**Exercise 4: Solution Reflection**

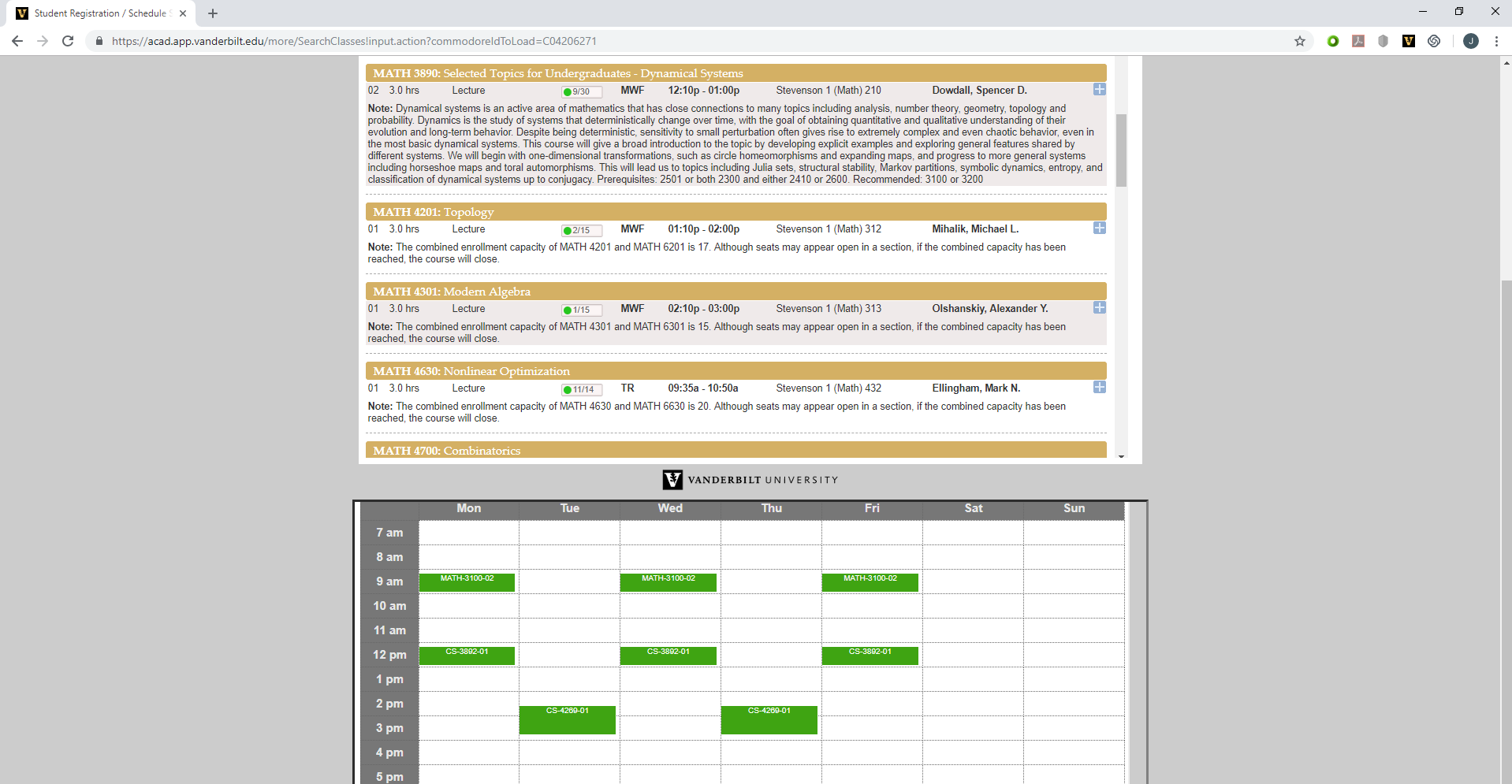
Last Updated 9.4.2018

Name: Aneesha Dasari, Irfaan Khalid, Jacob Lundy, Payton O’Daniels

Date: Due 12/11/2018

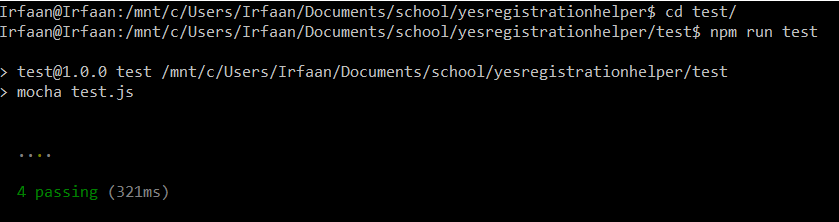
Course: Principles of Software Engineering, CS 4278

**In the box below, show your prototype, either with a photograph or a sketch. Then, include a full, written description of your prototype.**



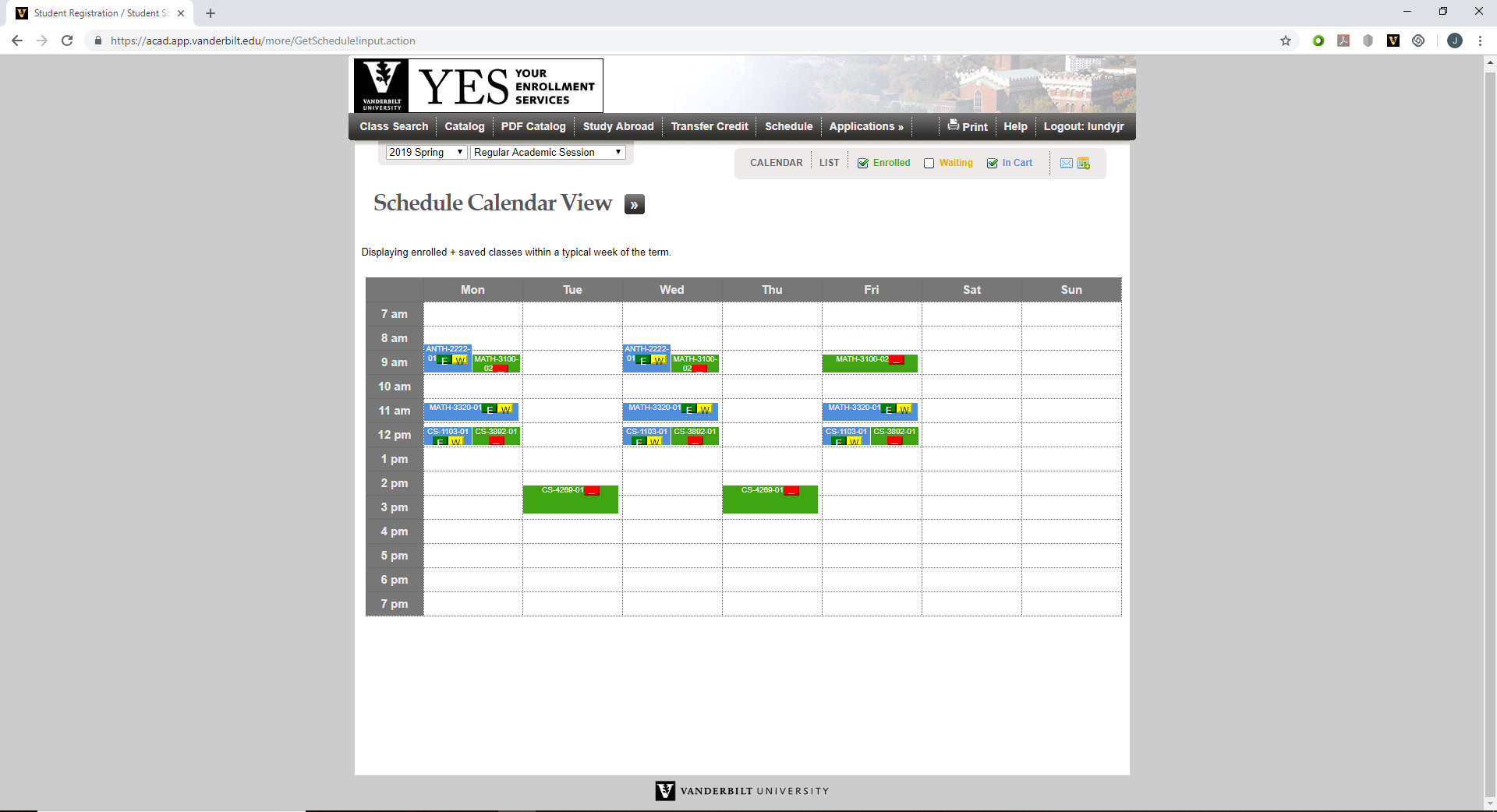
The initial prototype of the project was the basic functionality of allowing the schedule screen to load within an iframe. We initially had it where the user had to manually press a button to load the YES page, and in the user-review process, we decided it was more effective and useful to have the page load automatically underneath select YES domains. Several changes have been made to the original prototype design, including disabling clicking and scrolling, and rearranging the position of the schedule. Much of these changes were made with user-feedback, to make the application more usable. Without access to any support from the YES team, the code mostly focuses on manipulating the DOM, simulating user interaction with the page. In order to get the schedule screen to load beneath the normal page, we load the page in the background, and then extract the element containing the schedule, which we then present within a frame we load below the screen. All the other functionality in the project works in a similar manner, simulating functionality that the user could implement, trying to save time and offer convenience to the user.

**In the box below, show your test of your prototype, either with photographs or sketches of your testing. Then, include a full, written description of your testing.**



Tests were implemented using frameworks including Chai, Cheerio, Mocha and Sinon-Chrome. Chai is an assertion framework that pairs well with mocha, allowing assertion-based testing. Mocha is a JavaScript testing framework that seamlessly allows the creation of asynchronous testing scripts. Cheerio is an implementation of Jquery that allowed for the fast and efficient loading of DOM elements, so that we could have our application run scripts manipulating the elements. We would then use Chai to assert that the manipulations matched the expected behavior of our application. Sinon-Chrome is a testing tool designed for use with web browser extensions, mocking browser API calls, allowing chrome extensions to perform their intended functionality without loading them into a web browser. This assertion-based testing was used to ensure that our Chrome Extension provided the correct functionality. Most of our tests focused on validating that DOM manipulations occurred as expected, and those changes were successfully propagated across the various webpages loaded internally by our application. Because our application’s main function is DOM manipulation, ensuring that the DOM manipulations we expected were precisely what the application performed allowed for thorough unit and integration testing of our application.

**In the box below, describe in words or pictures your final solution.**



**Finally, tell us about at least one change you made to your prototype and why you made that change.**

One of the major ways we implemented changes to our prototype, including the changes we mentioned previously in the prototype section of this document, was to implement additional functionality that could benefit the user. Obviously when the user is already on the schedule screen, it’s not particularly useful to load a copy of the schedule beneath it. Instead, we added functionality that allowed the user to interact with courses they are enrolled in, waitlisted on, or have in their cart. The functionality provided allows them to conveniently drop, waitlist, or enroll from their project from the main screen. This functionality is also implemented in a way similar to the enrollment screen. When on the schedule screen, the enrollment screen and cart screen is loaded in a hidden iframe in the background. When the user interacts with the buttons, scripts are ran that appropriately fill out information on the hidden iframe, simulating a drop or enrollment action taken by the user on the cart page. Other functionality that we added was an “enroll all” button. When the user is on the cart page, they press the Chrome extension icon, which will automatically select “waitlist if full” on all the classes in the cart and attempt to enroll the user. This functionality was added because of the frequent user-story we heard related to YES, where important classes become full nearly immediately after enrollment opens. By allowing the user to enroll faster, they can both save time, and potentially secure a spot in class that a few seconds delay could cost.