FIG NEUTRONS
CSE 3901 Final Project

Anuj Kothari - Tanner Marshall - Jeff Morhous – Sneha Srinivasan

Content

- Requirements/ Implementation
- Database Schema Overview
- Demo
- Challenges
- Future Work

Project Requirement - Overview

Main Features

1

Student Application

- Enter CSE course grades
- Collect interested courses
- Fill out availability
- Edit grades/availability

2

Instructor Recommendations

- Adding endorsements for students
- Requesting a specific student grader

3

Admin Dashboard

- Display courses that require grader
- View instructor recommendations
- Select from student applicants

Starting Point

How did we in general approach this project?

- Discuss requirements/features and initial workflow of application
- Determine necessary models and design database schema
- Split up necessary features for each role
- Basic functionality → role-specific features
- Integration/testing

How did we approach the requirements that we didn't really know?

- Talk through potential options and solutions
- Request clarification

Project Implementation

- Used RoR, jQuery, and Bootstrap
- Scrape CSE course information initially
- MVC for student, instructor, and admin
- Login and registration system
  - Built in methods on Model, SessionHelper
  - Admin account creation?
- Student application form
  - Dynamic input form for adding grades,
  - Weekly-Schedule Plugin
- Instructor recommendation form
  - Editing different recommendations
- Admin Dashboard
  - Considers multiple factors and presents options

Content

- Requirements/Our Implementation
- Database Schema Overview
- Demo
- Challenges
- Future Work

Starting Point

How did we approach setting up the database?

- Determine what elements would require their own tables to organize their information
- Determine how those tables could be most efficiently connected to implement the features we needed

Overview: Tables

Admins

Availabilities

Courses

Graders

Instructors

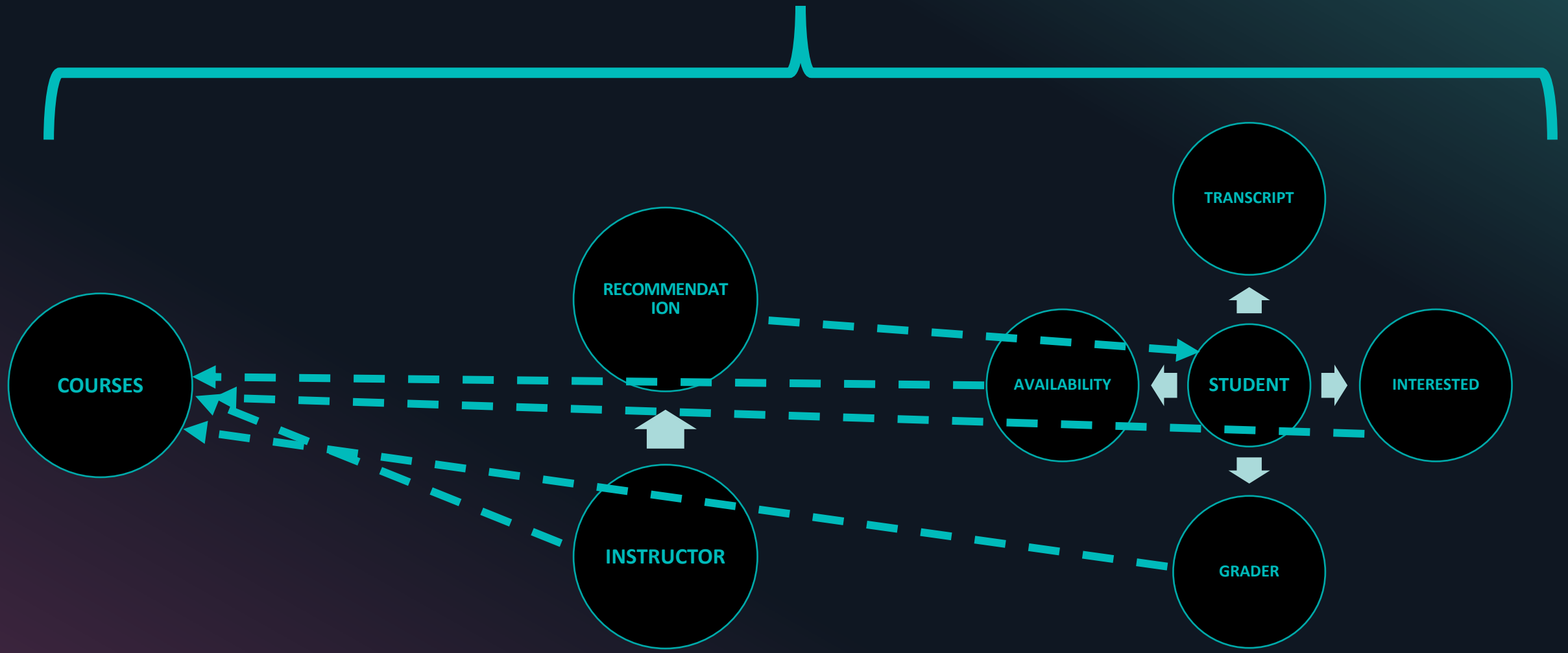
Interesteds

Recommendations

Students

Transcripts

ADMIN


TRANSCRIPT
RECOMMENDATION
COURSES
AVAILABILITY
STUDENT
INTERESTED
INSTRUCTOR
GRADER

Courses

- Populated by web scraping!

Data Type	Name
String	department
Datetime	created_at
Datetime	update_at
String	section_number
Integer	instructor_
String	course_number
String	location
String	start_time
String	end_time
String	days
Boolean	is_lab
Boolean	need_grader
Boolean	have_grader

Instructors

- Created by the user in the create account page.

Data Type	Name
String	first_name
String	last_name
String	email
String	password_digest
Datetime	created_at
Datetime	updated_at

Recommendations

- Created from the instructor pages.

Data Type	Name
String	recommendation
Datetime	created_at
Datetime	update_at
String	student_email
Integer	instructor_id
String	course_number
String	first_name
String	last_name
Boolean	request

Students

- Created by the user in create account page.

Data Type	Name
String	first_name
String	last_name
String	email
String	phone
String	password_digest
Datetime	created_at
Datetime	updated_at

Availabilities

- Created in the student application.

Data Type	Name
String	hour
Integer	student_id
Datetime	created_at
Datetime	updated_at
String	day

Interesteds

- Created in student application.

Data Type	Name
String	department
String	course
Datetime	created_at
Datetime	updated_at
Integer	student_id

Graders

- When a student is selected to grade (from admin), they're placed in this table.

Data Type	Name
Integer	semester
Datetime	created_at
Datetime	updated_at
Integer	course_id
Integer	student_id

Transcripts

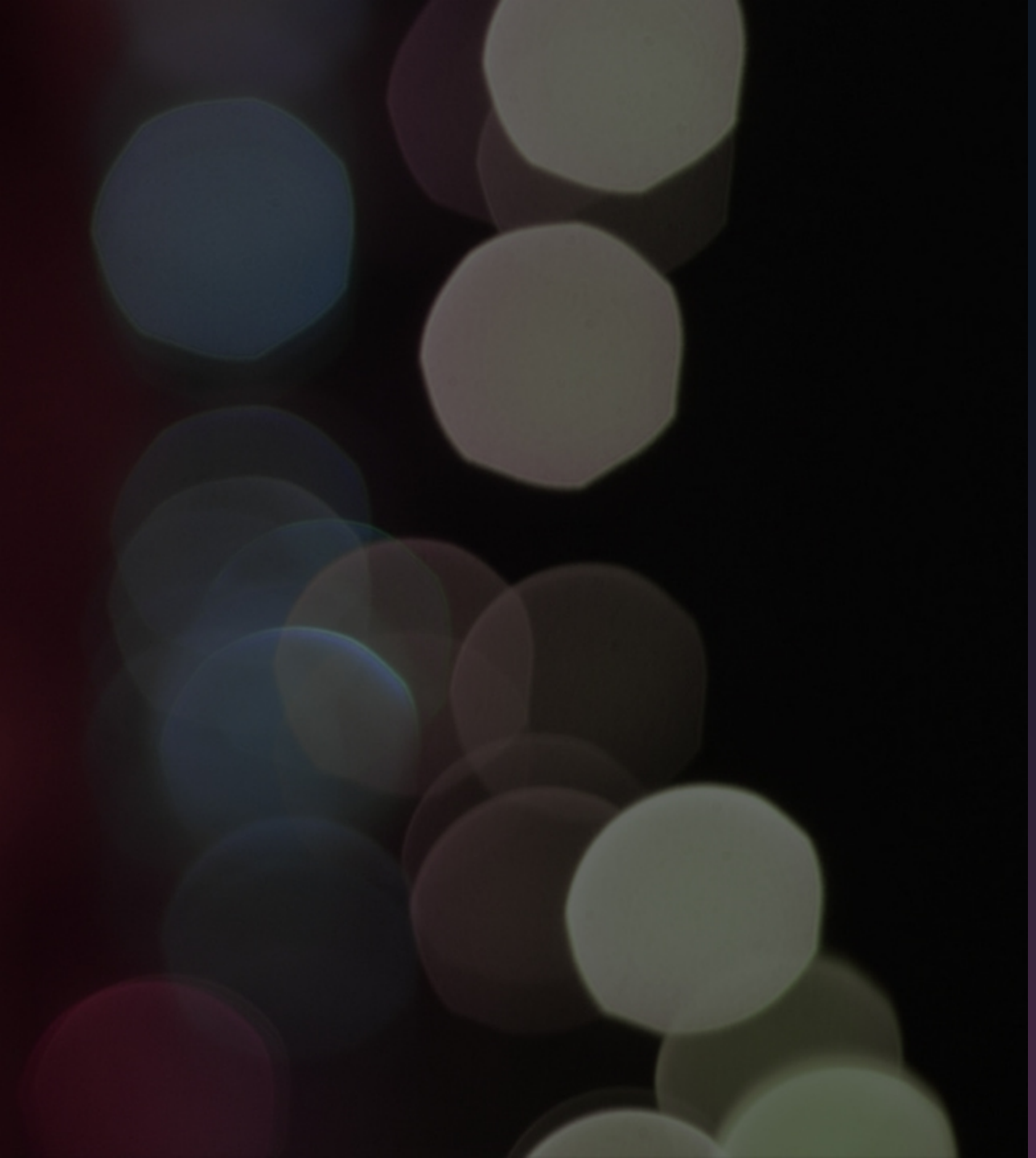
- Created in student application.

Data Type	Name
Integer	grader
Datetime	created_at
Datetime	updated_at
Integer	student_id
Integer	course_id

Admins

- Created outside of the application (Rails Console)

Data Type	Name
String	email
String	password_digest
Datetime	created_at
Datetime	updated_at

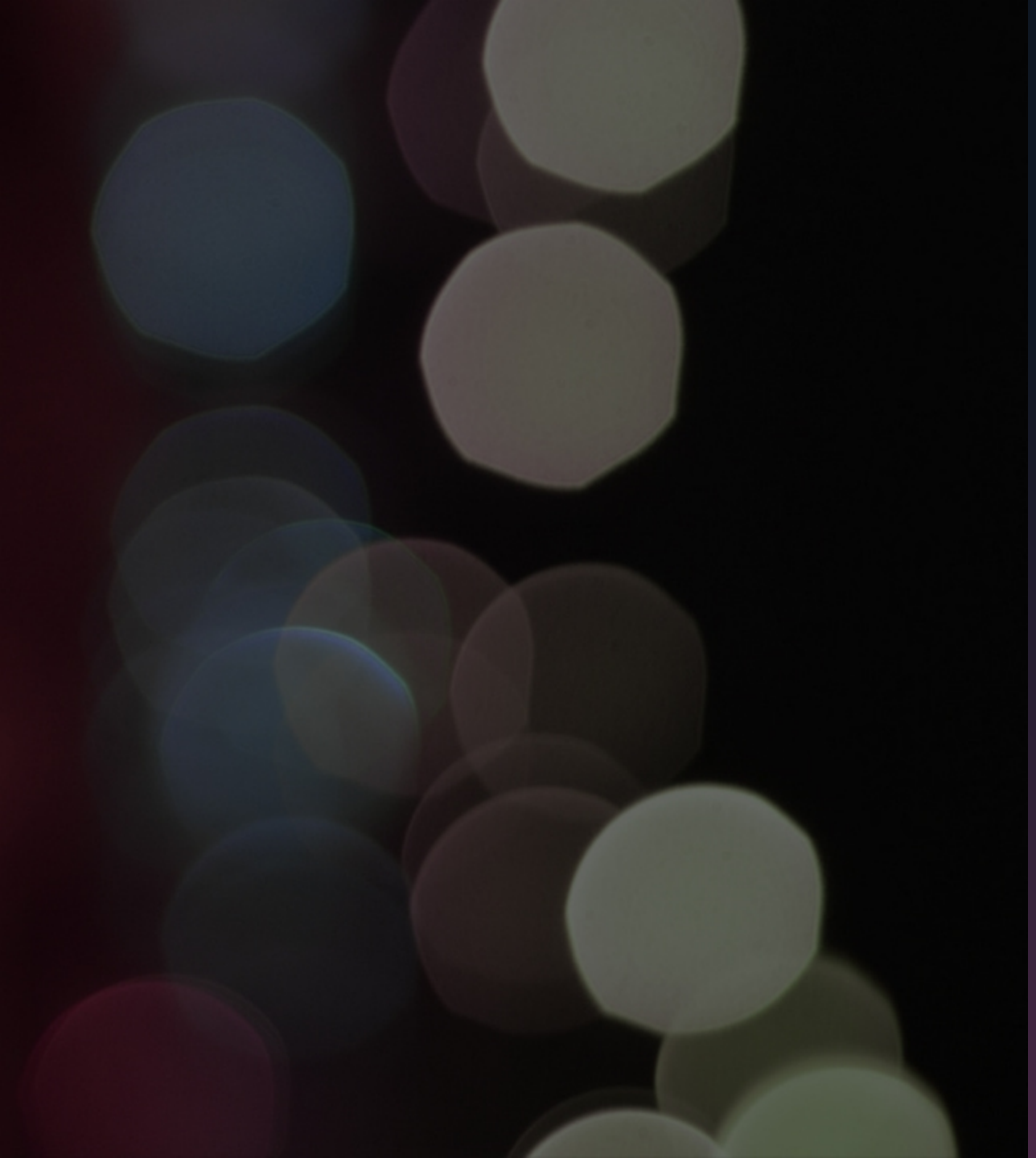


Content

- Requirements/Our Implementation
- Database Schema Overview
- Demo
- Challenges
- Future Work

DEMO

Click here



Content

- Requirements/Our Implementation
- Database Schema Overview
- Demo
- Challenges
- Future Work

Challenges

Modifying existing ruby gems to tailor them to our specific needs.

Gems exist to make our lives a little easier, but they're usually very general and modifying them to display exactly what and how we wanted them to required lots of time and a deeper understanding of the gem than we usually had

Needing to add a feature that would interfere with elements that had previously been implemented

Often, we would go back to change the way things were displayed or give users the ability to have more control over their information, and this caused us to have to rework a lot of previously completed work

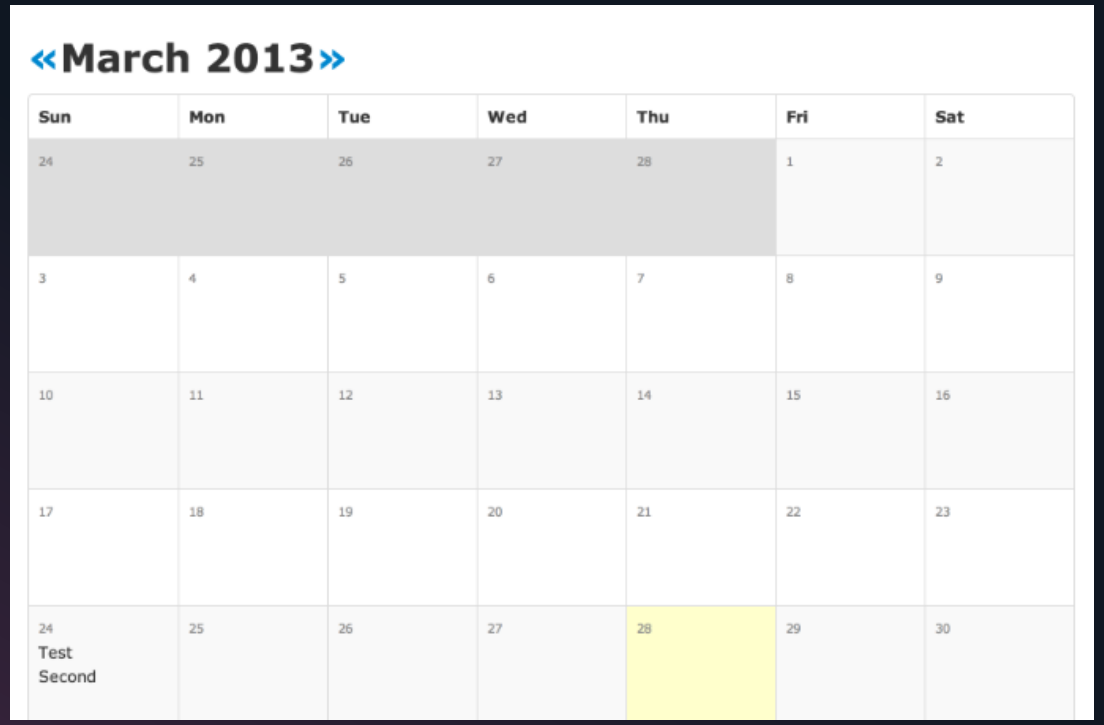
Incorporating custom JavaScript and Jquery to handle events in the view.

In order to render things in the way that we wanted to, we had to incorporate custom code, and this was not easy to figure out. Lots of trial and error and reading

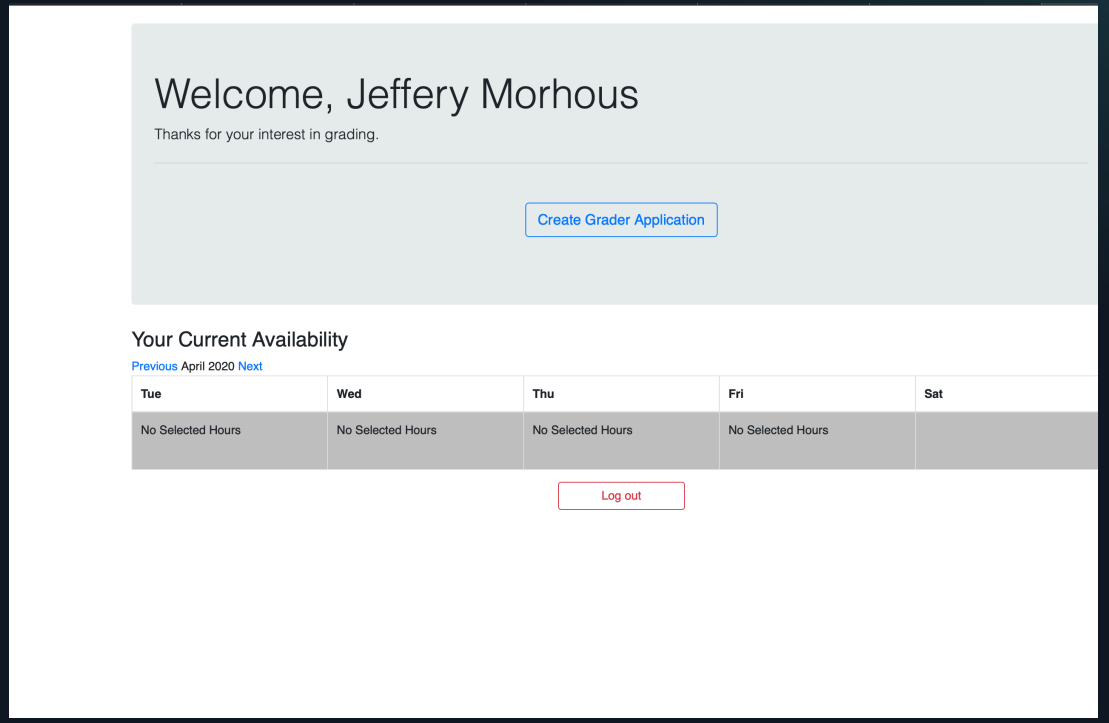
Creating a database schema that was easily usable across all the views and controllers – and understanding how to use the controller to connect the two parts

Several times we had to modify our database because the way/type we had originally determined columns to be was not compatible with all of the uses that we had for it , and the controller had to knit everything together which was hard to understanding going back and forth

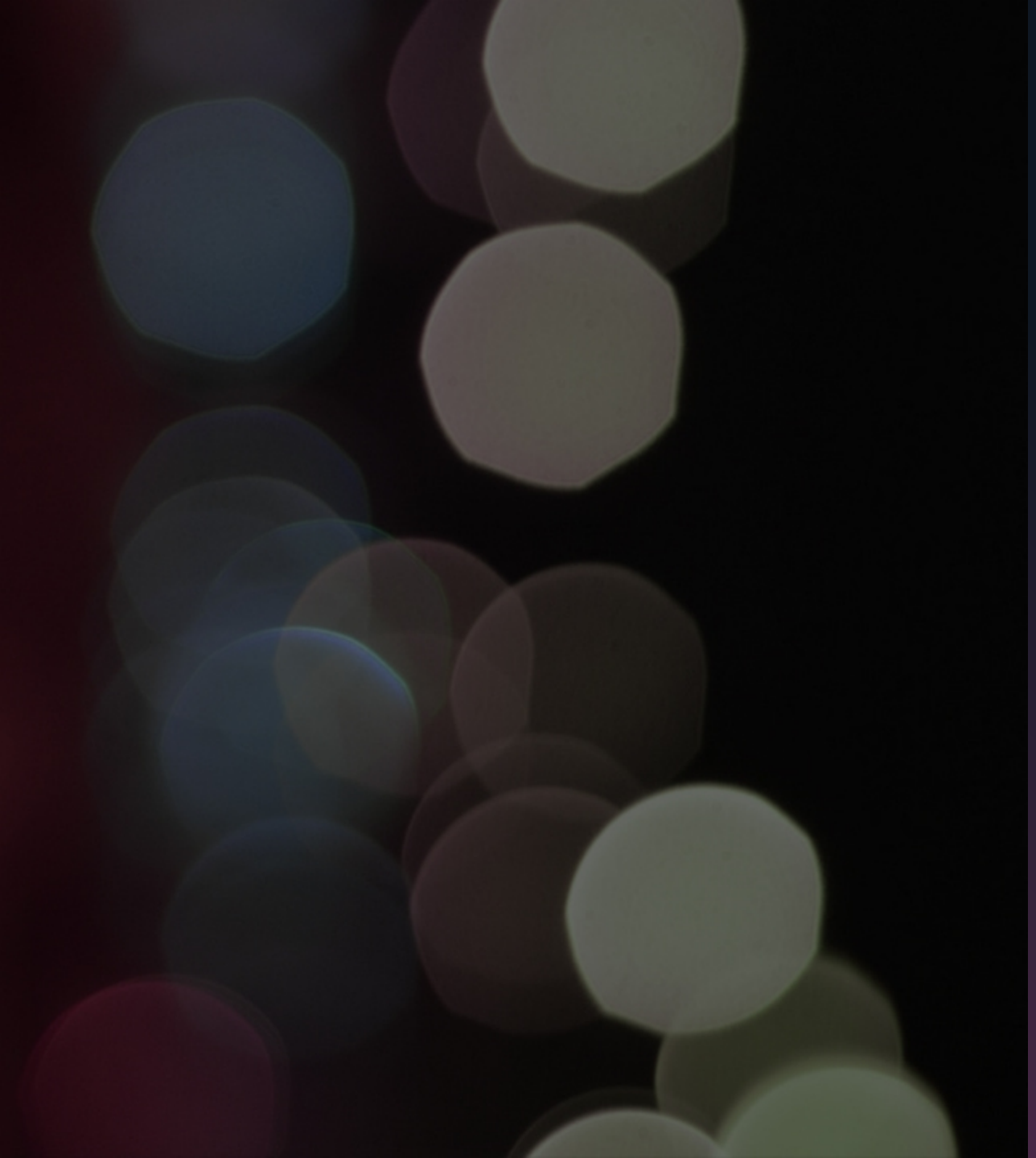
Before – The Original



After – Our Modifications



“Team work makes the dream work”



Content

- Requirements/Our Implementation
- Database Schema Overview
- Demo
- Challenges
- Future Work

What do we think we could have done better?

- Started With Testing
- Planned out the database better (Specifically minor details)
- Cleaned up organization of the entire application
- Made our solutions cleaner and more efficient instead of being as hacky as they are

In general, we would like to add….

- The ability for the admin page to select the best candidate for a class for you
- Grader Evaluations
- Use to find people tool from OSU to validate students and to auto populate information about them without them manually putting it in.
- Possible DUO logins?