LearnTrace

FINAL REPORT

Andrew Hartman, Andy Guibert, Jon Mielke, Travis Reed, Lucas Rohret
IOWA STATE | SENIOR DESIGN

CONTENTS

1 Project Design 3
1.1 Project Definition 3
1.2 Deliverables 3
1.3 System Description 3
1.4 Functional Requirements 4
1.5 Non-Functional Requirements 5
1.6 Module Diagrams 6
1.7 UI Screens 7
1.8 Interface Specifications 14
1.8.1 Server Specifications 14
1.8.2 Client Specifications 14
1.8.3 Software Specifications 14
1.9 Simulation And Modeling 15
1.10 Implementation and Issues 15
1.11 Testing, Procedures, And Specifications 16
1.12 Operating environment 16
1.13 User Interface Description 17
1.13.1 Login Page: 17
1.13.2 Main Page (Author): 17
1.13.3 Main Page (Student): 17
1.13.4 Registration Page: 17
1.14 Market And Literature 18
1.15 Resource Requirements ..... **Error! Bookmark not defined.**
1.16 Project schedule 19
1.17 Risks 20
2 Implementation 21
3 Testing 22
4 Appendix I : Operational Manual 23
4.1 Login/Registration ..... **Error! Bookmark not defined.**
4.2 Account Promotion 23
4.3 Course 23

4.3.1 Add Students 23
4.3.2 Archive 23
4.3.3 Unarchive 24
4.3.4 Create 24
4.3.5 Delete 24
4.3.6 Register 24
4.3.7 Registration Code 24
4.4 Grade 24
4.4.1 Cognitive 24
4.4.2 Correctness 25
4.5 Permission Scheme 25
4.5.1 Student 25
4.5.2 TA 25
4.5.3 Author 25
4.5.4 Admin 26
4.6 Task 26
4.6.1 Copy 26
4.6.2 Create 26
4.6.3 Delete 26
4.6.4 Edit 26
4.6.5 Export Data 27
4.6.6 Response 27
4.6.7 Transition 27
5 Appendix II : Alternative Versions 28
6 Appendix III : Miscellaneous Information 29
Lessons Learned 29
7 Appendix IV : Code 30

1 Project Design

1.1 Project Definition

Develop a web portal (WP) to diagnose learning styles. Handle user authentication for students and faculty to evaluate how students learn. Allow authors to create questions that contain supplementary learning materials. Keep track of how much time students spend on these questions, and reviewing each of the supplementary learning materials.
Supplementary learning materials may include (but are not limited to): YouTube videos, diagrams, and text materials.

1.2 Deliverables

1. An online application, where authors can create questions and evaluate responses
2. An online application, where students can answer questions and receive results

1.3 System Description

The system to be used for the web portal will consist of an Iowa State server to host the site itself with a SQLite database to house all of the login information and content to be used on the portal. Backend code will consist of python and the front end development will be in JavaScript and Html. Ajax calls in the JavaScript will allow communication between the user interface and the backend.

1.4 Functional Requirements

1) Authentication
   a) Simple login/logout system with basic auth
   b) User registration for students
   c) Have three tiers: Student/Author/Admin
      i) The student can view questions, submit responses, and view their own results
      ii) The author has all of the power of a student, but they can also create and modify questions, add students to their classes, and view responses from their classes
      iii) The admin has all of the power of an author, but they can also add authors to classes, and add authors to the system.

   d) Stretch goal - support authentication with blackboard credentials
2) Create Questions
3) Allow authors to create questions and assign these questions to specific assignments
4) Questions are made up of, the question, any potential responses, and supplementary material
      i) Supplementary material can include videos, text, or images

5) Answer Questions
   a) Students can answer questions from their assignments
   b) Students can view supplementary material from their assignments
   c) Students can only view one question at a time
   d) While students are answering questions, data will be collected about how the student answered the question, including things like time spent viewing supplementary material

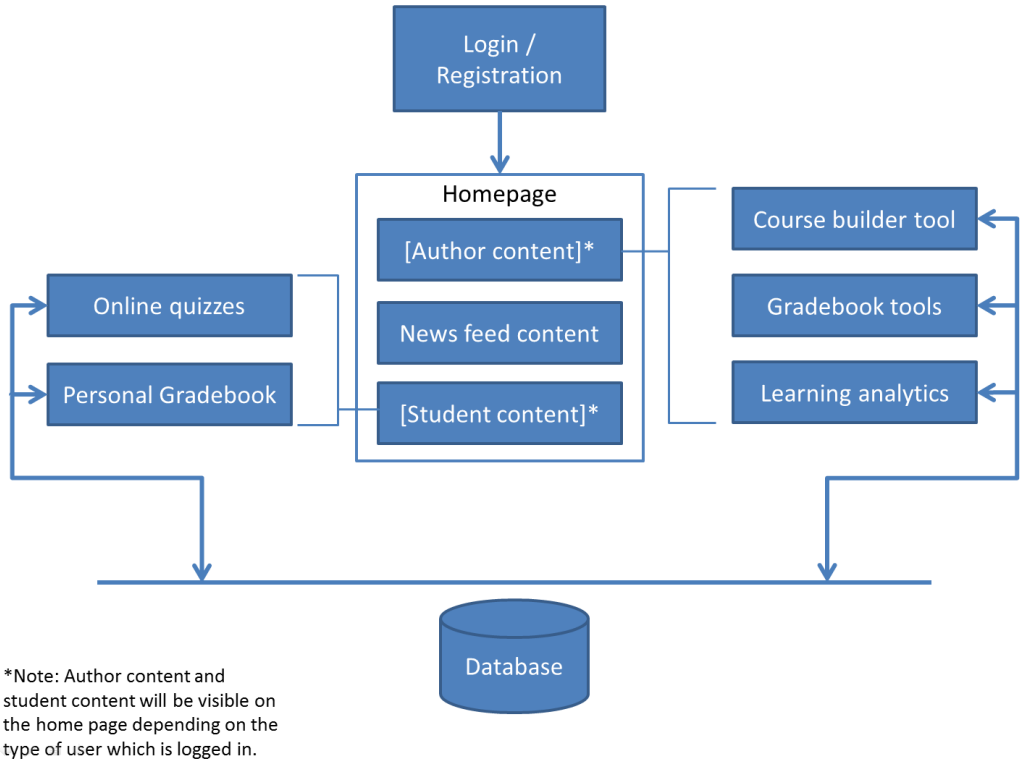
6) Evaluate Responses
   a) Authors can evaluate responses from their students
   b) This evaluation will include access to metrics including time spent on question, which supplementary materials were viewed, and for how long
7) Sign up for classes
   a) Students can sign up for the system
   b) Authors can add these students to their own classes
8) Grades
   a) Authors
      i) Can grade assignments
      ii) Can submit feedback
      iii) Can view analytics showing how students are answering questions, and whether or not they are thinking critically when they answer questions
   b) Students
      i) Students can view grades for courses and for individual assignments

1.5 Non-Functional Requirements

1) UI Design
   a) Design/answering buttons at most 3 clicks away
2) Speed
   a) Response calls from server come back within 1000 ms
   b) Load time of home page 1500 ms
3) Scalability
   a) Able to extend to other teaching communities
4) Maintainability
   a) Code readability
   b) Low Coupling

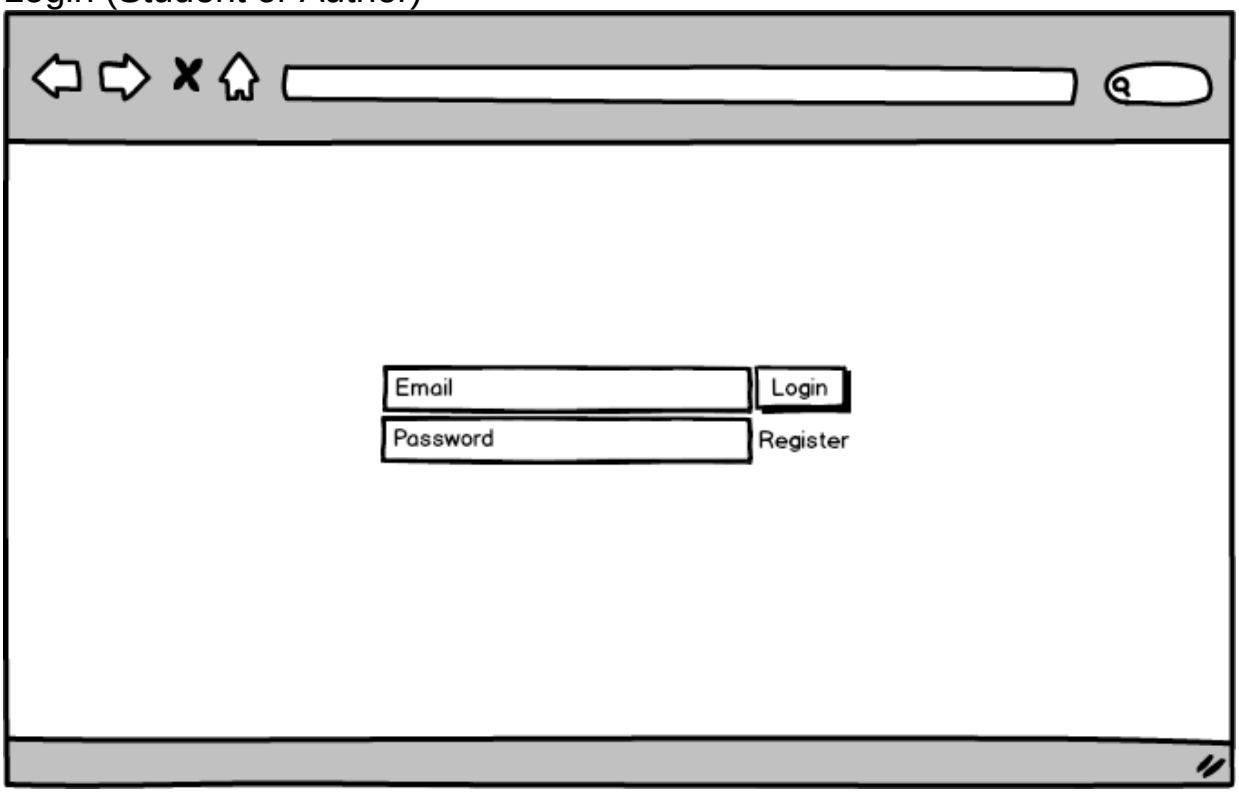
5) Testing
   a) 65% test code coverage (python)
   b) TDD (Test Driven Development)
6) Miscellaneous
   a) Analytics report generated after each answered question

1.6 Module Diagrams

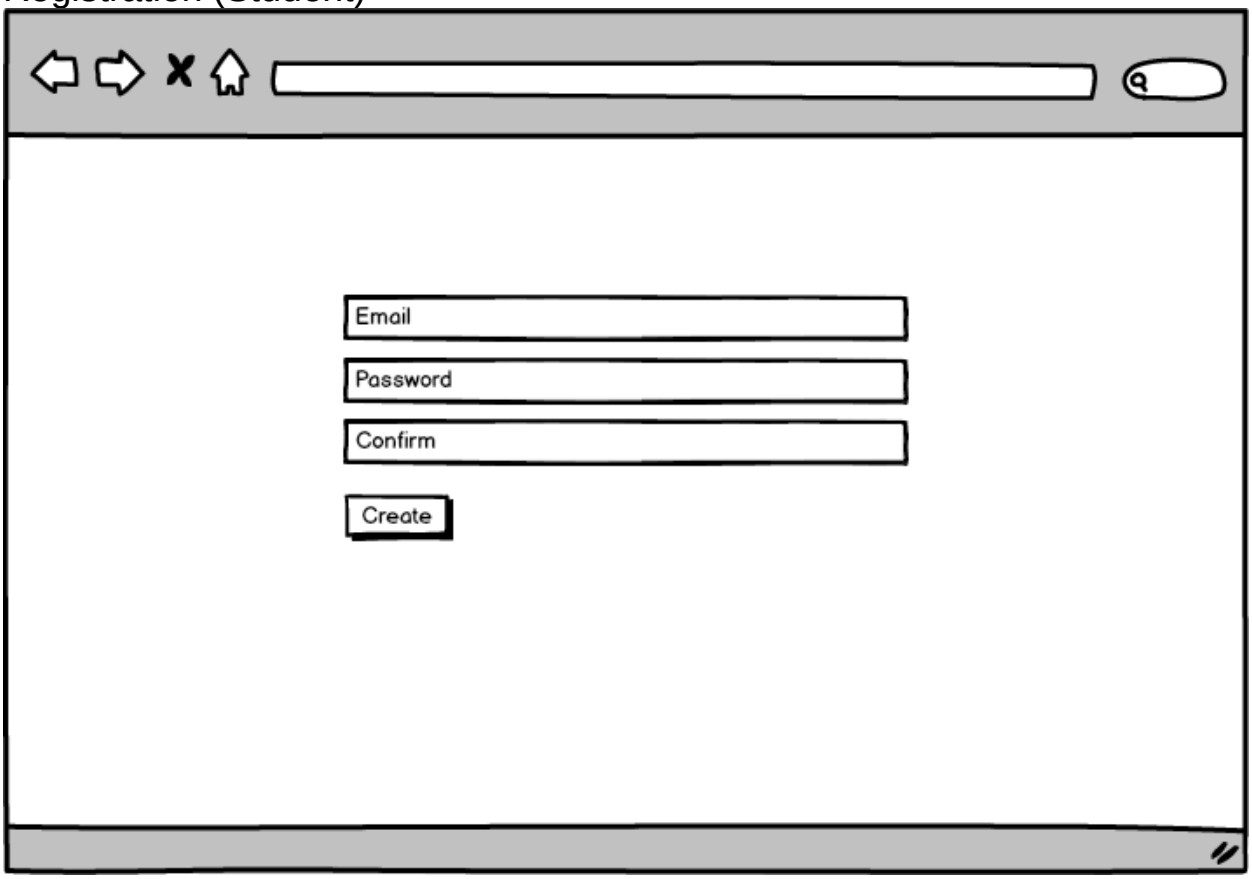


1.7 UI SCREENS

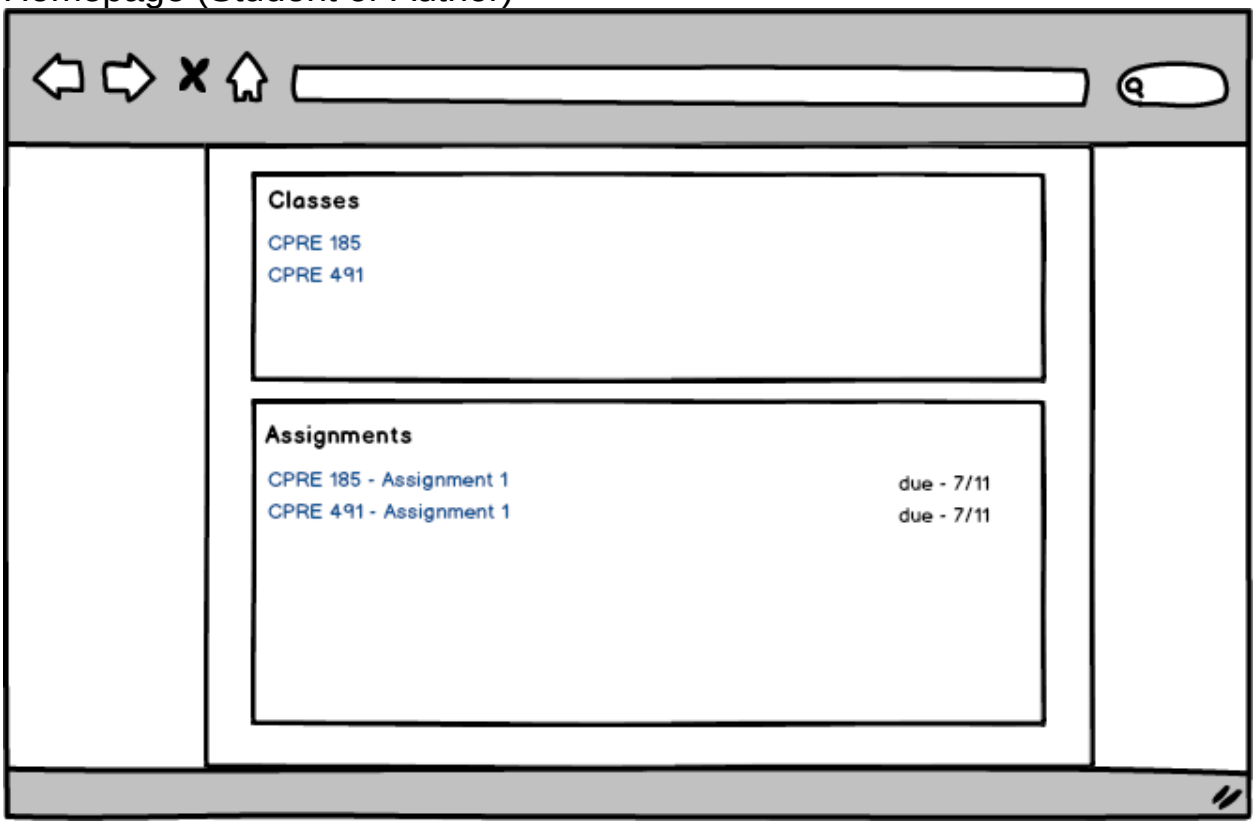
Login (Student or Author)



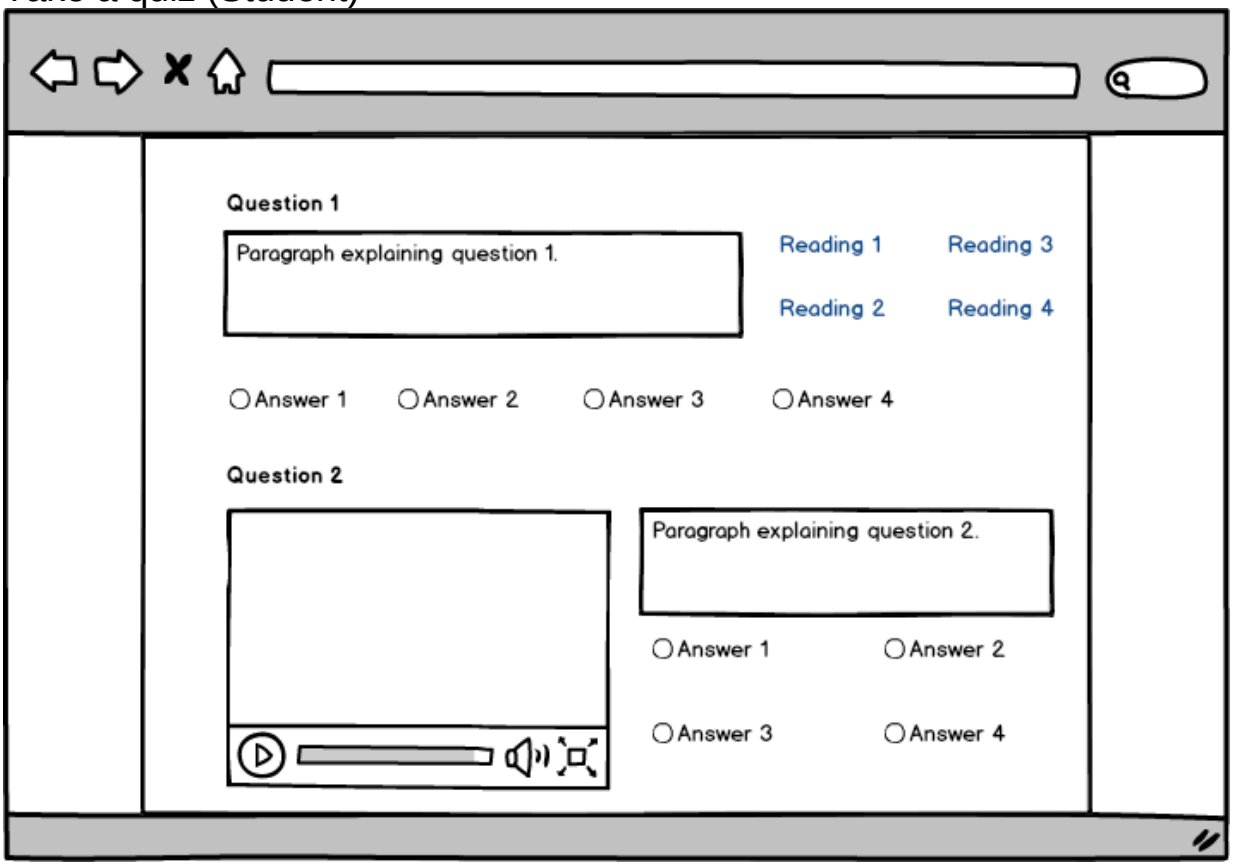
Registration (Student)



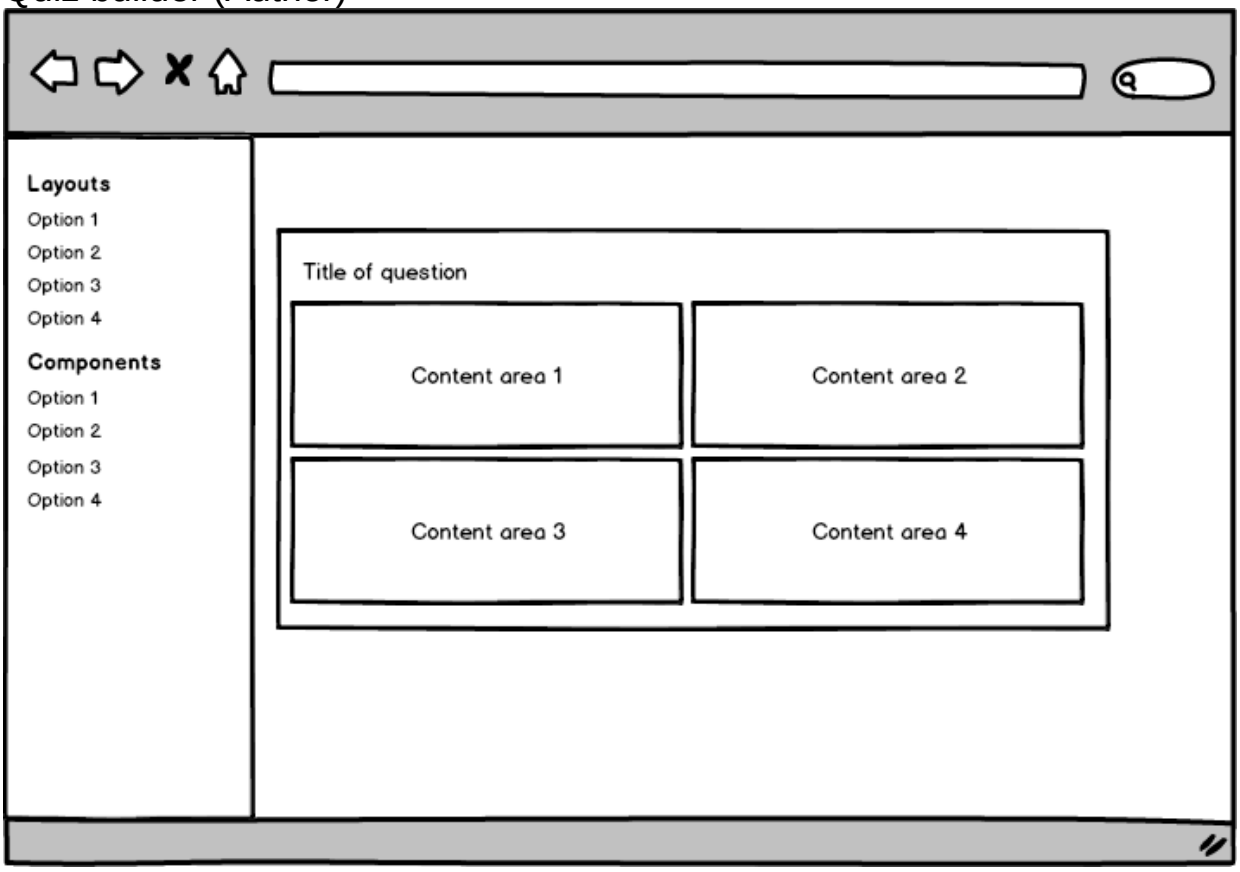
Homepage (Student or Author)



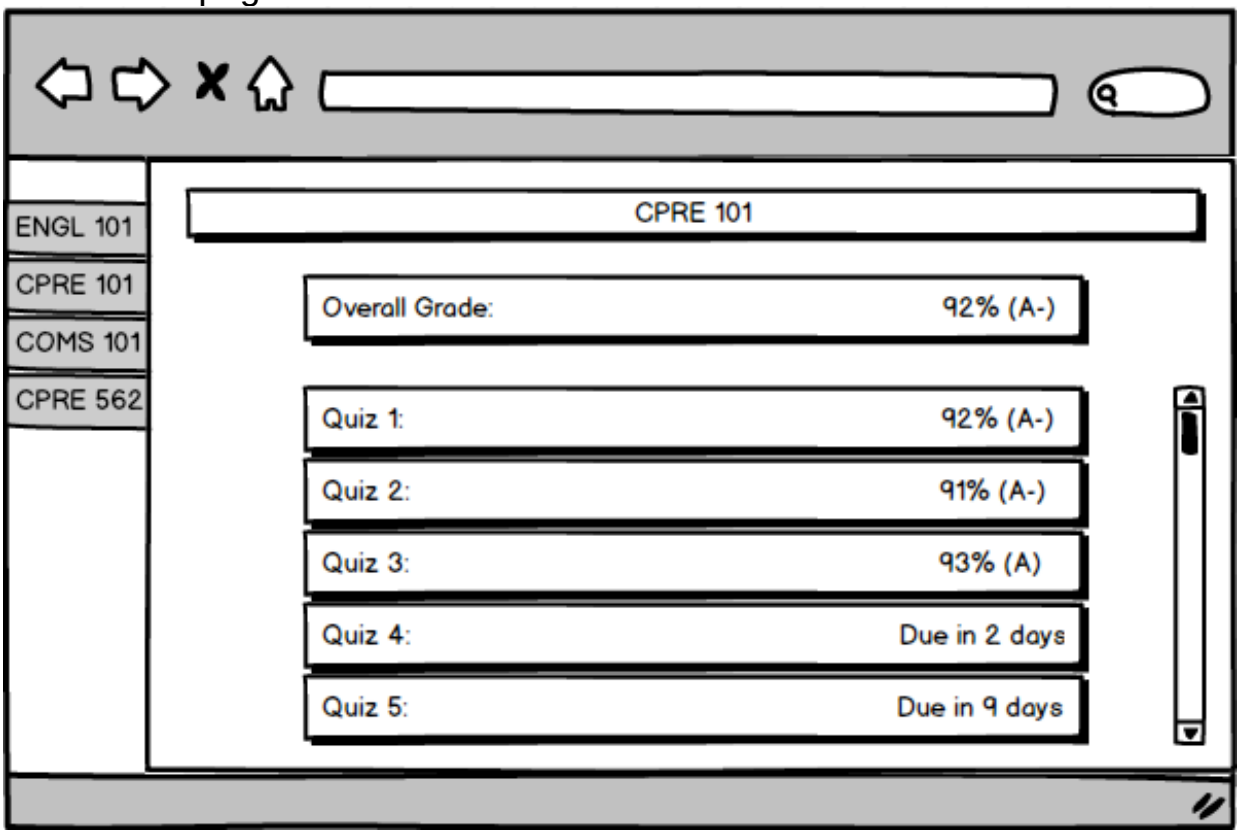
Take a quiz (Student)



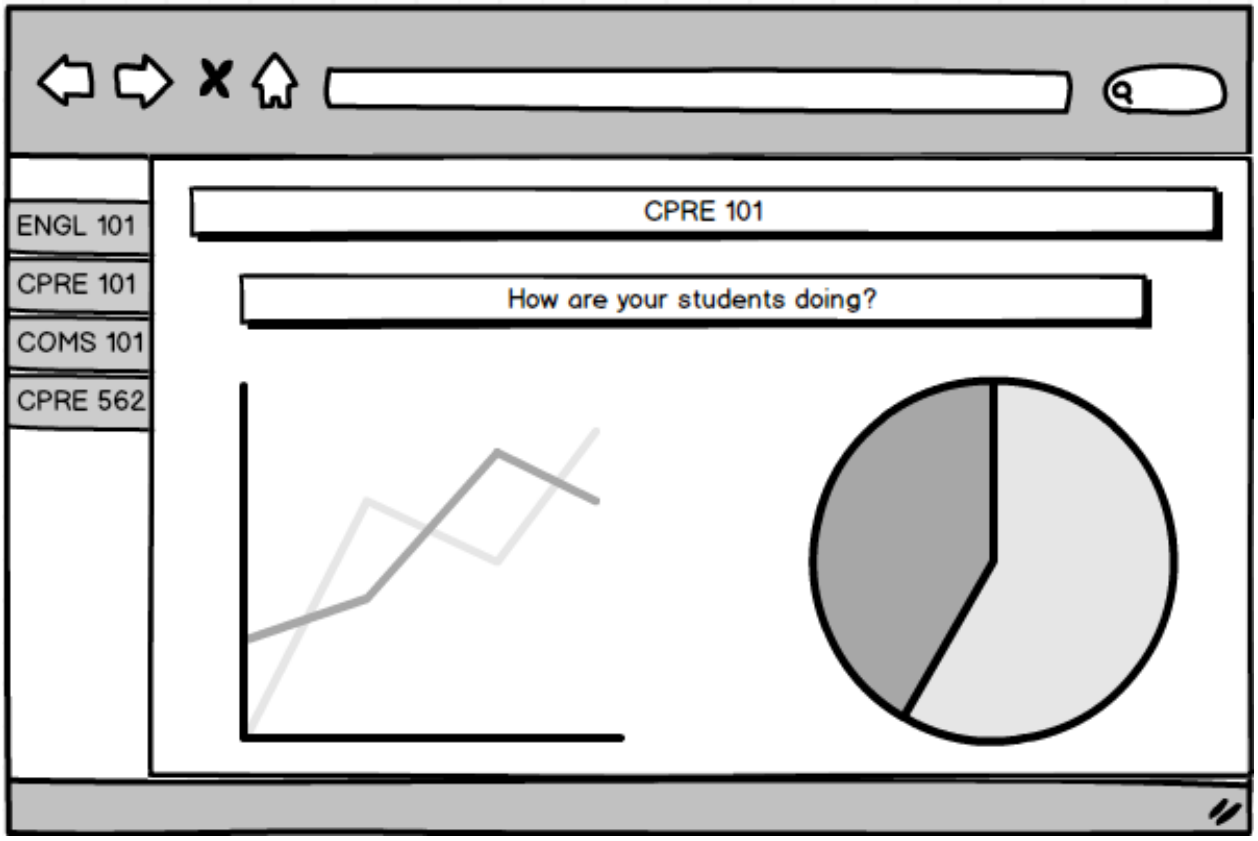
Quiz builder (Author)



Gradebook page



Learning Analytics - Author



1.8 Interface Specifications

1.8.1 Server Specifications

1. HTTP protocol
    1. Available ports to receive requests from client endpoints
2. MySQL Database
    1. 100 GB of initial storage for a relational database
    2. Compatibility with SQLAlchemy for python interface
3. Python
    1. Python 2.7 or higher
    2. Flask 10.1 or higher
    3. SQLAlchemy 9.6 or higher
4. High availability
    1. Server should have no more than 1 hour of downtime per week initially
    2. Put plan in place to have server downtime no more than 1 hour per month

1.8.2 Client Specifications

1. Web Browser
    1. login screen
    2. Register
2. Enrolled in a class that is active
    1. Input code provided by author
    2. Use link emailed by author
3. Valid Email Address
4. Active Internet Connection

1.8.3 Software Specifications

1. Web Browser
    1. HTML 5 Support
        1. Google Chrome
        2. Mozilla Firefox
2. Server
    1. Python 2.7 or higher
    2. SQL Lite 3
    3. SQLAlchemy 9.6 or higher

3. Imported Python Libraries
    1. Flask 10.1
    2. Jinja2

1.9 Simulation And Modeling

- Web Portal Hosted on Local Servers
- Web Portal Hosted on Iowa State provided Server

1.10 Implementation and Issues

Question builder - Allowing our users the flexibility they desire, while still keeping the interface clean will require a practiced and careful approach to the problem. We intend to ensure our success in this area by designing and implementing this system first, so that we can get a large amount of feedback on it as we develop it.

New Languages - Python is new to 80% of the team, and JavaScript is fairly new to 40% of the team. We have been completing several workshops as a team and are meeting twice a week to make sure that we can help each other learn the new languages and tackle engineering problems in a way that meets industry standards.

1.11 Testing, Procedures, And Specifications

Code quality will be ensured using three formal verification/validation processes.

- Code Reviews
- Unit Testing
- Design Document Review

All of our code will either be created using a pair programming approach, or will be code reviewed by at least 1 peer. That peer will be expected to review the code for quality, and review the software to ensure that it conforms to the requirements.

We will use the python library nose to unit test our code. By the end of our term we expect the code to have at least 65% code coverage, and 65% branch coverage (as measured by nose). All files written by our team will be included in these numbers (with the exception of models.py and run_sandbox.py), but external code will not be included in this calculation.

As we create design and project documents we will send these to each of our advisors for review. We will keep them up to date with our progress and will make sure that we make any changes that they require.

1.12 Operating environment

The environments capable of operating the web portal will be diverse. Anyone should be able to access this if the device in use takes advantage of a standard web browser either on a desktop, laptop, or mobile device. Web browsers that will be included in testing for functionality initially will include Google Chrome, Firefox, Internet Explorer, Safari, and Opera. The User Interface for all of these will have the same functionality, but may appear slightly different based on the operating system that is in use.

1.13 User Interface Description

1.13.1 Login Page:

This page will have username and password fields to allow users to login. There will also be a button for creating an account.

1.13.2 Main Page (Author):

The author will be given a blank page and will allowed to select a grid layout from a list of possible layouts. Inside of the grid there will be areas where an author can place elements of their page. A few examples would be YouTube videos, titles, or text fields for questions.

1.13.3 Main Page (Student):

The student will see one question per page based on what the author of the page has created for the student. The student will then have a method of answering the question (multiple choice, essay, buttons) based on what the author provides them. There will also be a submit button at the bottom of every page. At the top of every page there will be an account bar for login and account management.

1.13.4 Registration Page:

This page will contain lists and text fields for selecting courses.

1.14 Market And Literature

There is no question that education is a prominent path in our society, counting for at least 12 years of most people's lives. With such a wide audience, not everyone is able to be educated in the same way as others, as for there is not a foolproof education plan set. This undoubtedly allows space in the market for new more effective learning styles.

Trying to find a solution to help maximize a learning method is very difficult. There are many proven/disproven methods available to date. However the Web Portal project will be inheriting the Habermas and Grundy as a cognitive learning method. This learning style contains three main parts of cognitive development including technical aspects of learning (theoretical stage), practical stage, and the emancipator interest stage. The first stage is where the students learn while memorizing and following rules. The second stage, the students take the concepts they learned and applying them to various problems. The scopes of problems in this stage are narrowed, with reduced difficulty to that of real world problems. The third stage allows students questions from "what" to "why". This allows the student to broaden their mind sets and determine many factors that affect the problem. Most of the development and growth usually comes from this final stage.

1.15 Resource requirements

Physical capital:
Partition of a high availability server on the Iowa State that will host our learning portal website. There will be no significant speed requirements. We anticipate roughly 50GB of database space will be needed per semester. If classes will persist for multiple semesters, or be re-taught, the storage space on our server will need to expand to accommodate.

Software capital:
There will be no notable costs for purchasing or licensing software that will be used in this project. All of the tooling we will use is freely available. We expect that database software will be provided by Iowa State.

1.16 PROJECT SCHEDULE

Semester 1:
September: Design document, Screen Sketches
October: Setup server environment, Login / Logout / Registration functionality
November: Teacher perspective: Course and question builder
December: Teacher perspective: Course and question builder

Semester 2:
January: Teacher perspective: Course and question builder
February: Student perspective: Sign up for classes, and take questions
March: Teacher perspective: student response analytics
April: Teacher perspective: student response analytics
May: Final testing and bug fixes

1.17 RISKS

Risk	Probability of Occurrence	Criticality (1-100)	Risk Factor	Mitigation Strategy
Loss of team member (or non-contributing team member)	.05	60	3	Have solid group communication and document as much work as possible.
The project is not completed on time.	0.2	80	16	We are beginning the development work for this sprint early, and we have a development plan firmly in place for future sprints.
Student data is stolen	.1	100	10	We will keep as little student data as possible. We will research good authentication practices.

2 IMPLEMENTATION

This project was implemented using html, JavaScript, and python. We also relied upon a series of external technologies including:

- Flask – url routing, render templates
- Jinja2 – access python variables from html
- Flask-SQLAlchemy – database connection
- Bootstrap – css templating
- JQuery – simple JavaScript selectors

3 TESTING

Code quality was ensured using four formal verification/validation processes.

- Code Reviews
- Unit Testing
- Beta Testing
- Design Review

All of our code was either created using a pair programming approach, or was code reviewed by at least 1 peer. The peer who reviews the code looks for errors through searching the code in the pull request as well as testing with the code in their own branch when necessary.

We used a python library called Nose to unit test our code. Nose allows us to measure exactly how much branch and line coverage we achieved. At the end of the semester we met our goal of 65% branch coverage. All files written by our team were included in these numbers (with the exception of models.py and run_sandbox.py), but external code was not included in this calculation.

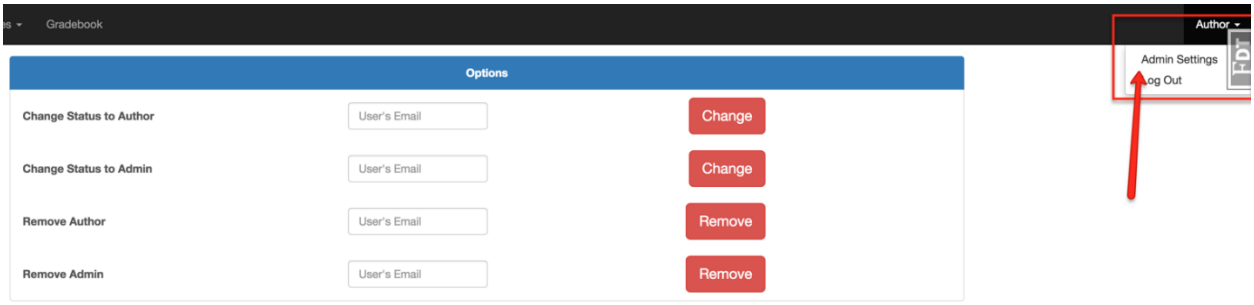
Nearing the end of the semester we were able to open up a beta testing session with one of Dr. Keren's classes for the purpose of initial stress tests and to simply see the system in use by users other than inside the team. This testing provided valuable insight into what we did right and wrong so far for the project as well as exposing bugs that were missed during our other testing efforts.

Throughout the semester we updated our advisor Dr. Keren to our progress on a weekly bases, giving a brief demo at the beginning of each meeting. This helped us to test functionality weekly on a broader scale than unit testing. During these meetings, end functionality would be decided and giving us a sense of direction to make changes where needed.

4 Appendix I : Operational Manual

4.1 Account Promotion

When an account is first created it has only student level permissions. An admin can promote an account to a higher permission level by clicking on their name in the top right, and selecting “Admin Settings”. From here, they have the option to promote accounts.



4.2 Course

4.2.1 Add Students

When creating a course you have the option to add students to it by uploading a .txt file of commas separated names, or by creating a registration code. (See “Registration Code”).
If you chose to upload a .txt file it should be in the following format:
<email1>, <email2>, ….
If you do not include a full email address, then @iastate.edu will be appended to each item in the list. This will auto-enroll students in the course. If the student you list does not have an account, a temp account will be created for them. They can register for the system the normal way, then when they log in for the first time they will have access to your course.

4.2.2 Archive

A course can be archived by its instructor, or by an admin. Simply navigate to the course page, and press the archive button at the bottom of the page. Once this is done, all data will be kept, but the author and students will no longer be able to access it. This will allow an author of the course or admin to retrieve the information from the course if needed.

4.2.3 Unarchive

Only an admin can unarchive a course. They do this by navigating to the course, and selecting the “Unarchive” option. At this point all data is restored, and it is as if the course were never archived.

4.2.4 Create

An author can create a course by submitting a name (MATH-101 section B for example) and a title (Introduction to Mathematics). They can also add students at this time, using the methods described in “Add Students”.

4.2.5 Delete

See “archive”

4.2.6 Register

A student can register for a course by clicking the “Course” dropdown in the navbar, then selecting “Register for Course”. To register, they must know the registration code created by the instructor.

4.2.7 Registration **Code**

An instructor can create a registration code for their course by navigating to their course and clicking “Registration Code”. This will create a 6 digit code that students use to register for the course.

4.3 GRADE

Grading is split into a couple of views. When a student clicks on gradebook, they see a list of all of tasks they have in all of the courses they are taking. Once a grade has been published they can see their correctness grade for that task, as well as the correct answer, and their score for each question.

When an author goes to gradebook, they are asked to choose a course first. Then they get a table of students and tasks for that course. They can update the grade for any task, and mark a solution as simple or critical.

4.3.1 Cognitive

The cognitive score is hidden to the student, and takes into consideration whether or not they viewed each supplementary material on a task for a sufficient amount of time. If the student viewed the material for enough time

they are given credit. If they did not view the material for enough time, they are not given credit.

4.3.2 Correctness

The correctness score is known to the student, once grades have been published. Each automatic question can give either 0 or 100 points, and each manual question can give 0-100 points.

4.4 Login/Registration

Before logging into the system you must register yourself. This can be done by inserting an email address, and a password. The password must have at least 8 characters in total with 1 number and 1 uppercase letter at a minimum.

Upon registering, you have a student account. If a professor previously included your email address when creating a course, you may already be a student in a class, otherwise you will need to join a class, or request that an admin promote your account. Once your account has been promoted to the author or admin level, simply log out of the system and log back in the normal way. You will now have access to whatever actions that are appropriate for your new permission level.

4.5 Permission Scheme

4.5.1 Student

Students can see courses they have registered for, and they can complete tasks for that course. Once grades for the task have been published, they can see their grades and any instructor feedback. They cannot see tasks from other students

4.5.2 TA

TAs have all the rights of students, and they can create/grade tasks for the courses they TA.

4.5.3 Author

Authors have full permissions to the courses that they are teaching. They can create courses, grade tasks, add and remove TAs, transition tasks, and archive the course. They cannot see courses that they are not teaching

4.5.4 Admin

Admins see everything. They can do everything an author can do, but they can do it for every course. They can also add new authors, unarchive courses, and even add new admins (with a special passcode).

4.6 Task

4.6.1 Copy

Once a task has been created, an admin or an author who has permission to see it, can copy it. Simply navigate to the home page, and next to a task, click the blue copy button. This copies over only the content of the task (questions, supp material etc.) but not any responses, or data associated with the task.

4.6.2 Create

A task can be created by any author or admin. Using the navbar click “TaskBuilder”. Building tasks is relatively straightforward. A palette of elements is located on the left column which can be dragged onto the center of the screen in order to build a list of questions. When the elements are dragged onto the screen, the author is responsible for filling in any empty fields that are present.

4.6.3 Delete

The author of a course, or any admin, can delete a task. Navigate to the home page, and next to a task, click the red “delete” button. All data associated with the task is then deleted. This is irreversible.

4.6.4 Edit

The author of a course, or any admin, can edit a task. Navigate to the home page, and next to a task, click the blue “edit” button. This should not be used to significantly change the content of a task after it has been released to students. Data is kept associated to the task, with no indication of whether a response came in before or editing the task.

4.6.5 Export Data

The author of a course, or any admin, can export task data. Click on a task from the course or home page, and click the “export data” button. This will download a Microsoft excel file with information about student who responded to the task. A time of response is listed, as well as information about the response, such as time spent on each supplementary material, answer for each question, and score for each question.

4.6.6 Response

Any student enrolled in the course can respond to a task associated with that course. The student needs to navigate to either the course page or the home page, then click on the task. Their response is stored in our database, including how much time they spent on each supplementary material. If they view a supplementary material more than once, the times are totaled together. If the tab is closed, all response data is lost, unless submit button has already been pressed. If they respond to a task more than once, only the most recent response is kept.

4.6.7 Transition

The author of a course, or any admin, can transition a task. When a task is first created it has the “Created” status, and is not available to any students. By navigating the task page, it can be transitioned then to “Available”, where it becomes available to students. Next it can be transitioned to “Closed” where responses are no longer accepted. Finally, it can be moved to “Published” where grades and answers are exposed to the students in the course.

5 APPENDIX II : ALTERNATIVE VERSIONS

When the idea of LearnTrace was first described to us, our thoughts immediately went to Blackboard Learn. Noting the similarities we wanted to add the analytics of LearnTrace to Blackboard Learn's existing quiz interface. This would have been the optimal solution to our project however we found that it would be very difficult to gain access to Blackboard Learn's API. It was at this time we knew that we would have to create our own platform for LearnTrace.

While working on this project we met with our Advisor Nir Keren once a week. He would advise us on design choices and provide feedback on our current progress. As the year progressed Nir would add small pieces of functionality that were not stated in the original project requirements. The end result is slightly different than defined in the beginning because of these pieces of added functionality.

From beginning to end of this project, we have been swapping, changing, and rewording parts of LearnTrace. For example, in the beginning of the project, we were calling it WebPortal instead of LearnTrace. Wording was discussed quite frequently in our weekly meetings with Dr. Keren, since it is incredibly important that our point gets across to the user without any confusion.

Throughout development of the project, the color schemes in the project was a white background with a near black navbar. This was joined with a more bootstrap styling with vibrant buttons and table headings. The schemes has since been changed to the style to date, with a more common theme.

6 Appendix III : Miscellaneous Information

Lessons Learned

Most of the team was new to the python programming language as well as the Flask micro-framework. Having the chance to spend a full year gaining real world experience with these technologies was very rewarding.

Interestingly, we started this project with a plan to use an agile oriented development lifecycle, but at the request of our advisor, we instead started with the waterfall methodology. However, throughout the semester new features were requested nearly every week - which meant that in reality our approach to development naturally gravitated toward the agile methodology that better serves this close connection to our client. The initial documentation was rarely referenced because it quickly became either out-of-date, or irrelevant because everyone had a strong enough picture of our end goal, to not need it when making design decisions. Regular weekly meetings and demos took place, which allowed for a smooth transition from concepts to demonstrable features.

At the start of senior design we were able to choose our top projects and the top people we would like to work with. Now as the ending of the project comes to an end, being in the right group is a vital component to the success of the project. The five developers on this project have known and have been friends with each other since freshman year in the learning community dorms. This allowed for an easy flow of communication throughout the year long process. Also, everyone in the group has a good sense of other's skillsets, making assigning tasks easier and more efficient.

7 APPENDIX IV : CODE

All of the code that we created for this project was compiled in a GitHub project throughout the year. This provided ease during implementation as well as for testing and review once the project was complete.

GitHub:
https://github.com/ISUWebPortal/491Project