AcaStat Team Report (NiddyGriddy)

Mark Doan, David Krotzer, Kristopher Honetschlager, Steeff Weber-Gonzales

CS 410-01: Software Engineering

Chi-Cheng Lin

5/3/2024

Table of Content

Product description, a two to three pages “marketing” material, including

- ### Introduction

Welcome to AcaStat, an educational platform for students to enroll and take classes taught by instructors of varying disciplines. This app features announcements, course catalogue and registration system, grade calculation, planner system, assignment system, and more. Once you’re added by one of our admins, you’re successfully able to use the system. Enroll in one of our offered courses and experience the clean and seamless quality our learning platform has to offer. Instructors of our platform enjoy the ability to create announcements and assignments for their classes. Admins of our system register new users and courses.

- ### Features:

  - #### Bulleted items of descriptions

- Login system: A fully integrated login system for users held within the database to be logged in.
- Home page: a home for all courses taken by the user and varying tools depending on the user’s role. Admins have admin tools like create new user and course, while students and instructors do not.
- Create new user system: A full system for admins to create new users of the system and add them to the database. Each new user has various details assigned to them from this page, users are created by role.
- Create new course system: A system for admins to create new courses, assigning an instructor to the course to be populated in the course catalogue for all users to see and register
- Course catalogue system: A system for all users to view available courses and register for them, registered courses appear on the home page and courses dropdown.
- Grade calculator page: A calculator for seeing what grade you need to get on varying assignments to get a good grade! Also a view for grades which was not implemented
- **Calendar and Planner Page:** The CalenderInterface contains a grid that represents days with each grid having a single month tied to it. Each box within the grid may contain information. The PlannerInterface contains a column of information designated by each individual course.
- **Course Page:** The CourseInterface is a visual representation of a course page that contains assignments, a time management interface, and a gradebook

- **Announcements:** Announcements section of course page created by instructor of the course and viewable to all users registered for the course. Instructors of courses can create new announcements for all enrolled users to see with the announcements creation view available via the create announcement button on the respective course page
- **Assignments:** The course page contains a button that will take the user to an assignment page and instructors to the same page with the option to add an assignment for enrolled users which will be populated on the same page
- Submission page: this page, accessible by clicking on an assignment via the assignment page will allow users to submit the assignment.

    - Features that the system should have, not what have been completed

- Provide an online learning platform that
    - Enrolls new students/teachers in enrolled courses
    - Created from the student perspective
    - Simplifies and centralizes class material
        - Reduces course page tabs
        - Better calendar for managing assignments/due dates/planning
        - Makes concurrent enrollment enmeshed
    - Allows dropbox functionality for submitting assignments
    - Allows for video calls for class time
- Offer a learning platform for students to
    - enhance their student experience/save time
    - easily navigate their course-load
- Provide a framework for Teachers to
    - Accurately represent their courses on the platform
    - Make announcements, assignments, and unified grades
- Allow universities to
    - Have an unobtrusive but beneficial presence on the platform
    - Host/promote extra-curricular activities

- ## Screenshots

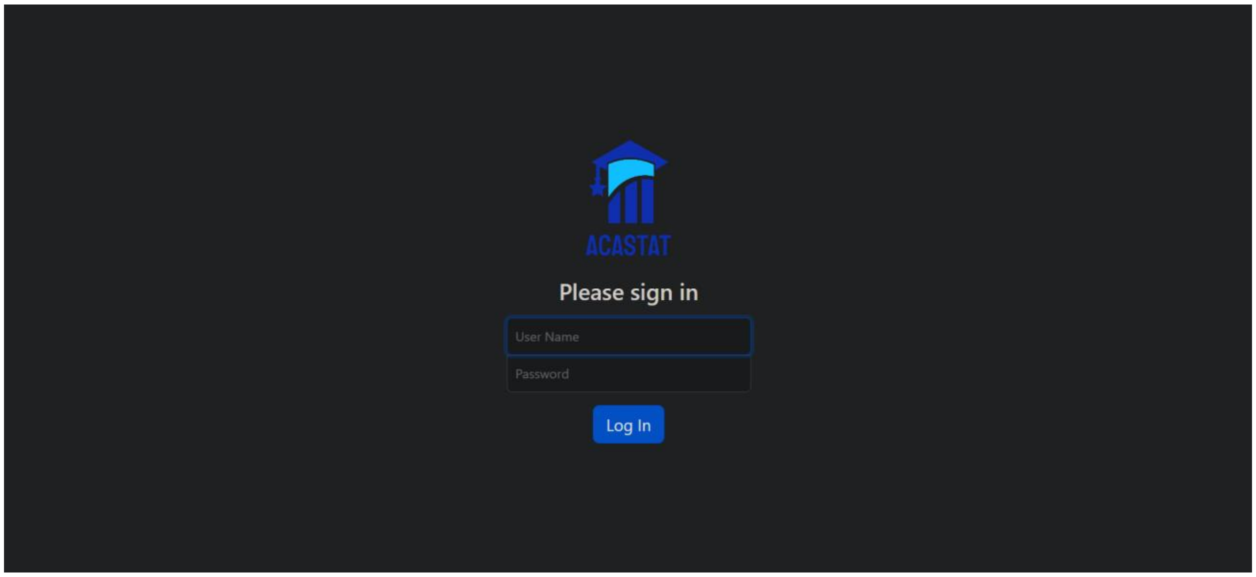
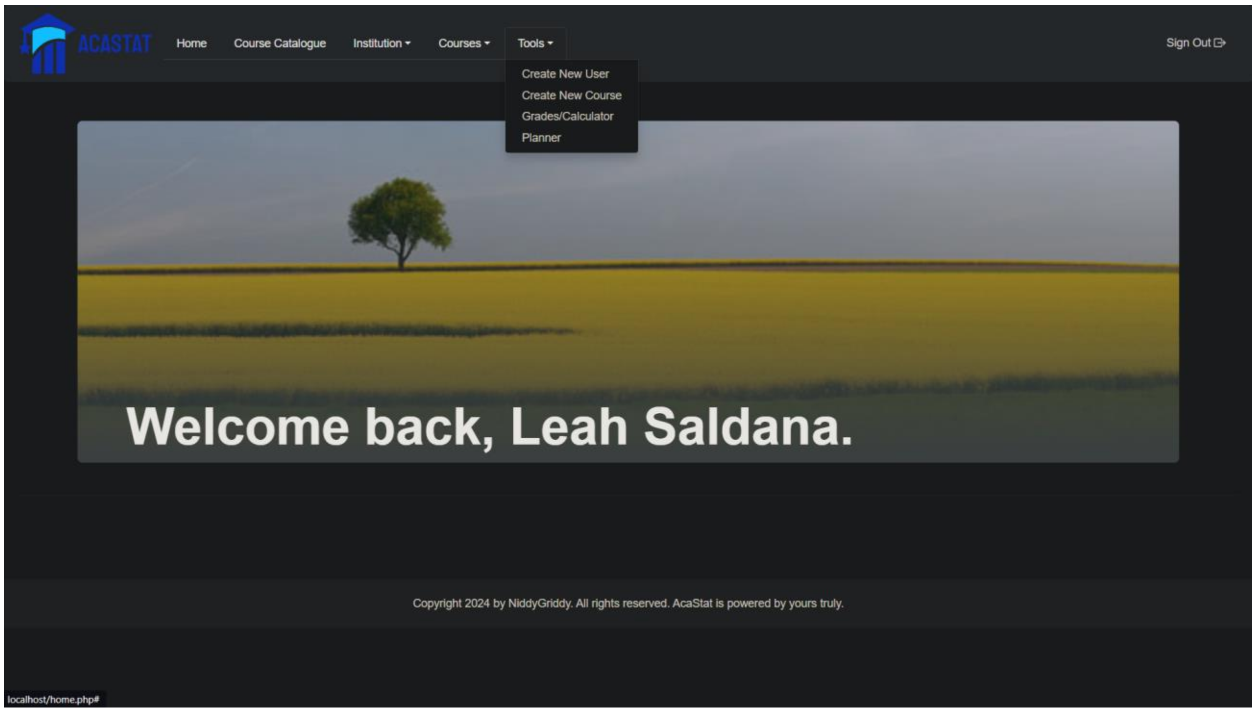


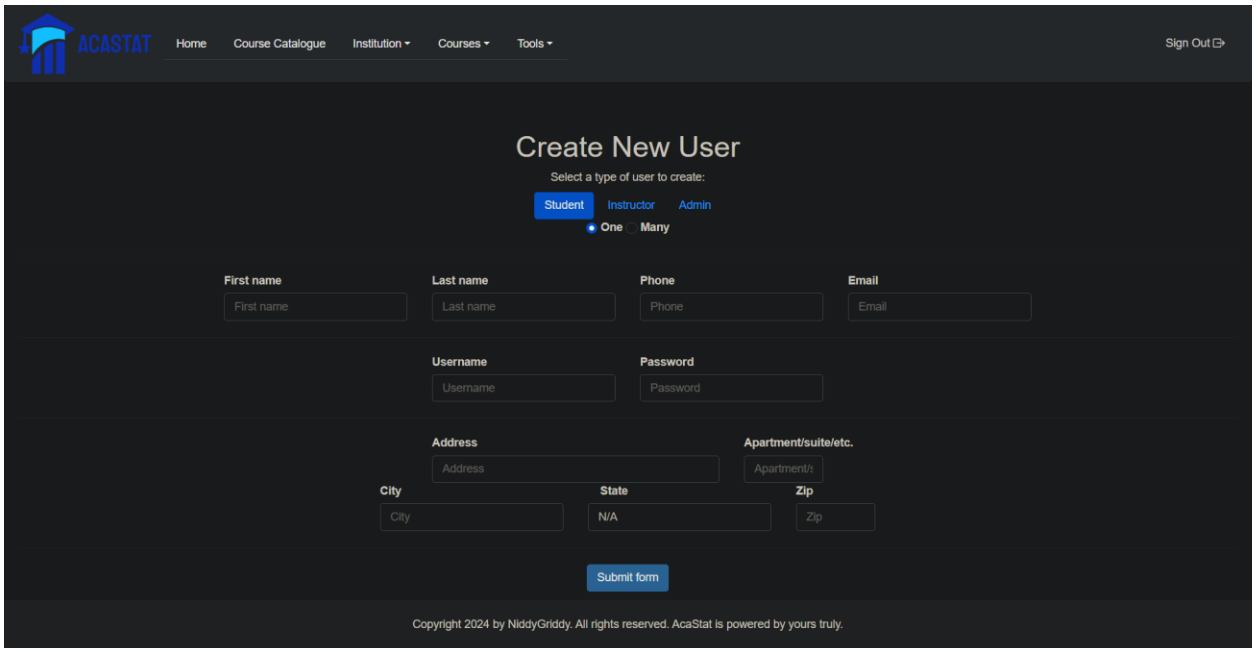
Figure 1 Login Screen (All Users)



Welcome back, Leah Saldana.

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 2 Home Page (Admins)



Create New User

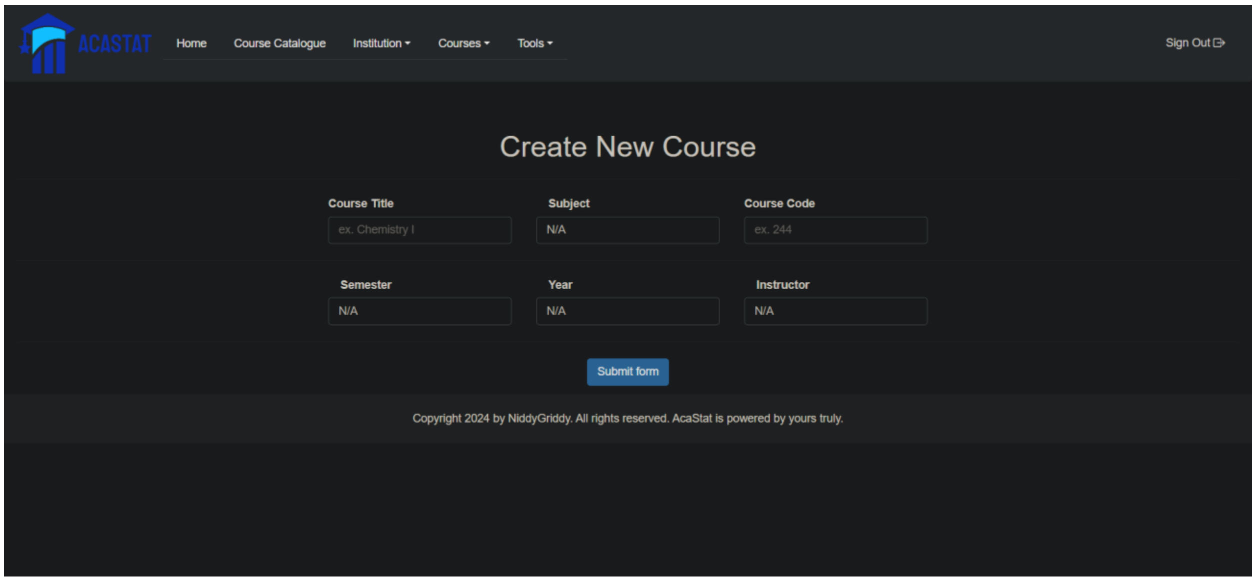
Select a type of user to create:

[✓ One] Many

First name	Last name	Phone	Email
First name	Last name	Phone	Email
Username	Password		
Username	Password		
Address	Apartment/suite/etc.		
Address	Apartment/s		
City	State	Zip	
City	N/A	Zip	

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 3 Create User (Admin)



Create New Course

Course Title	Subject	Course Code
ex. Chemistry I	N/A	ex. 244
Semester	Year	Instructor
N/A	N/A	N/A

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 4 Create Course (Admin)

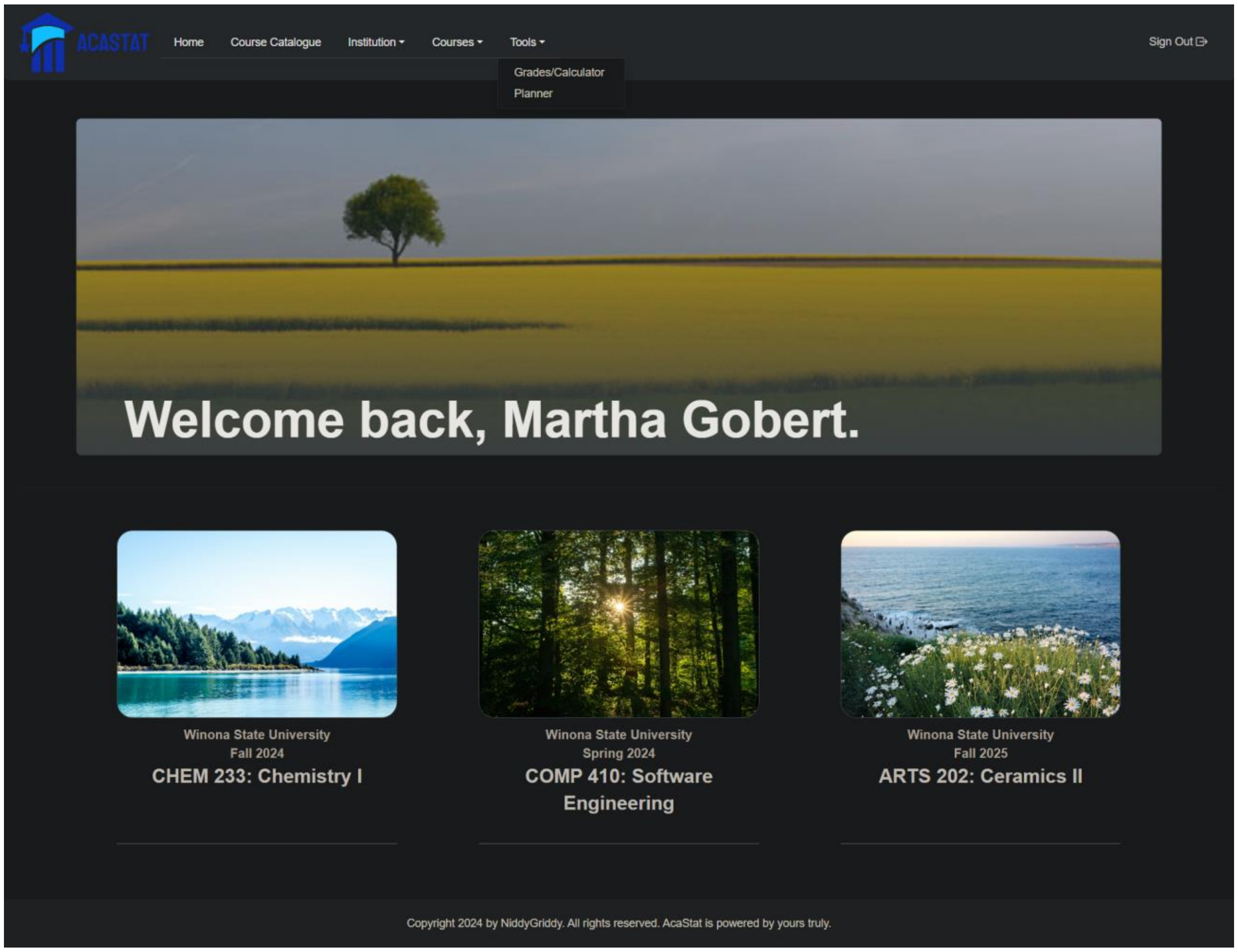
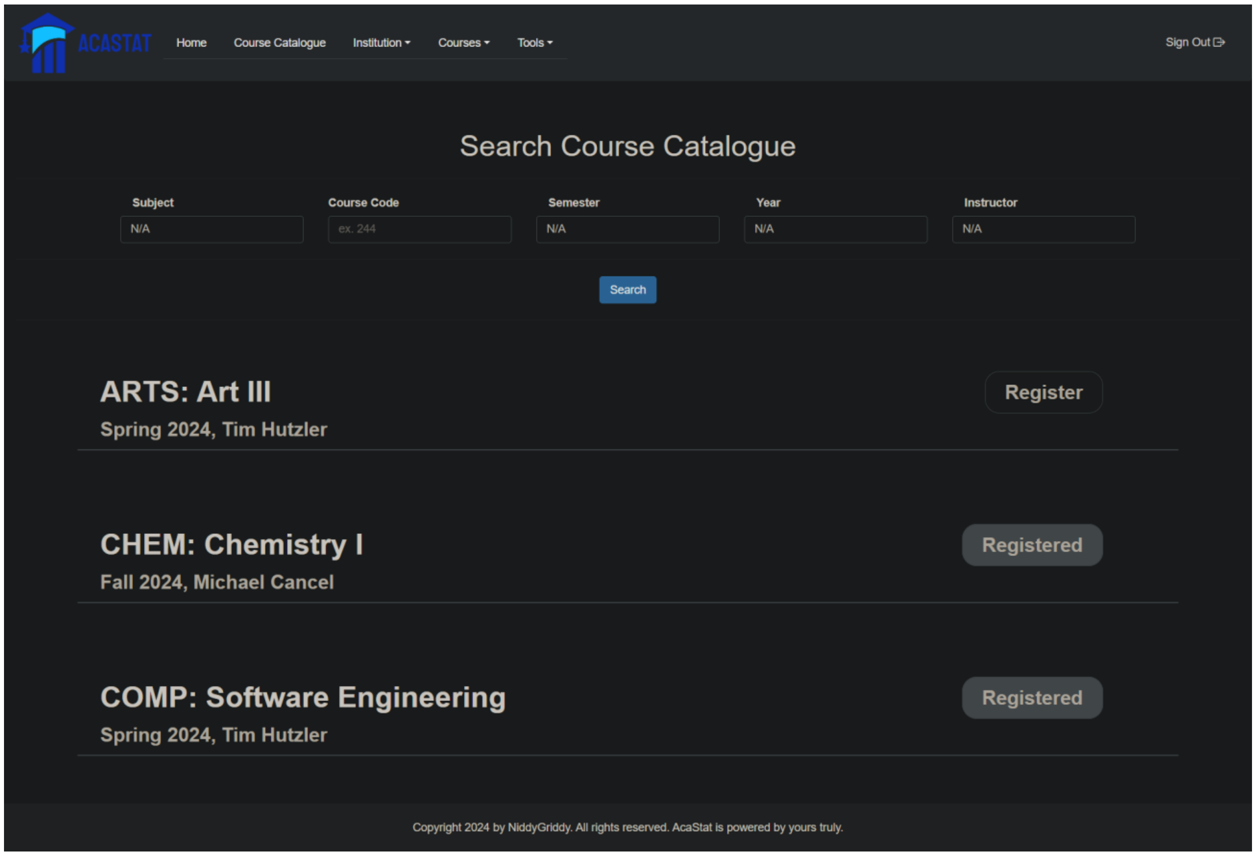


Figure 5 Home Page (Students/Instructors) [Note: Students and Instructors have same access to tools]



Search Course Catalogue

Subject	Course Code	Semester	Year	Instructor
N/A	ex. 244	N/A	N/A	N/A

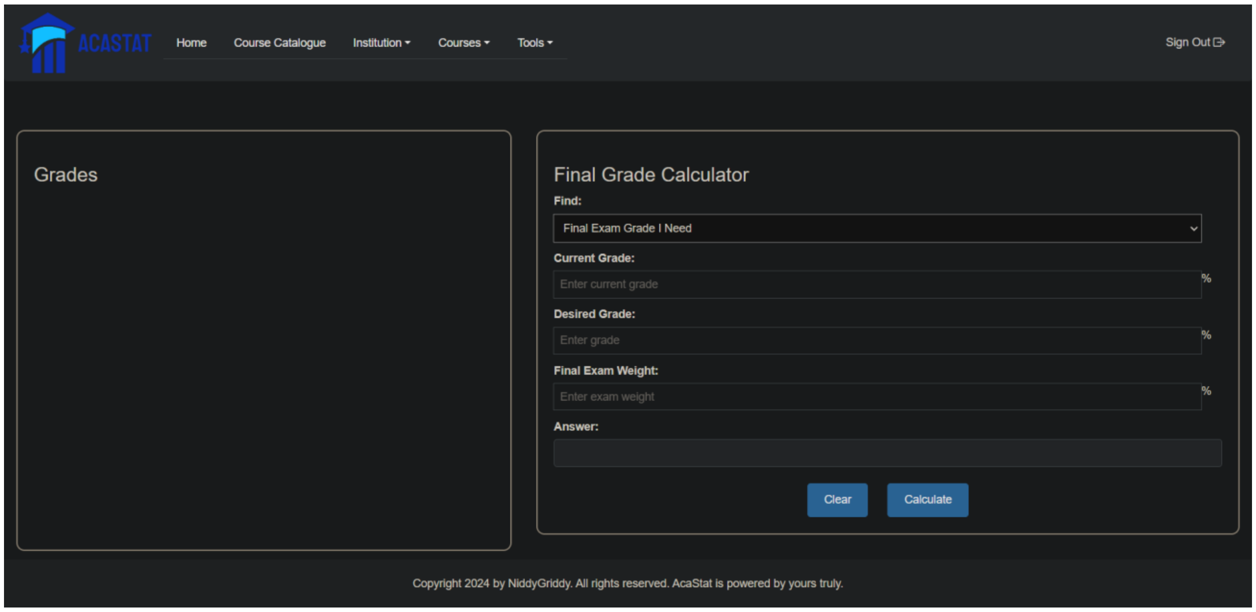
ARTS: Art III
Spring 2024, Tim Hutzler

CHEM: Chemistry I
Fall 2024, Michael Cancel

COMP: Software Engineering
Spring 2024, Tim Hutzler

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 6 Course Catalogue (All Users)



Grades

Final Grade Calculator

Find: Final Exam Grade I Need

Current Grade: Enter current grade %

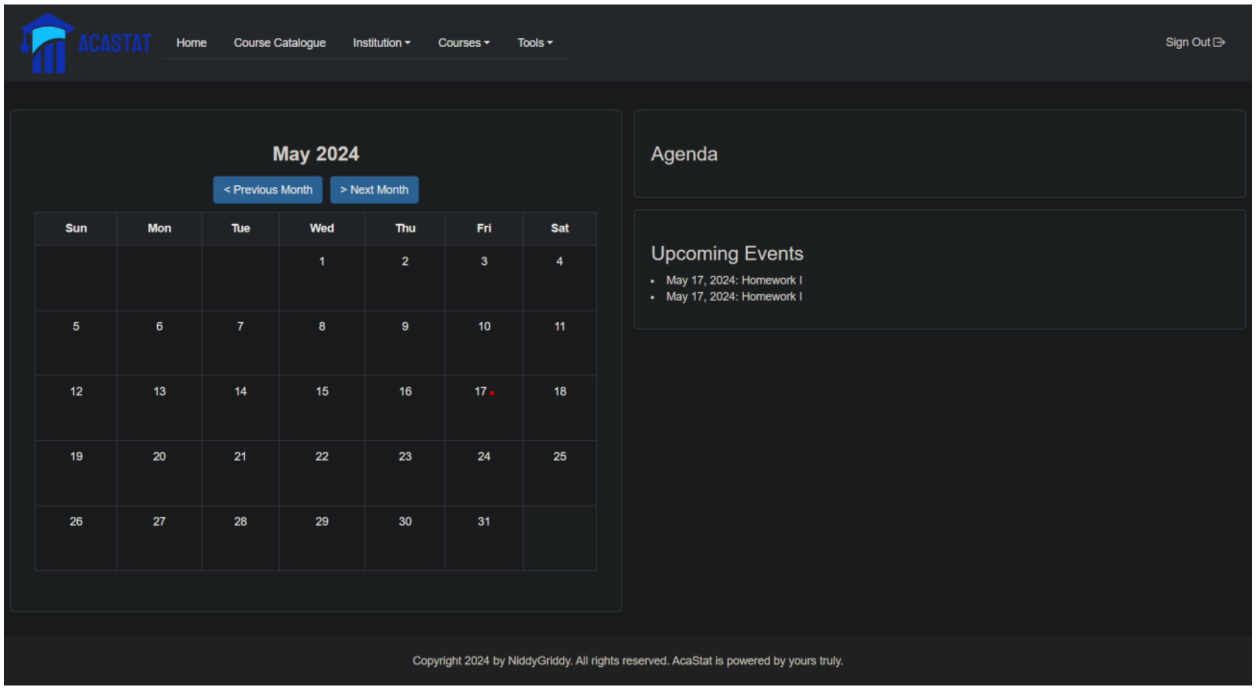
Desired Grade: Enter grade %

Final Exam Weight: Enter exam weight %

Answer:

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 7 Grade Calculator (All Users)



May 2024

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17 ●	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

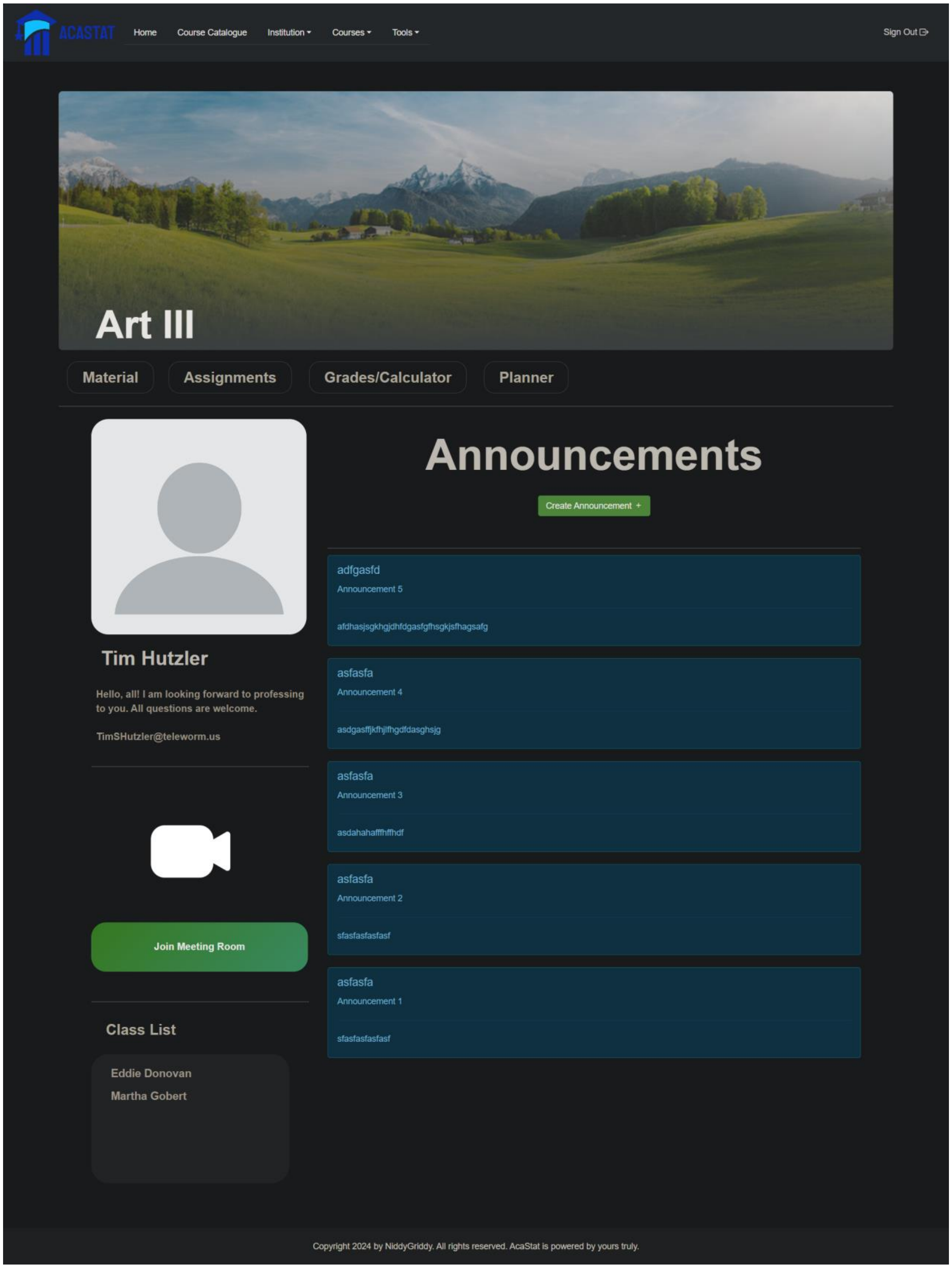
Agenda

Upcoming Events

- May 17, 2024: Homework I
- May 17, 2024: Homework I

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 8 Planner (All Users)



ACASTAT Home Course Catalogue Institution Courses Tools Sign Out

Art III

Material Assignments Grades/Calculator Planner

Announcements

Create Announcement +

adfgasfd
Announcement 5
afdhasjsgkhgjdhfdgasfgfhsgkjsfhagsafg

asfasfa
Announcement 4
asdgasffjkfhjlfhgdfdasghsjg

asfasfa
Announcement 3
asdahahafffhffhdf

asfasfa
Announcement 2
sfasfasfasfasf

asfasfa
Announcement 1
sfasfasfasfasf

Tim Hutzler

Hello, all! I am looking forward to professing to you. All questions are welcome.

TimSHutzler@teleworm.us

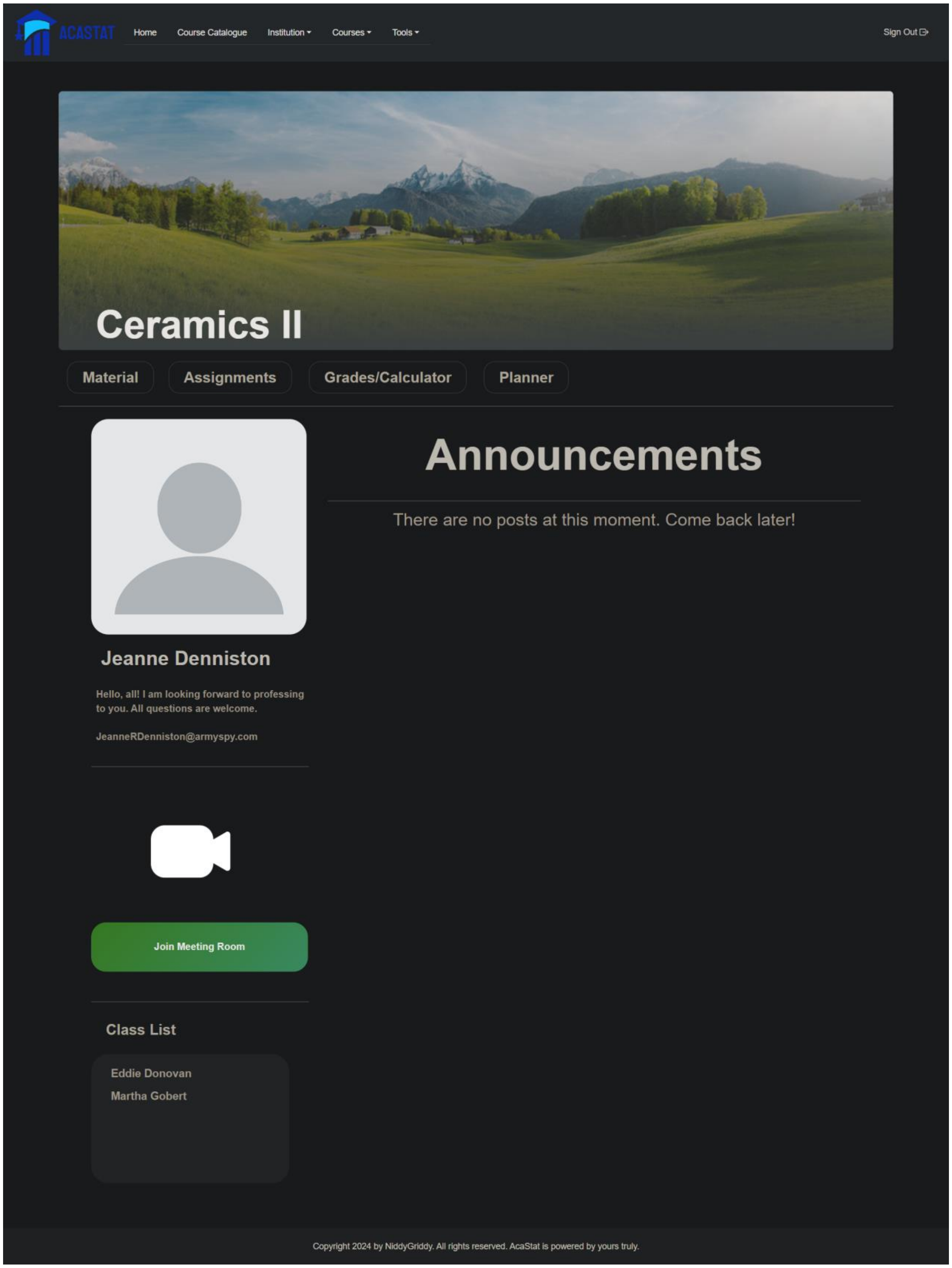
Join Meeting Room

Class List

Eddie Donovan
Martha Gobert

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 9 Course Page (Instructors) [Varies by Course and Create Announcement only appears on courses taught by logged in instructor]



Ceramics II

Announcements

There are no posts at this moment. Come back later!

Jeanne Denniston

Hello, all! I am looking forward to professing to you. All questions are welcome.

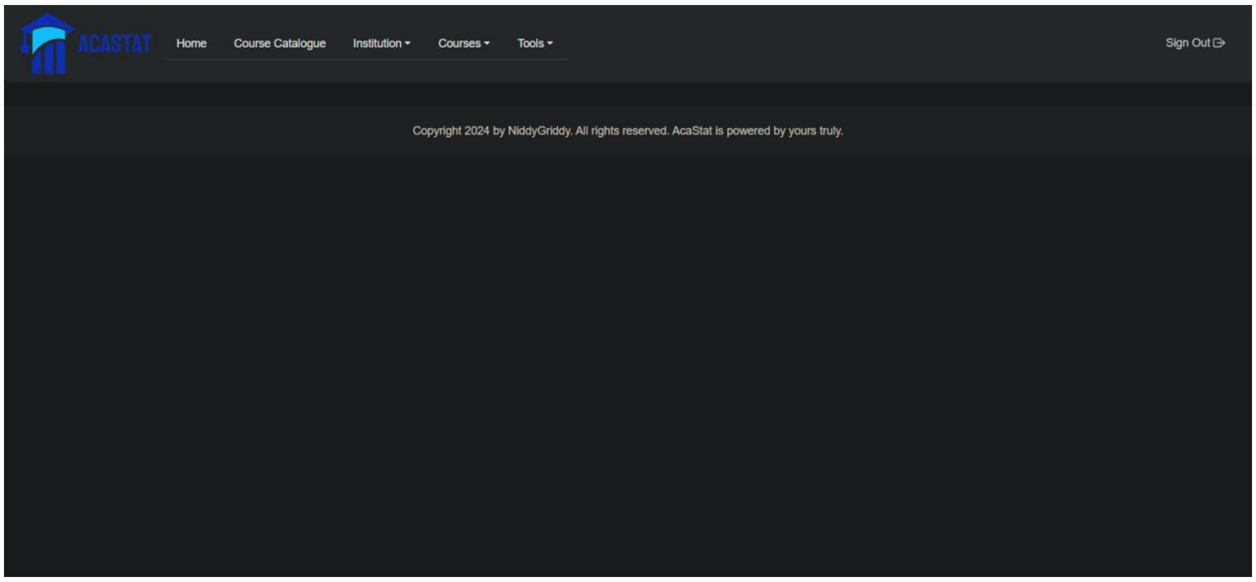
JeanneRDenniston@armyspy.com

Class List

Eddie Donovan
Martha Gobert

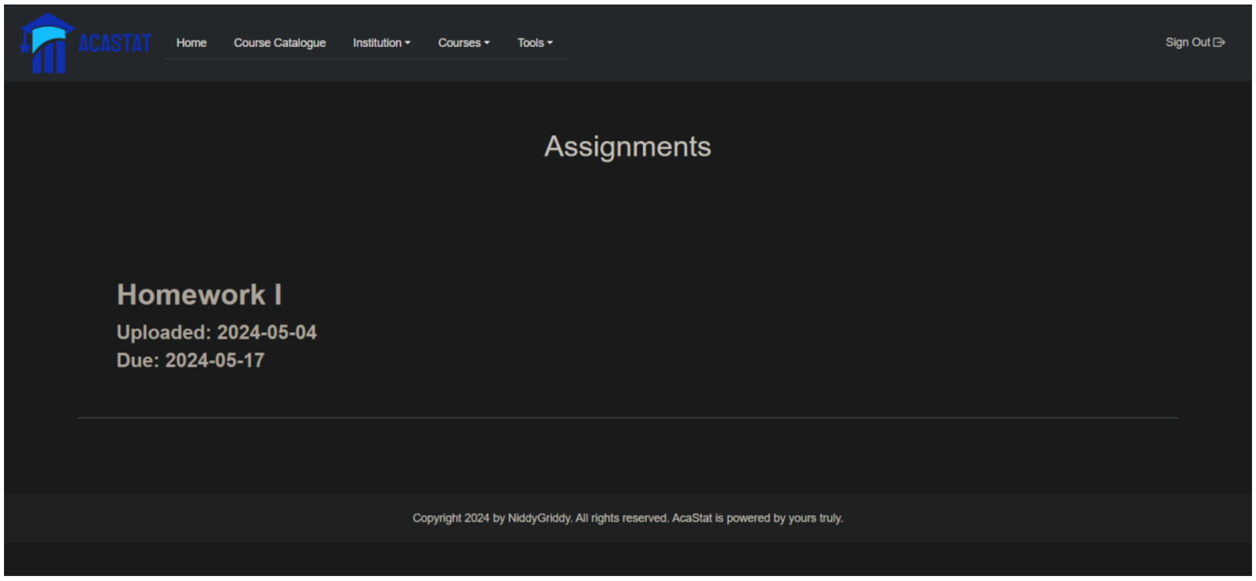
Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 10 Course Page (Students) [Varies by Course]



Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 11 Material (Students)

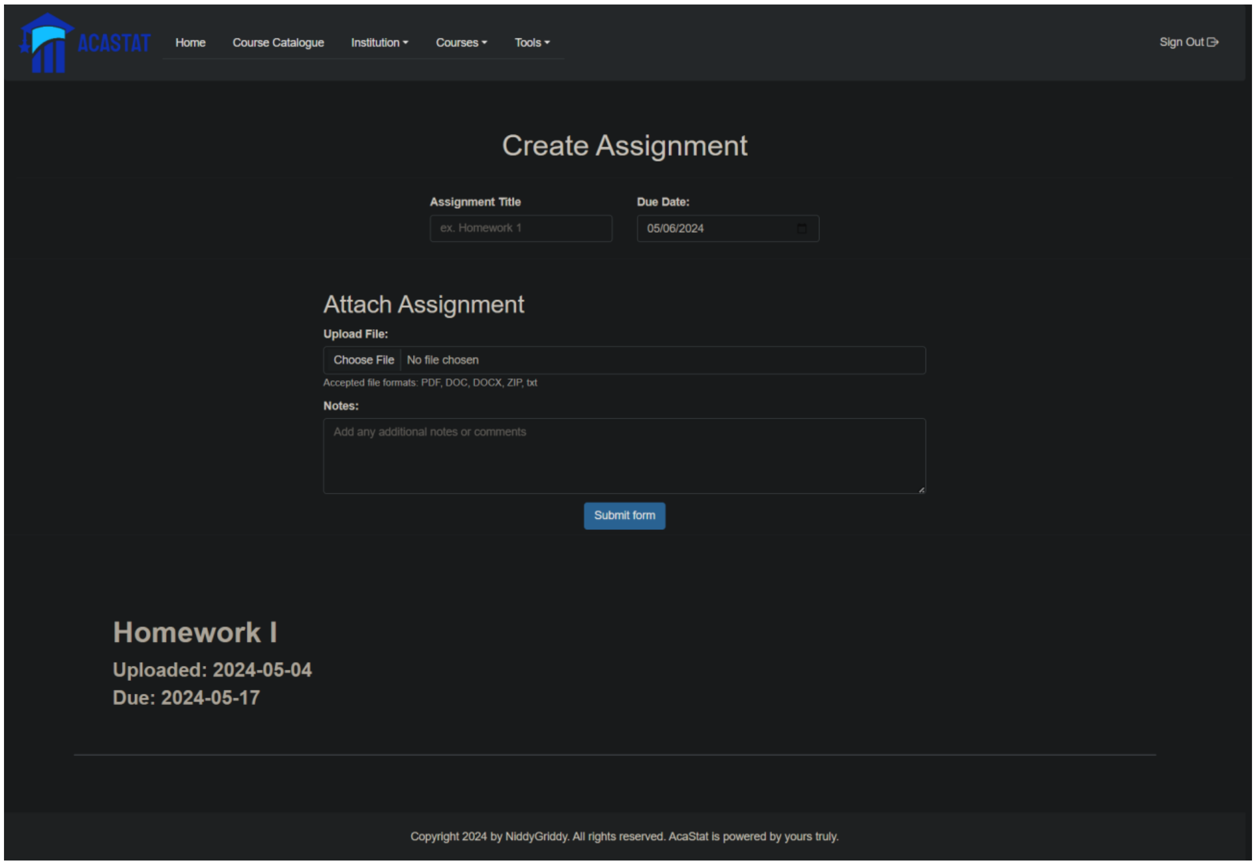


Assignments

Homework I
Uploaded: 2024-05-04
Due: 2024-05-17

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 12 Assignments (Students)



Create Assignment

Assignment Title: ex. Homework 1

Due Date: 05/06/2024

Attach Assignment

Upload File: Choose File No file chosen

Accepted file formats: PDF, DOC, DOCX, ZIP, txt

Notes: Add any additional notes or comments

Submit form

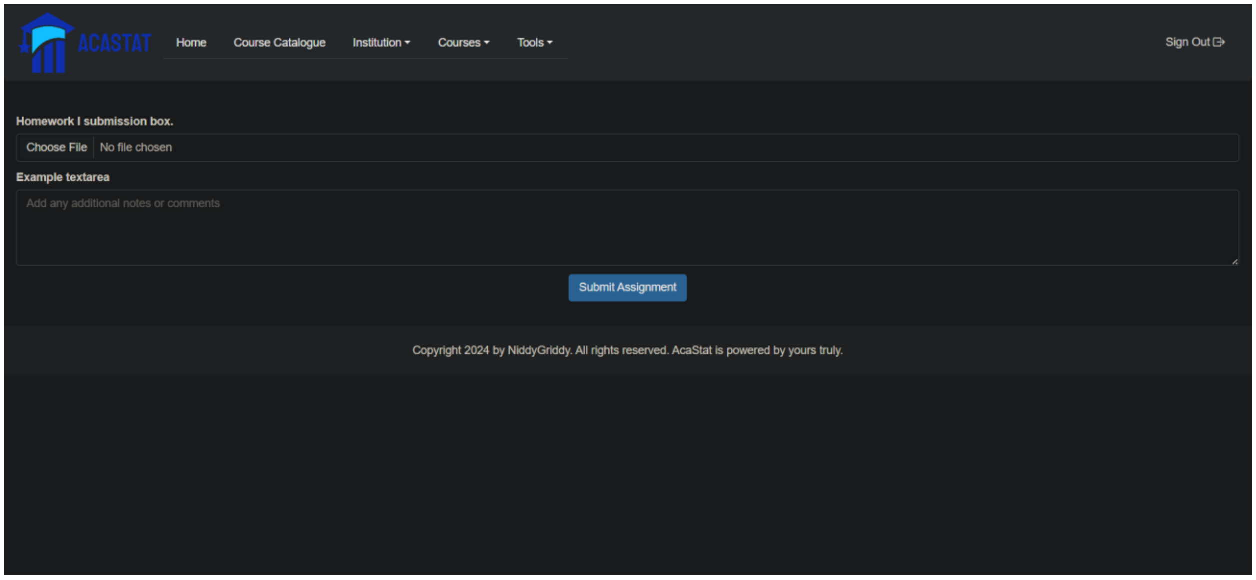
Homework I

Uploaded: 2024-05-04

Due: 2024-05-17

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 13 Assignment (Instructor)



Homework I submission box.

Choose File No file chosen

Example textarea

Add any additional notes or comments

Submit Assignment

Copyright 2024 by NiddyGriddy. All rights reserved. AcaStat is powered by yours truly.

Figure 14 Submission (Students)

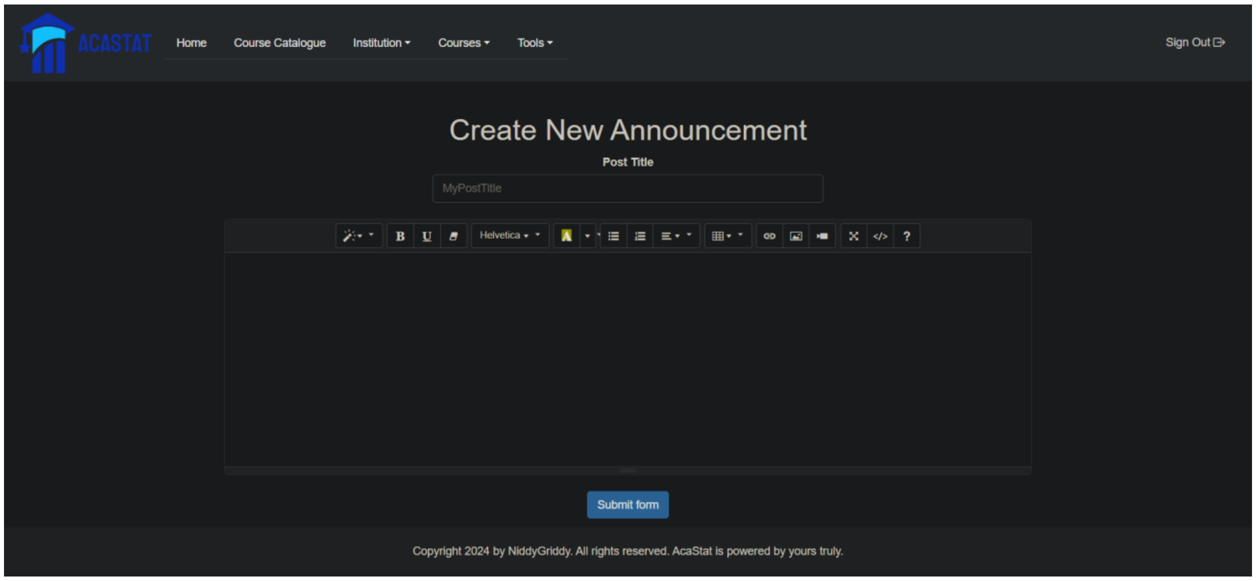


Figure 15 Create Announcement (Instructor)

Project documentation/artifacts

- ### Problem statement

The problem to be solved: (Kris)

The widespread accessibility of the internet and quick internet speeds has meant that more people can access online content on demand. One specific use for this speed and accessibility has been online classes. Teachers could post their content online, especially if the content is otherwise difficult to find, for students to easily access.

There's been a recent rise and offloading of students onto online platforms as opposed to in-person classes for the sake of accessibility and convenience, potentially also lesser cost. This has meant for the increasing need for easy-to-use centralized learning platforms. Given that learning platforms are a recent development in scholarly technology, they aren't necessarily as streamlined as they could be.

Important information can be cryptic, inconsistent, and scattered when it comes to these online learning platforms. Having an even easier means of accessing resources related to academic planning and course related material would be beneficial for students.

Description of current situations: (Mark)

- Popular educational platforms, although offering a centralized area to access course resources, allow for inconsistency with how users of the platform (dominantly instructors) format and display information, creating a noncohesive and messy platform. Along with this, many additional features or online tools are offered by existing platforms but are not fully developed to be useful or competitive with external tools.

• System architecture description, including the architectural style

Target Environment: (Steeff)

Our architectural style is model view controller, the one Apple uses

It's intended to be multi-system via a web app

- The environment in which the delivered system has to perform a specified set of system tests:
    - User-Focused Testing:
        - UI Response Times
            - Commands perform in less than 2 milliseconds in high traffic.
        - Security Vulnerabilities
            - Firewalls and intrusion detection.
        - Data Accuracy
            - Measuring effectiveness of learning strategies and tools.
    - Software Compatibility:
        - Chrome, Firefox, Safari, Opera, Edge
    - System Testing:
        - Testing on devices such as desktops, laptops, and mobile phones.
        - Stress testing for a surge in user activity such as exams.

• Specification

○ Functional and non-functional requirements

- Functional requirements:
    - Authentication – Users must be able to create accounts using a username and password and include account recovery measures.
        - Teacher accounts must be able to provide class info such as grades and exam dates.

        - Student accounts must be able to see their class info provided by the teachers.
    - Management – Admins must be able to manage accounts and perform administrative tasks.
    - Features – Users must be able to use features such as a grade calculator and a scheduler.
- Nonfunctional requirements:
    - Performance – The system must perform commands flawlessly and within time expectations such as no more than 2 milliseconds to calculate grades.
    - Security – The system must securely store all personal information and grades and encrypt this data to ensure privacy.
    - Scalability – The system must support an ever-growing user base starting with 5,000 concurrent users.

Availability – The website should be available 24/7 with exceptions to scheduled maintenance of one hour a week. In short, uptime must be 99.9 percent

Use cases, scenario, user stories, etc.

Use Cases:

PostAssignment (Instructor) (Steeff)

Use Case Name: PostAssignment
Description: Instructor posts an assignment on the site.
• *Participating Actors: Instructor (primary),* AcaStat
Flow of Events (With Numbered Steps): 1. *Instructor logs in to the site.* 2. *Instructor clicks on the appropriate class option.* 3. *Instructor clicks on "Post Assignment" through a dropdown menu.* 4. *Instructor provides details on the assignment such as a title, due date, and extra comments.* 5. *Instructor adds an appropriate link or media.* 6. *Instructor clicks on "Submit" at the bottom of the page.* 7. AcaStat *returns a successful confirmation message.*
Entry Conditions: Instructor logs into system successfully.
Exit Conditions: The assignment is successfully posted.
Alternative Flows (With Numbered Steps): 1. *Instructor navigates to submitted posted assignment.* 2. *Instructor clicks on an edit option.* 3. *Instructor changes details or media on the assignment.* 4. *Instructor clicks on "Save Changes".*
Exception Flows (With Numbered Steps):

1. *Instructor attempts to add an assignment with the same name as another posted assignment.* 2. AcaStat *notifies the instructor to change the assignment name as it is already taken.*

SubmitAssignment (student) (Steeff)

Use Case Name: SubmitAssignment
Description: Student submits an assignment on the site.
Participating Actors: Student (primary), AcaStat
Flow of Events (With Numbered Steps): 1. *Student logs in to the site.* 2. *Student clicks on the appropriate class option.* 3. *Student selects the assignment.* 4. *Student clicks on a "Submit Assignment" button.* 5. *Student uploads the appropriate file(s).* 6. *Student adds any additional comments.* 7. *Student clicks on "Submit" at the bottom of the page.* 8. AcaStat *returns a successful confirmation message.*
Entry Conditions: Student logs into system successfully.
Exit Conditions: The assignment is successfully submitted.
Alternative Flows (With Numbered Steps): 1. *Student navigate to submitted assignment.* 2. *Student clicks on a "Resubmit Assignment" button.* 3. *Student submits a revised assignment.*
Exception Flows (With Numbered Steps): 1. *Student attempt to submit an assignment past the due date.* 2. AcaStat *greys out the "Resubmit Assignment" button preventing the student from adding anything.*

CheckGrade (student) (Mark)

Use Case Name: CheckGrade
Description: Student launches grade page/s to view grades.
Participating Actors: - *Initiated by Student* - *Communicates with System*
Flow of Events (With Numbered Steps): 1. *Student* navigates the software to desired/relevant course. 2. *AcaStat* responds by displaying the appropriate interface for all steps. 3. *Student* uses 'Grade' interface to launch Grade page. 4. *System* displays grade page related to previous course page.

Entry Conditions: The *Student* is logged into *AcaStat*
Exit Conditions: - The desired grade page is fully loaded. - The appropriate error message is displayed if *AcaStat* is unable to load the page.
Alternative Flows (With Numbered Steps):
Exception Flows (With Numbered Steps):
Quality Requirements: - Pages should load within 1-2 seconds on user interaction. - Appropriate grade page is loaded according to selected course. - All grade information need be up to date to what is uploaded by the *Instructor* and properly formatted by *AcaStat*

SubmitGrade(Instructor) (Mark)

Use Case Name: SubmitGrade
Description: Instructor submits, edits, and/or removes grade on *AcaStat* for grade archival and displaying to the *Student*.
Participating Actors: - Initiated by *Instructor* - *Instructor* communicates with *AcaStat* - *Instructor* communicates with *Student* through *Acastat (?)*
Flow of Events (With Numbered Steps): 1. *Instructor* navigates to *Student* gradebook through AcaStat, navigating to the desired course. 2. *AcaStat* displays the gradebook from the chosen course. 3. *Instructor* selects gradebook of desired Student 4. *AcaStat* displays the gradebook of desired Student. 5. *Instructor* inputs grade/grade points determined by *Student* performance and grading method. 6. *Instructor* selects the 'Submit' button to save changes to the gradebook. 7. AcaStat sends and stores changes to database and displays updated information on the gradebook.
Entry Conditions: - *Instructor* is logged into *AcaStat*
Exit Conditions: - *AcaStat* displays a message communicating to Instructor that the submission is successful. - The appropriate error message is displayed if *AcaStat* is unable to load the page. - The appropriate error message is displayed if *AcaStat* is unable to save the submitted information.
Alternative Flows (With Numbered Steps): 1. *Instructor* navigates to *Student* gradebook through AcaStat, navigating to the desired course.

2. *AcaStat* displays the gradebook from the chosen course. *3.1. Instructor* selects gradebook of desired assignment *4.1 AcaStat* displays the gradebook of desired assignment. 5. *Instructor* inputs grade/grade points determined by *Student* performance and grading method. 6. *Instructor* selects the 'Submit' button to save changes to the gradebook. 7. AcaStat sends and stores changes to database and displays updated information on the gradebook.
Exception Flows (With Numbered Steps): *6.1. Instructor clicks 'cancel' or hits brower back button to cancel changes to gradebook. No changes are made and AcaStat displays the previous page.*
Quality Requirements - Pages should load within 1-2 seconds on user interaction. - Appropriate grade page is loaded according to selected course. - All grade information need be up to date to what is uploaded by the *Instructor* and properly formatted by *AcaStat.*

UsePlanner (students and instructors) (Mark)

Use Case Name: UsePlanner
Description: A set of actions that allow users of AcaStat to view, create, edit, and share tasks using the built-in planner.
Participating Actors: - *Student initiates use Planner* - *Instructor initiates use Planner* - *AcaStat communicates with Student* - *AcaStat communicates with Instructor*
Flow of Events (With Numbered Steps): 1. User navigates to *Planner* through AcaStat. 2. AcaStat displays planner. 3. User creates a 'task'. 4. AcaStat prompots the user for title, description, time and/or due date. 5. *User inputs all desired information for task then clicks 'submit' to save.* 6. *AcaStat sends and stores information to database and displays information on the Planner.*
Entry Conditions: - *User is logged into AcaStat.* - *AcaStat is not down.*
Exit Conditions: - *AcaStat* displays a message communicating to the User that the task submission is successful. - The appropriate error message is displayed if *AcaStat* is unable to load the page.

- The appropriate error message is displayed if *AcaStat* is unable to save the submitted information.

Alternative Flows (With Numbered Steps):

1. *User navigates to Planner through AcaStat.*
2. *AcaStat displays planner.*

3.1 User edits task.

3.1.1 AcaStat prompts user for title, description, time, and/or due date.

3.2 User shares task to users.

3.2.1 AcaStat prompts user to attach all desired users to share task with.

4. User inputs all desired information then clicks 'submit' to save.

5. AcaStat sends and stores information to database and displays information on the Planner.

Exception Flows (With Numbered Steps):

5.1. User clicks 'cancel' or hits brower back button to cancel changes to Planner. No changes are made and AcaStat displays the previous page.

Quality Requirements:

- Pages should load within 1-2 seconds on user interaction.
- All Planneri nformation need be up to date to what is uploaded by the *user and sharing users*.

LaunchGradeCalculator (students) (David)

Use Case Name: LaunchGradeCalculator

Description: Students launch the grade calculator, input values into the calculator, submit the inputs and receive respective outputs

Participating Actors: Students

Flow of Events (With Numbered Steps):

1. *Student signs in to AcaStat*
2. *Student launches the grade calculator*
3. *Student inputs various numbers*
4. *AcaStat checks if those inputs are valid*
5. *AcaStat sends back the result(s)*

Entry Conditions:

– *AcaStat is running*
– *Student is signed in*

Exit Conditions:

– *The outputs are sent back and displayed to the student*
– *The inputs were invalid and AcaStat sends back an error message(s)*
– *AcaStat is unresponsive and thus an error message will be sent back*

Alternative Flows (With Numbered Steps):

1. *Student signs in to AcaStat*
2. *Student launches the grade calculator*
3. *Student inputs invalid inputs*

4. *AcaStat checks if those inputs are valid* 5. *AcaStat sends back an error message*
Exception Flows (With Numbered Steps): 1. *AcaStat sends an error message after invalid inputs* 2. *Student closes the grade calculator* 3. *Student closes AcaStat* 4. *AcaStat is no longer responsive*

LaunchToDoList (students) (David)

Use Case Name: LaunchToDoList
Description: Students launch the To-Do list, view the displayed information, expand the information, adjust the viewing timeframe, create a listing, edit a listing, mark a listing complete, and delete a listing.
Participating Actors: Students
Flow of Events (With Numbered Steps): 1. *Student signs into AcaStat* 2. *Student launches the To-Do list* 3. *AcaStat displays the information* 4. *Student expands the information* 5. *Student adjusts the timeframe* 6. *Student creates a listing* 7. *Student edits a listing* 8. *Student marks a listing complete* 9. *Student deletes a listing*
Entry Conditions: – *AcaStat is running* – *Student is signed in*
Exit Conditions: – *The information is displayed to the student* – *The inputs were invalid, and AcaStat sends back an error message(s)* – *AcaStat is unresponsive and thus an error message will be sent back*
Alternative Flows (With Numbered Steps): 1. *Student signs into AcaStat* 2. *Student launches the To-Do List* **3. *AcaStat sends an error message information cannot be displayed*** 4. *Student creates a listing* 5. *Student edits a listing* 6. *Student deletes a listing* **7. *AcaStat sends an error message as listing is unable to be deleted***
Exception Flows (With Numbered Steps): 1. *AcaStat sends an error message as information cannot be displayed* 2. *Student exits the create mode*

3. Student exits the editor mode *4. Student exits the delete a listing action* *5. AcaStat sends an error message as listing is unable to be deleted*

LaunchCoursePage (Instructors/Students) (David)

Use Case Name: LaunchCoursePage
Description: Student and Instructors launch a course page from the main menu, AcaStat displays the selected course page.
Participating Actors: Student, Instructor
Flow of Events (With Numbered Steps): *1. Student/ Instructor sign into AcaStat* *2. Student/ Instructor launch desired course page* *3. AcaStat displays the corresponding course page*
Entry Conditions: *– AcaStat is running* *– Student/ Instructor is signed in*
Exit Conditions: *– The information is displayed to the student* *– AcaStat is unresponsive and thus an error message will be sent back*
Alternative Flows (With Numbered Steps): *1. Student/ Instructor sign into AcaStat* *2. Student/ Instructor launch desired course page* *3. AcaStat is unable to find the course and sends back an error message*
Exception Flows (With Numbered Steps): *1. AcaStat sends an error message as information cannot be displayed* *2. AcaStat is unable to find the course and sends back an error message*

CreateCourse (Admin) -> EnrollInCourse (Admin) -> CSV file of people in course from eServices (Kris)

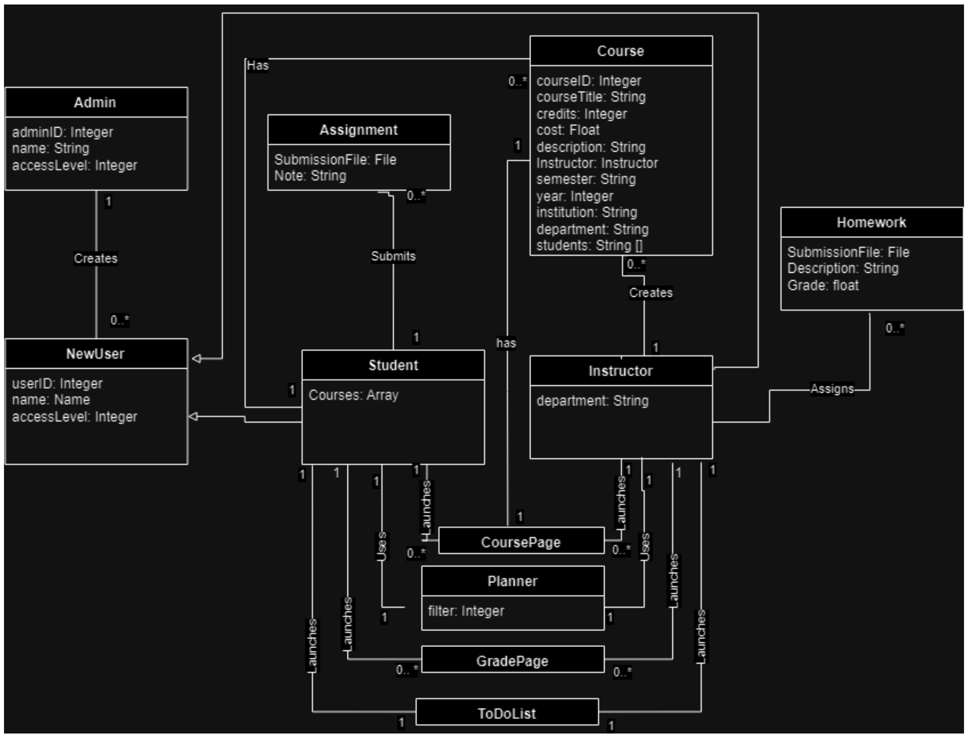
Use Case Name: CreateCourse
Description: Teacher creates a course that they are teaching
Participating Actors: Teachers, Students, Admin
Flow of Events (With Numbered Steps): *1. Teacher logs into system* *2. Teacher clicks settings/manage icon on home screen* *3. System displays settings/manage pop up* *4. Teacher clicks create new class* *5. System displays create new class screen* *6. Teacher enters class title, an image for the course, and the course number* *7. Teacher clicks create course*

8. System displays pop-up that course has been created
Entry Conditions: Teacher logs into system successfully and teacher has course information necessary
Exit Conditions: Course is created successfully or not
Alternative Flows (With Numbered Steps): *1. Teacher exits system* *2. Teacher closes settings window* *3. Teacher closes create class screen*
Exception Flows (With Numbered Steps): *1. Course not created successfully because required information is not filled out, system displays error message*

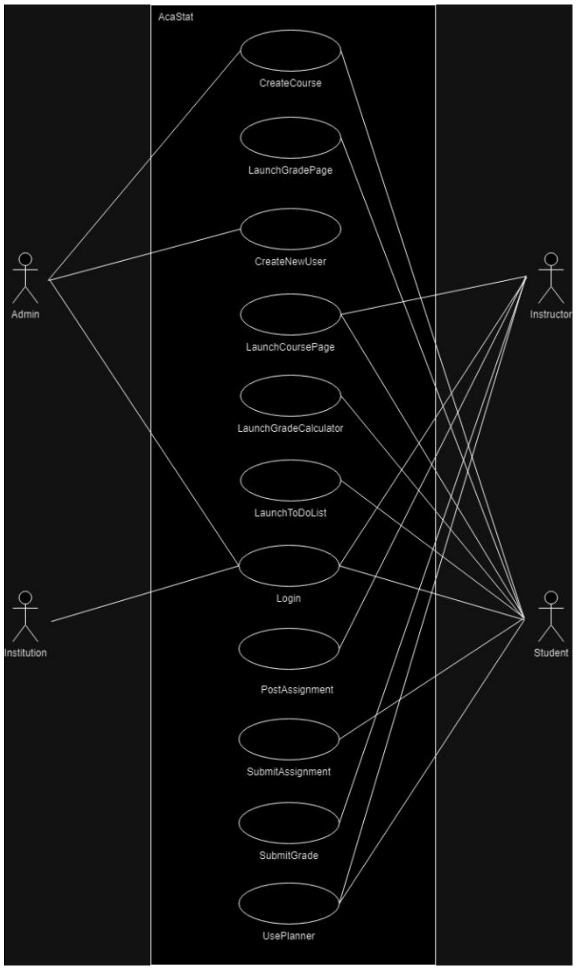
Use Case Name: CreateNewUser
Description: Process to register a new user to the system
Participating Actors: Admin
Flow of Events (With Numbered Steps): *1. The admin receives a list of new student(s) to be admitted to the system from eServices* *2. Admin logs into system (see Login use case)* *3. Admin clicks settings/manage icon on home screen* *4. System displays settings/manage pop up* *5. Admin clicks admit new users (submit a CSV or text file)* *6. A file explorer instance is opened (or OS equivalent)* *7. Admin submits CSV or text file of new users (username, password, and email) (Delimited by new line, space, or commas)* *8. If no errors occur, process is completed and an automated email is sent to new users informing them of their app enrollment* *9. The system displays a notification to the admin that the process completed successfully*
Entry Conditions: The admin has successfully entered the website by logging and has a list of users credentials to enter into the system (formatted properly)
Exit Conditions: The user credentials file is accepted or the user credentials file is not accepted
Alternative Flows (With Numbered Steps): *1. Admin exits website* *2. Admin closes settings* *3. Admin closes file explorer window*
Exception Flows (With Numbered Steps): *1. (see login use case)* *2. Admin submits improperly formatted csv/text file, system displays error message and location of error(s)*

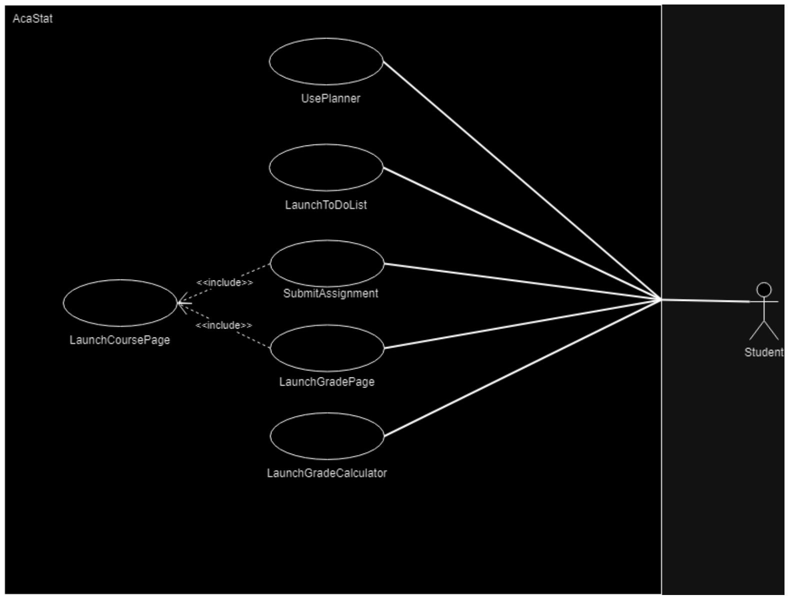
Use Case Name: Login
Description: Login screen confronted by all users when initiating the web program
Participating Actors: Students, Teachers, University, Admin
Flow of Events (With Numbered Steps): *1. Actors go to the website* *2) System responds by showing Login screen* *3. Actors enter credentials assigned to them respectively (username and password)* *4. Actor clicks login, and does if the credentials were proper* *a. System responds by displays home page for the respective actor*
Entry Conditions: The actor accesses the website
Exit Conditions: The actor is logged in OR the actor is not logged in and an error message is shown
Alternative Flows (With Numbered Steps): *1. If user exits the website, nothing happens*
Exception Flows (With Numbered Steps): *1. If username and/or password are improper, user is not signed in and system displays error message* *2. If database goes down, system displays error and user is not signed in* *3. If actor's internet has a problem, website will be inaccessible*

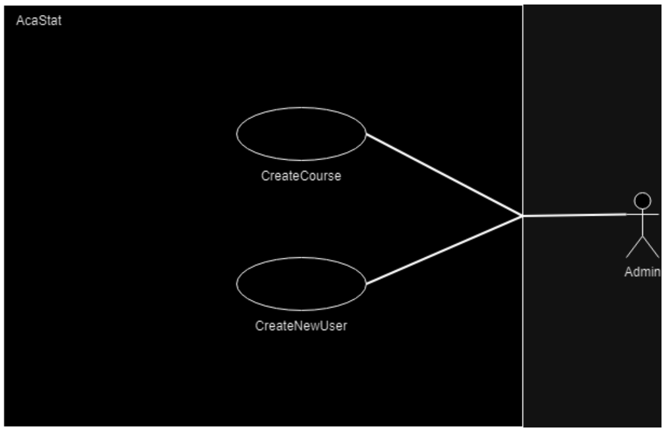
- Analysis and Design
    - Analysis models (UML diagrams)

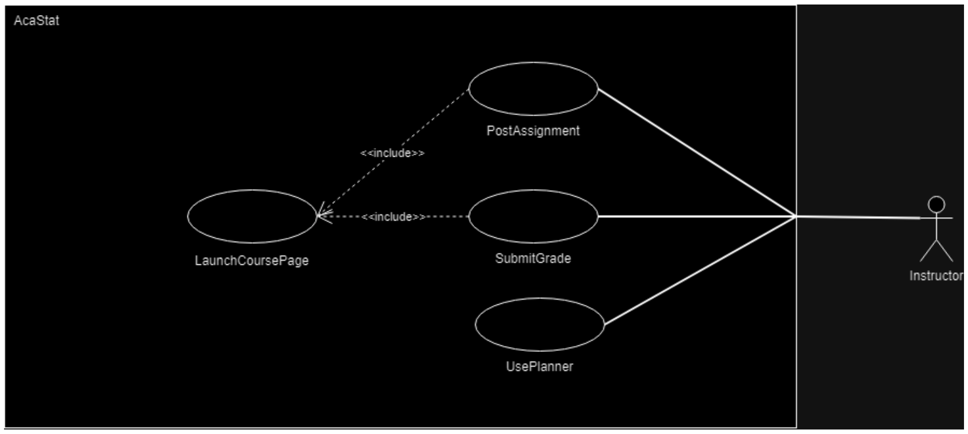


○ Design models (UML diagrams)









- ### Design patterns: patterns used or plan to use in the next delivery

We only intend to implement grades, our design pattern would likely not stray far from the pattern we used to implement the course registration

- ## Implementation (probably in the code)

Implementation is simple, go to this link and use the roles I mention on page 29/30: http://ec2-3-67-134-137.eu-central-1.compute.amazonaws.com/login.php

I, Kris, will continue to host the database until further notice.

My individual report details the implementation slightly better. Basically we just implemented using the WAMPP stack, the EC2 server is hosted on a windows Amazon AWS EC2 instance, the database is and RDS instance from Amazon AWS, we interface with the database via PHP, the webpages are programmed with a combination of PHP, JS, and CSS. We used bootstrap for the majority of styling and made sure to separate the JS, PHP, and CSS from the major web pages which are on the root of the project.

If you want to install XAMPP locally to view what the server is doing and what our files contain, follow these instructions (feel free to let me know if anything goes wrong):

```
#########################################################################
#####
#FOR VIEWING
#########################################################################
#####
```

install xampp 8.0.3 locally (for viewing database in relative real time [need to refresh page])
- - edit the file in the c drive under xampp/phpmyadmin/ after installation called config.inc.php
- - - paste in the following code under the next line after all of the first server instance in the aforementioned file but above the "End of Servers configuration" multi-line comment
-- go to xampp/htdocs/ and delete everything except the xampp, img, and dashboard folders and then paste our project code into the directory
- - launch xampp control panel and turn on apache/mysql and then click admin on both in the same control panel
-- a browser window should open redirecting you to our home page, the same is the case if you just use the hosted link, but for viewing what’s going on in the database you’ll need to do it this way.

```
/*
 * AWS RDS Database
 */
$i++;

/* Authentication type and info */
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'admin';
$cfg['Servers'][$i]['password'] = 'euDmg7+0Q4~';
$cfg['Servers'][$i]['extension'] = 'mysqli';
$cfg['Servers'][$i]['AllowNoPassword'] = false;
$cfg['Lang'] = '';

/* Bind to the localhost ipv4 address and tcp */
$cfg['Servers'][$i]['host'] = 'database-1.cs1hkdhivv1o.eu-central-1.rds.amazonaws.com';
$cfg['Servers'][$i]['connect_type'] = 'tcp';

/* User for advanced features */
$cfg['Servers'][$i]['controluser'] = 'pma';
$cfg['Servers'][$i]['controlpass'] = '';

/* Advanced phpMyAdmin features */
$cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';
$cfg['Servers'][$i]['bookmarktable'] = 'pma__bookmark';
$cfg['Servers'][$i]['relation'] = 'pma__relation';
$cfg['Servers'][$i]['table_info'] = 'pma__table_info';
```

```
$cfg['Servers'][$i]['table_coords'] = 'pma__table_coords';
$cfg['Servers'][$i]['pdf_pages'] = 'pma__pdf_pages';
$cfg['Servers'][$i]['column_info'] = 'pma__column_info';
$cfg['Servers'][$i]['history'] = 'pma__history';
$cfg['Servers'][$i]['designer_coords'] = 'pma__designer_coords';
$cfg['Servers'][$i]['tracking'] = 'pma__tracking';
$cfg['Servers'][$i]['userconfig'] = 'pma__userconfig';
$cfg['Servers'][$i]['recent'] = 'pma__recent';
$cfg['Servers'][$i]['table_uiprefs'] = 'pma__table_uiprefs';
$cfg['Servers'][$i]['users'] = 'pma__users';
$cfg['Servers'][$i]['usergroups'] = 'pma__usergroups';
$cfg['Servers'][$i]['navigationhiding'] = 'pma__navigationhiding';
$cfg['Servers'][$i]['savedsearches'] = 'pma__savedsearches';
$cfg['Servers'][$i]['central_columns'] = 'pma__central_columns';
$cfg['Servers'][$i]['designer_settings'] = 'pma__designer_settings';
$cfg['Servers'][$i]['export_templates'] = 'pma__export_templates';
$cfg['Servers'][$i]['favorite'] = 'pma__favorite';
```

Use the following roles to test the functionality of different things, these are all from the database:

```
Admins:
        Leah Saldana (15848455) {
                Username: Foriz1971
                Password: ie6ahzohBee
        }

Students:
        //takes 4 classes
        Eddie Donovan (54875814) {
                Username: Weireaddes
                Password: Quub2ieM
        }
        //takes 3 classes
        Martha Gobert (74655355) {
                Username: Sopichim
                Password: Sah2Teeth2
        }

Instructors:
        Molly Tarr (21900449) {
                Username: Nithe1969
                Password: uehav1Ei
        }
        //teaches 1 class
        Jeanne Deniston (75305948) {
                Username: Muce1982
                Password: eish7iowuiB
        }
        //teaches 1 class
        Michael Cancel (77716742) {
                Username: Caravered
                Password: FeebohV9
        }
        //teaches 4 classes
        //has 4 announcements in Art III
        Tim Hutzler (91377380) {
                Username: Abires
                Password: Vu5Ahghif0ah
        }
```

• Testing: what types of testing done

Tests were done to ensure basic functionality of all systems implemented: the following tests were considered

- Criteria for the system tests:
    - UI should be user-friendly and intuitive
    - Design is appealing and understandable

• Plan for next delivery

○ Requirements to be developed

All we would need is to create a new table for grades, implement two ways to view the grade page, one for users like admins and students then one for instructors, then include the relevant PHP scripting to make this happen,

○ Refactoring ideas, on both design and code

- Make everything apply to Bootstrap framework.
- Ensure that everything is shipshape code wise. Further get rid of PHP and JS in PHP files to make them more clean.
- Style things further,
- Etc.