CourseCraft

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

1 System Requirements

1.1 Functional Requirements

- **F1.** 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.
- **F2.** 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.
- **F3.** 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.
- **F4.** 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.
- **F5.** 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.
- **F6.** 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.
- **F7.** A user with the Student role will be able to access the Course Registration page and view their course schedule.
- **F8.** 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.
- **F9.** 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.
- **F10.** 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.

1.2 Non-Functional Requirements

- **NF1.** 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.
- **NF2.** 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.
- **NF3.** The system will generate and maintain identification numbers for users, assignments, announcements, and grades.
- **NF4.** The system shall support 1,000 simultaneous users at all times.
- **NF5.** The system shall run on any browser client, regardless of operating system.

2 Test Design

Test Case ID	**T01**
Purpose	Test sending announcement emails
Pre-conditions	Uses mocked email data instead of an actual database hit
Inputs/Test Data	(class_name="Test Class", subject="Test Announcement")
Expected Outputs	List of properly formatted announcement emails for all members of Test Class
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T02**
Purpose	Test sending assignment emails
Pre-conditions	Uses mocked email and assignment data instead of an actual database hit
Inputs/Test Data	(assignment_id=1)
Expected Outputs	List of properly formatted assignment emails for all members of the class corresponding to the assignment with ID 1
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T03**
Purpose	Test sending grade emails to a single student
Pre-conditions	Uses mocked email and assignment data instead of an actual database hit
Inputs/Test Data	(assignment_id=1, student_ids=1)
Expected Outputs	Properly formatted grade update email for assignment with ID 1 send to student with ID 1
Post-conditions	N/A
Design Technique	Requirements review, equivalence class partitioning

Test Case ID	**T04**
Purpose	Test sending grade emails to a class
Pre-conditions	Uses mocked email and assignment data instead of an actual database hit
Inputs/Test Data	(assignment_id=1)
Expected Outputs	Properly formatted grade update email for assignment with ID 1 send to student with ID 1
Post-conditions	N/A
Design Technique	Requirements review, equivalence class partitioning

Test Case ID	**T05**
Purpose	Test sending a grade submission email
Pre-conditions	Uses mocked email and assignment data instead of an actual database hit
Inputs/Test Data	(assignment_id=1, student_id=1)
Expected Outputs	Properly formatted grade submission email for assignment with ID 1 send to student with ID 1
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T06**
Purpose	Test a if a student can register for a class
Pre-conditions	Test student created , test class created
Inputs/Test Data	(full_name= test_student, email = test_student@test.edu, class_name=test_class)
Expected Outputs	Successful insertion into database, no output
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T07**
Purpose	Test if a student can submit an assignment
Pre-conditions	Test student created , test class created, test assignment created
Inputs/Test Data	(full_name= test_student, email = test_student@test.edu, class_name= test_class, assignment_name = test_assignment)
Expected Outputs	Successful insertion into database, no output
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T08**
Purpose	Test if a student can get assignment list
Pre-conditions	Test student created , test class created, test assignment created
Inputs/Test Data	(full_name= test_student, email = test_student@test.edu, class_name= test_class, assignment_name = test_assignment)
Expected Outputs	Dataframe containing 'test assignment'
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T09**
Purpose	Test if a Dean can create a class
Pre-conditions	Test Dean Created
Inputs/Test Data	(class_name = Test Dean Class, major = Math)
Expected Outputs	Successful insertion into database, no output
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T10**
Purpose	Test if a Professor can create an assignment
Pre-conditions	Test Professor created, Test Professor Class created
Inputs/Test Data	(assignment_name= Test Professor Assignment, file_name = test_prof_file, class_name= Test Professor Class, due_date = 2022-04-24 00:12:00)
Expected Outputs	Successful insertion into database, no output
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T11**
Purpose	Test if a professor can register to teach a course
Pre-conditions	Test Professor created, test user id, Test Professor Class created
Inputs/Test Data	(user_id= user_idt,class_name= Test Professor Class)
Expected Outputs	Dataframe containing open classes
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T12**
Purpose	Test if a professor can find available classes to teach
Pre-conditions	Test Professor created,
Inputs/Test Data	N/A
Expected Outputs	Successful insertion into database, no output
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T13**
Purpose	Test if a professor can get assignments
Pre-conditions	Test Professor created, test user id, Test Professor Class created
Inputs/Test Data	(class_name= Test Professor Class)
Expected Outputs	Dataframe containing course assignments
Post-conditions	N/A
Design Technique	Requirements review

Test Case ID	**T14**
Purpose	Test if a student can get a 4.0 gpa with 0 GPA currently
Pre-conditions	N/A
Inputs/Test Data	`student.calculate_future_gpa_method(0, 0, 15, 4.0)`
Expected Outputs	4.0
Post-conditions	N/A
Design Technique	Requirements review, equivalence class partitioning

Test Case ID	**T15**
Purpose	Test if a student can get a 0.0 gpa with 0 GPA currently
Pre-conditions	N/A
Inputs/Test Data	`student.calculate_future_gpa_method(0, 0, 15, 0.0)`
Expected Outputs	0
Post-conditions	N/A
Design Technique	Requirements review, equivalence class partitioning

Test Case ID	**T16**
Purpose	Test if a student can get a 3.5 gpa with 4.0 GPA with 15 creds currently

Pre-conditions	N/A
Inputs/Test Data	`student.calculate_future_gpa_method(4.0, 15, 15, 3.5)`
Expected Outputs	3.0
Post-conditions	N/A
Design Technique	Requirements review, equivalence class partitioning

Test Case ID	**T17**
Purpose	Test if a student can get a 0.0 gpa with 4.0 GPA with 15 creds currently
Pre-conditions	N/A
Inputs/Test Data	`student.calculate_future_gpa_method(4.0, 15, 15, 0)`
Expected Outputs	< 0
Post-conditions	N/A
Design Technique	Requirements review, equivalence class partitioning

Test Case ID	**T18**
Purpose	Test if a student can get a 4.5 gpa with 4.0 GPA with 15 creds currently
Pre-conditions	N/A
Inputs/Test Data	`student.calculate_future_gpa_method(4.0, 15, 15, 4.5)`
Expected Outputs	>4
Post-conditions	N/A
Design Technique	Requirements review, equivalence class partitioning

3 Traceability

Test Case #	List of Requirements Tested
T01	F10
T02	F10
T03	F10
T04	F10
T05	F10
T06	F6, F7
T07	F8
T08	F8
T09	F3
T10	F5
T11	F4
T12	F5
T13	F9
T14	F9
T15	F9
T16	F9
T17	F9
T18	F9