Spencer Roucoulet | Ryan Plante | Farid Hasanov
AS Capstone Technical Design Document

Models

User.cs

User model. This contains the fields that are used for the user. I am not sure if this is needed as User isn't a form, but it's good to create User as an object for best practices. It will contain the following fields:

- **userID**, int, PK
- **userFName**, string(32)
- **userLName**, string(32)
- **userPassword**, string(255)
- **userSalt**, string(255)
- **userDept**, int, *FK*
- **userPrivilege**, int
    - The privileges are tiered as followed:
    - 0: Student level. Students only have access to a read only schedule for when the labs are open.
    - 1: Lab monitor. This will allow monitoring for the department they are in.
    - 2: Department head: This will allow viewing reports of the department that they are in. They also are allowed to modify the schedule and add/edit lab monitors for the department.
    - 3: Admin: They have the same permissions as department head however they have the ability to modify department heads and modify lab room #s.
- **userFeedback**, string, used to display errors

UserDAL.cs

User data access layer
This connects to the SQL server to validate credentials and sets session variables to determine who is logged in.
Relevant functions:

- **ValidateCredentials(***int* userName, *str* userPassword)
    - This will perform an SQL query to compare userPassword with the password on the database to validate credentials. If they are valid, return the user database object. If they are not, return false.
- **ChangePassword(***User* user**,** *str* newPassword)
    - This will update the userPassword with the new supplied password in the database. It uses SHA-1 encryption with userSalt for security purposes.

- **GetPicture(***int* id**)**
  - This method will get the picture from the database given the userID is set. If it doesn't exist or there is an error loading the image, return the default picture, which will just be a gray glyphicon of a user account.
- **AddUser(***objUser* User**)**
  - This method isn't needed if the website was connected to TechNet, but we will add a User to the table if they don't exist in the database when entering them into the log. It will add them with the default password of password for demo purposes.
- **ChangeDept(***objUser* User, int dept**)**
  - This method also isn't needed if the website was connected to TechNet, but for demonstration purposes for this website, this will be used by Admins to assign departments for department heads.
- **ChangeToStudent**(int userID)
  - This method demotes the user permission level to a student (permission 0)
- **doesUserExist(int id)**
  - This method performs an SQL query to check if user exists by id
- **ChangeMonitorDept(int userID, int dept)**
  - This method wouldn't be needed if this site was connected to TechNet, but this is used to change the user permission level to 1 (monitor) and change their deptID. Because TechNet has their own implementation for departments, we had to make our own.
- **GetMonitors(***int* Dept**)**
  - This method will perform an SQL query that filters all lab monitors (permission level 1) in the users table with the given department id.

Lab.cs

Lab model. Lab is a form that is editable by admins so it will contain the following fields:

- **labID**, int, PK
- **labName**, string(32)
- **labRoom**, string(16)
- **deptHead**, int, FK
- **deptID**, int, *FK*

LabDAL.cs

Lab data access layer. Connects to and manages the Lab table.
Add, edit, delete, pull all, pull one
Relevant functions:

- *string* **GetConnected()**

    - Provides a string containing credentials + information to connect to the DB
- *void* **AddLab(***Lab* lab**)**
    - Inserts a single record in the Lab table using the object passed into the function that contains the needed information.
- *void* **EditLab(***Lab* lab**)**
    - Updates a single record in the Lab table using the object passed into the function that contains the needed information, such as the ID of the record to be changed and the new information.
- *void* **RemoveLab(***int* id**)**
    - Deletes the record that corresponds to the ID passed into the function.
- *Lab* **GetLabByID(***int?* id**)**
    - Pulls the record from the Lab table that corresponds to the lab ID (PK) passed into the function.
    - Returns as a Lab object
- *Lab* **GetLabByDeptID(***int?* id**)**
    - Pulls the record from the Lab table that corresponds to the department ID passed into the function
    - Returns as a Lab object
- *Lab* **GetEmployeeLab(***int?* id**)**
    - Pulls the record from the Lab table the corresponds to the user ID (department head ID) passed into the function
    - Returns as a Lab object
- *List<Lab>* **GetAllLabs()**
    - Pulls a list of all the records in the Lab table.
    - Returns as a list of Lab objects
- *bool* **DepartmentExists(***int* deptID**)**
    - Connects to Lab to check if there are any records with the department ID passed into the function
    - Returns true if there are, false if there are not
- *bool* **UserExists(***int* deptHead**)**
    - Connects to Lab to check if there are any records with the user ID (department head ID) passed into the function
    - Returns true if there are, false if there are not

Department.cs

Model for the table containing the departments. Contains the following fields:

- **deptID**, int, PK
- **deptName**, string(32)

DepartmentDAL.cs
Data access layer for the department table.

Relevant functions:

- *string* **GetConnected()**
  - Provides a string containing credentials + information to connect to the DB
- *Department* **GetDeptByID(***int?* id**)**
  - Connects to the Department table and pulls a single record that corresponds to the ID that was passed in
  - Returns the record as a Department object
- *List<Department>* **GetAllDepartments()**
  - Connects to the Department table and pulls all records stored in it
  - Returns the records as a list of Department objects

Log.cs

Model for the log. Contains the following fields:

- **logID**, int, PK
- **studentID**, int
- **studentName**, string
- **labID**, int
- **timeIn**, datetime
- **timeOut**, datetime
- **itemsBorrowed**, string(255)

LogDAL.cs

Log data access layer.
This model is in control of modifying and accessing the Log table.
Relevant functions:

- *string* **GetConnected()**
  - Provides a string containing credentials + information to connect to the DB
- *void* **AddLog(***Log* tba**)** tba = to be added
  - Connects to the Log table and adds a record with the information passed in by the Log object
    - Because timeOut is not required, there are two different SQL statements depending on if the passed object has a DateTime value in the timeOut field
- *void* **ModifyLog(***Log* tbu**)** tbu = to be updated
  - Connects to the Log table and updates the record that corresponds with the ID passed in with the Log object using the other data sent with the ID
- *void* **ClockOut(***int* id**)**
  - Connects to the Log table and sets the timeOut value of the passed ID's record to the current time
  - Used for the clock out buttons

- *List<Log>* **GetAllLogs(***int* dept**)**
    - Connects to the Log table and pulls all records that belong to the passed department ID. department ID needed because each department should only see their own logs
    - Returns records in a list of Log objects
- *List<Log>* **GetLogsBetween(***DateTime* start, *DateTime* end, *int* dept**)**
    - Connects to the Log table and pulls all records that:
        a) Fit between the start and end times passed in
        b) Match with the passed department ID
    - Returns as a list of Log objects
- *Log* **GetALog(***int* id**)**
    - Pulls a single record from the Log table that corresponds to the log ID passed into the function
    - Returns a Log object

Schedule.cs

Model for the table containing the times people are scheduled to work. Contains the following fields:

- **scheduleID**, int, PK
- **userID**, int
- **textSchedule**, string

ScheduleDAL.cs

Schedule data access layer.
This model is in control of modifying and accessing the Schedule and Unavailability table.
Relevant functions:

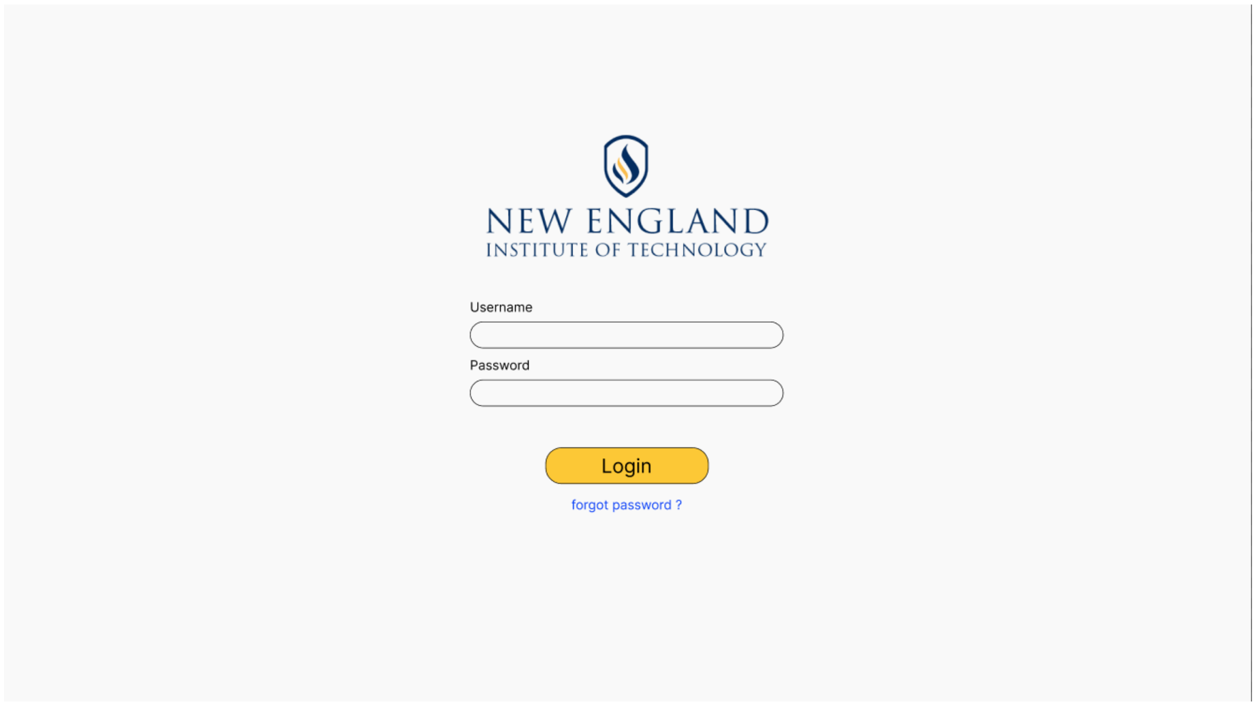
- **GetUserSchedule(***int* id**)**
    - Gets all schedule entries for specified user
- **GetUserUnavailbility(***int* id**)**
    - Gets all schedule entries for specified user
- **SetSchedule(***objSchedule* schedule**)**
    - Modifies schedule entry
- **GetDeptSchedule(***int* dept**)**
    - Returns all schedules filtered with department
- **DelSchedule(***int* id**)**
    - Deletes a schedule given schedule’s ID.
- **DelUserUnavailbility(***int* id**)**
    - Deletes a user’s unavailability record.
- **AddSchedule(***objSchedule* entry**)**

    - Adds a schedule record to the schedule table
- **AddUnavailbility(**_objUnavailability_ entry**)**
    - Adds an unavailability record to the unavailability table.

Pages

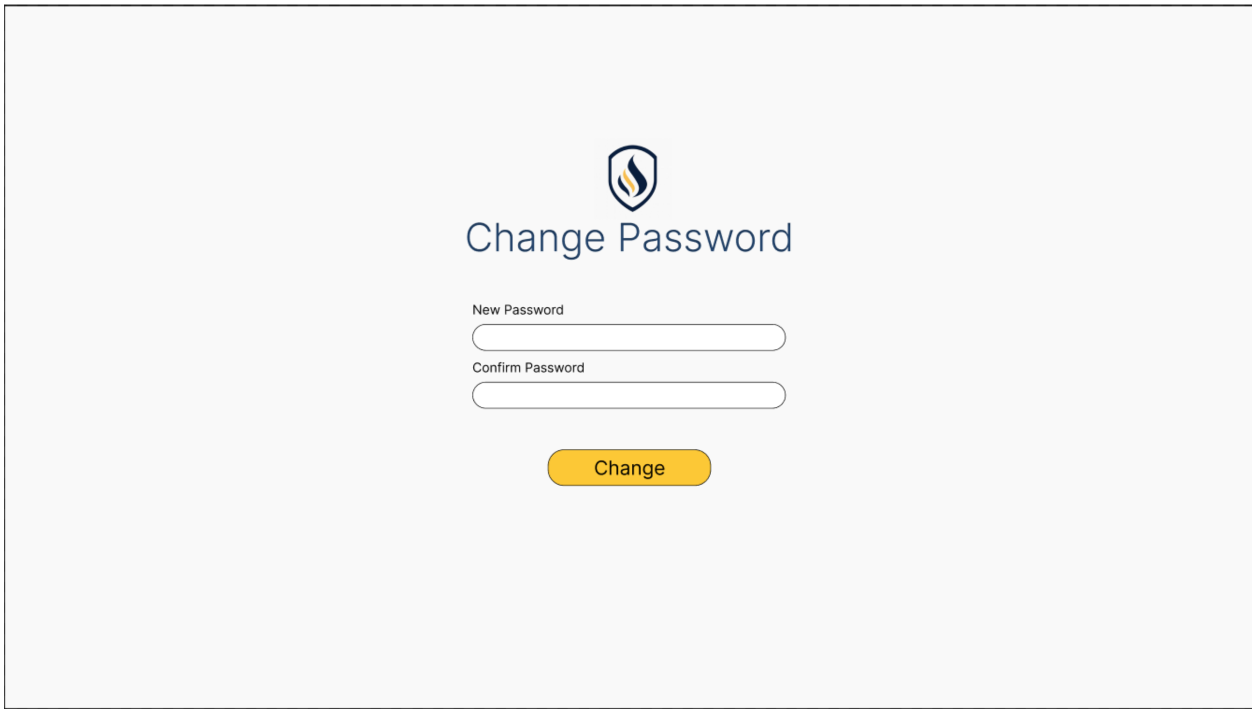
Login.aspx

This is the login page and will be the landing page for all users that are not signed in. It connects to the User data access layer (**UserDAL.cs**) and if the credentials are valid, it will set the session variables for the user name and access level. All of the pages will be internal to the school so if the session variables are not set, the website will send them to this site. Clicking the forgot password hyperlink will send an email to the user to the change password screen. This will be handled by TechNet so functionality is not implemented. If the credentials are valid, it will send the user to Landing.aspx

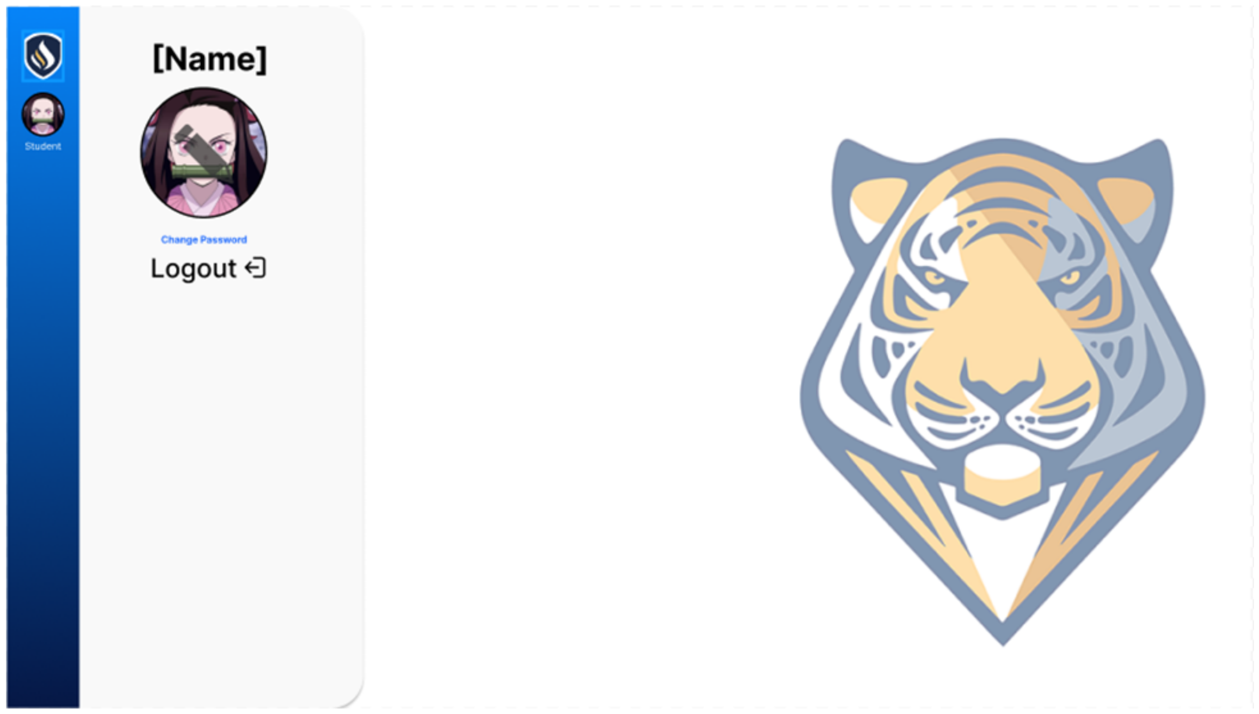


ChangePassword.aspx

This is the change password screen that will communicate with the user table to allow changing password. The only functionality that will be coded will be connecting to User data access layer to call the ChangePassword method if the new password and the confirm password text boxes are the same (to ensure that the user entered the correct password) This site is probably redundant since New England Tech already handles user login.



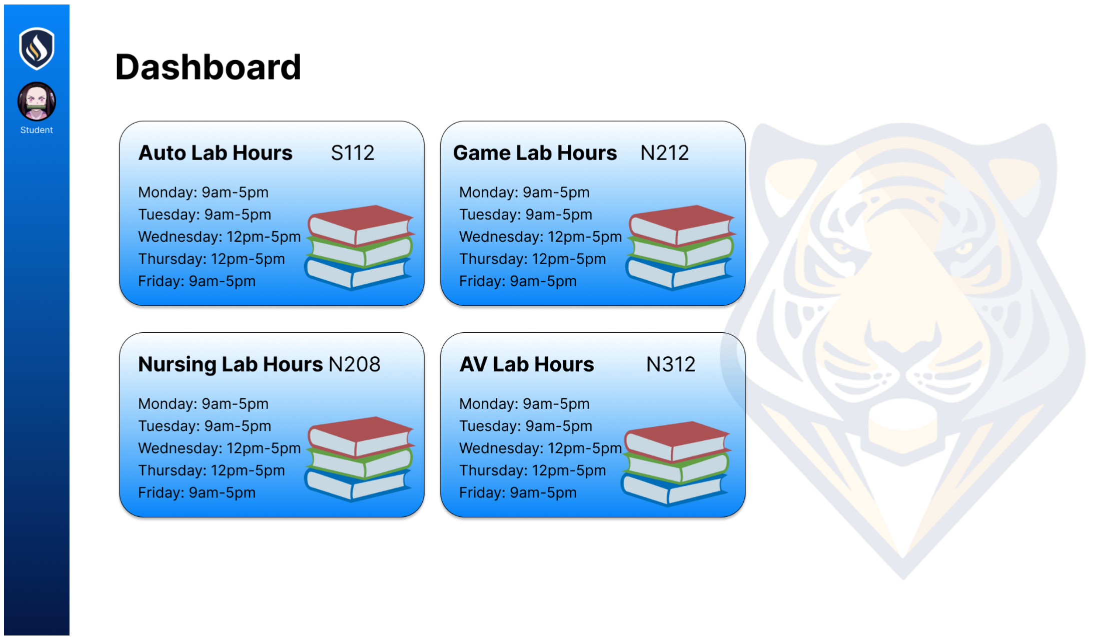
Example of expanded profile



This is a demonstration of how the user's slide-out menu will work. This sidebar will slide out when their profile picture icon is clicked on any page. The username they specified during registration will be displayed (or in the case of using a database, then the name of which is located there will be used). Under the name is the user's avatar. In the avatar there is a semi-transparent pencil, when pressed, the user will be able to change the avatar. The avatar photo will be stored and referenced using a string containing (userid + .jpeg or .png example: 008542316.png). Below that is a link to change your password. This link leads to ChangePassword.aspx. Lastly, there is a button to sign the user out of the website, which puts them back to the login page.

Landing.aspx

This is the landing page that the user will land on when they login to the site. The view changes based on the userPrivilege.



Dashboard

Auto Lab Hours S112
Monday: 9am-5pm
Tuesday: 9am-5pm
Wednesday: 12pm-5pm
Thursday: 12pm-5pm
Friday: 9am-5pm

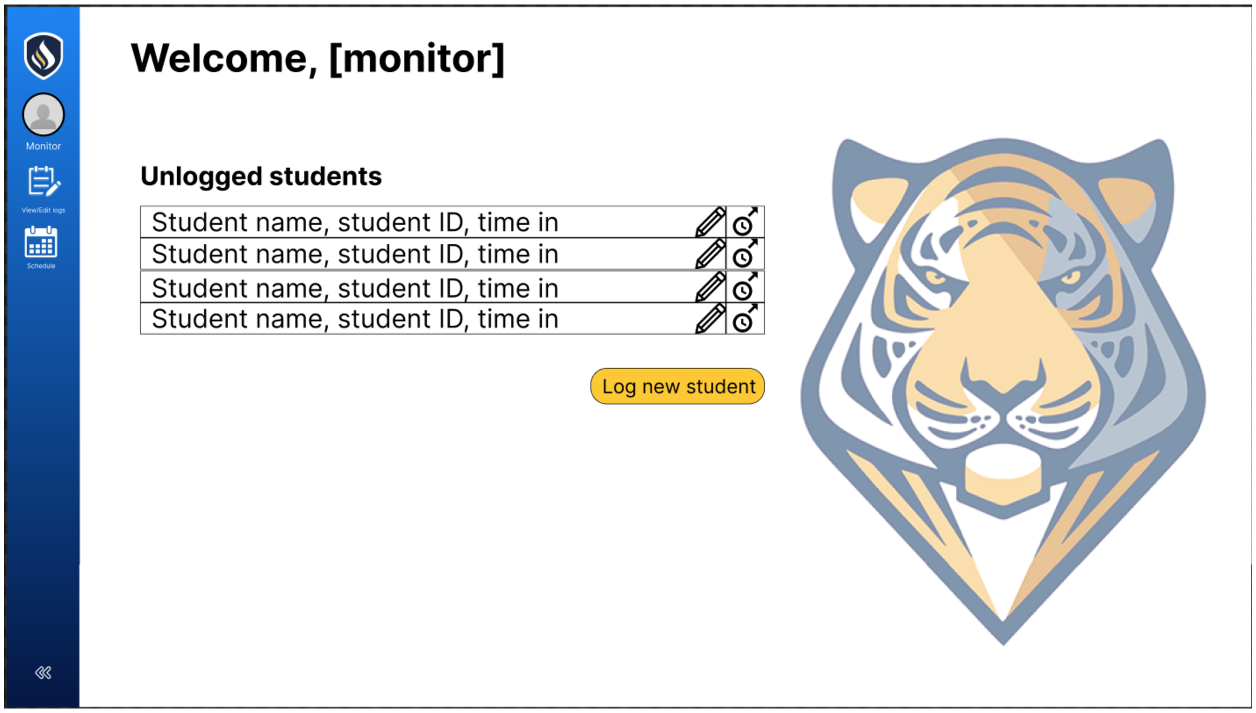
Game Lab Hours N212
Monday: 9am-5pm
Tuesday: 9am-5pm
Wednesday: 12pm-5pm
Thursday: 12pm-5pm
Friday: 9am-5pm

Nursing Lab Hours N208
Monday: 9am-5pm
Tuesday: 9am-5pm
Wednesday: 12pm-5pm
Thursday: 12pm-5pm
Friday: 9am-5pm

AV Lab Hours N312
Monday: 9am-5pm
Tuesday: 9am-5pm
Wednesday: 12pm-5pm
Thursday: 12pm-5pm
Friday: 9am-5pm

Student view (permission level 0)

The schedule boxes will populate based on the hours coming from the schedule table.

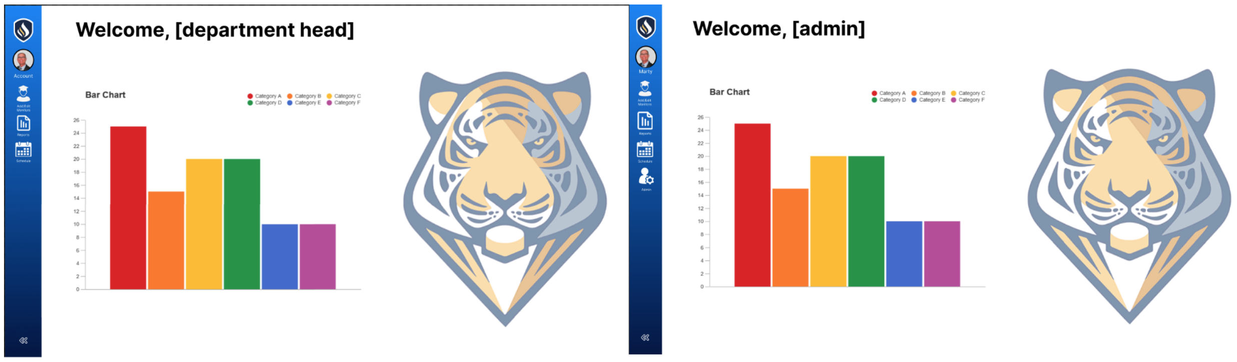


Welcome, [monitor]

Unlogged students

Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in

Monitor view (permission level 1)



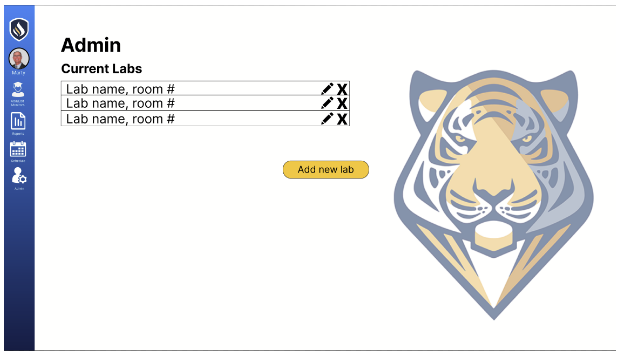
Department head(left) Admin (right)

Since only professors will be department heads/admins, their functionality is almost the same. Permission levels are 2 and 3 respectively. ASP.net has built in functionality to display charts, so we will be following their documentation.
The X axis displays the hour and the Y axis displays the frequency.

LabEdit.aspx

The one page exclusive to Admin (level 3) users.
This page checks to see if the user is logged in, and if not, they get returned to login.

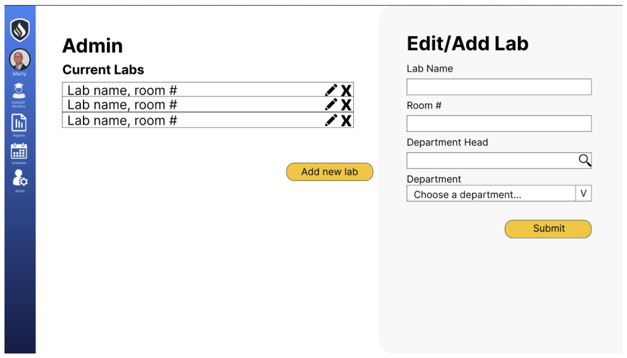
This page allows Admins to add, edit, or delete labs. On the page loading, it displays limited information about all of the current labs found in the database. Clicking the "add new lab" button or any of the pencil icons within records will pull up the form on the side. If the pencil is clicked, then the form will be pre-populated with the data from the corresponding record, allowing the user to change the information in the record.


Admin

Current Labs

Lab name, room # X
Lab name, room # X
Lab name, room # X

Add new lab


Admin

Current Labs

Lab name, room # X
Lab name, room # X
Lab name, room # X

Add new lab

Edit/Add Lab

Lab Name

Room #

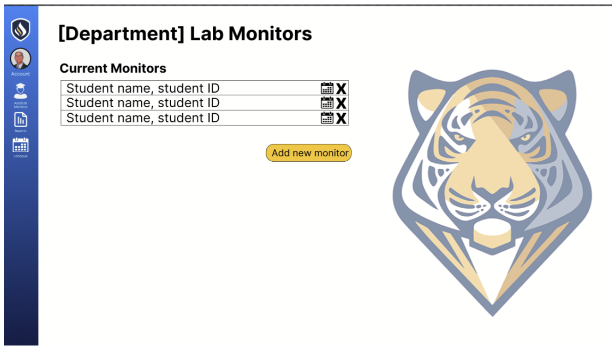
Department Head

Department
Choose a department... V

Submit

MonitorsEdit.aspx

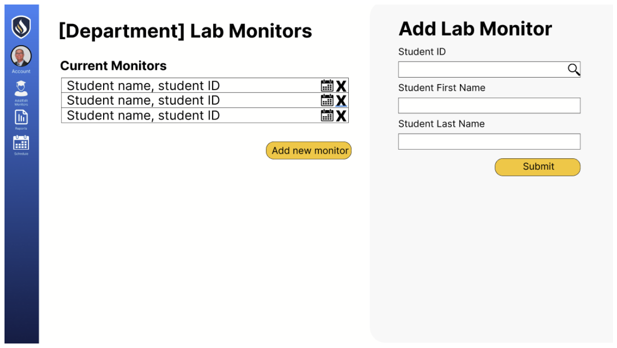
Used by department heads (level 2) and admins (level 3).
This page checks to see if the user is logged in, and if not, they get returned to login.


[Department] Lab Monitors

Current Monitors

Student name, student ID X
Student name, student ID X
Student name, student ID X

Add new monitor


[Department] Lab Monitors

Current Monitors

Student name, student ID X
Student name, student ID X
Student name, student ID X

Add new monitor

Add Lab Monitor

Student ID

Student First Name

Student Last Name

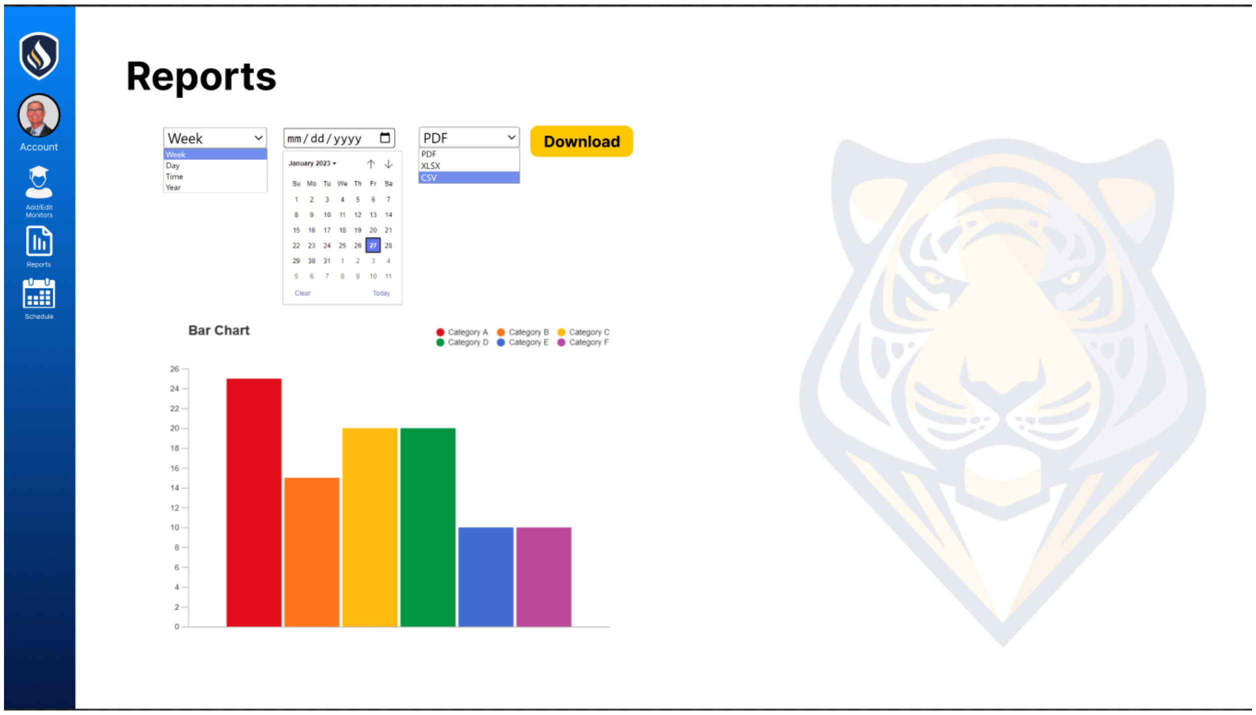
Submit

On loading, this page displays limited information about all monitors that belong in the user's department (department is a value stored in the database per user). The "add new monitor" button will open a form that searches for existing users by their ID, and changes their permission level to 1 to reflect their status as a lab monitor.
At the end of each record is an X. The X changes the monitor's status back to 0 (student).
The calendar icon leads to [schedule page] and highlights the selected user's schedule.[1]

Reports.aspx

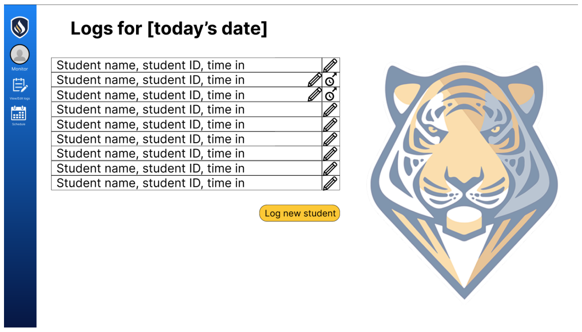
Used by department heads (level 2) and admins (level 3).
This page checks to see if the user is logged in, and if not, they get returned to login.

[1] The calendar icon and its functionality may not be implemented. We will do it if we have enough time and feel it would add value to our project.



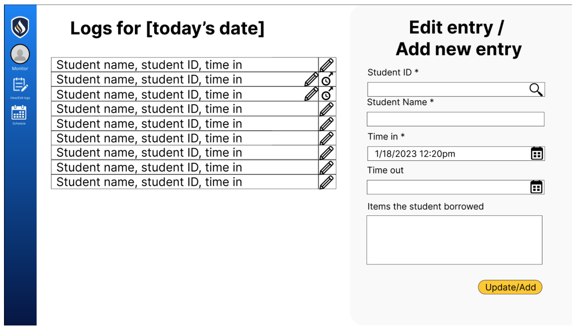
This page generates reports, and relies on 4 inputs to determine the values it will use to create the graph. The leftmost is a dropdown that changes the date range - Week, Day, Term, Year. The second and third are date pickers - start and end of date range respectively. The third is only visible and able to be interacted with when “Term” is the selected date range, as it essentially serves to be a custom date range, since we saw hard coding the terms as an unviable solution. The third input is still used even as it’s hidden; the computer will calculate the end date given the start date for the other 3 date range options. As the user interacts with the range and the start/end dates, the graph will update dynamically, reminiscent of a Power BI dashboard. The 4th input is also a dropdown menu, and changes the filetype that the download button will generate a report as.

LogView.aspx



Logs for [today's date]

Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in



Logs for [today's date]

Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in
Student name, student ID, time in

Edit entry / Add new entry

Student ID *

Student Name *

Time in *
1/18/2023 12:20pm

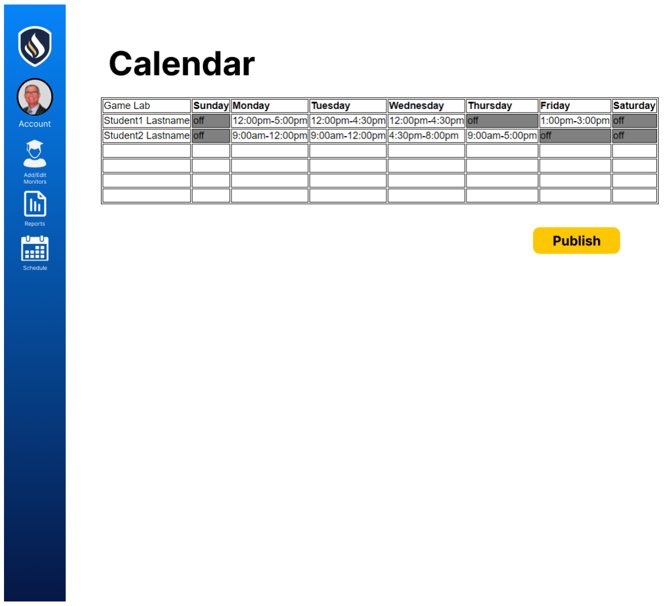
Time out

Items the student borrowed

This page will only be accessible by lab monitors, so it checks the permission level before entering this page. This connects with the Log data access layer (LogDAL.cs) to display all the log entries for today's date. Clicking on the edit button (pencil icon) or the log new student button, the edit entry/add new entry screen will display. The student ID, Student Name and Time in fields are required and the page will prevent the user from clicking the add button with a hidden feedback label that only displays if there are errors in the form. When the time out field is filled out, the timeout icon (clock icon) disappears. When the user clicks on the timeout icon, the record in the table automatically gets updated with the timeout filled as the current time.

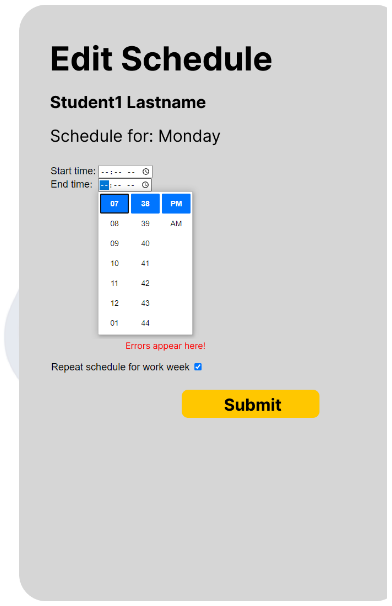
Calendar.aspx

This page will be accessible by lab monitors and department heads/admins. If the user is an admin/department head, they will get the view shown below, which allows them to modify the schedule.



Calendar

Game Lab	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Student1 Lastname	off	12:00pm-5:00pm	12:00pm-4:30pm	12:00pm-4:30pm	off	1:00pm-3:00pm	off
Student2 Lastname	off	9:00am-12:00pm	9:00am-12:00pm	4:30pm-8:00pm	9:00am-5:00pm	off	off



Edit Schedule

Student1 Lastname

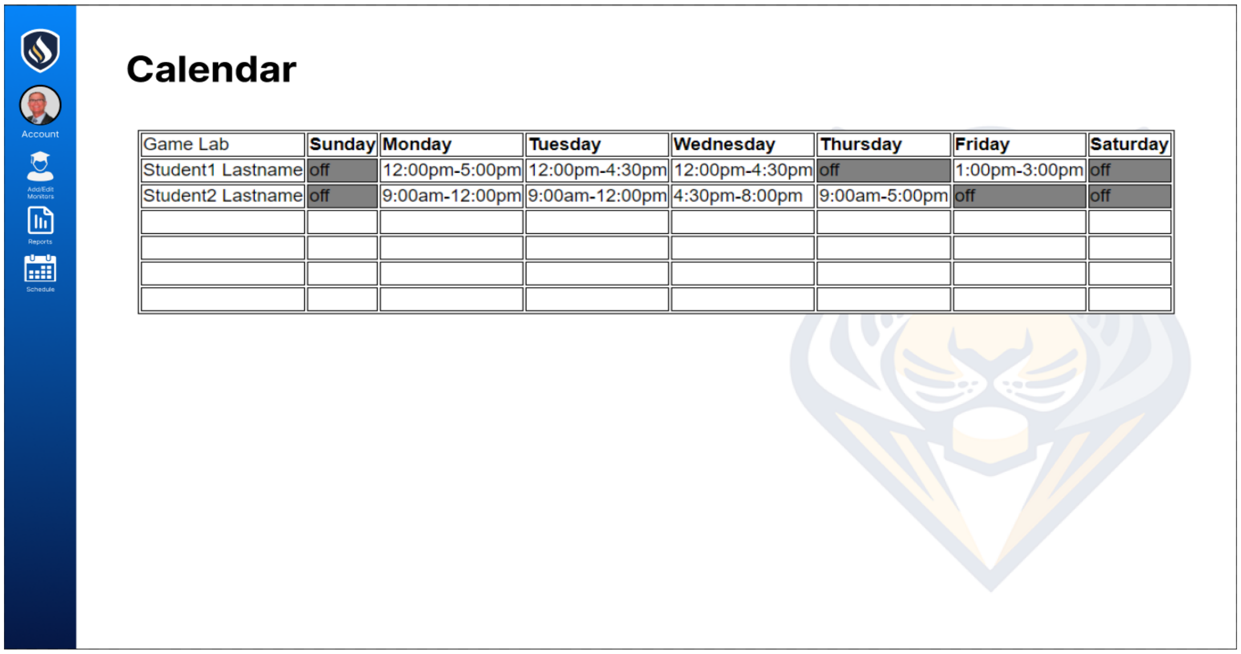
Schedule for: Monday

Start time: --:-- --
End time: --:-- --

Errors appear here!

Repeat schedule for work week ☑

The department head will be able to click on any of the cells in the schedule grid to modify that specific day. The form will pop out displaying which day and who's schedule they're modifying. This doesn't directly update, however. The user will get a pop up warning to make sure to publish changes if they try clicking on any of the nav links.



Calendar

Game Lab	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Student1 Lastname	off	12:00pm-5:00pm	12:00pm-4:30pm	12:00pm-4:30pm	off	1:00pm-3:00pm	off
Student2 Lastname	off	9:00am-12:00pm	9:00am-12:00pm	4:30pm-8:00pm	9:00am-5:00pm	off	off

Read-only schedule for the entire department. It is the same view as the department head/admin.