CSCE 606: Shockwave Kickers

Team members: Colton Simpson, Jacob Kelly, Kunal Vudathu, Rahul Shah, Stuart Nelson

Iteration 3

- **Team roles**: The team should elect a scrum master and product owner.
  - We have an agreed upon rotation that will shift with each iteration
  - Scrum Master: Stuart Nelson
  - Product Manager: Colton Simpson
- **Customer meeting date/time/place**: The team product owner should contact the customer and set up a regular weekly meeting.
  - Every Friday at 2pm in PETR
- **SCRUM meeting date/time/place**:
  - Every Tuesday and Thursday at 10am
- Github: https://github.com/Lightfoot-Heavy-Machinery/shockwave_kickers
- Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2595510
- Slack Workspace name:
  - Workspace: tamu.slack.com
  - Channel Name: csce606_shockwave_kickers
- Heroku Deployment: https://shockwave-kickers.herokuapp.com/courses

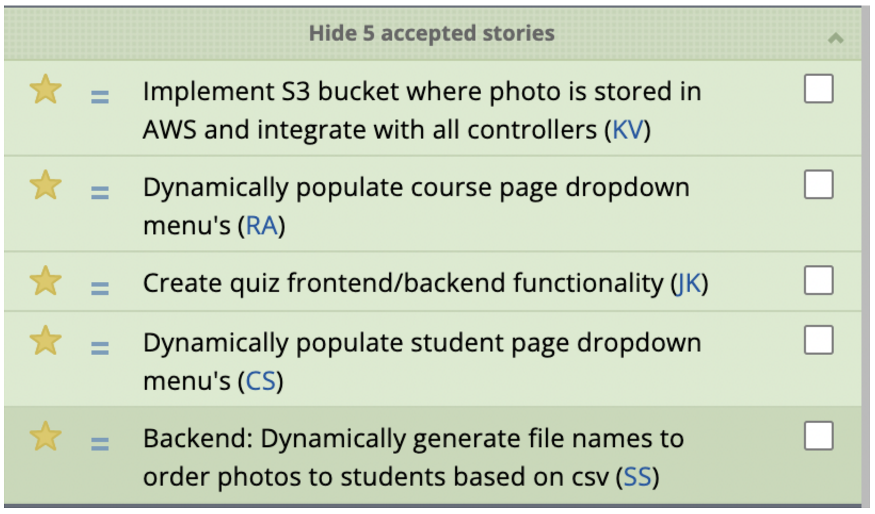
Work Accomplished:

- Implemented basic quiz functionality and UI with randomly selected students
  - Implemented multiple choice quiz format where image is shown and student names are shown. The user needs to pick whose picture it is.
  - Implemented score tracking and quiz history.
- Implemented ruby script that automatically renames image files to be used in student profile uploading
- Reverse lookup for semester and course implemented in Student table
  - Allows per-section identification for a given Student
- Dynamically populated dropdown menus in Course views and Student table
- Implemented image storage of Students to Amazon S3 bucket
  - Allows secure mass storage of Students using NOSQL db
  - Attach image blob to student model
  - Integrated with all controllers including upload controllers.
  - Support of local storage for development

User stories selected in this iteration.

- **Implement S3 bucket where photo is stored in AWS and integrate with all controllers**
    - Each student model needs a picture attached, so we used S3 bucket to store these images as blobs where each image is attached to each student object in postgreSql
    - As part of the upload process, each student will have an attached picture uploaded.
    - The work involves creating S3 bucket and making all controllers/views compatible with the S3
    - Implemented image transformation as all images should be of the same size
- **Create Quiz Backend/Frontend functionality**
    - This story implemented quiz view and CRUD operations
    - This work involves implementing the basic quiz algorithm to show a student image and show multiple names of students in the class and the user has to select one of them.
    - The work also tracks progress made on the quiz
    - You can create a quiz and get back to the incomplete quiz. View history of quizzes.
- **Dynamically the populate dropdown menus in the Course page**
    - When users want to filter the list of students in a certain class, the user will select filters from 3 dropdown menus based on semester, section and tags. These dropdown menus are now populated dynamically based on values in the database, without duplicates.
- **Dynamically the populate dropdown menus in the Student page**
    - When users want to filter the list of students in a certain class, the user will select filters from 3 dropdown menus based on semester, section and tags. These dropdown menus are now populated dynamically based on values in the database, without duplicates.
- **Backend: Dynamically generate file names to order photos to students based on csv**
    - First step in the upload algorithm is to parse the csv and attach photos to the respective customers. This is what this story is about, it makes sure correct photos are linked to the respective students.

Stories in Pivotal Tracker:


Hide 5 accepted stories

Implement S3 bucket where photo is stored in AWS and integrate with all controllers (KV)

Dynamically populate course page dropdown menu's (RA)

Create quiz frontend/backend functionality (JK)

Dynamically populate student page dropdown menu's (CS)

Backend: Dynamically generate file names to order photos to students based on csv (SS)

Stories Added/Changed:

- For iteration 4,
    - We split the course and student page end to end story, as we faced great difficulty with linking the student list to the dropdown menus. We made that its own story and pushed it to the next iteration, and delivered the dropdown menus for this iteration.
    - We also split frontend upload functionality story, as there was a lot of work on the algorithm.

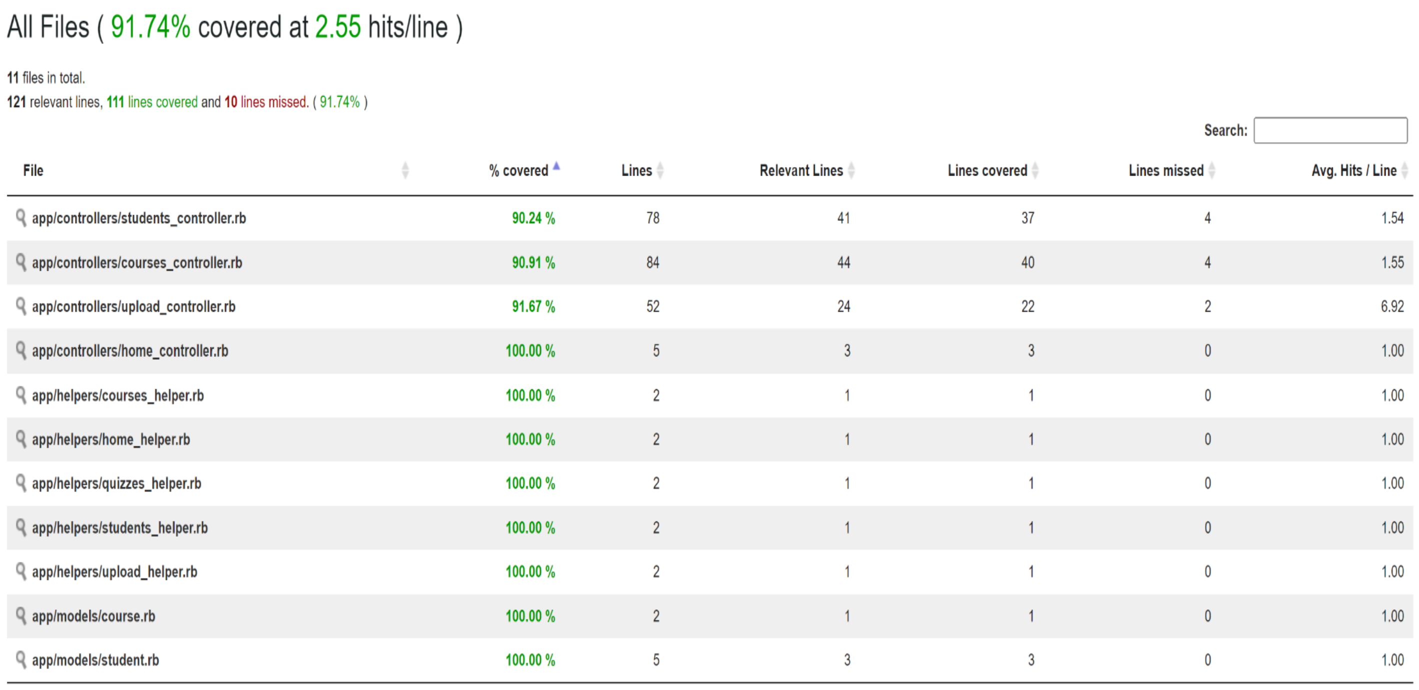
Iteration 4 work:

- Get the Student table and Course view to filter based on dynamically added dropdown menus
- Have a more complex algorithm for quiz, similar to how quizlet
- Finalize Student CSV file uploading with frontend and controller stichting
- Get higher test coverage using cucumber for frontend
- Get CI/CD pipeline working
- Remove duplicate entries out from the course and student view as we create one student entry per course and one course per section.

Test Coverage Report

Unit Cases Coverage:

We increased coverage coverage from 91.26% to 91.74%



All Files (91.74% covered at 2.55 hits/line)

11 files in total.
121 relevant lines, **111** lines covered and **10** lines missed. (91.74%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/controllers/students_controller.rb	90.24 %	78	41	37	4	1.54
app/controllers/courses_controller.rb	90.91 %	84	44	40	4	1.55
app/controllers/upload_controller.rb	91.67 %	52	24	22	2	6.92
app/controllers/home_controller.rb	100.00 %	5	3	3	0	1.00
app/helpers/courses_helper.rb	100.00 %	2	1	1	0	1.00
app/helpers/home_helper.rb	100.00 %	2	1	1	0	1.00
app/helpers/quizzes_helper.rb	100.00 %	2	1	1	0	1.00
app/helpers/students_helper.rb	100.00 %	2	1	1	0	1.00
app/helpers/upload_helper.rb	100.00 %	2	1	1	0	1.00
app/models/course.rb	100.00 %	2	1	1	0	1.00
app/models/student.rb	100.00 %	5	3	3	0	1.00