CourseCraft

Surreal Engine - Mikita Belausau, Ryan Coslove, John Hoban, Dorion Hobot, Nicholas Makar

1. Proposed System

In one centralized website, Deans, Professors, and Students will be able to access multiple features that Canvas and Webreg provide separately. Deans will be able to create courses. Professors will be able to register to teach those courses, assign the courses' meeting/lecture times, create assignments, input grades, create announcements, and allow overall clean professor-to-student communication. Students will be able to register for the courses, view their current GPA, calculate potential GPAs, view their grades, submit assignments, receive alerts and announcements. There will be a navigation bar at the top of each user's home page to transfer between different portals (example - from home to Degree Navigator to view GPA or from home to Course Management to view assignments for a class). The creation of new assignments or grades will have the system automatically contact the students of this notification.

This will be created with Dash and interact with a database on a virtual machine. The database will hold information such as user login credentials, students enrolled in classes, courses professors are teaching, student and course majors, etc.

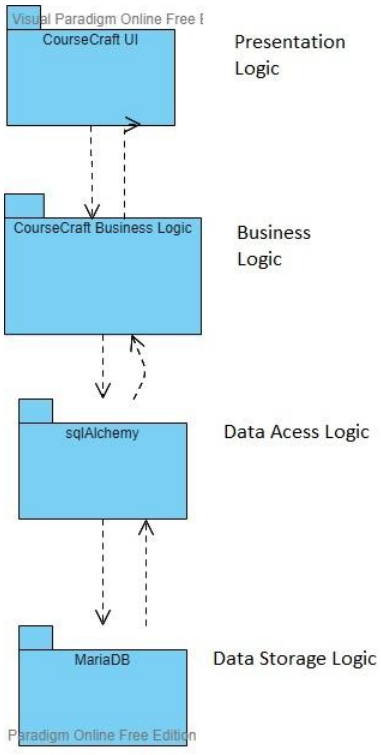
2.1 Functional Requirements

- A user will be able to securely enter valid login credentials in reference to a database (held on the virtual machine) to log in to a user's appropriate and personalized home page.
- A user will be able to navigate a homepage that allows a user to perform user-specific permissions and contains access to the user's personal information as referenced in a database.
- A user with the Dean role will be able to access the Course Registration page and create courses that will be stored in the database as accessible courses to professors and students.
- A user with the Professor role will be able to access the Course Registration page and register to teach a course in the database created by a Dean and assign the class time schedule, saved in the database as accessible to students.
- A user with the Professor role will be able to access the Course Management page and create announcements, create assignments, and input grades for their registered course(s) that will all be saved in the database and will notify the students registered for that course of the professor's action.
- A user with the Student role will be able to access the Course Registration page and register for courses from the database to create an appropriate schedule within the university's class-time, credit-intensive, and major-intensive limits.
- A user with the Student role will be able to access the Course Registration page and view their course schedule.

- A user with the Student role will be able to access the Course Management page and access their course's pages, view and submit assignments, view grades, all referenced and saved in the database.
- A user with the Student role will be able to access the Degree Navigator page and view their current GPA as referenced in the database, as well as calculate their potential GPA.
- A user with the Student role will automatically receive notification emails whenever a new assignment has been created, a grade has been imputed or changed, or a professor has sent an announcement, for their respective courses.

2.2 Non-Functional Requirements

- A user will be able to enter valid credentials safely stored in a database and be securely redirected to their appropriate home page and information without risk of losing private information, accessing another user's information, or accessing inaccurate information.
- A user will be able to access the software and their information at any moment in time, 24 hours a day, 365 days a year, without risk of failure.
- The system will generate and maintain identification numbers for users, assignments, announcements, and grades.
- The system shall support 1,000 simultaneous users at all times.
- The system shall run on any browser client, regardless of operating system.



Test design:

1. A user will be able to securely enter valid login credentials in reference to a database (held on the virtual machine) to log in to a user's appropriate and personalized home page.

I'm not sure how to automate a test that shows personalization of the home page, so we'd probably just boot up the webapp and actually type in correct credentials and view it ourselves, and incorrect ones to see if it doesn't log you in by mistake.

2. A user will be able to navigate a homepage that allows a user to perform user-specific permissions and contains access to the user's personal information as referenced in a database.

    Once again seems like end-to-end testing and I'm not sure how to do that. Would have to be a manual test.

3. A user with the Dean role will be able to access the Course Registration page and create courses that will be stored in the database as accessible courses to professors and students.

    The first half of this is end-to-end testing, but the second half you can create a unit test. Use the same logic that deans use to create a class, then make sure that class exists in the db after the logic is done running, if it does then it passes the test case.

4. A user with the Professor role will be able to access the Course Registration page and register to teach a course in the database created by a Dean and assign the class time schedule, saved in the database as accessible to students.

    You can do the same thing as deans creating a class. Just make a test that uses a specific professors name and a course you select, then query the db and make sure that its updated. Also make sure that there are no scheduling conflicts as well in your test.

5. A user with the Professor role will be able to access the Course Management page and create announcements, create assignments, and input grades for their registered course(s) that will all be saved in the database and will notify the students registered for that course of the professor's action.

    The test follows a similar idea, use the same logic used to create assignments, input grades and then make sure the DB recognizes those changes. For announcements we have a test email address used to test emails. So when they create the announcement check the email to ensure that it was sent.

6. A user with the Student role will be able to access the Course Registration page and register for courses from the database to create an appropriate schedule within the university's class-time, credit-intensive, and major-intensive limits.

This test will have to account for scheduling conflicts, and make sure that it updates the DB accordingly. Can be done using unit testing.

7. A user with the Student role will be able to access the Course Registration page and view their course schedule.

Seems like an eyeball test because I'm not entirely sure how you can compare graphs.

8. A user with the Student role will be able to access the Course Management page and access their course's pages, view and submit assignments, view grades, all referenced and saved in the database.

End-to-end testing where I'm unsure of how to automate this again. Would most likely be tested by logging in as a test student and making sure everything looks as expected.

9. A user with the Student role will be able to access the Degree Navigator page and view their current GPA as referenced in the database, as well as calculate their potential GPA.

Calculating potential GPA can be a unit test, if we figure out how to automate stuff we could also read the info off of their homepage and determine if the stored gpa value is the same as displayed. Otherwise it'd have to be a manual check.

10. A user with the Student role will automatically receive notification emails whenever a new assignment has been created, a grade has been imputed or changed, or a professor has sent an announcement, for their respective courses.

This is most likely going to be a manual check, because we have a test email address for these emails.

11. A user will be able to enter valid credentials safely stored in a database and be securely redirected to their appropriate home page and information without risk of losing private

Since this is a non-functional requirement, the testing here isn't really possible. But our knowledge on how to safely store credentials will ensure that this requirement is met.

12. A user will be able to access the software and their information at any moment in time, 24 hours a day, 365 days a year, without risk of failure.

If we figure out how to automate tests, then possibly we could have a ping to the server 24/7 for an entire day, but I don't think we can safely ensure that it'll be able to run 365 days a year. We'd have to have an oncall of some kind to ensure that if something goes sev2, then someone is around to fix it.

13. The system will generate and maintain identification numbers for users, assignments, announcements, and grades.

    The MariaDB already has auto increment for these specific things, so that ensures that it'll be unique id numbers.

14. The system shall support 1,000 simultaneous users at all times

    If we can automate tests then we could login 1000 people at once, but realistically we'll be able to ensure that we can have 5 users at any time, as we have 5 people in the team as well. Maybe 15 if we run 3 instances of the webapp.

15. The system shall run on any browser client, regardless of operating system.

    We have access to windows, linux (Rutgers), and if one of us has a mac then we cover most of our bases.