



# ISUWEBPORTAL DESIGN DOCUMENT

Presented By: Andrew Hartman, Travis Reed, Jon Mielke, Andy  
Guibert, and Lucas Rohret

Senior Design 491

## TABLE OF CONTENTS

1 - Project Definition .....	2
2 - Deliverables .....	2
3 - System Description .....	2
4.1 - Functional requirements .....	3
4.2 - Non-functional requirements .....	4
5.1 - Module Diagrams.....	4
5.2 Use Case Diagrams.....	5
5.3 - UI Screens .....	5
6 Interface specifications .....	8
6.1 Server Specifications .....	8
6.2 Client Specifications .....	9
6.3 Software specification.....	9
7 Simulations and modeling.....	9
8 Implementation Issues and Challenges .....	9
9 Testing, procedures and specifications.....	10
Appendix A.....	11

## 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.

## 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

## 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.

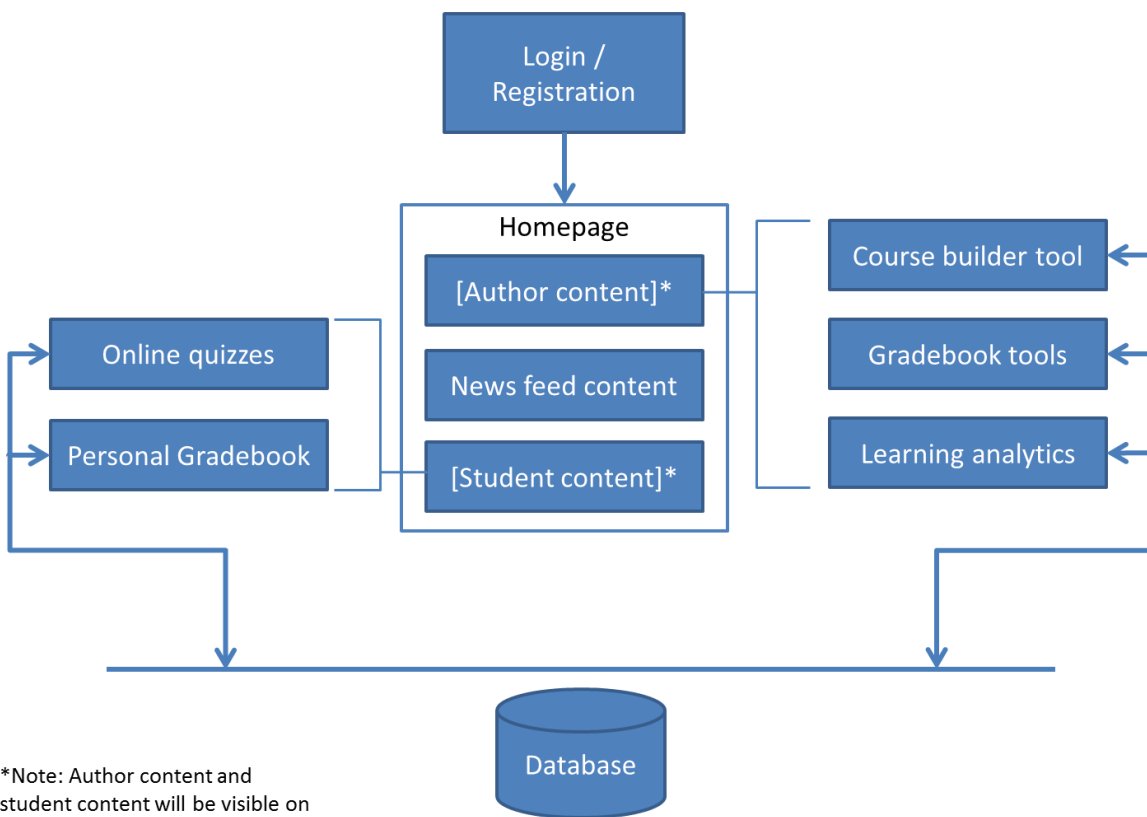
## 4.1 - FUNCTIONAL REQUIREMENTS

- 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
  - a) Allow authors to create questions and assign these questions to specific assignments
  - b) Questions are made up of, the question, any potential responses, and supplementary material
    - i) Supplementary material can include videos, text, or images
- 3) 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
- 4) 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
- 5) Sign up for classes
  - a) Students can sign up for the system
  - b) Authors can add these students to their own classes
- 6) 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

## 4.2 - 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

## 5.1 - MODULE DIAGRAMS



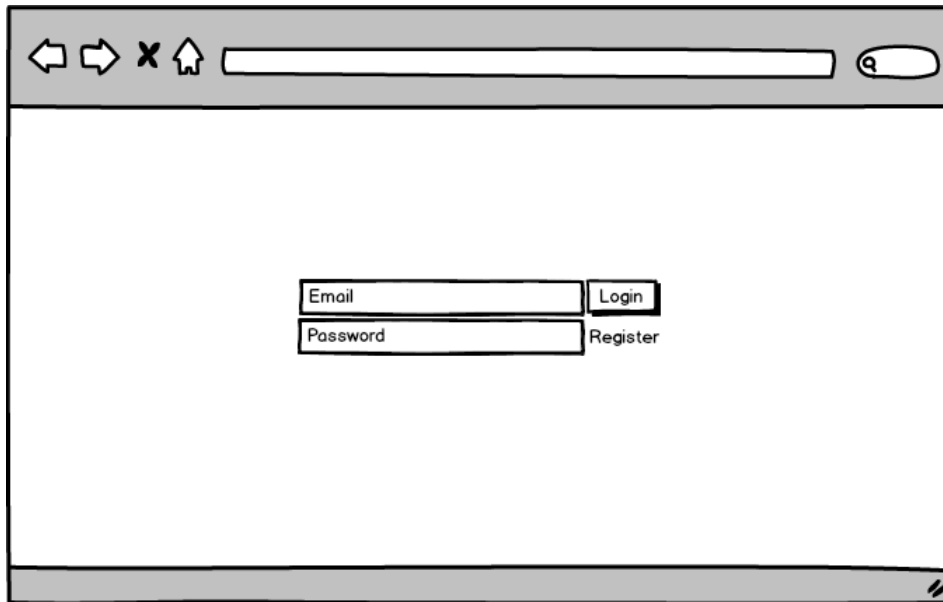
\*Note: Author content and student content will be visible on the home page depending on the type of user which is logged in.

## 5.2 USE CASE DIAGRAMS

See Appendix A for Use Case Diagrams.

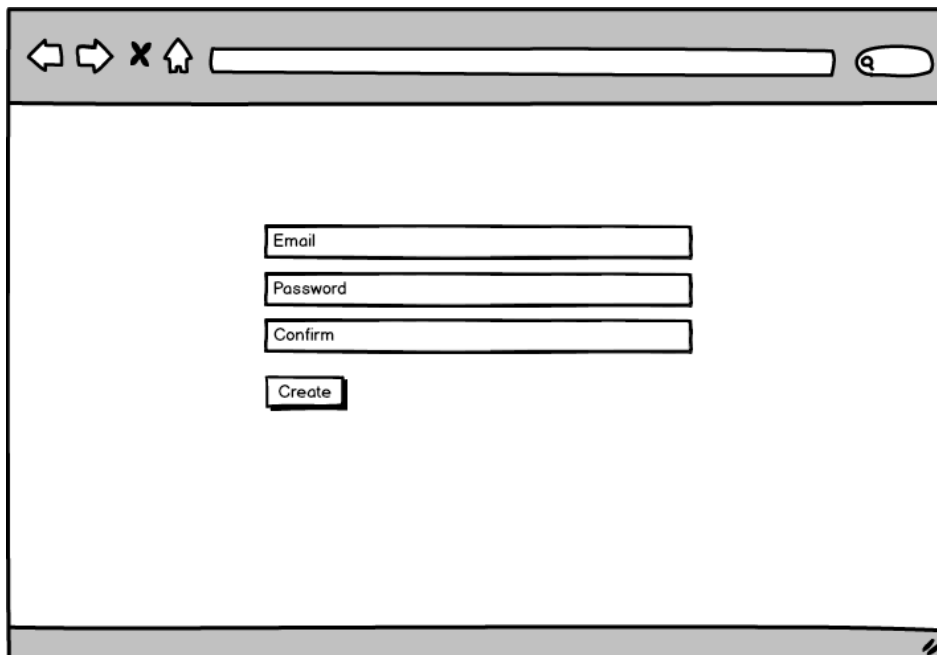
## 5.3 - UI SCREENS

Login (Student or Author)



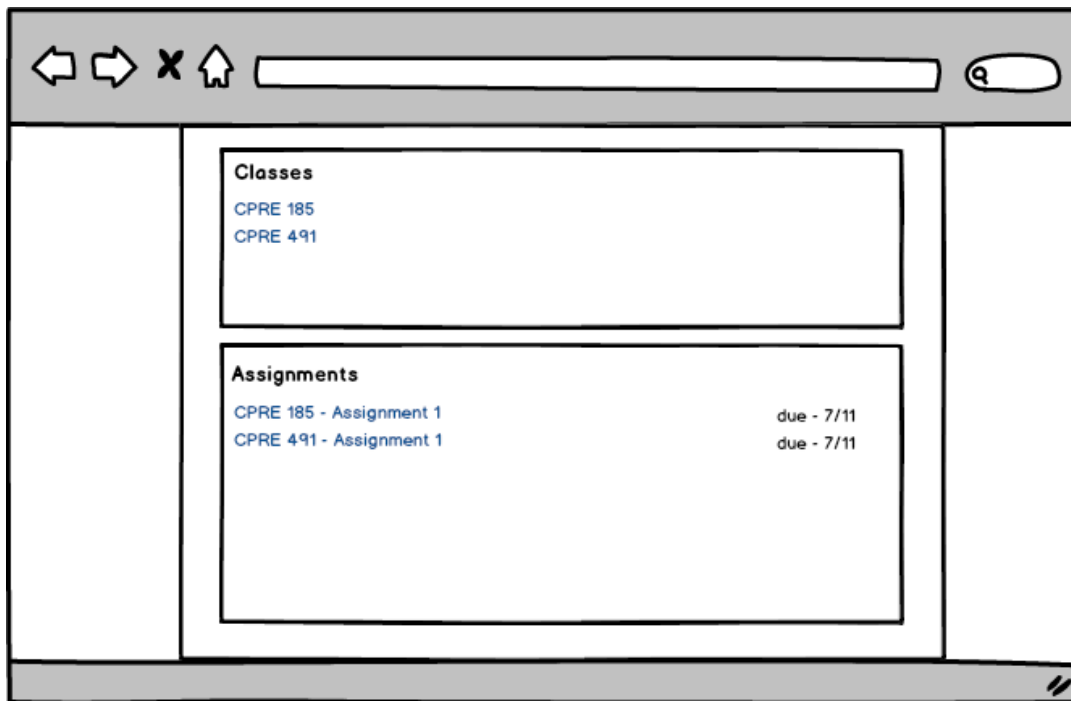
A hand-drawn sketch of a login form within a browser window. The browser window has a grey header bar containing navigation icons (back, forward, close, home) and a search bar. The main content area is white and contains two input fields: "Email" and "Password". To the right of the "Email" field is a "Login" button, and to the right of the "Password" field is a "Register" button. The browser window has a grey footer bar with a double-slash icon.

Registration (Student)

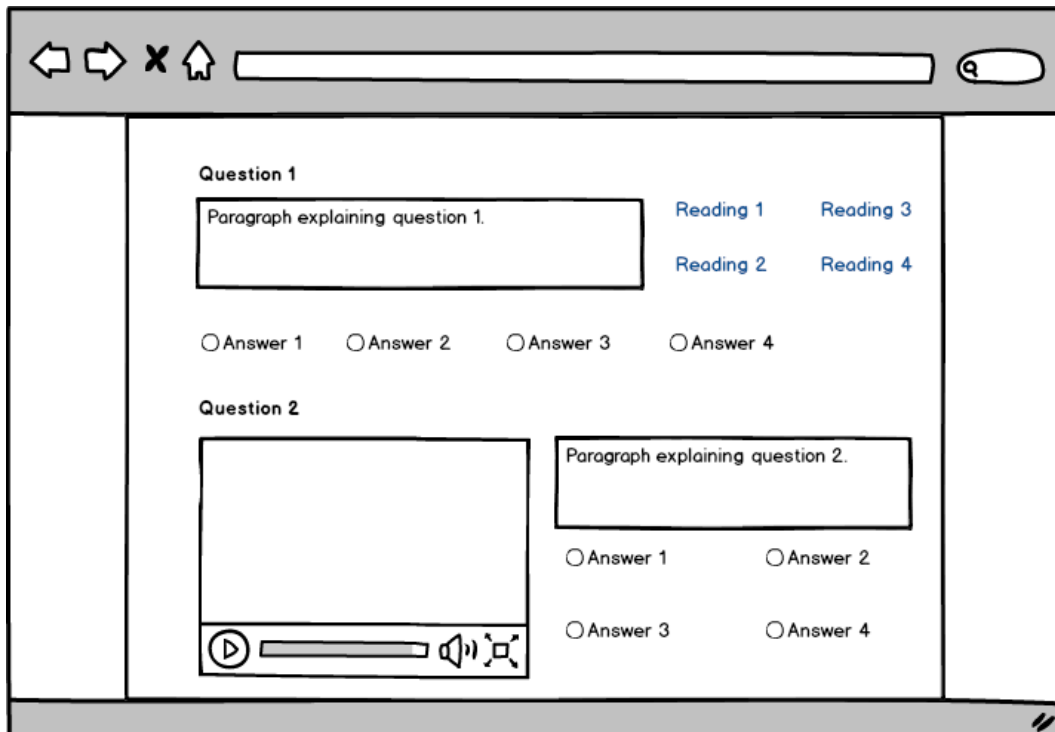


A hand-drawn sketch of a registration form within a browser window. The browser window has a grey header bar containing navigation icons (back, forward, close, home) and a search bar. The main content area is white and contains three input fields: "Email", "Password", and "Confirm". Below these fields is a "Create" button. The browser window has a grey footer bar with a double-slash icon.

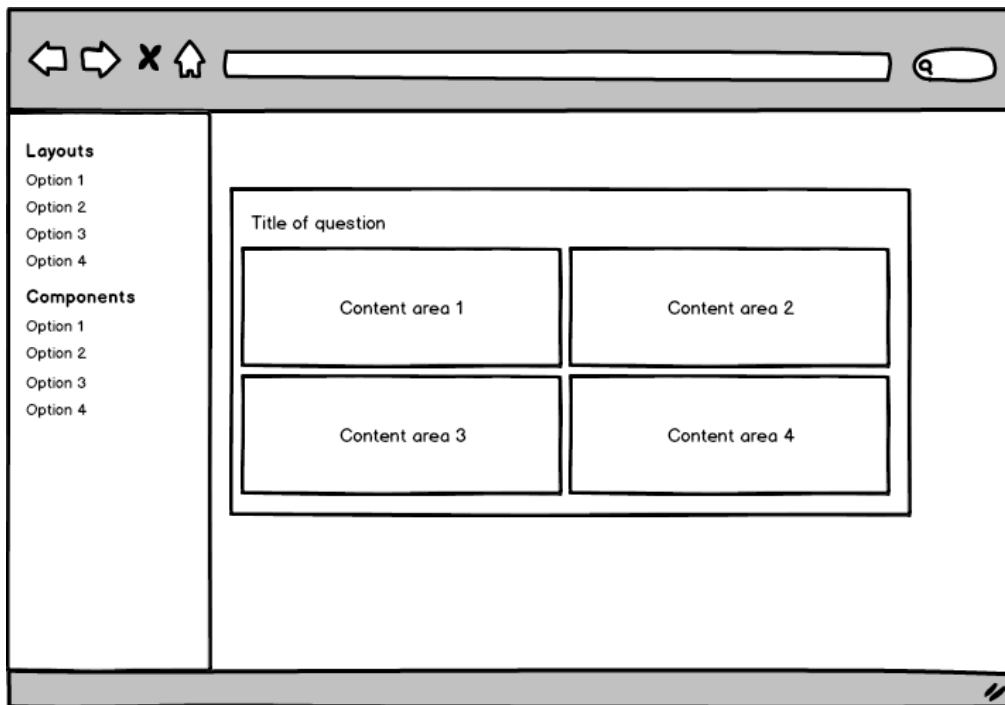
Homepage (Student or Author)



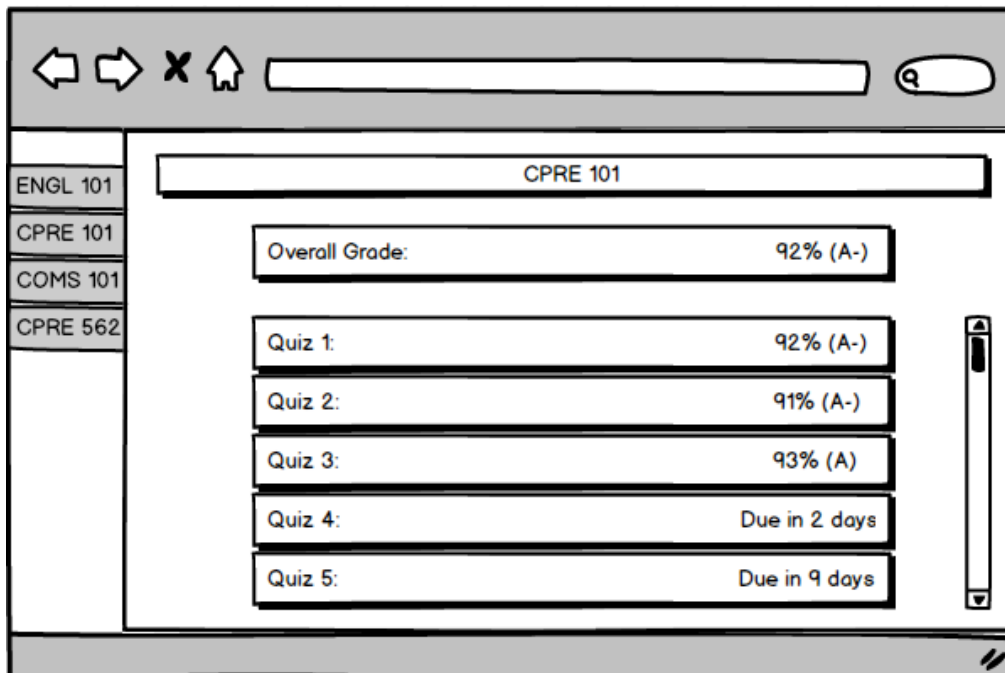
Take a quiz (Student)



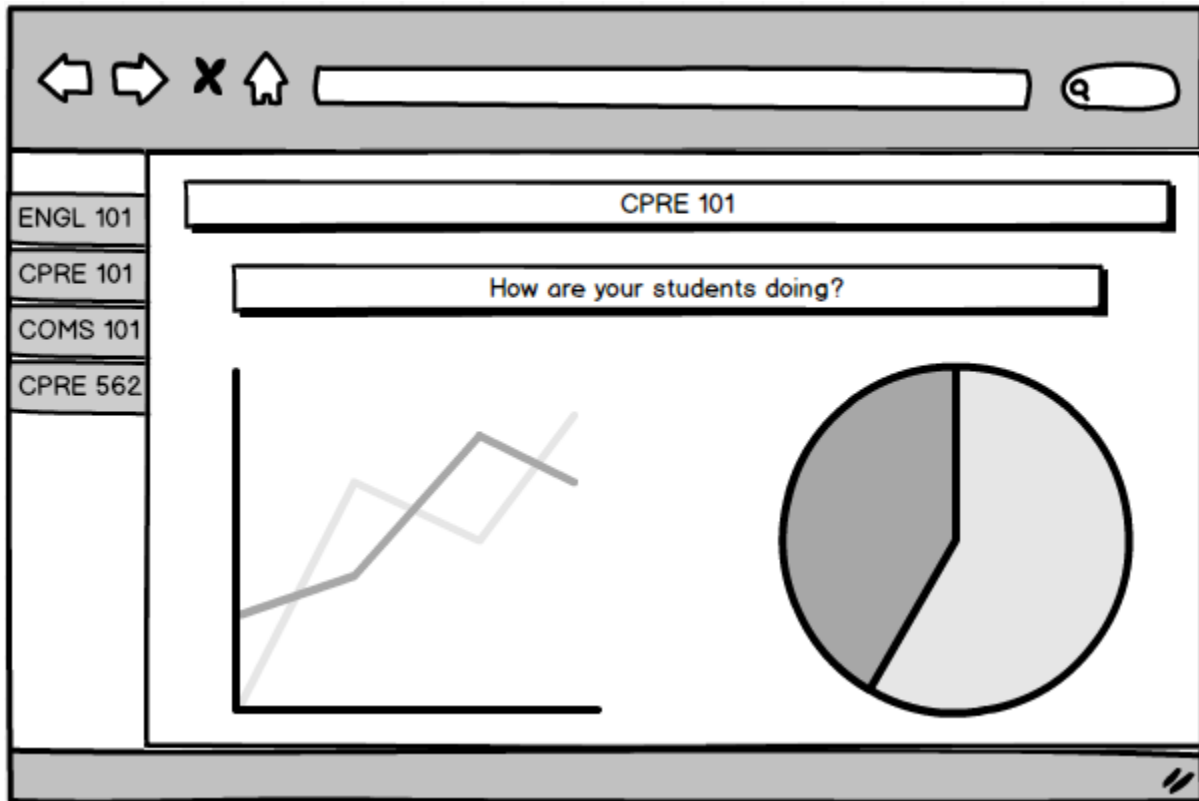
Quiz builder (Author)



Gradebook page







## 6 INTERFACE SPECIFICATIONS

### 6.1 SERVER SPECIFICATIONS

1. HTTP protocol
  - 1.1. Available ports to receive requests from client endpoints
2. MySQL Database
  - 2.1. 100 GB of initial storage for a relational database
  - 2.2. Compatibility with SQLAlchemy for python interface
3. Python
  - 3.1. Python 2.7 or higher
  - 3.2. Flask 10.1 or higher
  - 3.3. SQLAlchemy 9.6 or higher
4. High availability
  - 4.1. Server should have no more than 1 hour of downtime per week initially
  - 4.2. Put plan in place to have server downtime no more than 1 hour per month

## 6.2 CLIENT SPECIFICATIONS

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

## 6.3 SOFTWARE SPECIFICATION

1. Web Browser
  - 1.1. HTML 5 Support
    - 1.1.1. Google Chrome
    - 1.1.2. Mozilla Firefox
2. Server
  - 2.1. Python 2.7 or higher
  - 2.2. SQL Lite 3
  - 2.3. SQLAlchemy 9.6 or higher
3. Imported Python Libraries
  - 3.1. Flask 10.1
  - 3.2. Jinja

## 7 SIMULATIONS AND MODELING

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

## 8 IMPLEMENTATION ISSUES AND CHALLENGES

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.

## 9 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 60% code coverage, and 60% 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.

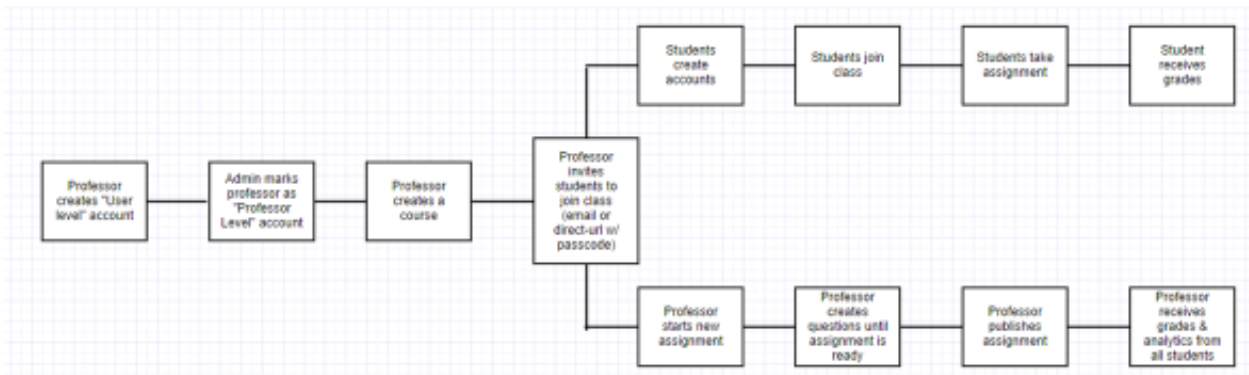
## APPENDIX A

The Use case diagrams are displayed below or in attached document below

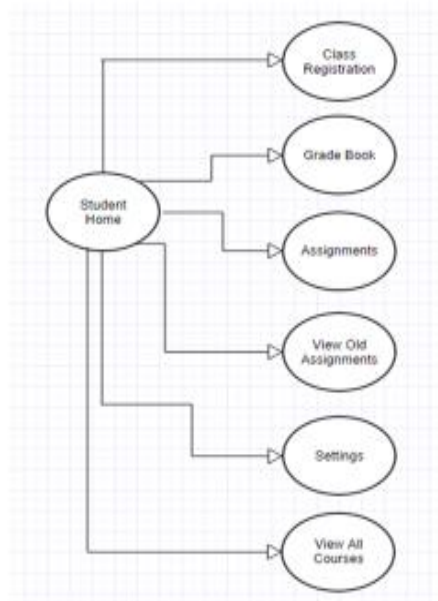


Use Cases.pdf

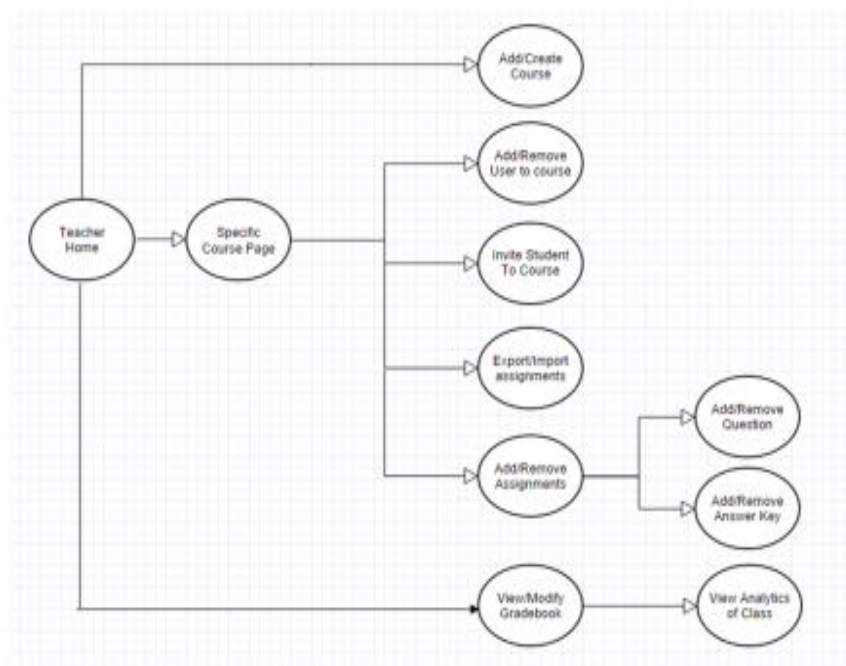
# Overall Overview



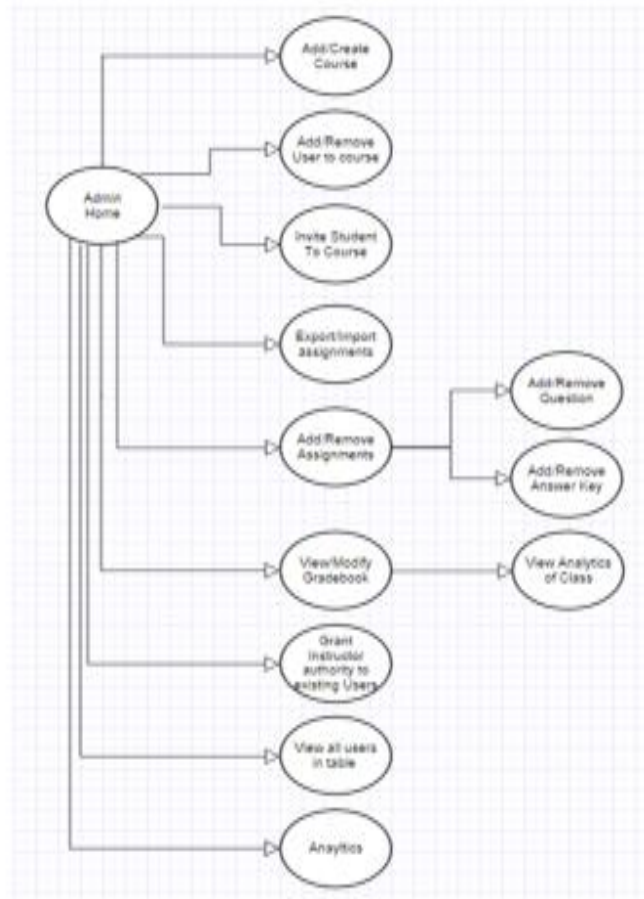
# Student Overview



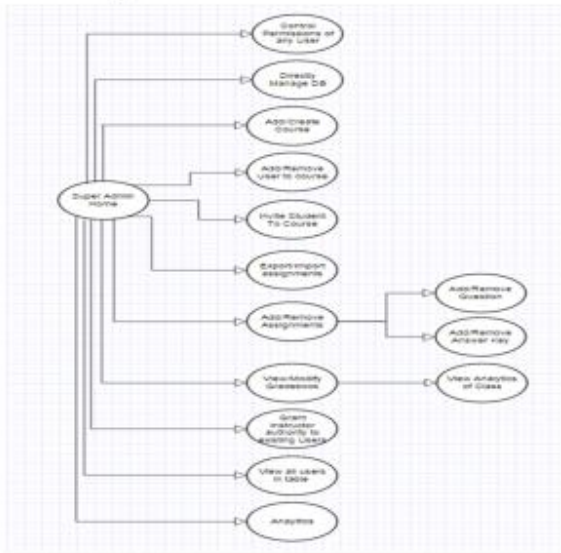
# Author Overview



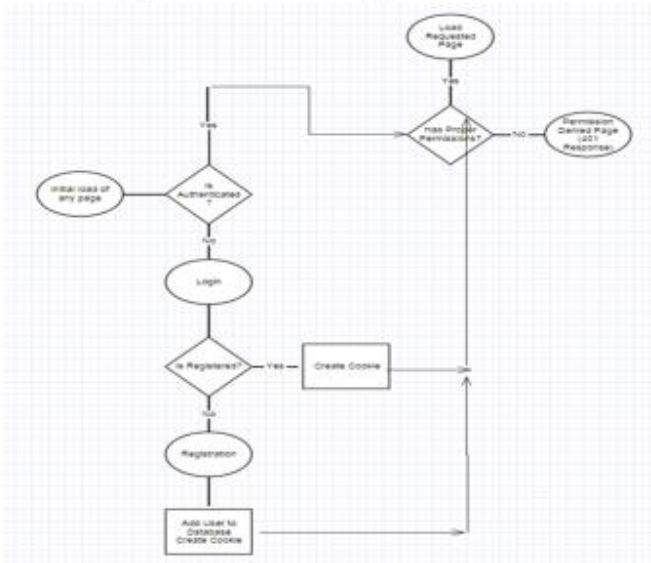
# Admin Overview



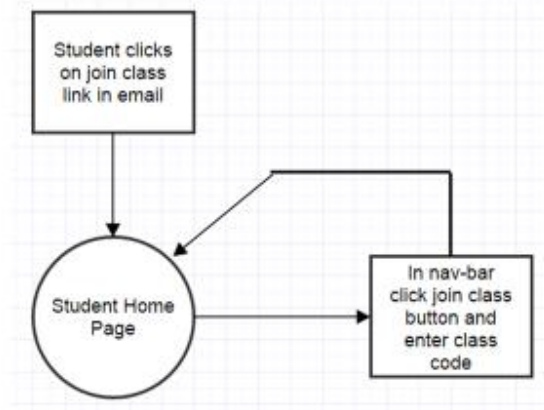
# Super Admin Overview



# Login and Registration



## Student Class Sign-Up

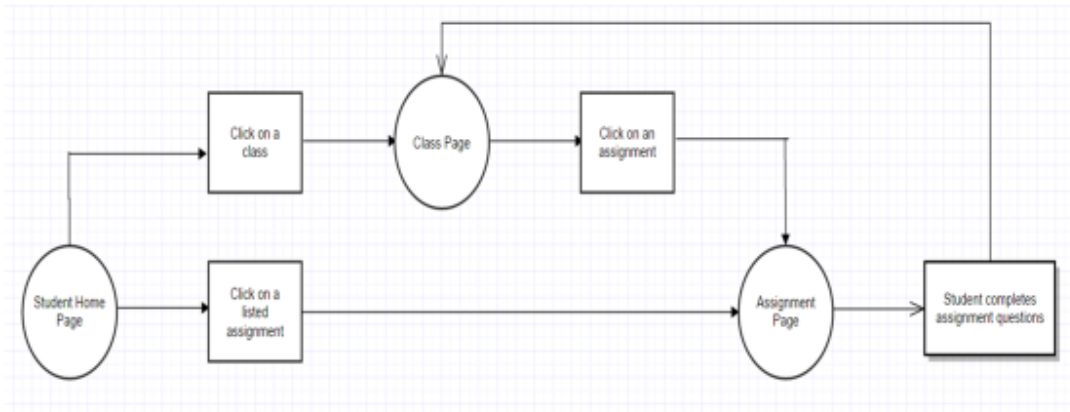


## View Student Gradebook

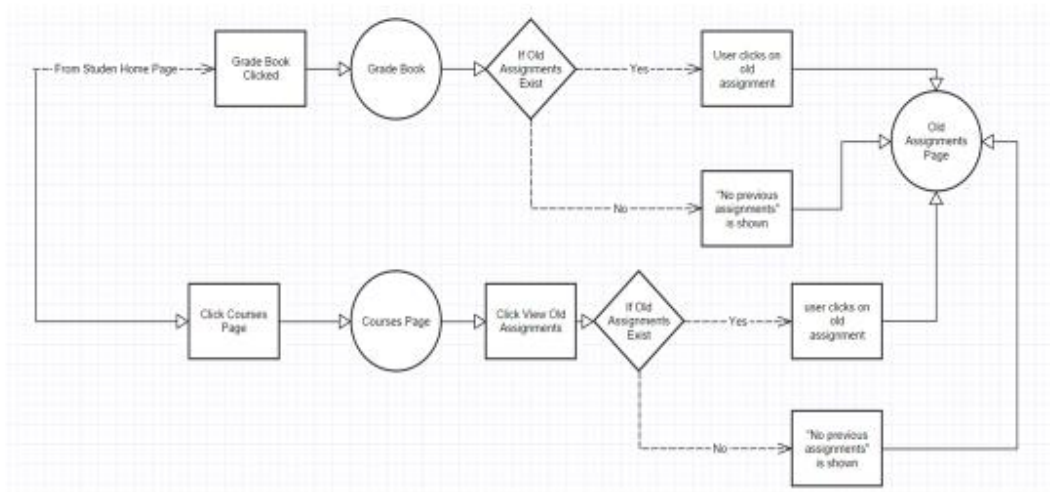




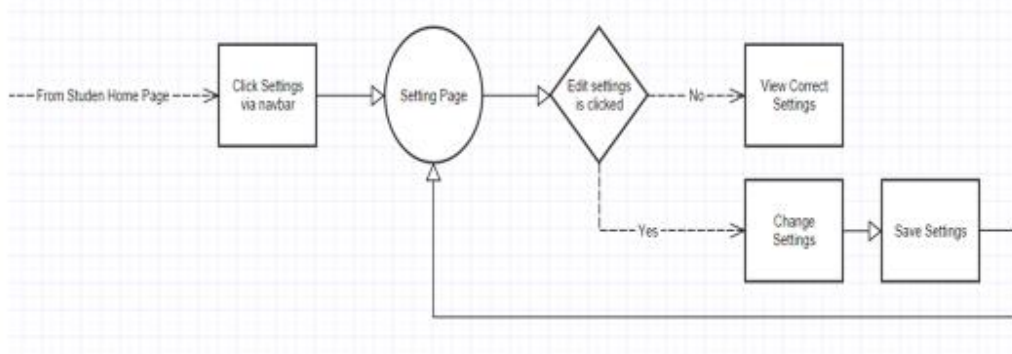
# Take an Assignment



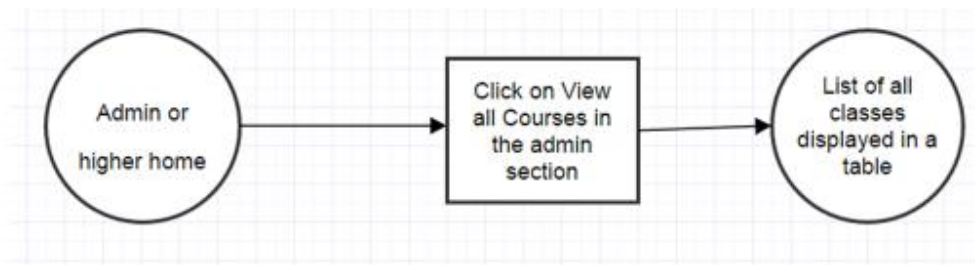
# View Old Assignments



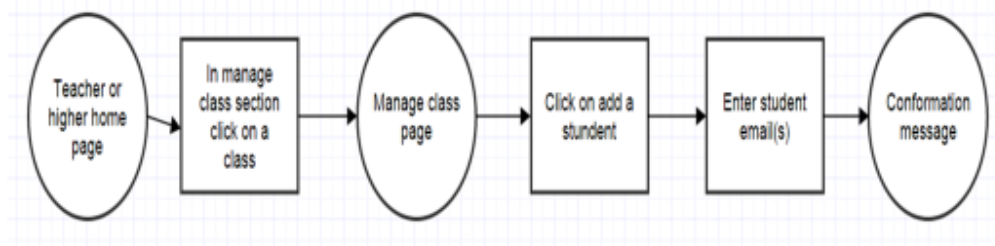
## View/Edit Settings



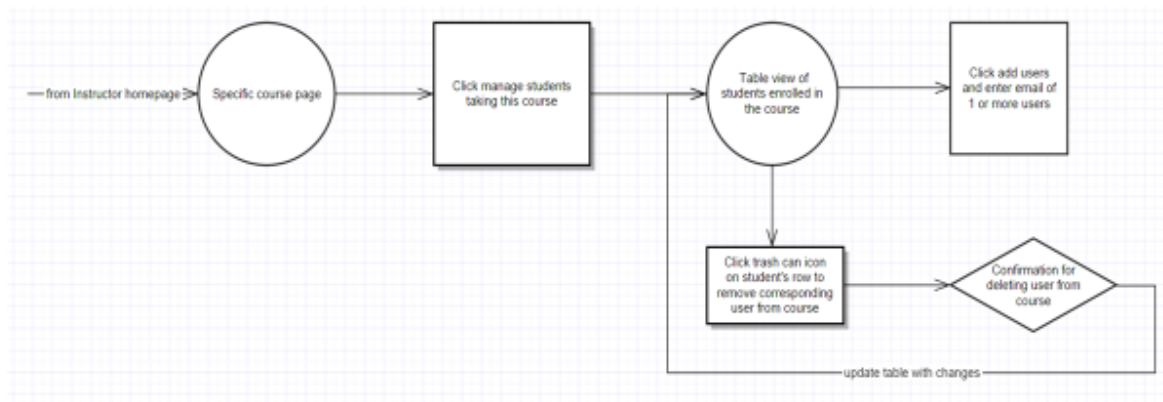
## View Listing of all Courses(w/Filters)



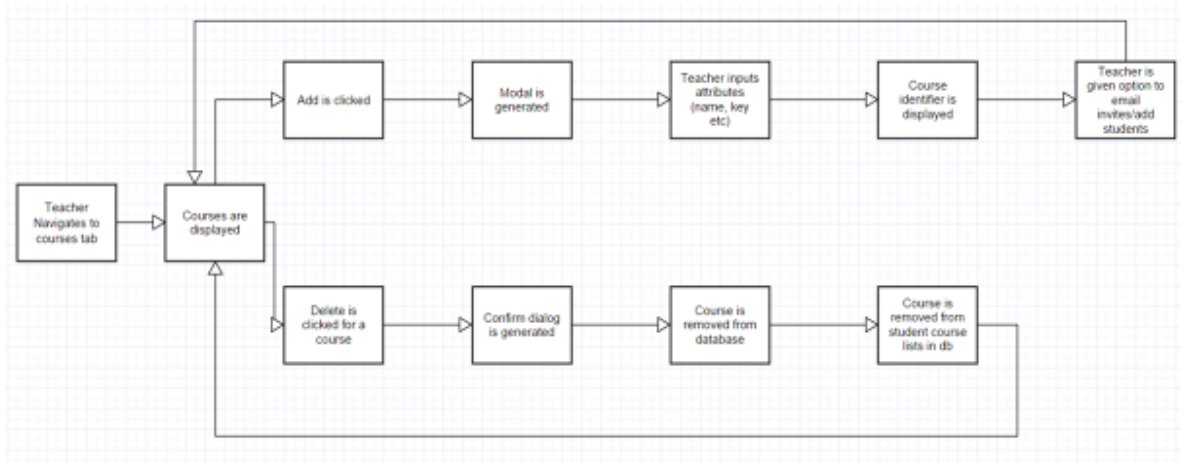
## Invite Students to Join a Course



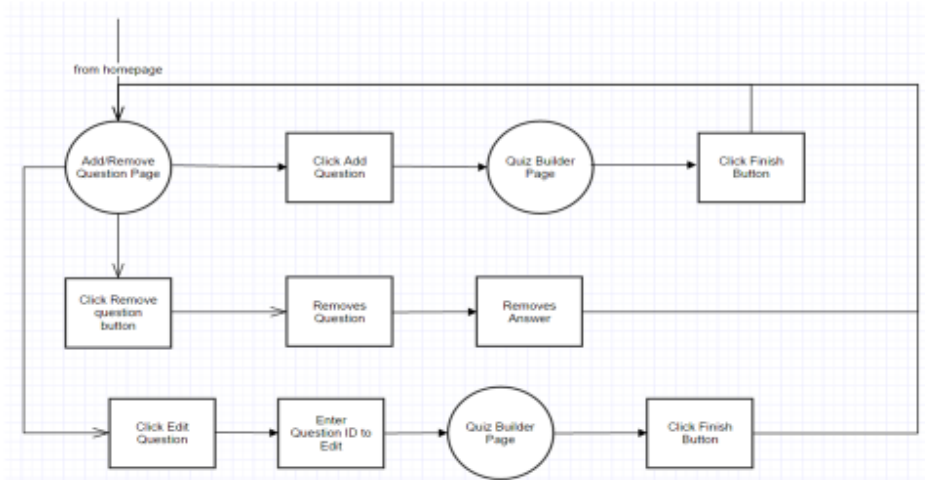
## Author Add/Remove Users to a Course



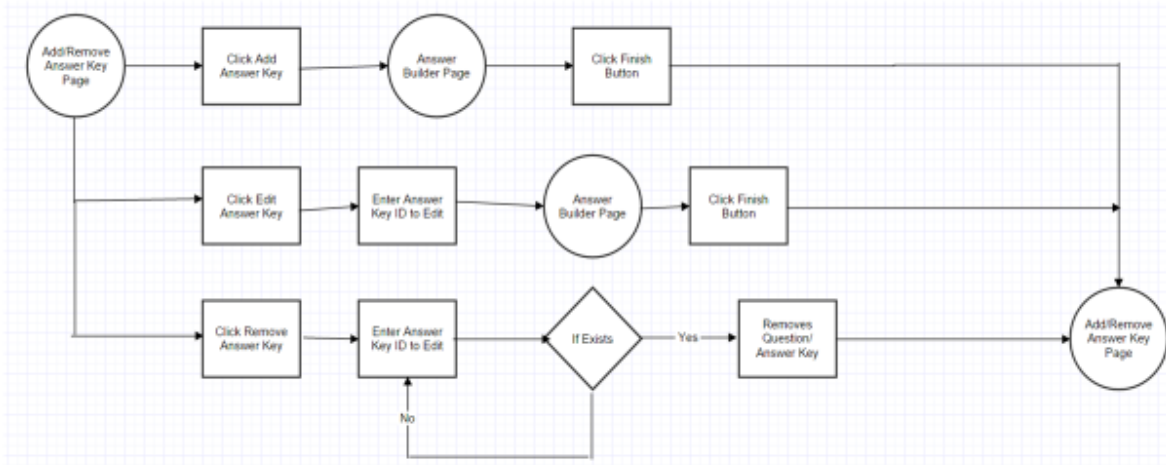
## Author Create/Remove Course



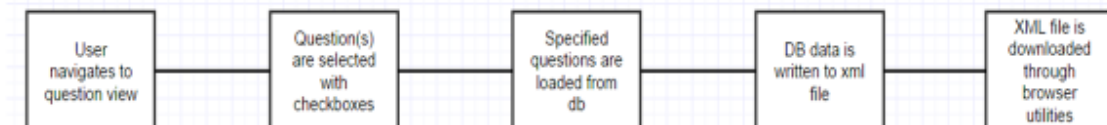
## Create/Remove Question



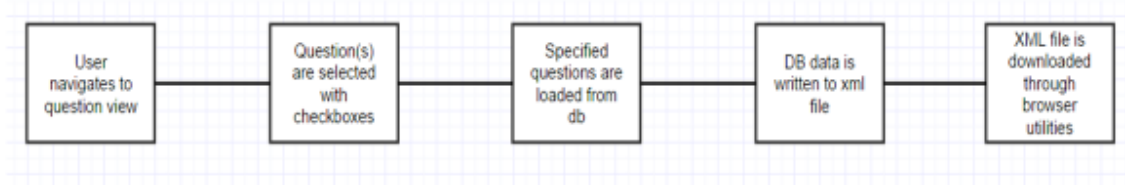
## Create/Remove Answer Key



## Export/Import from XML File



## Export/Import from XML File



## View/Edit User Information of all Users

