Sprint 1 Retrospective

Links:
1) Github
2) Project Management page
3) Deployed App

Timespan: Feb 3rd - Feb 14th

Team Contribution:

Team Member Name	Fair Share Percentage
Evan Wu	22%
Jasmin Pena	18%
Haodong Luo	24%
Xiaoyun Chu	18%
Adnan Moheddin	18%

Sprint goal:
Deploy the application and create the basic features for a MVP. Connect our application to a heroku server, and implement testing for our current features.

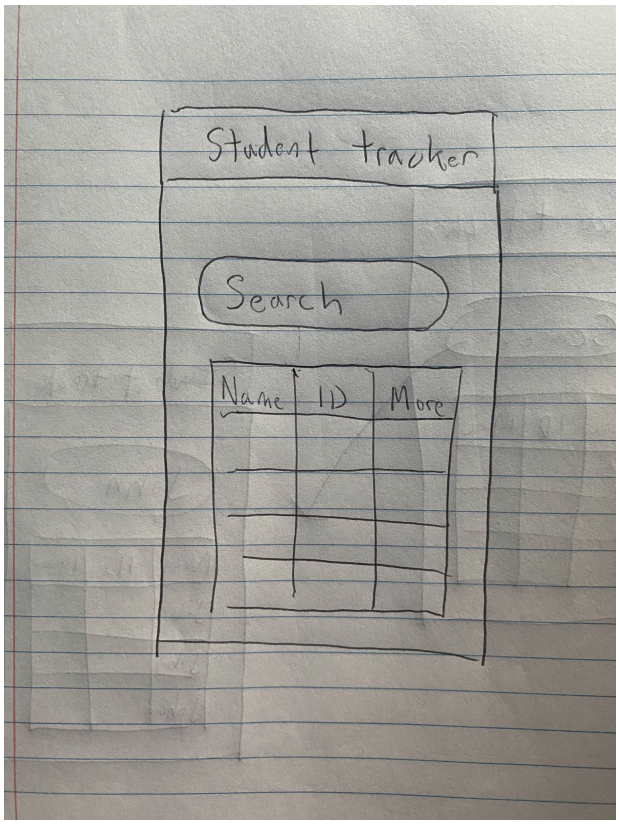
Sprint Achievements:
We successfully deployed the app and added basic CRUD functionality for the student records database. This completed our goal of deploying a basic MVP for the project. We also added testing and achieved over 90% code coverage.

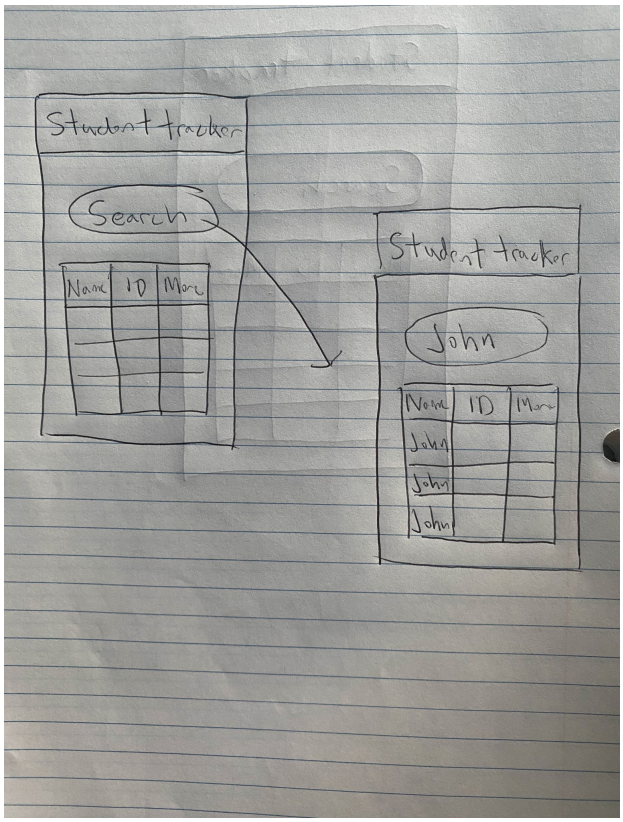
Sprint Backlog and Status:
Summary: We completed all desired features in the backlog for sprint 1. Our main priority was connecting the software on github with the heroku server.
- “Online Deployment of the App”: We created a heroku server and connected our application through github, allowing us to deploy the project.
* “Peer teacher Backlog”: We weren’t able to finish this feature in time so we’re moving it for Sprint 2.
+ “upload student records button”: We implemented a button to allow us to upload the student records.
+ “create student records table in database”: We made the “student records” table in the database itself.

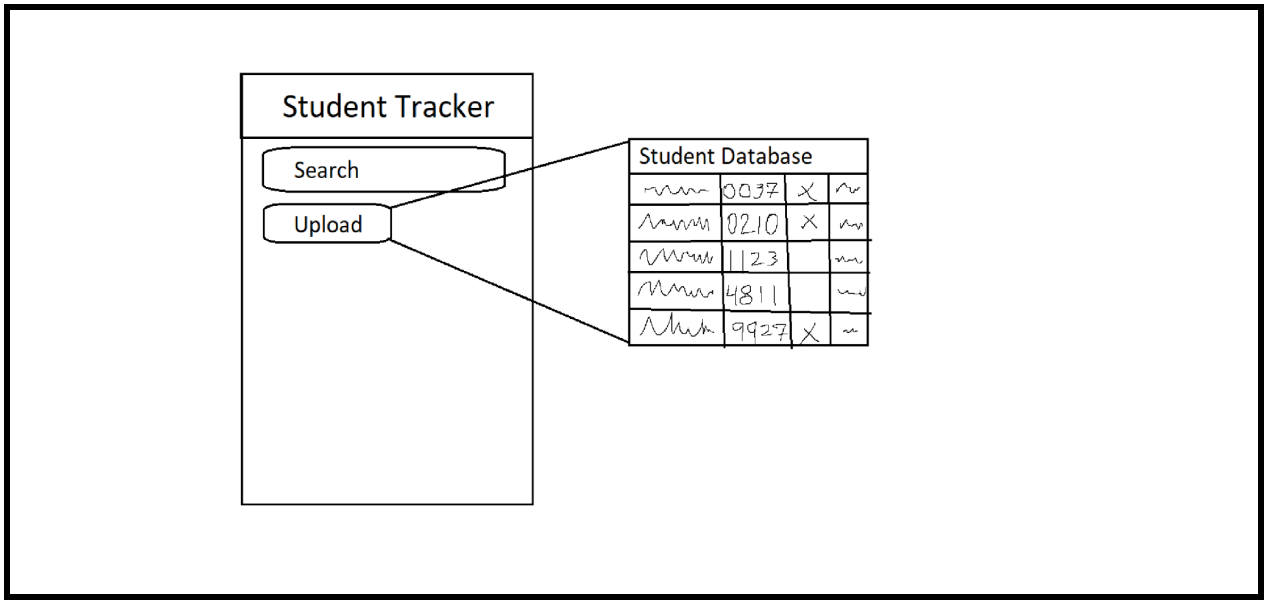
Burndown:

We didn't save the dates when we completed a feature so our burndown chart doesn't look impressive at the moment. The team switched to Jira as it automatically generates a burndown chart when we mark a task as "completed". Hopefully this will solve the problem in the future. https://tamu-team-p9tvl2yg.atlassian.net/jira/software/projects/SCRUM/boards/1/reports/burndown?source=overview

Design diagrams:







Documentation of changes:

We decided to use Jira to track our backlogs instead of using the Github project, so our sprint 1 burndown chart doesn't look very good.

Evaluations of code and test quality:

All Files (96.43%) | Controllers (94.44%) | Channels (100.0%) | Models (100.0%) | Mailers (100.0%) | Helpers (100.0%) | Jobs (100.0%) | Libraries (100.0%)

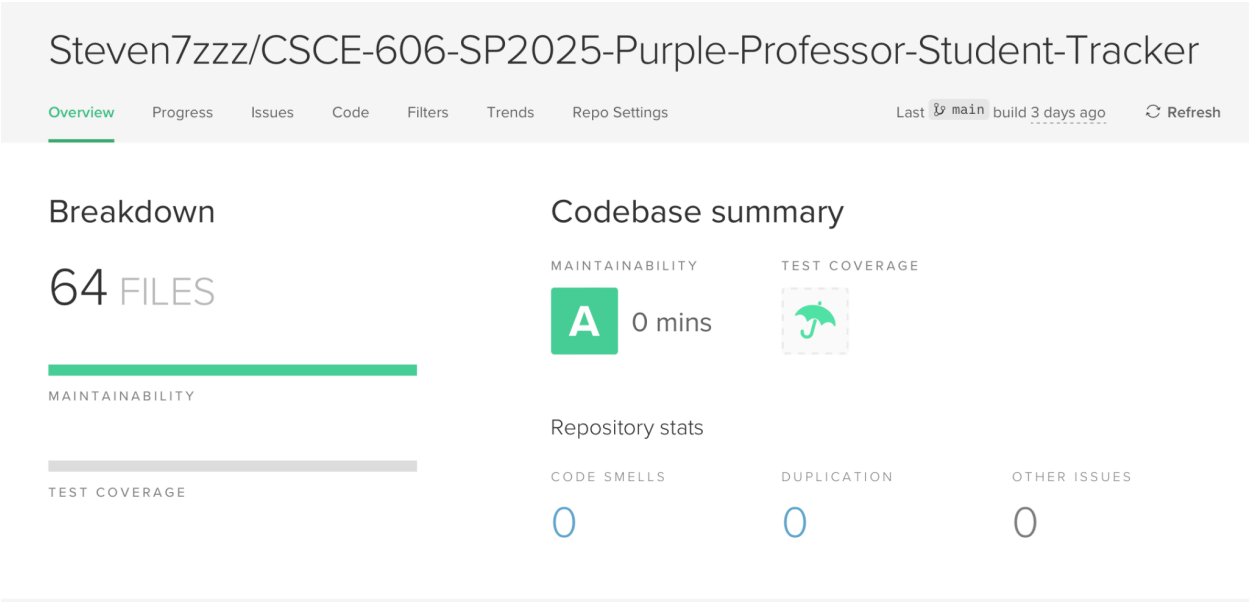
All Files (96.43% covered at 1.11 hits/line)

10 files in total.
28 relevant lines, **27** lines covered and **1** lines missed. (96.43%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/controllers/students_controller.rb	93.33 %	27	15	14	1	1.20
app/controllers/application_controller.rb	100.00 %	2	1	1	0	1.00
app/controllers/home_controller.rb	100.00 %	4	2	2	0	1.00
app/helpers/application_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/students_helper.rb	100.00 %	2	1	1	0	1.00
app/jobs/application_job.rb	100.00 %	7	1	1	0	1.00
app/mailers/application_mailer.rb	100.00 %	4	3	3	0	1.00
app/models/application_record.rb	100.00 %	3	2	2	0	1.00
app/models/student.rb	100.00 %	2	1	1	0	1.00

Code Climate Report



Customer Meeting:
Initial meeting went well. We went over the Lofi sketches of how the application would look like and any desired features to be implemented in the future. We were given the initial .csv files with student data to use and build the initial program around.
We will meet with our customer on 2/21 at 4 PM over zoom demo our sprint 1 MVP and discuss future steps.

BDD & TDD:
Feature: Online Deployment of the App
Cucumber Scenario:
When you use the link to our app
Then you should see the home screen
When you click on the buttons
Then it should redirect you to the correct pages

No rspec test needed for this one

Feature: Upload student records button
Cucumber Scenario:
Given that you're on the homepage of the purple prof. project.
When you click "add student".
Then You will be redirected to the students/new page.
And you can add a new student based on current information.

```
RSpec.describe StudentsController, type: :controller do
  describe 'POST #create' do
    let(:valid_attributes) do
```

```
    {
      student: {
        uin: '123456789',
        first_name: 'John',
        last_name: 'Doe',
        major: 'Computer Science',
        email: 'john.doe@example.com'
      }
    }
  end
  context 'with valid parameters' do
    it 'creates a new Student' do
      expect {
        post :create, params: valid_attributes
      }.to change(Student, :count).by(1)
    end
    it 'redirects to the created student' do
      post :create, params: valid_attributes
      expect(response).to redirect_to(Student.last)
    end
  end
 end
end
```

Feature: Create student records table in database

Cucumber Scenario:

Given that you added a student through the “add student” feature.
Then the student information should go to the records table in the database.
And the information should be saved for next time.
When you go back to the homepage.
Then you should be able to see the new student information displayed.

```
RSpec.describe "StudentsNew", type: :request do
  describe "GET /students/new" do
    it "returns a successful response" do
      get new_student_path
      expect(response).to have_http_status(:success)
    end
  end
end
```