

Sprint 3 Report



ByteMe

By Alex Blahnik, Elizabeth
Leuenberger, Natasha
Scannell, Eonshik Kim

This report outlines the details surrounding the development, implementation and features of the TA scheduling application designed by the group called “Byte Me” at the University of Wisconsin, Milwaukee (UWM). The TA scheduling application was created in order to provide the Computer Science Department at UWM with a replacement for their current, antiquated method for scheduling TAs. The application was developed using Python and the web framework Django.

The application supports several features, described subsequently, that will greatly improve the TA scheduling system at UWM. The main features of the application are grouped into three categories: Account features, Course features and Scheduling features.

The application relies on the creation and alteration of accounts in order to properly schedule TAs within the department. The application maintains two “superuser” accounts: the administrator account and the supervisor account. These two accounts are to be assigned to faculty/ staff members within the department. The power to create accounts lies with the administrator and supervisor. Instructor accounts and TA accounts can be created. Once created, the account owners can edit and update their personal information. The supervisor and administrator can also edit or update any account’s information, as well as delete accounts. Accounts are password protected and allowed various actions based on the account owner’s level. The creation of accounts linked to specific instructors or TAs is pivotal to the implementation of the main purpose of the application: assigning TAs to labs.

In addition to creating, maintaining and altering accounts all based on account owner level, the application also supports various features surrounding the creation of courses. The supervisor and administrator accounts have the capability to create courses. Courses that are both on campus (in person) and online can be created. When creating a course, the supervisor or administrator can specify a course name, course number, course meeting days and course times. The application supports several safeguard features that prevent against inappropriate information being accidentally or purposefully entered. After a course is created, it is possible for the supervisor account or the administrator account to create sections for the newly created course. Sections can either be a lecture section or a lab section. Similar to the creation of courses, section creation involves a section number, section meeting day(s) and section times. The creation of courses and then sections is necessary for the main purpose of the application, which is to assign TAs to lab sections.

After accounts, courses and sections have been created, the final main feature of the application can be utilized: course/section assignments. The supervisor has the ability to assign both instructors and TAs to courses and sections. Instructors and TAs can be assigned to courses without being assigned to any sections. More than one instructor can be assigned to a course and more than one TA can be assigned to a course. Instructors can be assigned to lecture sections, but not lab sections. TAs can be assigned to lab sections, but not lecture sections. After the supervisor assigns TAs and instructors to courses, it is possible for any instructor assigned to a certain course to then assign the TAs associated with the same course to any lab sections. Only one TA can be assigned to any lab section. After the assignments take place, the account associated with the assignment will be linked with the specific course/section it was assigned to. An account owner can see his or her assignments via the “course assignments” link.

The TA scheduling application will be a useful, easy-to-use tool for the Computer Science Department at UWM. It focuses on the creation and modification of instructor and TA accounts in order to assign specific accounts to courses and sections. The application involves the use of permissions based on account level in order to prevent misuse. In conclusion, the application supports three main features including: Account features, Course features and Assignment features in order to facilitate TA scheduling.

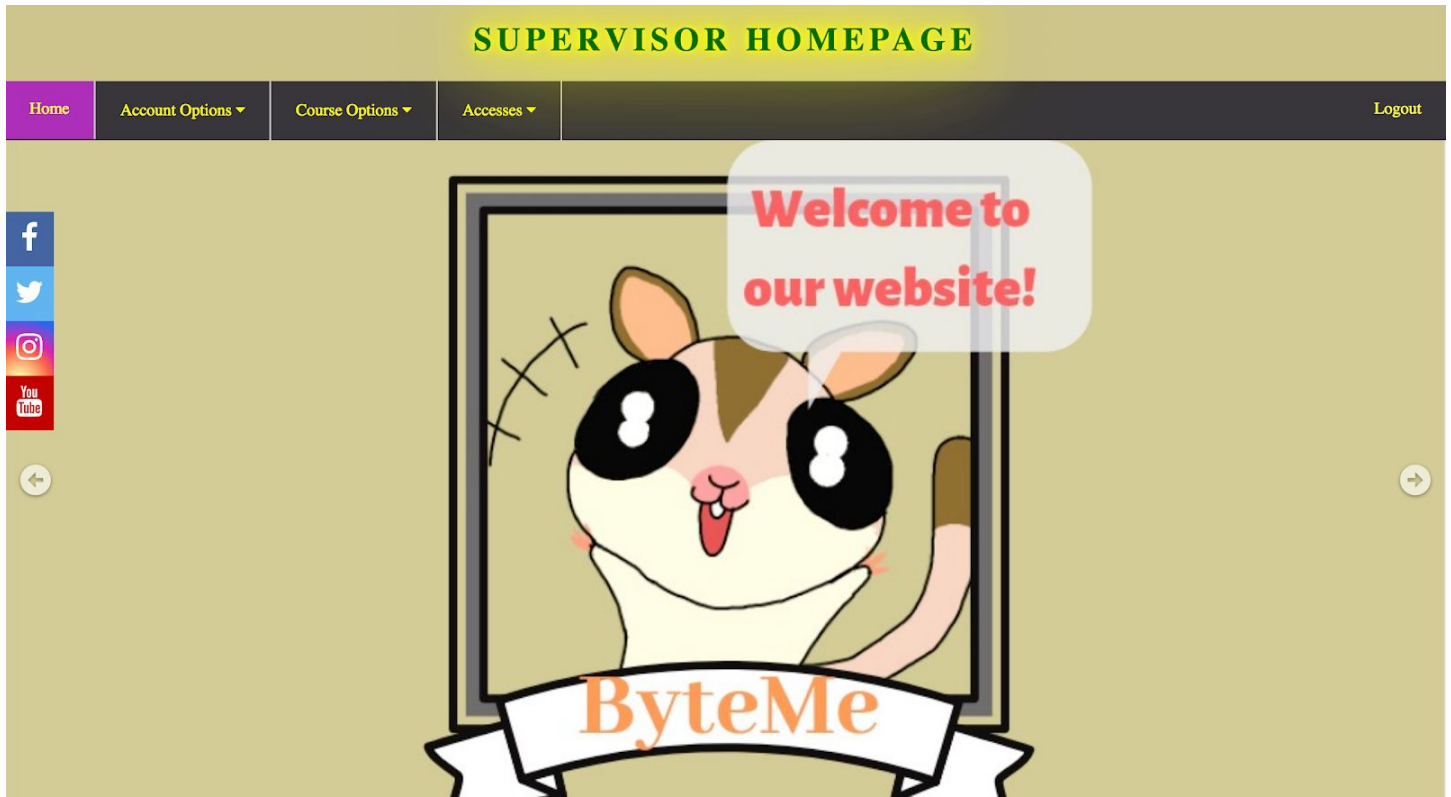
SCREENSHOTS OF WEB PAGES WITH FUNCTIONAL DESCRIPTIONS

Login Page

The image shows a login page for a website called "ByteMe". The page has a light green background. At the top center, there is a pink rectangular box containing a cartoon cat's face with orange and white fur, and the text "ByteMe" below it. Below this box, there are two white input fields with black borders. The first field is labeled "UserName" and the second is labeled "Password". Below these fields is a blue rectangular button with the word "LOGIN" in white capital letters. Below the button, there is a checkbox labeled "Stay Signed in" and a link that says "Can't sign in? [Contact us](#)". At the bottom of the page, there is a large yellow silhouette of a cat standing on top of the word "BYTEME" in a bold, yellow, sans-serif font.

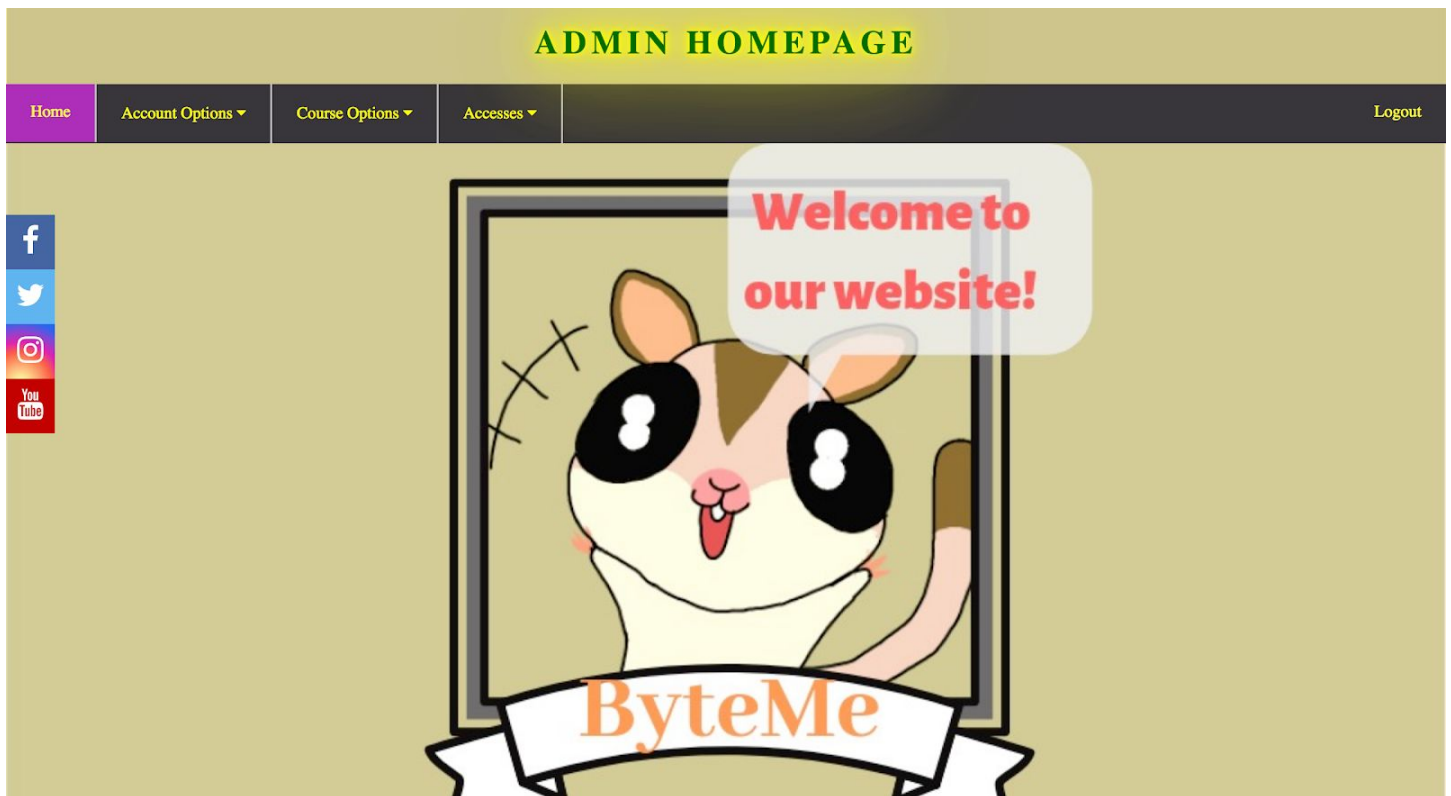
When a user first enters the website they will be directed to the login page. This page allows users to enter in their login information so that they may enter the website.

Supervisor Homepage



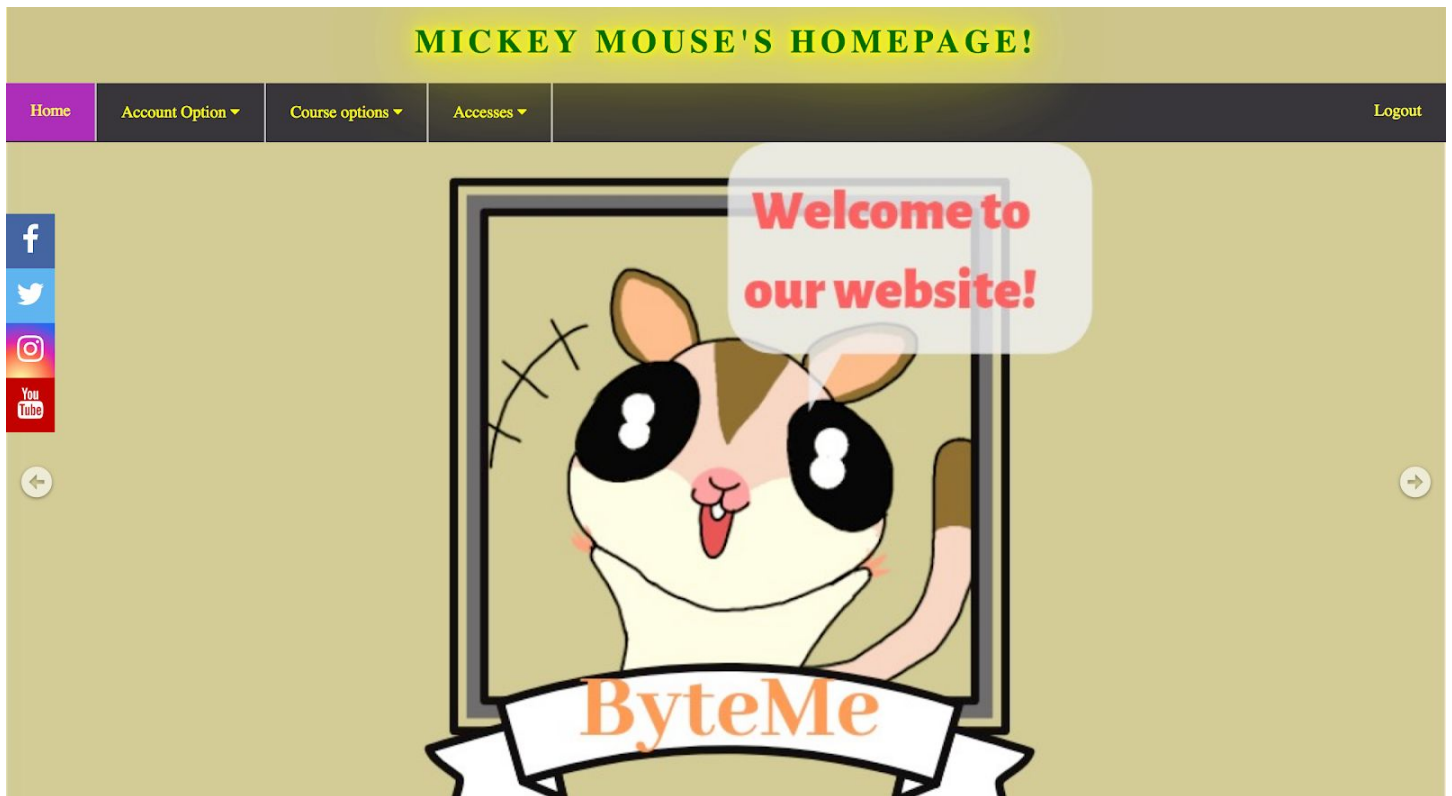
When a supervisor logs into the website they are directed to the supervisor homepage. The homepage allows supervisors to access supervisor only features as well as the public features. From this page Supervisors can access create account, delete account, edit account, create course, create section, assign account to course, assign account to section, course assignments, and the directory page.

Administrator Homepage



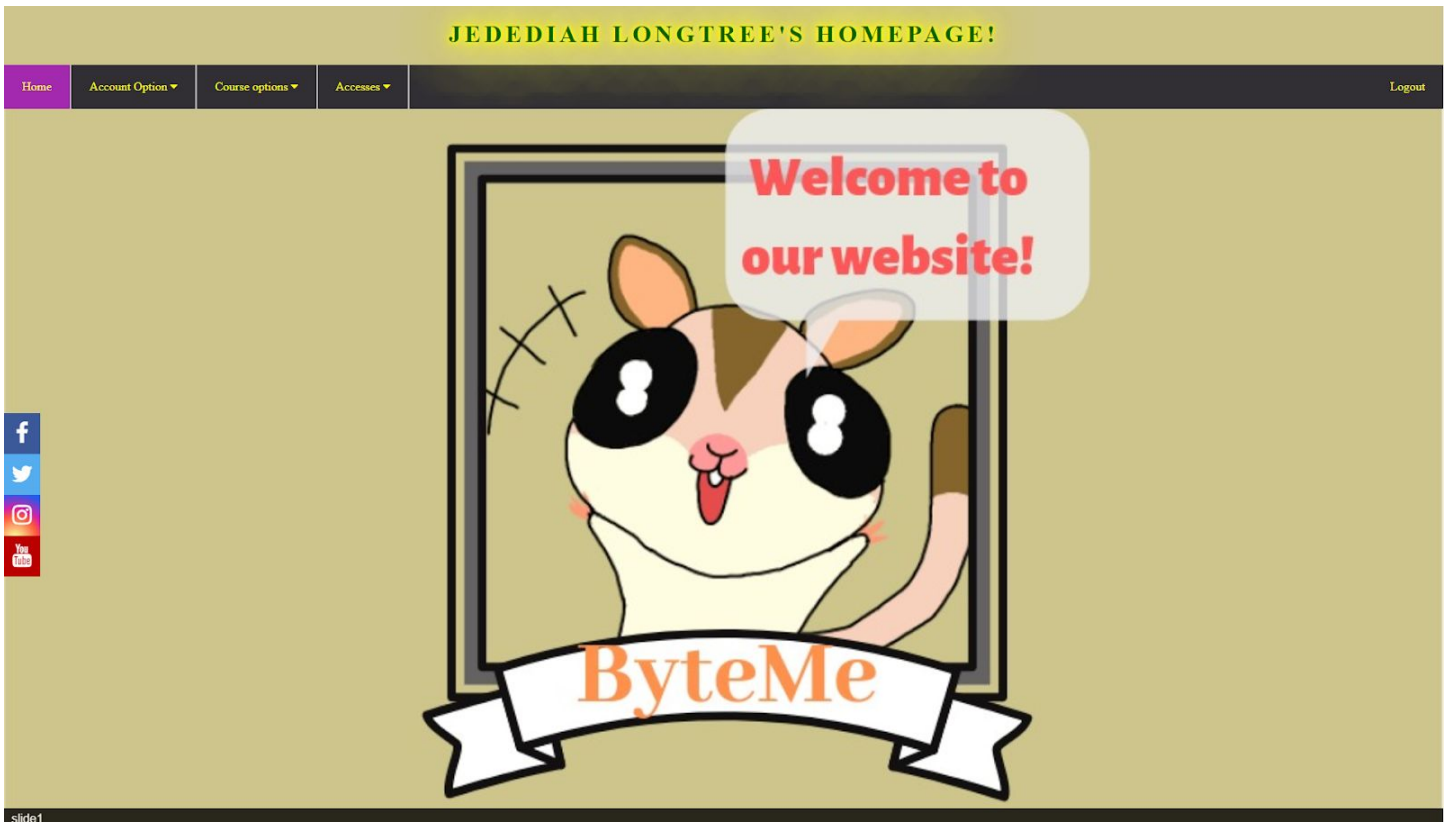
When an administrator logs into the website they are directed to the administrator homepage. The admin homepage allows administrators to access admin only features as well as the public features. From this page Administrators can access create account, delete account, edit account, create a course, create a section, course assignments as well at the directory page.

Instructor Homepage



When an instructor logs into the website they are directed to their homepage. This page allows instructors to access instructor only features as well as the public ones. From this page instructors can access edit contact info, assign TA to section, view course assignments, and the directory page.

Teaching Assistant Homepage



When a TA logs into the website they are directed to their home page. This page allows Teaching assistants to access the TA features. From this page TAs can access edit contact info, view course assignments, and the directory page.


Create Account

CREATE AN ACCOUNT

[Home](#) [Account Options ▼](#) [Course Options ▼](#) [Accesses ▼](#) [Logout](#)

First Name	<input type="text"/>
Last Name	<input type="text"/>
<input type="radio"/> TA	<input type="radio"/> Instructor
UserName	<input type="text"/>
Email	<input type="text"/>

SUBMIT



UWM

When a supervisor or admin selects create account from their homepage they will be directed to the create account page. From this page, administrators and supervisors can enter new account information and add accounts into the database.


Delete Account

DELETE AN ACCOUNT

Home Account Options ▾ Course Options ▾ Accesses ▾ Logout

james kirk

SUBMIT


UWM

When a supervisor or admin selects delete account from their homepage they will be directed to the delete account page. From this page, administrators and supervisors can select an account from a list of all TA and instructor accounts and then delete it from the database.

Edit Other Users Information


EDIT INFORMATION

[Home](#) [Account Options ▾](#) [Course Options ▾](#) [Accesses ▾](#) [Logout](#)

james kirk

⬇

SUBMIT


UWM

When a supervisor or admin selects edit an account from their homepage they will be directed to the edit information page. From this page, administrators and supervisors can select an account from a list of all TA and instructor account and then edit that accounts information.

Edit Information

EDIT INFORMATION

Home

Account Options ▾

Course Options ▾

Accesses ▾

Logout

User Name	disney
First Name	Wait
Last Name	Disney
Email	disney@uwm.edu
Password	disney456
Home Phone	
Address	
City	
State	
Zip Code	0
Office Number	0
Office Phone	
Office Days	0
Office Hours Start	0
Office Hours End	0

SUBMIT QUERY



This is the edit information page. A user can arrive at this page by selecting Edit Information in the drop down box under Account Options. A user will see and be able to edit their own information. A user that is not a supervisor or administrator will only be able to change their own information. A supervisor or administrator will see and be able to edit the user’s information that they selected on the “edit other users information” page. A supervisor or administrator will be able to change the fields of any user in the database.

Create Course

CREATE A COURSE

[Home](#) | [Account Options ▾](#) | [Course Options ▾](#) | [Accesses ▾](#) | [Logout](#)


Course Name

Course Number

☐ On Campus

☐ Online

SUBMIT


UWM

When a supervisor or admin selects create course from their homepage they will be directed to the create course page. From this page, administrators and supervisors can enter in information for a new course and then add it to the database.


Create Section

CREATE A COURSE SECTION

[Home](#) | [Account Options ▾](#) | [Course Options ▾](#) | [Accesses ▾](#) | [Logout](#)

Course	warp theory ▾
<input type="radio"/> Lecture	<input type="radio"/> Lab
Section Number	
Days	
Start Time	
End Time	

SUBMIT


UWM

When a supervisor or admin selects create section from their homepage they will be directed to the create course section page. From this page, administrators and supervisors can enter in information for a new course section and then add it to the database.

Assign Instructor Course


ASSIGN AN INSTRUCTOR TO A COURSE

[Home](#) [Account Options ▾](#) [Course Options ▾](#) [Accesses ▾](#) [Logout](#)

James Kirk

Warp Theory

SUBMIT


UWM

When a supervisor selects assign instructor to course from their homepage, they will be directed to the assign an instructor to a course page. Form this page, supervisors can select an Instructor and a course from the database and assign that instructor to the course. After being assigned to a course, an instructor can then assign the teaching assistants to the lab sections.

Assign TA Course


ASSIGN A TA TO A COURSE

[Home](#) [Account Options ▾](#) [Course Options ▾](#) [Accesses ▾](#) [Logout](#)

Hikaru Sulu

Warp Theory

SUBMIT


UWM

When a supervisor selects assign TA to course from their homepage, they will be directed to the assign a TA to a course page. From this page, supervisors can select a TA and a course from the database and assign that TA to the course. After being assigned to a course, instructors and supervisor can then assign the TAs to Lab sections for that course.

Assign User to Section


ASSIGN A USER TO A COURSE SECTION

HomeAccount Options ▾Course Options ▾Accesses ▾Logout

Choose a course.

Warp Theory

SUBMIT


UWM

ASSIGN A USER TO A COURSE SECTION


HomeAccount Options ▾Course Options ▾Accesses ▾Logout

Warp Theory

James Kirk

LEC 401

SUBMIT


UWM

When a supervisor or Instructor selects assign account to a section they will be directed to the Assign A User To A Course Section Page. When a supervisor views this page they will be able to select any course in the database, from there they can then select any account that is assigned to that course and then assign it to a section. When a Instructor views this page they will only be able to select courses that they are assigned to.

Course Assignments

COURSE ASSIGNMENTS

Home

Account Options ▾

Course Options ▾

Accesses ▾

Logout

Warp Theory CS432

Instructors:
James Kirk

Teaching Assistants:
Hikaru Sulu

LAB 256

Start: 1400

End: 1500

Days: T

Hikaru Sulu

LEC 401

Start: 800

End: 900

Days: MW

James Kirk

Temporal Mechanics CS648

Instructors:
James Kirk

Teaching Assistants:
Hikaru Sulu


LEC 401

Start: 1000

End: 1100

Days: TR

James Kirk



UWM

When any users selects view course assignments from their homepage they will be directed to the Course Assignments page. This page allows all users to view all of the Courses and which Users are assigned to which course. If a course has sections, the accounts assigned to those sections will be displayed below the section.

Directory

DIRECTORY

[Home](#) | [Account Options ▾](#) | [Course Options ▾](#) | [Accesses ▾](#) | [Logout](#)

James Kirk	Instructor
Email: kirkj@uwm.edu	Office Number: 0 Phone: 9999999 Days: 9999999 Hours: 0 - 0
Address: 0000000 qwdsd we 0	Home Phone: 000000

Hikaru Sulu	TA
Email: sulu@uwm.edu	Office Number: 0 Phone: 9999999 Days: 9999999 Hours: 0 - 0
Address: 23 wefkj we 0	Home Phone: 9999999



When any user selects directory from their homepage they will be directed to the Directory page. This page allows all users to view information about all of the accounts in the database. When a Ta or Instructor views this page the names,title,email,and office information is display. When an administrator or supervisor visits this page the addresses and home phone numbers of every account will be displayed on top of all other information.

SCRUM MEETINGS AND TASK BOARDS

Miscellaneous meeting notes:

During this Scrum Meeting we refined our user stories in preparation for Sprint 3. We updated our task board to reflect changes made to the various user stories. We decided that the sprint will mostly focus on functionality surrounding assigning instructors and TAs to lecture and lab sections. We added a Sprint 3 report task to the board.

Elizabeth:

What I did since the last meeting: Nothing pertaining to the project

What I will do until next meeting: I will work on refining our create course function, working on more acceptance tests, starting the report, etc. etc.

What impedes me: Other assignments from other classes

Alex:

What I did yesterday: I converted our form pages into tables so they will line up a little better. I only converted the course assignment page to use tables as well.

What I will do today: I want to convert the directory page to use tables and if I get bored of that I will write more tests.

What Impedes me: I am not creative. I have no idea how the directory page should look.

Eonshik:

What I did since the last meeting: I made slide shows. I put a logo on the login page. I made SNS buttons on account pages. I created the footer and put privacy, terms links in it.

What I will do until the next meeting: I will add the menu bar for each page. I will create the contact page if I have enough time.

What impedes me: To make contact form actually works(to be able to check it is actually sent), I found that I need to create the new app. However, I think it would be better to just make the fake page otherwise, it could mess up our code.

Natasha:

What I did: not really anything

What I will do: create section, assign instructor to course, assign instructors and tas to sections, add more tests

What impedes me: life and other classes

Adrian:

Did not submit the answers to his questions on D2L

Tuesday, April 30th
Corresponding Task Board

User Story	New	In Progress	Ready For Test	Closed	Assigned To
Sprint 3 Report	Description of Software's major features				Elizabeth
	Screenshots of each web app page with function desc.				Elizabeth
	UML case diagram for each major use case				Eonshik
	Daily Scrum Minutes				Elizabeth
	Screen Shot of Task Board after every scrum				Alex
	Class Diagrams of database design				Natasha
	List of completed PBIs				Eonshik
	List of acceptance tests				Elizabeth
	Class diagrams of software classes				Natasha
	List of unit tests				Elizabeth
	Test page				Natasha
	Test analysis				Alex
View TA Assignments	View Screen				Natasha
Assign TA Section	TA cannot be assigned to a section of a course if they are not assigned to the course				Natasha
	TAs can only be assigned to 200 level sections				Natasha
	Supervisors can assign TAs to any section				Natasha
	Instructors can assign TAs to their own sections				Natasha
	No more than 1 TA per section				Natasha
Assign Instructor Section	Supervisors can assign instructors to sections				Natasha

Tuesday, April 30th
Corresponding Task Board

	Instructors can only be assigned to 400 level sections				Natasha
	Only one instructor per section				Natasha
Create Section	Lab and lecture sections can be created for courses				Natasha
Storyless Tasks	Delete Course				Elizabeth
	Delete Assignments				Elizabeth
	Responsive HTML				Eonshik
	Contact Form for login page				Eonshik
	Footers for account pages				Eonshik
	SNS buttons for accounts pages				Eonshik
	Navigation bar for each page				Eonshik
	slide shows for account pages				Eonshik
	More edit info functionality				Elizabeth
	Updated create Course with no days/times				Elizabeth
	Better directory page				Alex

Elizabeth:

What I did since the last meeting: I added functionality to edit information so that users cannot enter office hours without entering office days and vice versa. I changed the course models to no longer include days or times since we are now creating lecture and lab sections. Updated tests to reflect these changes. I wrote the "description of software" page for our spring 3 report. I listed all current tests in the sprint 3 report (acceptance and unit tests)

What I will do until next meeting: Continue refining and refactoring code. Work more on the sprint 3 report.

What impedes me: Nothing

Eonshik:

What I did since the last meeting:

1. I created the navigation bar for an each page. The navigation bar only contains the home and logout buttons.
2. I added the contact form. I moved the contact from from account pages to the login page. Since, it's on the login page, I put the Google Recaptcha v2 to avoid spamming.
3. I added the custom select boxes.

What I will do until next meeting:

1. I will re-design the contact form.
2. I will make the responsive custom select boxes.

What impedes me:

Select boxes move up when the browsers are not full screen. I need to make the responsive select boxes to fix it.

Alex:

What I did yesterday: I worked on some acceptance tests that probably should have been done earlier.

What I will do today: I want to properly format the directory page.

What impedes me: I can't think of a good way to format the directory page. I'm not creative

Natasha:

What I did: Implemented create section, assign users to sections, and assign users to courses. Updated user pages to reflect the changes. Refactored some code in commands.

What I will do: More refactoring, reviewing functionality to see if other updates make sense.

What impedes me: Just other classes

Adrian:

Did not submit answers on D2L

Thursday, May 2nd
Corresponding Task Board

User Story	New	In Progress	Ready For Test	Closed	Assigned To
Sprint 3 Report				Description of Software's major features	Elizabeth
	Screenshots of each web app page with function desc.				Elizabeth
	UML case diagram for each major use case				Eonshik
		Daily Scrum Minutes			Elizabeth
		Screen Shot of Task Board after every scrum			Alex
	Class Diagrams of database design				Natasha
	List of completed PBIs				Eonshik
		List of acceptance tests			Elizabeth
	Class diagrams of software classes				Natasha
		List of unit tests			Elizabeth
	Test page				Natasha
	Test analysis				Alex
View TA Assignments	View Screen				Natasha
Assign TA Section				TA cannot be assigned to a section of a course if they are not assigned to the course	Natasha
				TAs can only be assigned to 200 level sections	Natasha
				Supervisors can assign TAs to any section	Natasha
				Instructors can assign TAs to their own sections	Natasha
				No more than 1 TA per section	Natasha
Assign Instructor Section				Supervisors can assign instructors to sections	Natasha

Thursday, May 2nd
Corresponding Task Board

				Instructors can only be assigned to 400 level sections	Natasha
				Only one instructor per section	Natasha
Create Section				Lab and lecture sections can be created for courses	Natasha
Storyless Tasks	Delete Course				Elizabeth
	Delete Assignments				Elizabeth
	Responsive HTML				Eonshik
		Contact Form for login page			Eonshik
				Footers for account pages	Eonshik
				SNS buttons for accounts pages	Eonshik
				Navigation bar for each page	Eonshik
				slide shows for account pages	Eonshik
				More edit info functionality	Elizabeth
				Updated create Course with no days/times	Elizabeth
	Better directory page				Alex

Elizabeth -

What I did since the last scrum: I made it so the "home" button displayed on each page takes a user to their actual homepage, rather than just back a page. I began inputting screenshots of each web page into the sprint 3 report. Added background to edit info success page. Talked to Apoorv about the sprint 2 review.

What I will do until the next scrum: Continue work on the report, specifically the functional descriptions of each web page.

What impedes me: Getting conflicting information about the design.

Alex -

What I did yesterday: I updated the directory page so it no longer says "something" and it now no longer displays a home phone and address for TAs and Instructors.

What I will do today: If I can come up with a better layout I might change the course assignments page. I will also be starting and hopefully finishing the test analysis for the sprint 3 report.

What impedes me: Short attention span.

Eonshik -

What I did since the last scrum: I fixed the issue that the select box kept moving up on the small screen. It has displayed correctly on each page now. I also worked on PBI completed lists on our paper.

What I will do until the next scrum: I will work on UML use case diagram and change the design of the contact and other pages.

What impedes me: Custom select box didn't look neat and organize with "submit" button. It was my bad idea.

Natasha -

What I did: some code refactoring; updated tests that were using the wrong type for online sections and added a test; fixed bugs with time.

What I will do: review UML for database and software; function description key; refactoring

What impedes me: life/other classes

Adrian:

Did not submit answers on D2L

Sunday, May 5th
Corresponding Task Board

User Story	New	In Progress	Ready For Test	Closed	Assigned To
Sprint 3 Report				Description of Software's major features	Elizabeth
		Screenshots of each web app page with function desc.			Elizabeth
	UML case diagram for each major use case				Eonshik
		Daily Scrum Minutes			Elizabeth
		Screen Shot of Task Board after every scrum			Alex
	Class Diagrams of database design				Natasha
		List of completed PBIs			Eonshik
		List of acceptance tests			Elizabeth
	Class diagrams of software classes				Natasha
		List of unit tests			Elizabeth
	Test page				Natasha
		Test analysis			Alex
View TA Assignments	View Screen				Natasha
Assign TA Section				TA cannot be assigned to a section of a course if they are not assigned to the course	Natasha
				TAs can only be assigned to 200 level sections	Natasha
				Supervisors can assign TAs to any section	Natasha
				Instructors can assign TAs to their own sections	Natasha
				No more than 1 TA per section	Natasha
Assign Instructor Section				Supervisors can assign instructors to sections	Natasha

Sunday, May 5th
Corresponding Task Board

				Instructors can only be assigned to 400 level sections	Natasha
				Only one instructor per section	Natasha
Create Section				Lab and lecture sections can be created for courses	Natasha
Storyless Tasks	Delete Course				Elizabeth
	Delete Assignments				Elizabeth
				Responsive HTML	Eonshik
			Contact Form for login page		Eonshik
				Footers for account pages	Eonshik
				SNS buttons for accounts pages	Eonshik
				Navigation bar for each page	Eonshik
				slide shows for account pages	Eonshik
				More edit info functionality	Elizabeth
				Updated create Course with no days/times	Elizabeth
				Better directory page	Alex
	Make an app for one of the functionalities (maybe) - per Apoorv's review				TBD
				CSS edit info success	Elizabeth

Elizabeth -

What I did since the last meeting: I worked with Alex to move one of our functionalities to its own app. I continued working on the Sprint 3 report. Added more acceptance tests.

What I will do until next meeting: Will continue working on sprint 3 report. Write more tests. Make sure everything is in order. Start making outline for presentation.

What impedes me: Nothing.

Eonshik -

What I did since the last meeting:

1. I redesigned the contact form page.
2. I updated the UML use case diagram.
3. I deleted the phone number on the error page and changed the sentence for the picture of the second on our slideshows. (to avoid being sued from Daniel Kondos Law Offices)
4. I added the hit/ visitor counter for the login page.

What I will do until next meeting:

I will update the the lists of completed PBIs and check the design of every page. I will also revise the UML use case diagram.

What impedes me: Time constraint, final exams are coming!

Alex-

What I did yesterday: I went through all of the html files and made it so they extend other templates now. This means that the proper navigation bar will be on every page.

What I will do today: Updating the html broke a lot of the acceptance tests. Should probably fix those.

What Impedes me: N/A

Natasha -

What I did: fixed a couple of issues, worked on UML for database and included it in the report, started editing the UML for software

What I will do: Finalize UML for software, function descriptions, find anything else to add to project

What impedes me: other classes

Tuesday, May 7th
Corresponding Task Board

User Story	New	In Progress	Ready For Test	Closed	Assigned To
Sprint 3 Report				Description of Software's major features	Elizabeth
				Screenshots of each web app page with function desc.	Alex
			UML case diagram for each major use case		Eonshik
		Daily Scrum Minutes			Elizabeth
		Screen Shot of Task Board after every scrum			Alex
				Class Diagrams of database design	Natasha
		List of completed PBIs			Eonshik
		List of acceptance tests			Elizabeth
		Class diagrams of software classes			Natasha
		List of unit tests			Elizabeth
	Test page				Natasha
		Test analysis			Alex
Assign TA Section				TA cannot be assigned to a section of a course if they are not assigned to the course	Natasha
				TAs can only be assigned to 200 level sections	Natasha
				Supervisors can assign TAs to any section	Natasha
				Instructors can assign TAs to their own sections	Natasha
				No more than 1 TA per section	Natasha
Assign Instructor Section				Supervisors can assign instructors to sections	Natasha
				Instructors can only be assigned to 400 level sections	Natasha

Tuesday, May 7th
Corresponding Task Board

				Only one instructor per section	Natasha
Create Section				Lab and lecture sections can be created for courses	Natasha
Storyless Tasks	Delete Course				Elizabeth
	Delete Assignments				Elizabeth
				Responsive HTML	Eonshik
			Contact Form for login page		Eonshik
				Footers for account pages	Eonshik
				SNS buttons for accounts pages	Eonshik
				Navigation bar for each page	Eonshik
				slide shows for account pages	Eonshik
				More edit info functionality	Elizabeth
				Updated create Course with no days/times	Elizabeth
				Better directory page	Alex
				Make an app for one of the functionalities (maybe) - per Apoorv's review	Alex/Elizabeth
				CSS edit info success	Elizabeth
				Refactor HTML	Alex

Elizabeth -

What I did since last scrum: I worked a lot on refactoring the edit pub info method, so now it can determine multiple incorrect fields at one time and change the correct ones while not changing the incorrect ones. Wrote tests for this functionality.

What I will do until next time: Finishing touches. Work on presentation.

What impedes me: Nothing.

Eonshik -

What I did since last scrum:

1. I finished UML use case diagram and PBI lists for our report. I double checked them to make sure I did correctly.
2. I reformatted HTML and CSS codes.
3. I revised the contact form to be able once the user submits the form I will be able to receive it on my email account.

What I will do until next time: I will work on the presentation.

What impedes me: It looks like when I use Gmail or Outlook, It sometimes takes a long time to receive the email for the contact form. Yahoo mail works fine though.

Alex -

What I Did Yesterday: I worked on my part of the report. Adding in Pictures of the web pages. Added super awesome animations to some of the elements.

What I will do today: Work on the test page of the report. Maybe add more super awesome animations.

What Impedes me: nothing

Natasha -

What I did: Worked on report: UML diagrams for database and software, writing function descriptions for everything covered in diagrams

What I will do: finish the function descriptions, anything else that needs to be completed

What impedes me: Other classes

Thursday, May 9th
Corresponding Task Board

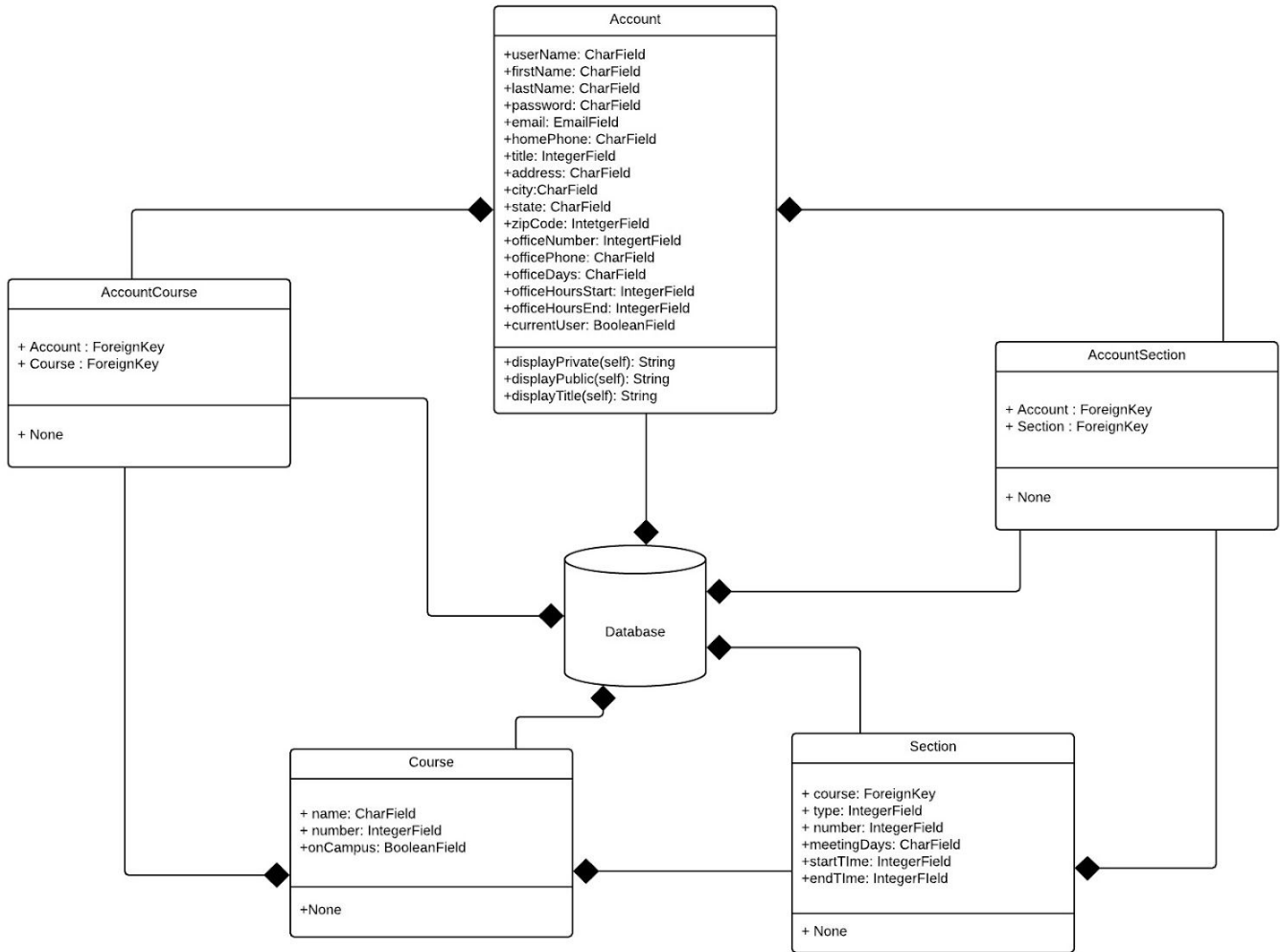
User Story	New	In Progress	Ready For Test	Closed	Assigned To
Sprint 3 Report				Description of Software's major features	Elizabeth
				Screenshots of each web app page with function desc.	Alex
				UML case diagram for each major use case	Eonshik
				Daily Scrum Minutes	Elizabeth
				Screen Shot of Task Board after every scrum	Alex
				Class Diagrams of database design	Natasha
				List of completed PBIs	Eonshik
		List of acceptance tests			Elizabeth
				Class diagrams of software classes	Natasha
		List of unit tests			Elizabeth
	Test page				Natasha
				Test analysis	Alex
Assign TA Section				TA cannot be assigned to a section of a course if they are not assigned to the course	Natasha
				TAs can only be assigned to 200 level sections	Natasha
				Supervisors can assign TAs to any section	Natasha
				Instructors can assign TAs to their own sections	Natasha
				No more than 1 TA per section	Natasha
Assign Instructor Section				Supervisors can assign instructors to sections	Natasha
				Instructors can only be assigned to 400 level sections	Natasha

Thursday, May 9th
Corresponding Task Board

				Only one instructor per section	Natasha
Create Section				Lab and lecture sections can be created for courses	Natasha
Storyless Tasks			Delete Course		Alex
	Delete Assignments				Elizabeth
				Responsive HTML	Eonshik
				Contact Form for login page	Eonshik
				Footers for account pages	Eonshik
				SNS buttons for accounts pages	Eonshik
				Navigation bar for each page	Eonshik
				slide shows for account pages	Eonshik
				More edit info functionality	Elizabeth
				Updated create Course with no days/times	Elizabeth
				Better directory page	Alex
				Make an app for one of the functionalities (maybe) - per Apoorv's review	Alex/Elizabeth
				CSS edit info success	Elizabeth
				Refactor HTML	Alex
			Make Edit info better		Elizabeth

DATABASE DESIGN

TaScheduleizer
Byte Me | May 8, 2019



ACCEPTANCE TESTS

test_login_wrong_password
test_login_username_does_not_exist
test_createAccount_success
test_createAccount_render
test_createAccount_alreadyexists
test_createAccount_invalid_email
test_createAccount_invalid_title
test_deleteAccount_success
test_deleteAccount_render
test_deleteAccount_doesnotexists
test_createCourse_success
test_createCourse_invalidNumber
test_createCourse_course_exists
test_createCourse_render
test_createCourse_invalid_days
test_createCourse_invalid_times
test_createCourse_invalid_locations
test_createCourse_times_out_of_order
test_createCourse_name_exists
test_createSection_success
test_createSection_invalidNumber
test_createSection_course_not_exists
test_createSection_not_onsite_class
test_create_section_invalid_sectNum
test_createSection_Lab_render
test_create_section_invalid_days
test_create_section_invalid_times
test_editPubInfo_firstName
test_editPubInfo_render
test_editPubInfo_lastName
test_editPubInfo_two_fields
test_editPubInfo_homephone_invalid
test_editPubInfo_officephone_invalid
test_editPubInfo_zipcode_invalid
test_editPubInfo_officenum_invalid
test_editPubInfo_firstname_invalid
test_editPubInfo_lastname_invalid
test_editPubInfo_city_invalid
test_editPubInfo_city_invalid2
test_editPubInfo_state_invalid
test_editPubInfo_state_invalid2
test_editPubInfo_officetimes_invalid
test_editPubInfo_start_no_end
test_editPubInfo_end_noStart
test_editPubInfo_times_noDays
test_editPubInfo_days_noTimes
test_assignInsCourse_success
test_assignInsCourse_course_does_not_exists
test_assignInsCourse_user_does_not_exist
test_assignTACourse_user_does_not_exist
test_viewPublicInfo_not_logged_in
test_viewPublicInfo_success
test_viewTAHome_Success
test_viewInstructorHome_success

test_directory_Ta_View
test_directory_Instructor_View
test_directory_Supervisor_View
test_courseAssignments_view
test_sendOutNotification_TA_success
test_sendOutNotification_TA_not_success
test_sendOutNotification_ins_not_success
test_viewTaHome_noLogin
test_viewTaHome_InstructorLogin
test_viewTaHome_AdminLogin
test_viewTahome_SupervisorLogin
test_viewInstructorHome_nologin
test_viewInstructorHome_TaLogin
test_viewInstructorHome_AdminLogin
test_viewInstructorHome_SupervisorLogin
test_viewAdminHome_nologin
test_viewAdminHome_TaLogin
test_viewAdminHome_InstructorLogin
test_viewAdminHome_SupervisorLogin
test_viewSupervisorHome_nologin
test_viewSupervisorHome_TaLogin
test_viewSupervisorHome_Instructorlogin
test_viewSupervisorHome_AdminLogin
test_createAccount_nologin
test_createAccount_TaLogin
test_createAccount_InstructorLogin
test_viewCourseAssignmenets_nologin
test_deleteAccount_nologin
test_deleteAccount_TaLogin
test_deleteAccount_InstructorLogin
test_assignInsCourse_nologin
test_assignInsCourse_TaLogin
test_assignInsCourse_InstructorLogin
test_assignInsCourse_AdminLogin
test_assignTaCourse_nologin
test_assignTaCourse_TaLogin
test_assignTaCourse_InstructorLogin
test_assignTaCourse_AdminLogin
test_directory_nologin
test_editPubInfo_nologin
test_createCourse_nologin
test_createCourse_TaLogin
test_createCourse_InstructorLogin
test_editUserInfo_nologin
test_editUserInfo_TaLogin
test_editUserInfo_InstructorLogin
test_deleteCourse_nologin
test_deleteCourse_TaLogin
test_deleteCourse_InstructorLogin
test_deleteAccount_Success
test_deleteAccount_accountNotFound
test_deleteCourse_Success
test_deleteCourse_notFound

UNIT TESTS

Assign Account Course

test_assign_success
test_assign_invalid_course
test_assign_nonexistent_acct
test_assign_invalid_title

Assign Account Section

test_assign_success
test_assign_invalid_course
test_assign_nonexistent_section
test_assign_invalid_section
test_assign_nonexistent_acct
test_assign_invalid_title
test_assign_not_assigned
test_assign_already_assigned

Create Account

test_account_successfully_created
test_success_post
test_account_already_exist
test_account_already_exists_post
test_invalid_email
test_invalid_email_post
test_invalid_title
test_invalid_title_post

Create Course

test_course_successfully_created
test_create_course_already_exists
test_create_course_invalid_courseNum
test_invalid_location
test_invalid_days
test_invalid_times
test_times_out_of_order
test_course_with_name_exists

Create Section

test_createSection_success
test_createSection_invalid_course
test_createSection_courseDNE
test_createSection_invalidType
test_createSection_onlineLab_fail
test_createSection_onlineLecture_success
test_createSection_invalid_sectionNumber
test_createSection_alreadyExists
test_createSection_invalidTimes

Display Course Assignments

test_displayCourseAssign_Empty
test_displayCourseAssign_1_Lec
test_displayCourseAssign_1_Lab
test_displayCourseAssign_1_Lec_1_lab
test_displayCourse_2_lect_2_Lab
test_displayCourse_1_Instructor
test_displayCourse_2_Instructor

test_displayCourse_1_TA
test_displayCourse_2_TA
test_displayCourse_1_instructor_1_TA

Display Private Info

test_displayPrivate_TA
test_displayPrivate_Instructor
Display Public Info

test_displayPublic_TA
test_displayPublic_Instructor

Edit Public Information

test_change_firstName_success
test_change_lastName_success
test_change_email_success
test_change_password_success
test_change_homephone_success
test_change_address_success
test_change_city_success
test_change_state_success
test_change_zipcode_success
test_change_officePhone_success
test_change_officeNum_success
test_change_officedays_success
test_change_officeStart_success
test_change_officeend_success
test_change_email_invalid
test_change_firstname_invalid
test_change_lastname_invalid
test_change_city_invalid
test_change_state_invalid
test_change_homephone_invalid
test_change_officedays_invalid
test_change_officephone_invalid
test_change_zip_invalid
test_change_officeNum_invalid
test_change_times_invalid
test_change_times_start_nofinish
test_change_times_finish_nostart
test_times_no_day
test_days_no_times

View Course Assignments

test_viewCourseAssign_accountNotFound
test_viewCourseAssign_noAssignments
test_viewCourseAssign_Instructor_1Course
test_viewCourseAssign_Instructor_2Course
test_viewCourseAssign_TA_CourseLab
test_viewCourseAssign_Ta_2CourseLab
test_viewCourseAssign_Ta_2CourseLab_1Course
test_viewCourseAssign_TA_noLabs
test_viewCourseAssign_Ta_1CourseLab_1Course

TEST PAGE

Acceptance Tests:

Test Results	1	test_createSection_Lab_success	test_editPublInfo_correct_email_days_no_times_def
Tests.AcceptanceTests.test_web.Test_web	1	test_createSection_Lec_invalidNumber	test_editPublInfo_correct_email_days_no_times_em
test_assignInsCourse_AdminLogin		test_createSection_Lec_invalidNumber2	test_editPublInfo_days_noTimes_default
test_assignInsCourse_InstructorLogin		test_createSection_Lec_invalidNumber3	test_editPublInfo_end_noStart_default
test_assignInsCourse_TaLogin		test_createSection_Lec_invalidNumber4	test_editPublInfo_end_noStart_emptString
test_assignInsCourse_course_does_not_exists		test_createSection_Lec_success	test_editPublInfo_firstname
test_assignInsCourse_nologin		test_createSection_TaLogin	test_editPublInfo_firstname_invalid
test_assignInsCourse_render		test_createSection_exists	test_editPublInfo_homephone_invalid
test_assignInsCourse_success		test_createSection_invalidDays	test_editPublInfo_lastname
test_assignInsCourse_user_does_not_exist		test_createSection_invalidEnd	test_editPublInfo_lastname_invalid
test_assignTACourse_user_does_not_exist		test_createSection_invalidEnd2	test_editPublInfo_nologin
test_assignTACourse_AdminLogin		test_createSection_invalidEnd3	test_editPublInfo_officenum_invalid
test_assignTACourse_InstructorLogin		test_createSection_invalidStart	test_editPublInfo_officenum_invalid2
test_assignTACourse_TaLogin		test_createSection_invalidStart2	test_editPublInfo_officenum_invalid
test_assignTACourse_nologin		test_createSection_invalidStart3	test_editPublInfo_officetimest_invalid
test_createAccount_InstructorLogin		test_createSection_nologin	test_editPublInfo_officetimest_invalid2
test_createAccount_TaLogin		test_createSection_onlineCourseLab	test_editPublInfo_officetimest_invalid3
test_createAccount_alreadyexists		test_createSection_onlineCourseLec	test_editPublInfo_officetimest_invalid4
test_createAccount_invalid_email		test_deleteAccount_InstructorLogin	test_editPublInfo_officetimest_invalid5
test_createAccount_invalid_title		test_deleteAccount_InstructorLogin2	test_editPublInfo_render
test_createAccount_nologin		test_deleteAccount_Success	test_editPublInfo_start_no_end2_emptString
test_createAccount_render		test_deleteAccount_TaLogin	test_editPublInfo_start_no_end_default
test_createAccount_success		test_deleteAccount_TaLogin2	test_editPublInfo_state_invalid
test_createCourse_InstructorLogin		test_deleteAccount_accountNotFound	test_editPublInfo_state_invalid2
test_createCourse_TaLogin		test_deleteAccount_doesnotexists	test_editPublInfo_times_noDays_default
test_createCourse_course_exists		test_deleteAccount_nologin	test_editPublInfo_times_noDays_default_emptStrin
test_createCourse_invalidNumber		test_deleteAccount_nologin2	test_editPublInfo_two_fields
test_createCourse_invalidNumber2		test_deleteAccount_render	test_editPublInfo_zipcode_invalid
test_createCourse_invalidNumber3		test_deleteAccount_success	test_editPublInfo_zipcode_invalid2
test_createCourse_invalidNumber4		test_deleteCourse_InstructorLogin	test_editUserInfo_InstructorLogin
test_createCourse_invalid_locations		test_deleteCourse_Success	test_editUserInfo_TaLogin
test_createCourse_invalid_locations2		test_deleteCourse_TaLogin	test_editUserInfo_nologin
test_createCourse_invalid_locations3		test_deleteCourse_nologin	test_login_username_does_not_exist
test_createCourse_name_exists		test_deleteCourse_notFound	test_login_wrong_password
test_createCourse_nologin		test_directory_nologin	test_sendOutNotification_TA_not_success
test_createCourse_render		test_editPublInfo_FN_LN_email_HP_CITY_invalid	test_sendOutNotification_TA_success
test_createCourse_success		test_editPublInfo_FN_LN_invalid	test_sendOutNotification_ins_not_success
test_createSection_InstructorLogin		test_editPublInfo_change_multipleFields_five	test_sendOutNotification_ins_success
test_createSection_Lab_invalidCourseNumber		test_editPublInfo_change_multipleFields_four	test_viewAdminHome_InstructorLogin
test_createSection_Lab_invalidNumber		test_editPublInfo_change_multipleFields_six	test_viewAdminHome_SupervisorLogin
test_createSection_Lab_invalidNumber2		test_editPublInfo_change_multipleFields_three	test_viewAdminHome_TaLogin
test_createSection_Lab_invalidNumber3		test_editPublInfo_city_invalid	test_viewPublicInfo_not_logged_in
test_createSection_Lab_invalidNumber4		test_editPublInfo_city_invalid2	test_viewPublicInfo_success
test_createSection_Lab_render		test_editPublInfo_correct_FN_incorrect_LN	test_viewSupervisorHome_AdminLogin
			test_viewSupervisorHome_Instructorlogin
			test_viewSupervisorHome_TaLogin
			test_viewSupervisorHome_nologin
			test_viewTAHome_Success
			test_viewTAHome_AdminLogin
			test_viewTAHome_InstructorLogin

Unit tests

Test_assignAccCourse

Test Results	14 ms
Tests.UnitTests.test_assignAccCourse.TestAssignAccC	14 ms
test_assign_invalid_course	1 ms
test_assign_invalid_title	4 ms
test_assign_nonexistent_acct	1 ms
test_assign_success	8 ms

Test_createSection

Test Results	11 ms
Tests.UnitTests.test_createSection.TestCreateSection	11 ms
test_createSection_alreadyExists	3 ms
test_createSection_courseDNE	1 ms
test_createSection_invalidTimes	1 ms
test_createSection_invalidType	0 ms
test_createSection_invalid_course	0 ms
test_createSection_invalid_sectionNumber	3 ms
test_createSection_onlineLab_fail	0 ms
test_createSection_onlineLecture_success	2 ms
test_createSection_success	1 ms

Test_editPubInfo

Test Results	9 ms
Tests.UnitTests.test_editPubInfo.Test_editPubInfo	9 ms
test_change_address_success	0 ms
test_change_city_invalid	1 ms
test_change_city_success	0 ms
test_change_email_invalid	0 ms
test_change_email_success	0 ms
test_change_firstName_success	0 ms
test_change_firstname_invalid	0 ms
test_change_homephone_invalid	0 ms
test_change_homephone_success	0 ms
test_change_lastName_success	0 ms
test_change_lastname_invalid	0 ms
test_change_officeNum_invalid	0 ms
test_change_officeNum_success	0 ms
test_change_officePhone_success	1 ms
test_change_officeStart_success	1 ms
test_change_officedays_invalid	0 ms
test_change_officedays_success	1 ms
test_change_officeend_success	0 ms
test_change_officephone_invalid	0 ms
test_change_password_success	0 ms
test_change_state_invalid	0 ms
test_change_state_success	1 ms
test_change_times_finish_nostart	0 ms
test_change_times_invalid	0 ms
test_change_times_invalid2	0 ms
test_change_times_start_nofinish	1 ms
test_change_zip_invalid	0 ms
test_change_zipcode_success	0 ms
test_days_noHours_correct_FN	0 ms
test_days_no_times	0 ms
test_incorrect_FN_LN_email_HP_correct_city	0 ms
test_incorrect_FN_LN_correct_email	0 ms
test_incorrect_FN_LN_email_correct_homephone	1 ms
test_incorrect_FN_correct_LN	0 ms
test_incorrect_FN_correct_Zipcode	1 ms
test_times_no_day	1 ms

Test_assignAccoutSection

Test Results	60 ms
Tests.UnitTests.test_assignAccSection.TestAssignAccS	60 ms
test_assign_already_assigned	12 ms
test_assign_invalid_course	3 ms
test_assign_invalid_section	10 ms
test_assign_invalid_title	2 ms
test_assign_nonexistent_acct	1 ms
test_assign_nonexistent_section	8 ms
test_assign_not_assigned	4 ms
test_assign_success	20 ms

Test_deleteAccount

Test Results	2 ms
Tests.UnitTests.test_deleteAccountCom.TestDeleteAcco	2 ms
test_deleteAccountCom_notFound	0 ms
test_deleteAccountCom_success	2 ms

Test_deleteCourse

Test Results	2 ms
Tests.UnitTests.test_deleteCourseCom.TestDeleteAccor	2 ms
test_deleteCourseCom_notfound	1 ms
test_deleteCourseCom_success	1 ms

test_createCourse

Test Results	2 ms
Tests.UnitTests.test_createCourse.Test_CreateCourse	2 ms
test_course_successfully_created	1 ms
test_course_with_name_exists	0 ms
test_create_course_already_exists	0 ms
test_create_course_invalid_courseNum	0 ms
test_invalid_location	1 ms

Test_deleteAccount

Test Results	57 ms
Tests.UnitTests.test_createAccount.TestCreateAccoun	57 ms
test_account_already_exist	0 ms
test_account_already_exists_post	24 ms
test_account_successfully_created	2 ms
test_invalid_email	0 ms
test_invalid_email_post	11 ms
test_invalid_title	0 ms
test_invalid_title_post	10 ms
test_success_post	10 ms

TEST ANALYSIS

Acceptance Tests:

Def test_sendOutNotification_ins_success:

When a user attempts to send an a message to an instructor they should get some sort of feedback saying that their message was sent. This test fails because this feature was not implemented in out project. In order to get it to pass we would need to write a function to send out messages are create a webpage that would allow users to send messages.

Def test_sendOutNotification_ins_not_success:

When a user attempts to send an a message to an instructor that does not exist, they should get some sort of feedback saying that their message was not sent. This test fails because this feature was not implemented in out project. In order to get it to pass we would need to write a function to send out messages are create a webpage that would allow users to send messages.

Def test_sendOutNotification_ta_success:

When a user attempts to send an a message to a TA they should get some sort of feedback saying that their message was sent. This test fails because this feature was not implemented in out project. In order to get it to pass we would need to write a function to send out messages are create a webpage that would allow users to send messages.

Def test_sendOutNotification_ta_not_success:

When a user attempts to send an a message to a TA that does not exist, they should get some sort of feedback saying that their message was not sent. This test fails because this feature was not implemented in out project. In order to get it to pass we would need to write a function to send out messages are create a webpage that would allow users to send messages.

LIST OF COMPLETE PBIs

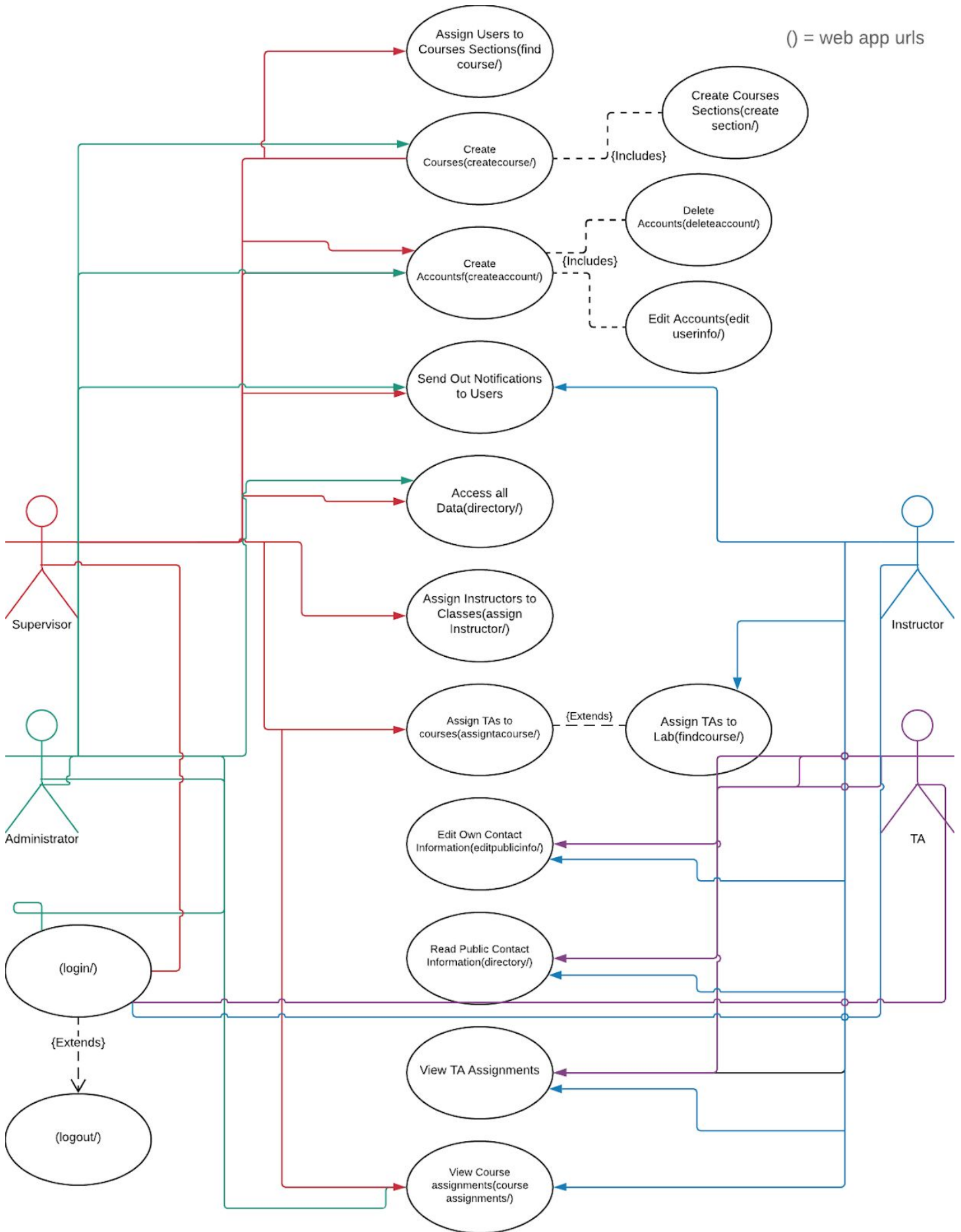
PBI	Acceptance Criteria
Assign TAs to Courses	<p>The supervisor can assign TAs to courses so that the instructors can assign them to labs</p> <ul style="list-style-type: none"> Warning given if there is assignment conflicts
Logout	<p>A user should be able to logout so that another user can log in</p> <ul style="list-style-type: none"> Should not be able to log out if no account is logged in Log out command
Assign TAs to Labs	<p>Instructors can assign TAs to labs so that there is someone to teach the labs</p> <ul style="list-style-type: none"> Instructor can't be assigned to labs Lab section must be exist Assign a TA to a particular Lab section
Assign Instructors to Courses	<p>The supervisor can assign instructors to classes so that there is someone to teach classes</p> <ul style="list-style-type: none"> Assign Instructor to course command Ability to assign an instructor to a course Feedback if the course trying to be assigned already exists
View Course Assignments	<p>Instructors can view course assignments so that they know which classes they are assigned to</p> <ul style="list-style-type: none"> Display the Courses and or labs assigned to an Account
Login	<p>A user should be able to login to the system so they can execute commands</p> <ul style="list-style-type: none"> Login command Ability to enter username and password Feedback if password or username is incorrect Should not be able to access any commands until a User is logged in
Create Courses	<p>The supervisor and administrator can create courses so that TAs can be assigned to them</p> <ul style="list-style-type: none"> Ability to enter the necessary information needed to create a course Feedback if the course trying to be created already exists A create course command
Create Labs	<p>Supervisors and administrators can create labs associated with a course so that TAs can be assigned to them by instructors</p> <ul style="list-style-type: none"> Create lab command Feedback if the lab trying to be created

	already exists
Create Accounts	<p>The supervisor and administrator can create accounts for instructors and TAs so that the supervisor and administrator can assign the TAs and instructors to classes and so that information can be entered</p> <ul style="list-style-type: none"> • Feedback if the account trying to be created already exists • Ability to enter the basic info needed to create account • Create Account command • Ability to create a certain type of account
Delete Accounts	<p>The supervisor and administrator can delete accounts for instructors and TAs so that there aren't frivolous accounts floating around wasting space</p> <ul style="list-style-type: none"> • DeleteAccount command • Option to delete an account • Feedback if the account trying to be deleted not exists • Message confirming deletion
Read Public Contact Information	<p>TAs and instructors can read public contact information so that they can contact each other if they need to</p> <ul style="list-style-type: none"> • Search feature to find who they are looking for • View screen
Edit Accounts	<p>Supervisors and administrators should be able to edit accounts for TAs and instructors so that there is always up-to-date information in the system and to be able to account for their errors</p> <ul style="list-style-type: none"> • Ability to change TA/instructor class assignments • Ability to update contact information for instructors and TAs
Edit Own Contact Information	<p>TAs and instructors can edit their own contact information so that up to date information is always available</p> <ul style="list-style-type: none"> • Webpages • View current information • User Functions • Option to edit • Course Functions
Assign TA Section	<p>Assign ta to a section of a course</p> <ul style="list-style-type: none"> • TA cannot be assigned to a section of a course if they are not assigned to that course • TAs can only be assigned to 200 level

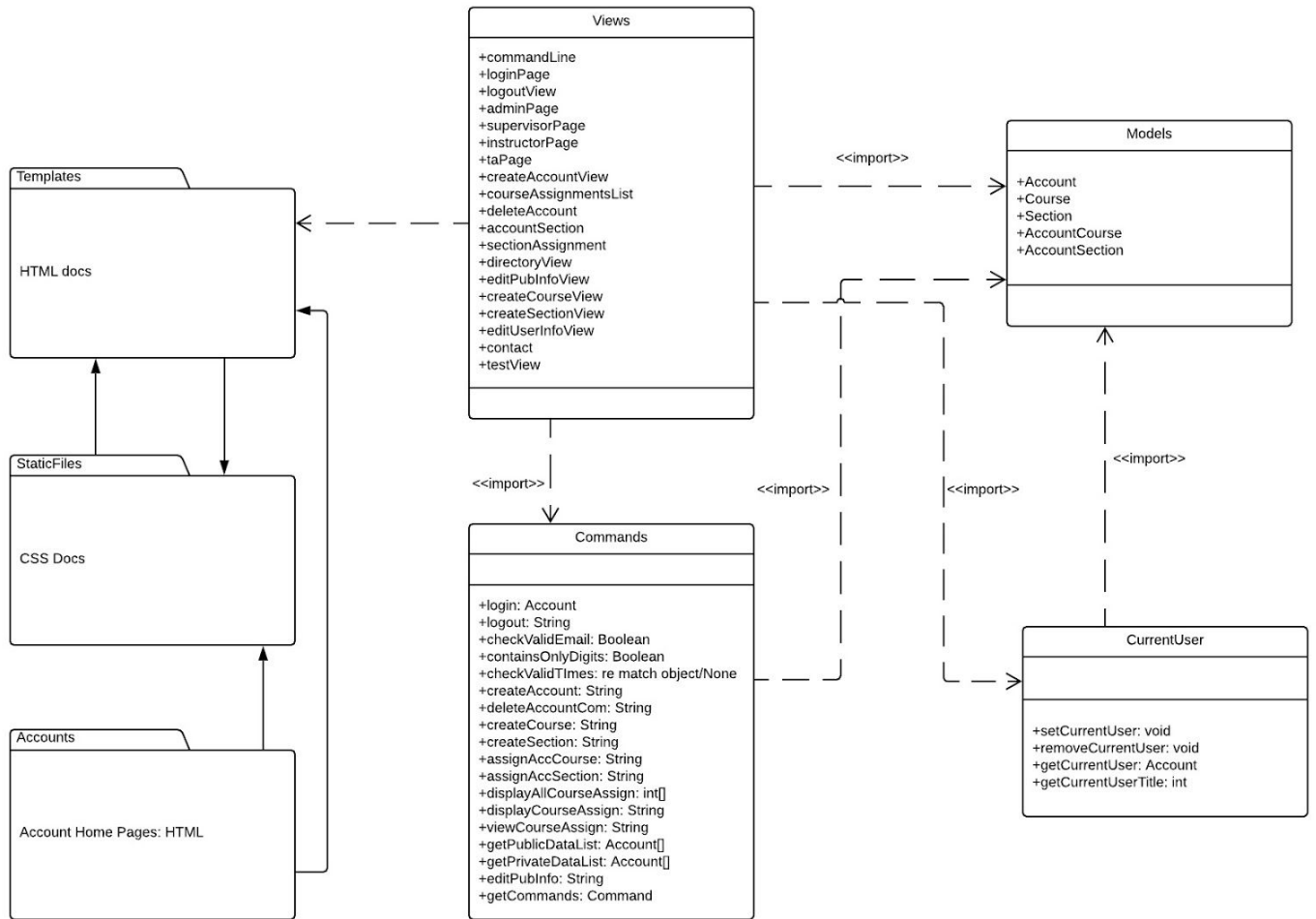
	<p>sections</p> <ul style="list-style-type: none"> • Supervisors can assign TAs of any course to Sections • Instructors can only assign their TAs to their own sections • No more than 1 TA to a section
Assign Instructors to a section	<p>Assign instructor to a section</p> <ul style="list-style-type: none"> • Supervisors can assign instructors to sections • Only one instructors per section • Instructors can only be assigned to 400 level sections
Create Section	<p>Admins/Supervisors can successfully create a section for any course</p> <ul style="list-style-type: none"> • Successfully create a section for any course

UML USE CASE DIAGRAM

() = web app urls



SOFTWARE DESIGN



Commands.py

def login(userName, password):

This function takes in a user name and password as parameters as provided by a user through the login page. The current user is set by finding the account in the database using the user name. If the account is found, the password is checked to see if it matches the one saved for that account. If the password is not correct, an exception is raised. If the account was never found, an exception is raised. If the login was successful, the current user is returned.

def logout(user):

This function takes in an account as a parameter. The account's currentUser flag is set to false. If everything is successful, a String is returned to express this. However, if the account does not exist in the database, an exception will be caught and a String is returned. If there are multiple accounts returned, an exception is caught and a String is returned to communicate this.

def checkValidEmail(email):

This function takes in an email as a parameter and splits it starting at the @ symbol. It then verifies that the substring after @ is "uwm.edu", the only valid ending for a UWM email address. The function returns a boolean based on this.

def containsOnlyDigits(argument):

This function checks to make sure that the argument passed in contains only digits by using re.match and an appropriate regex to test against. This function returns a boolean based on the outcome of re.match.

def checkValidTimes(time):

This function uses re.match to ensure that a time passed in as a parameter is valid military time. This means that the time begins with a 0 or 1, followed by any digit [0-9], any digit [0-5], and any digit [0-9]. If the time begins with a 2, the only digits allowed for the second index are [0-3]. This function returns a re.match object or None.

def checkValidDays(days):

This function checks the validity of the days passed in as a parameter. Days are represented as a String. First, all spaces are removed. Then, the function checks to make sure that the String passed in only contains valid characters contained in "MTWRFN", "N" referring to a course or section that is online. The function returns a boolean based on the result of this check. If an invalid character is found, False is returned.

def createAccount(firstName, lastName, userName, title, email):

This function creates a new account in the database based on the information provided as parameters, which contains the base minimum information required to set up a user. Initially, the function searches the database using the username to ensure the account does not already exist. If it passes this test, the email is checked using checkValidEmail() to ensure it is valid. If this passes, the function creates a new account object, verifies that title is valid, and fills the account's fields. A temporary password is created and saved to the database. An appropriate String message is returned after each failed test and at the end.

def deleteAccountCom(userName):

This function takes in a username as a parameter, then attempts to retrieve an account with that username. If the account is not found, an error message is displayed. If the account is found, it is deleted and a success message is returned.

def deleteCourseCom(courseName):

This function takes in a course name as a parameter, then attempts to retrieve a course with that name. If the course is not found, an error message is displayed. If the course is found, it is deleted and a success message is returned.

def createCourse(name, number, online):

This function takes in a course name, number, and whether the course is online or on campus as arguments. First it checks that the course number is valid (3 digits and numeric). If it fails, an error message is returned. Then the function checks if the course already exists by checking the number and the name. Error messages are returned if the course already exists. If all of the tests pass, a new course is created and added to the database. A success message is returned.

def createSection(courseNumber, type, sectionNumber, days, start, end):

This function takes in the information needed to create a section as parameters. First the course number is checked to make sure it is valid. If not, it returns an error message. Then the function tries to retrieve the course. If it fails, it returns an error message. The type is then verified. If it is a "0", it is a lab section. The course number must then be 3 digits long, numeric, and beginning with a 2. If the type is "1", it is a lecture section. The course number must then be 3 digits long, numeric, and begin with a 4. If it fails this test, an error message is returned. The function makes sure that a section being added to an online course is a lecture and not a lab. Days are checked to make sure they are valid in MTWRF or N if it is online. The start and end times are checked to make sure they are valid and the start time comes before the end time. If the section already exists, an error is returned. Otherwise, the section is created.

def assignAccCourse(userName, courseName):

This function takes in a username and course name and checks to see if these exist in the database. If they do not, an error is printed. The course is retrieved. If the user is already assigned to the course, an error message is returned. Next, the title is checked. The title must be 1 or 2. If it is not, an error message is returned. If these tests pass, the assignment is created.

def assignAccSection(userName, courseNumber, sectionNumber):

This function takes in a username, course number, and section number as parameters. It retrieves the account. If it does not exist, an error message is returned. The title is checked to make sure the user is an instructor or TA. Then the course is retrieved if it exists. The function checks to make sure that the user is already assigned to the course (a user must be assigned to a course before they can be assigned to a section of said course). Then the section is retrieved from the database. The function checks to make sure that an instructor is being assigned to a lecture and that a TA is being assigned to a lab section. Finally, the function checks to see if the section was already assigned. If it was, an error message is returned. Otherwise, the assignment is made.

def displayAllCourseAssign():

The function gets a list of all the courses in the database. A new list is created that contains the result of calling displayCourseAssign(courseNumber) for each course. This new list is returned.

def displayCourseAssign(courseNumber):

The function takes a course number as a parameter and retrieves the course. A string is created with the course name concatenated with " CS", the course number, and a new line. Then a list of all of the account-course assignments is created for that course. A list is created of all of the accounts in those assignments that are instructors and they are added to the returned string. A list is created of all of the accounts that are TAs and they are also added to the returned string. Then lists are created for the lab and lecture sections and they are added to the string, if they exist. Finally, this function returns the string created.

def viewCourseAssign(userName):

This function receives a username as a parameter. First, the account is retrieved or an error message is displayed. Lists of the courses and sections that user is assigned to are created and concatenated with the user's name. This string is returned.

def getPublicDataList():

This function creates a list of all instructors, a list of all TAs, and then combines those into a list. Then a list is created by appending the result of calling displayPublic() on each account. The list is sorted, then returned.

def getPrivateDataList():

This function does the same as the previous, except calls displayPrivate() on each account.

def editPubInfo(user, dict):

This function takes in a user and a dictionary as parameters. The value of each key in the dictionary is validated. If there is an error, a message is added to a list. Each value is handled individually so that valid changes can happen whilst invalid edits are handled separately. Finally, the function makes all of the updates and saves the account. It returns the errors or a success message if no errors occur.

def getCommands()

This function returns a list of the commands for the command line interface.

CurrentUser

def setCurrentUser(self, account, request):

This function takes in an account and request as parameters and sets the current user to the account's username.

def removeCurrentUser(self, request):

This function takes in a request as parameters and deletes the current user.

def getCurrentUser(self, request):

This function takes in a request as a parameter and retrieves the account with the matching username from the database. If unsuccessful, an error message is displayed.

def getCurrentUserTitle(self, request):

This function takes in a request as a parameter. The current user is retrieved with getCurrentUser(). If a current user exists, their title is returned.

def getTemplate(self, request):

This function takes in a request as a parameter and retrieves the current user's title. It then returns the corresponding template with the appropriate menu and header.

instructorCourse(View):**def get(self, request):**

The title of the current user is verified. If no one is signed in, a message displays saying the user must log in. If the person trying to access the page is not a supervisor, an error message is displayed stating the user doesn't have permission to view the page. A list of instructors and a list of courses are made. The template for the page to assign an instructor to a course is rendered.

def post(self, request):

The username of the instructor being assigned and the course they are being assigned to are retrieved from post. `assignAccCourse()` is called and the result is displayed on the page.

taCourse(View):**def get(self, request):**

The title of the current user is verified. If no one is signed in, a message displays saying the user must log in. If the person trying to access the page is not a supervisor, an error message is displayed stating the user doesn't have permission to view the page. A list of TAs and a list of courses are made. The template for the page to assign a TA to a course is rendered, including these lists for the form.

def post(self, request):

The username of the TA being assigned and the course they are being assigned to are retrieved from post. `assignAccCourse()` is called and the result is displayed on the page.

commandLine(View):**def get(self, request):**

The command line interface is rendered.

def post(self, request):

The command is retrieved from post. The response from calling `command()` with the input is displayed on the screen.

def redirect_login(request):

The user is redirected to the login page.

loginPage(View):**def get(self, request):**

The current user is removed and the login screen is rendered.

def post(self, request):

The username and password are retrieved from post. login() is called with this information, the user is set, and their title is retrieved. The function redirects to the appropriate page based on the user's title. If it is unsuccessful, an error is displayed.

logoutView(View):**def get(self, request):**

The current user is logged out and removed. The logout page is rendered with the appropriate message returned from logout().

adminPage(View):**def get(self, request):**

The current user, their title, and base template are retrieved. If no one is logged in, an error page is displayed to instruct the user to login. If the user is not an admin, an error message is displayed. Otherwise, the admin's home page is rendered.

supervisorPage(View):**def get(self, request):**

The current user, their title, and base template are retrieved. If no one is logged in, an error page is displayed to instruct the user to login. If the user is not a supervisor, an error message is displayed. Otherwise, the supervisor home page is rendered.

instructorPage(View):**def get(self, request):**

The current user, their title, and base template are retrieved. If no one is logged in, an error page is displayed to instruct the user to login. If the user is not an instructor, an error message is displayed. Otherwise, the instructor's home page is rendered.

taPage(View):**def get(self, request):**

The current user, their title, and base template are retrieved. If no one is logged in, an error page is displayed to instruct the user to login. If the user is not a TA, an error message is displayed. Otherwise, the TA home page is rendered.

createAccountView(View):**def get(self, request):**

The current user and their title are retrieved. If no one is logged in, an error page is displayed to instruct the user to login. If the user is not a supervisor or admin, an error page is displayed. Otherwise, the create account page is rendered with the appropriate base.

def post(self, request):

The current user and base template are retrieved. The username, first name, last name, email, and title for a new account are retrieved from post. The page is rendered with the return value from createAccount().

courseAssignmentsList(View):

def get(self, request):

The current user and title are retrieved. If no one is logged in, an error page is rendered. Lists of sections, courses, account to course assignments, and account to section assignments are created. The page to view all course assignments is rendered with these lists.

deleteAccount(View):

def get(self, request):

The current user, their title, and base template are retrieved. A list is made of all instructors and TAs. These lists are combined. If no one is logged in or the user is not an admin or supervisor, an error page is rendered. Otherwise, the delete account page is rendered using the list of instructors and TAs.

def post(self, request):

The current user is retrieved, and the username of the account to delete is retrieved from post. An updated list is created. The delete account page is rendered with the returned string from deleteAccountCom() and the updated account list.

accountSection(View):

def get(self, request):

The current user, title, and base template are retrieved. If the current user is a supervisor, a list of all courses is created (they can make assignments for all courses). If the current user is an instructor, a list of all of the courses that instructor is assigned to is created (instructors can assign the TAs for their own courses only). If the current user is not one of these, an error page is rendered. A page is rendered allowing the user to choose the course they want to make assignments for.

def post(self, request):

The course name is retrieved from post. The course, current user, current user's title, and base template are retrieved. If the user is a supervisor, a list of all accounts assigned to the chosen course and a list of all sections in the course are created. If the user is an instructor, the account list only contains the TAs assigned to the course and the section list only contains lab sections. The assign section page is rendered using the account and section lists.

sectionAssignment(View):

def post(self, request):

The current user is retrieved. The account name, section, and coursename are retrieved from post. The correct account, course, and section are found using the retrieved information. The assign section page is rendered using the return string from assignAccSection().

directoryView(View):

def get(self, request):

The current user and title are retrieved. If no one is logged in, an error page is rendered. Lists of TAs and instructors are created, then combined in one list. The directory page is rendered using this list and the appropriate base template.

editPubInfoView(View):

def get(self, request):

The current user, title, and base template are retrieved. If no one is logged in, an error page is rendered. Otherwise, the page to edit public information is rendered.

def post(self, request):

A dictionary is created with all of the account information retrieved from post. The user is retrieved using the username. A new page is rendered with the result of editPubInfo() and the updated user information.

def makeUserDictionary(user):

This function creates a dictionary of all of a given user's information.

createCourseView(View):

def get(self, request):

The current user and their title are retrieved. If no one is logged in or the user is not a supervisor or admin, an appropriate error page is rendered. Otherwise, the create course page is rendered with the appropriate base template.

def post(self, request):

The current user is retrieved. The course name, number, and whether it is online or on campus is retrieved from post. The create course page is rendered with the returned string from createCourse() and the appropriate base template.

createSectionView(View):

def get(self, request):

The current user, title, and list of all courses is retrieved. If the user is not logged in or is not a supervisor or admin, the appropriate error page is rendered. Otherwise, the create section page is rendered with the list of courses and appropriate base template.

def post(self, request):

The current user and list of all courses is retrieved. The course name, section type, number, meeting days, start and end times are retrieved from post. The create section page is rendered with the message returned from createSection() and the appropriate base template.

editUserInfoView(View):

def get(self, request):

The current user and title are retrieved. If no one is logged in or the user is not a supervisor or admin, the appropriate error page is displayed. A list of all instructors and TAs is made. The edit user information page is rendered with the list of accounts and the appropriate base template.

def post(self, request):

The account is retrieved with the username from post. makeUserDictionary() creates a dictionary of the updated information. The edit public information page is rendered with the dictionary and appropriate base template.

contact(View):

def get(self, request):

If a current user exists, it is removed and the contact page is rendered.

def post(self, request):

The contact page is rendered.

testView(View):

def get(self, request):

The current user and title are retrieved. A test “create account” page is rendered with the user and appropriate base template.

deleteCourseView(View):

def get(self, request):

The current user, title, base template, and a list of all courses are retrieved. If no one is logged in or the user is not a supervisor or admin, the appropriate error page is rendered. Otherwise, the delete course page is rendered with the course list and base template.

def post(self, request):

The current user, updated course list, and base template are retrieved. The course name is retrieved from post. The delete course page is rendered with the message returned from deleteCourseCom() and updated course list.

Account(models.Model):

def displayTitle(self):

This function returns the string “TA” or “Instructor” depending on what the account’s title is.

def displayPrivate(self):

This function returns a string containing all of the account’s information.

def displayPublic(self):

This function returns a string containing all public contact information for the account.