAssignment(thisUser, Ins, TA): boolean
validate(thisUser, user, course): boolean
chair_assignToCourse.py
populateUsers(thisUser): list
populateCourses(thisUser): list
id_to_name(courselist): list
makeAssignment(thisUser, user, course): boolean
assignTA(user, course): boolean
localVerify(thisUser, user, course): boolean
instructor_assignToCourse.py
populateUsers(thisUser): list
populateCourses(thisUser): list
id_to_name(courselist): list
makeAssignment(thisUser, user, course): boolean
localVerify(thisUser, user, course): boolean
menu.py
menuPopulator(request): string

Use Case Diagram



Scrum Minutes

Meeting One (12/5/19)

Rob-

did: a lot of HTML cleanup, added some other new pages, fixed up static files

will do: acceptance tests, a bit more HTML

obstacles: nothing short term

Joe-

did: plotted out database changes

will do: unit tests & code skeleton, maybe some models changes

obstacles: nothing short term

Sriram-

did: unit tests

will do: updating course model, skeleton & unit tests

obstacles: studying for 361 exam

Michael-

did: refactoring left over from sprint 2

will do: skeleton & unit tests for TA schedule

obstacles: studying for 351 exam

Hal-

did: other homework

to do: work on edit course errors

obstacles: some homework

Meeting Two (12/12/19)

Rob-
did: continued to update html
will do: fill out tests more
obstacles: finals

Joe-
did: filled in the validate unit tests & implementation
will do: clean up acceptance tests, maybe tweak some models, refactor validate & unit tests if it ends up skronky once we can actually test it
obstacles: working the next two days and have another assign due wednesday, not all the external stuff validate needs in order to test is in master yet

Sriram-
did: implemented main python course assignment overhaul
will do: integrating view/html for course assignment
obstacles: some exams

Michael-
did: worked on models & html for TA schedule
will do: finish up implementation
obstacles: another group project and on last homework assignment

Hal-

Did: link up all menu items to their corresponding URLs and pages, and fix some sessions crashing issues

will do: work on some acceptance tests, and fix the hours in edit course

obstacles: None, my last final is friday

Meeting Three (12/19/19)

Rob-

Worked on: Refactored many acceptance tests, refactored much html, site design and testing

Going to work on: Sprint w3 report

obstacles: I didn’t sleep

Joe-

Worked on: I filled in the validate unit tests & implementation

Going to work on: Clean up acceptance tests, maybe tweak some models, refactor validate & unit tests if it ends up skronky once we can actually test it

obstacles: I'm working the next two days and have another assign due wednesday midnight, not all the external stuff validate needs in order to test is in master yet

Sriram-

Worked on: Created chair_assignToCourse (chair to be able to assign both instructors and TA's to lectures) and instructor_assignToCourse(Instructors to be able to assign TA's to labs).

Going to work on: Fix the breaks and check the functionality. Code is working so far, need to run regression tests.

Barrier: I have 2 other exams this week, I am working on them as well.

Michael-

Worked on: Updated break model, put code into taEditSchedule.html

Going to work on: Finishing schedule/ break so that can get pushed asap.

Barrier: I have another group project due at 5pm tomorrow that I'm working on at the same time. CS337 assignment due tomorrow night

Hal-
did: worked on other homework things
Plan to do: work on edit course errors, and anything else that gets assigned to trello.
Roadblocks: only 2 more HW assignments in all classes.

Trello PBIs

TAs will be able to edit their schedules

Will show example of formatted text

Menu options:

- Enter new break
- Edit existing break

New breaks will require:

- Break Name
- Days of the week
- Start time
- End time

Users can validate schedules to make sure that there are no conflicts

- Code skeleton
- Unit Tests
- Implementation
- Refactoring

- Refactor Acceptance Tests
- Pass Unit tests
- Pass acceptance tests

Fix TA assignment: course and instructor

Refactoring

- Switch to checkboxes for adding/ editing courses/ schedules
- refactor acceptance tests for interface changes, database changes
- DB cleanup

Test Analysis

Acceptance Tests

Our acceptance tests test the user interface and functionality of this web application. We have tests for Every menu item in each of the 3 user type menus, and tests for access of each of these menu items when no user is logged in, as well as the wrong type of user logged in trying to access pages they should not see.

For the Chair menu, We have tests for addUser, and addCourse, which tests the errors the Chair user might encounter when entering users and courses, as well as that the correct message is displayed when the correct information format is entered. We also have acceptance tests for out validate function, which creates scheduling conflicts in a dummy database, and makes sure the correct information is displayed for the respective user type.

For the instructor menu, we test that the instructor can assign TAs to their lab sections, and that the dropdown lists populate correctly.

We're passing 60 of our 63 acceptance tests. Two failures are due to some re-direct issues inside of the tests (not sure how to write the tests). We have one failed test from an error in add Course, where we could not figure out how to implement comparisons of time. Test_bad_format_time tests a start time that is after the end time, passes when it should fail. We do have files for acceptance tests for users changing passwords, but we have not yet re-factor them from our command line implementation. On shipment of product, we would focus on passing these last few tests, and a bit more robust testing of newly implemented features.

Unit Tests

Our unit tests are built to test individual parts of our web application, class by class, and method by method. These exist mainly to test parts of the software that rely heavily on the backend, so functionality that works mainly through django is not tested here (see acceptance tests).

Tested is addCourse, which tests the main and helper methods in the addcourse.py file. Add users does much the same. test_assignToCourse does a bit more, setting up a dummy database to test its functionality against, as does chair_assignToCourse. Test_validate tests all helper methods and main functionality inside of validate. Test addUser tests the main and helper methods in the adduser.py file. Similarly, addcourse sets up a dummy database to test the functionality individually. Instructor_Assigntocourse tests the main and the individual helper methods for instructor_assigntocourse.py. To organize our project we split the functions of the site into individual python files in a directory 'sitelogic'. For every file, and every class and method in this file we have a corresponding suite of unit tests that serves as a framework for our design and a regression testable measure of our progress. Tests are fairly self explanatory, as they use SOLID principles and consistent naming schemes to increase clarity.