Code Inspection Report

Boardman Computer Science Lab Web Portal

Client: Mr. Christopher Dufour

Development Company: In-House Operations

Development Team:
Klei Bendo
Jack Brisson
Alex Landry
Samuel Morse
Aaron Schanck
Forrest Swift

Date: 3/08/2022
Version 1.0

Boardman Computer Science Lab Web Portal
Code Inspection Report

Table of Contents

1. Introduction

This section introduces the purposes in creating this document and cites important prior documentation integral to its design and implementation. Further, this section provides the conventions for coding and commenting In-House Operations has used in the development of the Boardman Computer Science Lab Web Portal and a checklist of known defects to date.

1.1 Purpose of This Document

This Code Inspection Report is intended first and foremost for the In-House Operations team to look inwards at the physical development we have produced to date. An internal inspection is integral to ensure we are adhering to the utmost quality and uniformity for future development, readability, and maintenance. Within this document is detailed the results of our inspection including our conventions, defects, the process by which we conducted our inspection, and the specific modules that were inspected. If all is included correctly and adequately, the goal is for anyone to be able to read this document and be able to contribute to the application seamlessly in content and style.

1.2 References

Bendo, K., Brisson, J., Landry, A., Morse, S., Schanck, A., & Swift, F. (2021, November 20). *In-house operations UIDD v.1.* Google Docs. Retrieved December 13, 2021, from https://docs.google.com/document/d/1ExOExrAWAgdNB8ay1yYSWllcMcLxWPr_ylOSGEBPz1U/edit?usp=sharing

Bendo, K., Brisson, J., Landry, A., Morse, S., Schanck, A., & Swift, F. (2021, November 30). *In-house operations critical design review document*. Google Docs. Retrieved February 7, 2022, from https://docs.google.com/document/d/11XAon5JmpzevMZ7L10jm5781Aio1FbbRcJ2E6BakDUY/edit?usp=sharing

Bendo, K., Brisson, J., Landry, A., Morse, S., Schanck, A., & Swift, F. (2021, October 18). *In-house operations SRS*. Google Docs. Retrieved December 13, 2021, from https://docs.google.com/document/d/1YIFScQdYOcsTWcKpfTEac3g4aTRo3XtSTO1Uay2CQv4/edit?usp=sharing.

Bendo, K., Brisson, J., Landry, A., Morse, S., Schanck, A., & Swift, F. (2021, October 28). *In-house operations SDD v. 1.0*. Google Docs. Retrieved November 29, 2021, from https://docs.google.com/document/d/1qvNbDHLXdX5J8jHynGA8STqZW0qGLAZs1bayzoYAhoY/edit?usp=sharing

Dufour, C., & Rheingans, P. (2021, September 16). dufour_help-Resource-Scheduling. Orono.

Morse, S. (2021, November 21). *Figma Design*. Retrieved December 12, 2021, from https://www.figma.com/file/YvE53I7e9N8KaubB23RP2m/Main(e)-Site?node-id=0%3A1

Schanck, A. (2021, October 4). *Team 17 capstone proposal*. Google Docs. Retrieved December 13, 2021, from https://docs.google.com/document/d/19nm8LNdbCEEdSQNdVRj570LcsRJC7rjRumRwU4srtBE/edit?usp=sharing.

1.3 Coding and Commenting Conventions

1.3.1 Coding Conventions

File organization is organized by application within our project. Each application contains its own models, views, and templates (as needed). Static files are organized inside the main application (boardmanlab) and contain our image libraries and stylesheets.
The template (html) naming convention is camelCase to maintain readability. We used Kabab-case for css classes and ids to be consistent with the CSS property names. CamelCase was used for context variables inside views. Classes maintain uppercase first letter convention for almost all object oriented programming languages. Template tags (functions for use inside templates) use snake_case to help readability inside the template (and differentiate them from context variables). We tried to keep line length in python code capped at 80 characters, and used a combination of Black for Python and djLint for Django enabled HTML for formatting our code.
Continuous integration was used through GitHub actions to ensure that builds were successful with each push to main. Testing will be added to the CI workflow once a significant portion of the unit tests have been written.

1.3.2 Commenting Conventions

Our commenting conventions consist mostly of directional markers for web page placement. In our HTML files, we explicitly state where code will be rendered on the web page in terms of content blocks. This is very helpful in terms of organization so that we can categorize our HTML files and be able to easily jump between code blocks since the render view is on a different page. Additionally, in our testing files the commenting outlines what app each block of tests is for. For example there is a code block with a comment above it stating its for boardman lab url tests. Finally, unused code and code under development is left commented for the time being to hold onto the information in case it becomes needed or useful in later development. Our commenting conventions could be improved greatly for future development. In our rush to produce a full application in a short timeframe, commenting in terms of functionality has been rather lax, but this inspection has been a wakeup call in this regard and we will strive to continuously improve our conventions to convey the utmost clarity, readability, and ease of maintenance moving forward.

1.4 Defect Checklist

1.4.1 Code conventions:

- Ambiguous variable names
- Similar variable names
- Improperly capitalized variable/model names
- Code not linted
- Lack of comments
- Unused code segments
- Redundant code segments

1.4.2 Functional (logic errors):

- Urls/views/models not imported:
- Data transfer failure (forms not updating, displaying correctly, etc.)
- Invalid data access
- Unconditional infinite loop

- Context improperly passed to template
- POST request reference errors
- Referencing incorrect variable
- Crashes/timeouts

1.4.3 Security Oversights:
- Unprotected user information
- Database Inferential Attacks
- Execution of malicious code through forms
- Unauthorized access to helper/administrator tools
- Sql injection

1.4.4 Display Errors:
- HTML errors
- CSS incompatibility with certain browsers
- Rendering issues
- Style inconsistencies
- Resize errors
- UI bugs/distortions
- Lack of feedback to the user upon using a function (success page)

2. Code Inspection Process

This section outlines the process by which we conducted our inspection. It includes a specific description of our inspection process, our impressions of the process itself, and an overview of the meetings that were held to inspect our code.

2.1 Description

We started by assigning roles to each of our team members, but had overlap as each of us contributed to the codebase. We worked together to choose a list of potential defects that we felt had happened or could happen in our project, and planned to run through the checklist on each file. The style of the review was very open and we took turns being the reader, inspectors, scribes and moderators. The reader would share their screen over Discord and go through the module, stopping when there were questions or comments by the other team members. It was conversational and the scribes took down notes as we proceeded. We organized each of our modules into categories so that we could analyze similar code in groups: HTML, Forms, Views, Models, and Misc. For each module we read through the file together, asking questions when necessary, commenting on flaws found from our defect checklist, and ensuring that everyone understood each piece of the code. During the reading process we suggested improvements that should be applied in the future. Some code was found to be redundant, and was removed where appropriate. We noted some code to review further in the future, either to check for potential redundancy or to refactor for efficiency or intended function.

The code inspection uncovered many flaws, the most obvious being a near complete lack of comments. The next most common issue was redundant or outdated code that either needed to be trimmed or reworked. In fact, most of the flaws uncovered were related to coding conventions. We did uncover a few major flaws, such as security vulnerabilities in unsecured forms in the calendarDay template, some necessary changes in our settings file, and a major crash in the recurring reservations function. We believe we largely stuck to our “ideal” process, although as the meetings went long we tended to be less sharp in our bug catching abilities.

2.2 Impressions of the Process

Impressions were generally positive. We were able to detect a lot of defects from each module, and understand the modules and their functionality much better than before. Having established roles in the process helped us to keep organized and to keep moving forward. With one reader, we were able to scan the code as a group more effectively and record any comments or issues efficiently. We also had a recorder directing the reader so that everyone
Was on the same page which saved time and effort. We thought, however, that it would have been more effective if there were an outside party reviewing the code, at least in conjunction, we would be able to catch more issues as they wouldn’t be used to the quirks and tricks of the system. Due to the nature of this class this is not viable, but in a real world setting this would be a valuable resource.

If we were to redo any segment of the code inspection process, we would change our sessions schedule. We were able to complete the inspection in only one day with two sessions, however it was an arduous task for a virtually single push and left us fatigued by the end. With this amount of code to sift through, it would be pertinent to allocate sessions over two days if not a third as well. This will allow us to analyze the information from these sessions more easily.

The best modular units in our program were the models. We only found one small defect, and the rest of the code looked good. This means our database is storing the fields for each app properly. The worst units in our program were: templates, and the style.css file. We found plenty of defects that covered all of the defect checklist in code conventions, functional errors, security oversight, and display errors. The style.css file in particular did not follow any of the code conventions to its full potential and contained a lot of redundancies. The style.css sheet somehow managed to hit every defect in our code conventions section of the defect checklist. Ambiguous, similar, and improperly cased variable names and a lack of meaningful comments make it difficult to track elements down in the html. There is also a plague of redundancy in the sheet, with many lines placed in multiple children elements when they could be placed in one parent. If given enough time, one of the largest quality of life updates would be to refactor the style.css sheet.

2.3 Inspection Meetings

Inspection Meeting 1

Date: 3/8/2022
Location: virtual
Time Started: 4:00pm
Time Ended: 7:05pm
Participation: full team
Roles:
Reader - sam
Inspectors - Forrest, Aaron, Sam
Scribe - Alex, Klei, Forrest
Units Covered: all HTML pages

Inspection Meeting 2 -

Date: 3/8/2022
Location: virtual
Time Started: 7:35pm
Time Ended: 9:40pm
Participation: full team
Roles:
Reader - sam
Inspectors - Forrest, Aaron,sam
Scribe - alex, klei, forrest
Units covered: Forms, Views, Models, Misc

3. Modules Inspected

This section outlines modules that were examined in our inspection including the module name, description, functionality, and differences, if any, to their original design. Included additionally are comparisons between original plans for major design sections and how they were actually implemented.

3.1 Modules Overview

3.1.1 HTML

Module: Index, Login, Home
Description:

- Index: Unused file
- Login: handles authentication for Google, using the allauth library
- Home: Landing page

Where it fits in:

SDD: 2.2.1 Sign In/Out, this module displays the user interface for logging into the application. Login page is used as an entrance point that allows access to the rest of the website.

Inspection:

Q: Is the user variable inside login.html a model instance?
A: No, it references the session instance

- 1.4.1 Unused code segment - Index.html, Home.html
- 1.4.2 Urls not imported
- 1.4.3 Home.html can be unintentionally accessed
- 1.4.1 Unused code segment

Module: base, base_simple
Description:

Starting point for the rest of templates, contains the background and sidebars; base_simple: same as base but no side navs

Where it fits in:

base and base_simple are the starting points for every other template in the application, i.e. each template extends either base or base_simple (not included in SDD). The SDD was written before the team had knowledge of the conventions of the Django framework and did not know to include certain developmental pages.

Inspection:

- 1.4.4 Background image might be too large (issues with rendering)
- 1.4.1 Lack of comments

Module: calendarDay, calendarDay_content

Description:

- calendarDay: Calendar tab selected, contains links to Sessions, and Manage tabs if the user is admin. Contains the content for the day calendar view.
- calendarDay_content: Contains the content strip for filtering sessions by month, scroll through and display help sessions, edit help sessions that you own

Where it fits in: SDD 2.2.9 calendar view, change_view().

Inspection:

- 1.4.1 calendarDay unused code on line 2 (removed)
- 1.4.1 Improperly capitalized template name
- 1.4.1 calendarDay_content.html indentation error line 53,60-72
- 1.4.3 calendarDay_content.html line 62-66, potential security issue (not a proper Django form)
- 1.4.1 calendarDay_content.html line 70, redundant if statement
- 1.4.1 lack of comments

Module: calendarMonth, calendarMonth_content

Description:

- calendarMonth: Calendar tab selected, contains links to Sessions, and Manage tabs if the user is admin. Contains the content for the month calendar view.

- calendarMonth_content: Contains the content strip for filtering sessions by month, displays a calendar grid of days for a month, with day labels on top. If there are help sessions for a particular date, the number of help sessions is displayed inside the cell for each date.

Where it fits in:

SDD 2.2.9 calendar view, change_view()

Inspection:

- 1.4.1 Redundant code: Social account load unnecessary
- 1.4.1 Improperly capitalized template name
- 1.4.1 Redundant code calendar month content line 2 - 4
- 1.4.1 Lack of comments

Module: createHelpSession, recurHelpSession, editHelpSession

Description:

- createHelpSession: displays a form for creating a help session with the following fields: Helper, Time, Duration, Topic, Mode (In-person, remote), Notes.

- editHelpSession: displays the same form as createHelpSession, but draws data from the model to display an already saved session, and allows the helper and admin users to edit it.

- recurHelpSession: allows helpers and admin users to create recurring help sessions.

Where it fits in:

SDD 2.2.3 Store Help Session Data, SDD 2.2.4 Display Reservation, SDD 2.2.9 Edit Help Session. recurHelpSession was added later as a way to store multiple help sessions on a weekly, monthly, or daily basis.

Inspection:

Q: Why is Form.errors needed after every field?
A: Verifies field input

Q: Who can access and edit a help session?
A: Helpers and admins

- 1.4.1 Improper Indentation throughout - fixed
- 1.4.1 Redundant code - no need to pass back user (line 26)
- 1.4.1 Lack of comments
- 1.4.1 Improper capitalization of a template

Module: manageHelpSessions, filter_manageHelpSessions
Description:

- manageHelpSessions: displays the help sessions owned by the helper/admin user, allows the helper/admin to edit/remove sessions, contains links to download csv report files.

- filter_manageHelpSessions: helper/admin can filter their sessions by month.

Where it fits in:

SDD 2.2.3 Store Help Session Data. In the SDD we didn’t include a method for displaying help sessions, only reservations. We figured we needed a way for helpers to manage their own help sessions as well as filter through them by month.

Inspection:

- 1.4.1 Redundant Code Sections
- 1.4.1 Improper Indentation - fixed
- 1.4.1 Lack of comments

Module: helpSessions, helpSessionFeedback, filter_helpSessions
Description:

- helpSessions: Display each reservation made by the student user for a help session. Also displays a provide feedback form.

- helpSessionFeedback: After the student user has made a reservation and clicked the provide feedback button, their reservation is displayed including a feedback field for the student to fill.

- filter_helpSessions: Filter reservations made by the student user by month.

Where it fits in:

SDD 2.2.9 Submit Feedback. SDD 2.2.5 Search Reservation. These templates display the feedback field and the reservations made by a student. The SDD modules mentioned above provided the foundation for the design of these templates.

Inspection:

Q: How did you get the feedback box to look so much better
A: CSS to style to form input box

- 1.4.1 Lack of comments
- 1.4.1 Improper Indentation - fixed
- 1.4.1 Improperly capitalized template reference

Module: success, Error templates (400,403,404,500)
Description:

- success: Displayed when the user has edited a help session, created a help session, provided feedback for a help session, edited feedback for a help session.

- Error (400, 403, 404, 500): all errors extend error_base, which loads the background for the error to be displayed with its proper code.

Where it fits in:

The success and error templates are related to most other templates either through successful use of a function (create or edit help sessions for example) or the failed attempt to access a page. The success page is related to the calendarMonth page, where it redirects the user to. The success or error pages are not referenced in the SDD, but are helpful in giving the user feedback whether their attempt to use a function was successful or not.

Inspection:
- Specifically counters 1.4.4 - Lack of feedback to user
- 1.4.1 Redundant code: error_base no longer needed, using base_simple instead

3.1.2 Forms
Module: FormCreateHelpSession
Description:

Django form for holding the field values of a help session. Also customizes those fields.

Where it fits in:

SDD 2.2.3 Store Help Session Data. This is the form for the help session.

Inspection:

TODO TASK: Set up Postgres within our CI/CD environment

- 1.4.1 Lack of comments

Module: FormDeleteHelpSession
Description:

Takes a helpSession ID as an input.

Where it fits in:

SDD 2.2.9 Helper.remove_Help_Session()

Inspection:

- 1.4.1 Lack of comments
- 1.4.1 Redundant code - FormFeedbackButton, FormEditButton, FormDeleteHelpSession are identical

Module: FormEditButton, FormEditHelpSession
Description:

- FormEditHelpSession: Similar to FormCreateHelpSession. It updates the help session object, instead of creating a new one, and it also includes an attendance field for the helper to record attendance.
- FormEditButton: Takes a helpSession ID as input.

Where it fits in:

These forms ensure that we can perform data validation on POST requests and are not referenced directly in the SDD but are necessary to maintain security in the application.

Inspection:

- 1.4.1 Lack of comments
- 1.4.1 Redundant code - FormFeedbackButton, FormEditButton, FormDeleteHelpSession are identical

Module: FormFeedbackButton, FormEditHelpSessionFeedback
Description:

- FormFeedbackButton: Takes a helpSession ID as input.
- FordEditHelpSessionFeedback: a ModelForm that updates the feedback field of a help session.

Where it fits in:

SDD 2.2.3 Store Help Session Data. The model was changed from the original design specification in the SDD - instead of storing feedback in a help session, a reservation model is used. The reservation model points to both a user and a help session instance, and contains the feedback from that user. This form and form button allow users (students) to add feedback to a specific help session that they have attended.

Inspection:

- 1.4.1 Redundant code - FormFeedbackButton, FormEditButton, FormDeleteHelpSession are identical
- 1.4.1 Lack of comments

Module: FormFilterDate
Description:

Takes a month and year as input, and applies it to context for filtering the date.

Where it fits in:

SDD 2.2.4 Display Reservation - this filter is used in other contexts as well, including filtering help sessions in the manage tab. Though not specified directly in the SDD, this module was useful in creating a more user friendly interface.

Inspection:

- 1.4.1 Lack of comments
- 1.4.2 Longevity issue, will only work until 2025

Module: FormRecur
Description:

Form for setting up recurrent help sessions for a specified start date to an end date, based on daily, weekly or monthly frequency.

Where it fits in:

SDD 2.2.3 Store Help Session Data and SRS 2.2.3 HSREQ-4: The system shall allow requests to schedule recurring meetings. This module implements a form to store help session data in recurrence.

Inspection:
- 1.4.2 Crashes/timeouts
- 1.4.1 Lack of comments

3.1.3 Views

BoardmanLab.views:

Module: index, login,

Description:
- index: renders index.html template, requires login
- login: renders login.html template

Where it fits in:

SDD 2.2.1 Sign In/Out. This is the template (view) for the sign in page.

Inspection:

Nothing to report.

Module: calendarMonth

Description:

URL is specified by year, month, day. Shows help sessions on a given day for a particular month. Renders the calendarMonth.html template based on context by year, month, day, and number of sessions.

Where it fits in:

SDD 2.2.9 Calendar view: The calendarMonth view was developed based on this module of the SDD. What we added was displaying the number of sessions for any given day in the calendar.

Inspection:
- 1.4.1 Lack of comments - some added
- 1.4.1 Ambiguous variable names - dayz

Module: calendarDay

Description:

Same format as calendarMonth. Checks for a POST request by the student user, then gets a key from the POST dictionary to create a help session. If POST request includes ‘delete’ it removes the help session from the database. If POST request includes ‘attend’, a new reservation is created then saved to the database, unless it has been created before for the same help session. Renders the calendarDay.html template and displays sessions based on day/month.

Where it fits in:

SDD 2.2.9 Calendar view: The calendarDay view was developed based on this module of the SDD.

Inspection:
- 1.4.1 Ambiguous variable names - key/pk
- 1.4.1 Inconsistent case of variable names

Module: helpsessions

Description:

Displays reservations for a student user, filtered by day/month. Renders the helpSessions.html template. If the user wants to access feedback, it renders the helpSessionFeedback.html for a particular reservation.

Where it fits in:
SDD 2.2.4 Display Reservation: This view displays the reservations for a student user just as described by this SDD module. We also added filters by day/month to it.

Inspection:

- 1.4.1 Inconsistent case of variable names
- 1.4.1 Lack of comments - #TODO comments still left in code

Module: managehelpsessions
Description:

Displays the help sessions that the helper owns. Can filter help sessions by day/month, then renders the manageHelpSessions template. The helper can delete a help session, and also edit it, in which case it renders the editHelpSession template.

Where it fits in:

SDD 2.2.9 Helper.add_Help_Session(), Helper.edit_Help_Session(), Helper.remove_Help_Session(). This view allows the helper to create, edit, and delete a help session just as the SDD module describes. We also added filters by day/month.

Inspection:
Q: Do we need to specify both admin and helper
A: Yes we should specify both

- 1.4.1 Inconsistent case of variable names
- 1.4.1 Lack of comments
- 1.4.1 Redundant code

Module: helpSessionFeedback
Description:

Saves the feedback into the database, and renders success.html template. If there is no POST request, render the helpSessionFeedback template.

Where it fits in:

SDD 2.2.9 Submit Feedback: This view allows the student to enter text into the feedback box and submit it, which saves it to the database. What we also added was a success.html template to let the student know that their submission went through.

Inspection:

- 1.4.1 Lack of comments - #TODO comment left in
- 1.4.1 Ambiguous variable names - helpSessionFeedbackEdit

Module: editHelpSession
Description: When editing a help session from the calendar day or manage tab, users can populate the fields for a help session, then the session is saved to the database and the success.html template is rendered. If there is no POST request, render the editHelpSession template.

Where it fits in:

SDD: 2.2.9 Helper.edit_Help_Session(): This view shows the interaction between users and the database, saving information fields and being able to manipulate fields after being saved in the same way as the SDD model.

Inspection:
- 1.4.1 Redundant code - No longer redirects to manage help session, now redirect to success
- 1.4.1 Lack of comments - some added
- 1.4.2 POST request reference errors - Always will be a POST request but code implies it could be something else

Module: createHelpSession
Description:
Initializes the date to the current date for the help session. The helper can then create a help session by filling the help session form, then save it to the database and render the success.html template. Helper may specify if the help session is recurrent, in which case recurHelpSession.html is rendered, then the helper can select the recurrence options. Based on the options selected, multiple help sessions may be saved to the database by this view. The user can cancel from the recur view into the manage help session view.

Where it fits in:
SSD 2.2.9 Helper.add_Help_Session(). We also added a success.html template to let the helper know that the help session was created successfully, as well as recurrence options for the helper, so that multiple help sessions can be created based on a daily, weekly, monthly basis.

Inspection:
- 1.4.1 Redundant code - unnecessary context variables

Module: recurHelpSession
Description: N/A

Where it fits in:
SRS 2.2.3 Functional Requirements HSREQ-4: The system shall allow requests to schedule recurring meetings.

Inspection:
- 1.4.1 Redundant code - removed

Module: success, error, errors
Description:
- success renders success.html template
- error renders error.html template
- error_404 renders error_404.html template…

Where it fits in:
The success or error requests are not referenced in the SDD, but are helpful in giving the user feedback whether their attempt to use a function was successful or not.

Inspection:
- 1.4.1 Redundant code - removed old error request

Reservations.views
Module: new_help_session
Description:
This function was moved to the boardmanlab.views and was renamed and refactored.
Where it fits in:

N/A

Inspection:

- 1.4.1 Redundant code - No longer needed

Users.views

Module: update_user_social_data

Description:

This request was not included in our SDD, but we found it was necessary to store some user information, and thought it might be necessary to update that information.

Where it fits in:

N/A

Inspection:

- 1.4.1 Redundant code - Not in operation or needed

Reports.views

Module: reports.course_freq_csv

Description:

Generates a count of reservations per topic

Where it fits in:

SDD 2.2.6 Generate Report

Inspection:

- Good comments

Module: reports.helper_freq_csv

Description:

Generates a count of reservations per helper

Where it fits in:

SDD 2.2.6 Generate Report

Inspection:

- Good comments

Module: reports.time_freq_csv

Description:

Generates a count of reservations per time window

Where it fits in:

SDD 2.2.6 Generate Report

Inspection:

- Good comments

3.1.4 Models

Module: helpSession

Description:

models the helpSession table in our database.

Where it fits in:

SDD 3.1 Database Descriptions

Inspection:

- 1.4.1 Improperly cased model name

Module: Reservation

Description:
  models the Reservation table in our database.
Where it fits in:
  SDD 3.1 Database Descriptions
Inspection:
  Nothing to report.

Module: User
Description:
  models the User table in our database.

Where it fits in:
  SDD 2.2.9 User Hierarchy

Inspection:
  Nothing to report.

Module: Topic
Description:
  models the Topic table in our database

Where it fits in:
  SDD 3.1 Database Descriptions

Inspection: Nothing to report.

3.1.5 Miscellaneous

Module: calendar_tags.py
Description:

- get_month(): returns current month, current year?
- get_day_string(year, month, day): returns string for a particular date
- gen_month_string(): returns string for a particular month
- increment_day(), decrement_day(): returns the date after performing the increment/decrement
- increment_month(), decrement_month(): next/previous month as an int
- increment_month_alt(), decrement_month_alt(): next/previous month as an int
- increment_year(), decrement_year(): next/previous year as an int
- add_minutes(): returns a string of time + duration
- time_format(): formats time into a string object
- set_pk(): N/A
- index(): indexes a list from an indexable list and an index
- get_student_sign_up(): returns number of students signed up for a particular help session
- get_freq(): checks if some parameter matches the string ‘days’

Where it fits in:
  Tags store html code fragments, which are hidden from the clients (not included in SDD). The SDD was written before the team had knowledge of the conventions of the Django framework and did not know to include certain developmental pages.

Inspection:

- 1.4.2 get_month does two things.

- 1.4.1 Lack of comments - comments added’
- 1.4.1 Redundant code - decrement_month and decrement_month_alt similar, possibly redundant
- set_pk() - complex
- 1.4.1 Redundant code - get_freq

Module: urls.py
Description:
- boardmanlab.urls: contains the paths for each view inside the boardman app
- Inhouse.urls: contains the paths for each view inside our django project

Where it fits in:
URL paths for our Django apps (not included in SDD)

Inspection:
- 1.4.1 Redundant code - some urls unused, boarmanlab.urls is potentially unneeded
- 1.4.1 Inconsistent casing - helpsessions, csv urls not camelcase

Module: settings.py
Description:
The configuration file that holds all the values for the django project to work such as: apps, database settings, static files, configurations, external APIs, authentication user models, middleware, social account providers.

Where it fits in:
Django configuration file (not include in SDD). The SDD was written before the team had knowledge of the conventions of the Django framework and did not know to include certain developmental pages.

Inspection:
- 1.4.3 Unprotected information - secret key in plain text, database login and passwords in plain text, debug mode is on
- 1.4.1 Lack of comments
- Lots of security features located here (middleware)
- Static files not ready for deployment in current state
- 1.4.1 Redundant code - Social Account Provider (line 161) possibly redundant

Module: style.css
Description:
- Dynamic content
- Styles the calendar content, sidebars, forms, scrollbar
- Styles the headers, footers, head-strips, logos, arrows, buttons, tabs, left and right content
- Styles the login, logout container, errors
- Styles the help session container

Where it fits in:
Styling the elements of our templates (not included in SDD). The SDD was written before the team had knowledge of the conventions of the Django framework and did not know to include certain developmental pages.

Inspection:
It is immediately apparent that the code needs serious refactoring. The code meets every possible code convention defect.
- 1.4.1 Redundant code - Insane amount of redundant code
- 1.4.1 Inconsistent casing - Kebab-case intended, many names do not abide this convention

- 1.4.1 Ambiguous variable names - Many names are not easy to locate in html
- 1.4.1 Lack of comments
- 1.4.1 Similar variable names - Many names difficult to tell apart
- 1.4.1 Code not linted
- 1.4.1 Unused code segments - Many divs initialized but are redundant or otherwise unnecessary
- 1.4.4 CSS incompatibility
- 1.4.4 Style inconsistencies

3.1.6 Unfinished Modules

Module: profile, editProfile (views and templates and possible model configuration)

Description:

Shows the user their current profile and an option to edit things like: interests, course list, preferred name, pronouns etc.

Projected Completion Date:

March 11, 2022

Module: feedback, reports (views and templates)

Description:

A view and template that generates an overview of data being collected as well as options to download raw data for processing.

Projected Completion Date:

March 11, 2022

Module: ExtendedFilter

Description:

A view and template that filters information displayed to the user by datetime picker form.

Projected Completion Date:

March 11, 2022

Module: helpSessionDetailView

Description:

A view and template that shows more fine grained detail about a help session, including current reservations, date of sign up from students (ordered for first come first serve).

Projected Completion Date:

March 11, 2022

Module: unitTest

Description:

unit tests

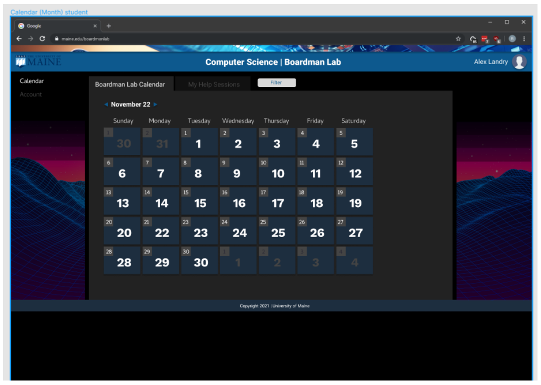
Projected Completion Date:

March 20, 2022

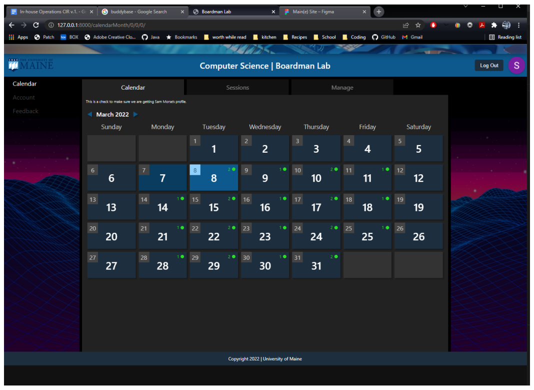
3.2 Design Evolution

3.2.1 UI Change

Planned module interface:



Actual module interface:



The actual module interface is loyal to the planned interface, but there were several design changes that were made in order to increase usability. For instance, the actual interface displays if there are help sessions on a given day, and how many. There was a good amount of dogfooding involved during development which revealed some flaws in design such as the inability to filter help sessions by date - this was added in the actual interface.

3.2.2 Data Structures

Planned data structures: A single reservations table containing all the data
Actual data structures: reservations, help sessions, user, topic tables

The planned data structures were insufficient to store all the data and have the proper relationships that we wanted to have for the actual data.

3.2.3 I/O

Planned user and database I/O:

- User Input: Help Session Fields, Login credentials, Filter options for calendar views, Filter options for help sessions, Feedback for help session, Profile bio, Profile class list
- Database Input: reservations, help sessions, user, topics
- Database Output: reservations, help sessions, reports

Actual user and database I/O:

- User Input: Help Session Fields, Login credentials, Filter options for calendar views, Filter options for help sessions, Feedback for help session, Profile bio, Profile class list
- Database Input: reservations, help sessions, user, topics
- Database Output: reservations, help sessions, reports

Our actual user and database I/O corresponds to what we planned from the beginning.

3.2.4 Algorithms
Planned Algorithms: Queries to help session based on frequency by helper, topic, time window. Queries to help session based on user/date.
Actual Algorithms: Queries to help session based on frequency by helper, topic, time window. Queries to help session based on user/date.

We made no changes to the queries we intended to use, only on the structure of the database tables and the methods with which we invoke the queries.

3.2.5 Tradeoffs
Tradeoffs:

- Time vs Memory
- Compressed vs Uncompressed Data
- Re-rendering vs Stored Images

4. Defects

This section outlines the known defects in our code. It includes the module the defect originated from, description, and defect category.

4.1 Defects Expanded

Name: Recur crash (FormRecur, recurHelpSession, createHelpSession)
Description: Recur can create absurd amounts of help sessions either through malicious action or a simple type (creating a daily session for one year but typing “3023” instead of “2023” will create 365,000 help sessions. This can cause intense server lag, system unavailability, and untested display issues.
Category: Potential system failure

Name: Unauthorized sign-in redirect
Description: Sign in page sometimes redirects to sign up page (should not be allowed)
Category: Security vulnerability i.e. sql injection

Name: Longevity issue with FormFilterDate (FormFilterDate)
Description: Only lasts until 2025 due to hard coded variables.
Category: Maintenance and future proofing

Name: get_month does two things (calendar_tags.py)
Description: get_month, a possibly unreferenced call, has two functions tied to it.
Category: Maintenance and future proofing

Name: settings.py plain text secrets (settings.py)
Description: settings.py file has secret keys and database login information in plain text.
Category: Security vulnerability

Name: Pre-deployment issues (settings.py)
Description: Debug mode, other pre-deployment settings pose a massive security vulnerability if left unchanged when deploying.

Category: Security vulnerability

Name: Inconsistent casing (views, templates, css)
Description: Several of our variables, views, elements, models, etc. names are inconsistently cased.
Category: Coding convention violation - inconsistent casing

Name: Lack of comments (templates, views, et al.)
Description: The majority of our files have severely lacking or nonexistent comments.
Category: Commenting violation - missing comments

Name: Redundant code (urls, templates, views, css, et al.)
Description: Several functions, css elements, html elements, etc. are redundant and could be trimmed for readability and ease of future maintenance.
Category: Maintenance and future proofing

Appendix A – Team Review Sign-off

This section denotes that all members of the In-House Operations development team have reviewed this document and agree on its content and format. If any minor disagreements in content and format are present, they are listed below the development team signatures.

Name: Klei Bendo
Date: 3/9/22
Signature: [illegible]
Klei Bendo (Mar 9, 2022 19:41 EST)

Name: Jack Brisson
Date: 3/9/22
Signature: John h Brisson
John h Brisson (Mar 9, 2022 19:39 EST)

Name: Alex Landry
Date: 3/9/22
Signature: [illegible]
Alex landry (Mar 9, 2022 19:37 EST)

Name: Samuel Morse
Date: 3/9/22
Signature: [illegible]
Samuel Morse (Mar 9, 2022 19:35 EST)

Name: Aaron Schanck
Date: 3/9/22
Signature: [illegible]
aaron schanck (Mar 9, 2022 20:08 EST)

Name: Forrest Swift
Date: 3/9/2022
Signature: Forrest Swift
Forrest Swift (Mar 9, 2022 20:03 EST)

Minor Disagreements in Content and Format (if any):

Appendix B – Document Contributions

This section denotes the contributions of each team member to this document. It includes the sections each member worked on and their percentage contributed in parentheses.

Name: Klei Bendo
Sections worked on (percentage contributed to document):
Sections 2, 3, 4 (16%)

Name: Jack Brisson
Sections worked on (percentage contributed to document):
Sections 1, 2, 3 (16%)

Name: Alex Landry
Sections worked on (percentage contributed to document):
Sections 1, 2, 3 (16%)

Name: Samuel Morse
Sections worked on (percentage contributed to document):
Sections 1, 2, 3, 4 (16%)

Name: Aaron Schanck
Sections worked on (percentage contributed to document):
Sections 1, 2, 3, 4 (20%)

Name: Forrest Swift
Sections worked on (percentage contributed to document):
Sections 2, 3, 4 (16%)