TEXAS A&M UNIVERSITY

CODE JUDGE

Spring-2023

Anmol Anand
Druva Kumar Gunda
Loksubhash Pulivarthi
Nachiket Umesh Naganure
Nagarjuna Kolloju
Praveen Kumar Maddikunta

Table of Contents

Project Summary

Code Judge is a legacy project site for Texas A&M instructors to assign programming problems to their students. While it provides some of the core functionalities, we are asked to implement some new features to enhance the usability of the Code Judge application. Some of the requirements from the customer include assigning multi-tags to problems, adding students to class groups, adding problems to class groups, student rating system, filter based on tags, filter based on difficulty, language restriction, dynamic faculty scoring for problems based on student submissions, access code to join a class, tagging functionality in questions, Bulk upload of test cases and student upload their solution.

We have started with the story of filters based on problem tags and difficulty, which tests some of the core functionalities that are necessary for other feature stories. For that, we have changed the necessary backend schema and created new UI screens for page rendering. Following that, we concentrated on student grouping, multi-tag systems, and so on. We have accomplished almost all the functionalities, but due to time constraints, we couldn't complete a few stories and statistics on student groups. Regular meetings with clients helped us concentrate on the high-priority stories and kept us on the right track.

User stories

Completed user stories

1. Language restriction dropdown menu for a problem - 184562729 (2 points): This story was created to add a language restriction dropdown menu while creating a problem so that the instructor can choose whether that problem has a language restriction. If the instructor chooses 'None', it means the instructor is not enforcing any language restriction on it.
2. Populating tags and language restriction in problems table - 184562837(2 points): This story was created in order to add support for language restriction and tag support in the problems table.
3. Restrict problem creation to once in a class - 184674914 (2 points): The instructor should not add an existing or new problem multiple times to the same class.
4. Display appropriate tags for problem list - 184674811(2 points): For problem lists, we should display appropriate tags in both student and instructor views.
5. Add a difficulty level to the problem based on the number of attempts. - 184676523 (2 points): The difficulty level of a problem should dynamically change based on the wrong attempts made by the students.
6. Filtering problems based on difficulty level- 184452819(2 points): Once the difficulty level is computed, we should be able to filter the problems based on the difficulty level.
7. Multiple tags to a problem - 184893205 (3 points): A problem can have multiple tags; hence the instructor can add multiple tags to the problem while creating it. We should be able to filter the problems based on tags as well.
8. Evaluate the submissions using compiler - 184893184 (2 points): When a student submits the code using the attempts page. The compiler needs to compile the code and result in the final o outcome.

9. Add a new page for user submissions and student ratings - 185057112(2 points): Create a new page rating, so that the instructor can view the ratings of students.
10. Add a user message for students in case a particular class is not found by the student - 184674766 (3 points): Throw an error message when a class is not found by the students while joining with the wrong class code.
11. Enforce Language restriction when specified by the instructor - 184674795(3 points): While creating a problem. if the instructor specifies a language in which the student needs to submit the solution, the student should submit the solution in the mentioned language only.
12. Restrict the creation of problems without title - 184674855 (2 points): Instructors should not be able to add or create new problems if the title is empty.
13. Fix search in instructor and student views - 184674756 (2 points): Search functionality is not working as expected. Search results show all the available problems irrespective of the search query.
14. Calculate student rating based on a number of correct attempts - 184893188 (3 points): If a problem is attempted by the student, the compiler evaluates the code and records whether it is correct or not. Once it is done, the rating is computed based on its correctness.
15. Add Rspec tests - 185057110 (1 point): Add Rspec tests to the newly added features respectively.
16. Add cucumber tests- 185057134 (1 point): Add cucumber scenarios for various features and implement their step definitions respectively.
17. Add a class description to class details - 185057119 (1 point): Display class description and class details when the instructor opens a class.

Completed bugs

1. Resolve Rspec issue - 184676524 (2 points): Whenever we run spec or cucumber test for this legacy project, it is trying to upload the local code coverage report to codecov(external tool). Codecov was trying to connect to our repository, but it needed a GitHub repo token. Eventually, it led to the failure of Rspec and cucumber.
2. Deleting a class throws an error - 184674844 (2 points): When we delete a class, it doesn't get deleted, instead it is throwing an error.
3. Editing a problem throws an error - 184674721 (2 points): When we log in as an instructor and try to edit a problem, it results in an error.
4. Project Deployment in the local environment - 184554317 (2 points): Set up the application locally and fix any bugs during the deployment.

Ice boxed user stories

In this section, you'll find user stories that were created during the first iteration but were neither prioritized nor assigned points. Due to the rapidly approaching deadline, these stories were subsequently "ice boxed." However, the following team may find them useful as a source of inspiration for incorporating additional features.

1. Add Google signup authentication - 185112332
2. Implement time restriction for the run time of problem submissions - 185112339

3. Can suggest a class to use a particular language - 184554216

Modifications to legacy project

Schema changes

Columns added

- We have added tags and language columns to the problems table so that while creating a problem, the instructor can add tags and language restrictions.
- We have added a column rating to the user's table so that it can be used to compute the rating of the students.
- We have added a difficulty column to the problems table to record the hardness of the problem based on correct submissions.

Tables created

- We have created a table of difficulty_levels to store the difficulty levels of the problems.

Tables modified

- We have modified problem_tags and difficulty_levels tables and changed them to have difficulty_level_id as a foreign key.
- We made the default value of the difficulty column in the problems table: 11
- We made the default value of the rating column in the user's table: nil

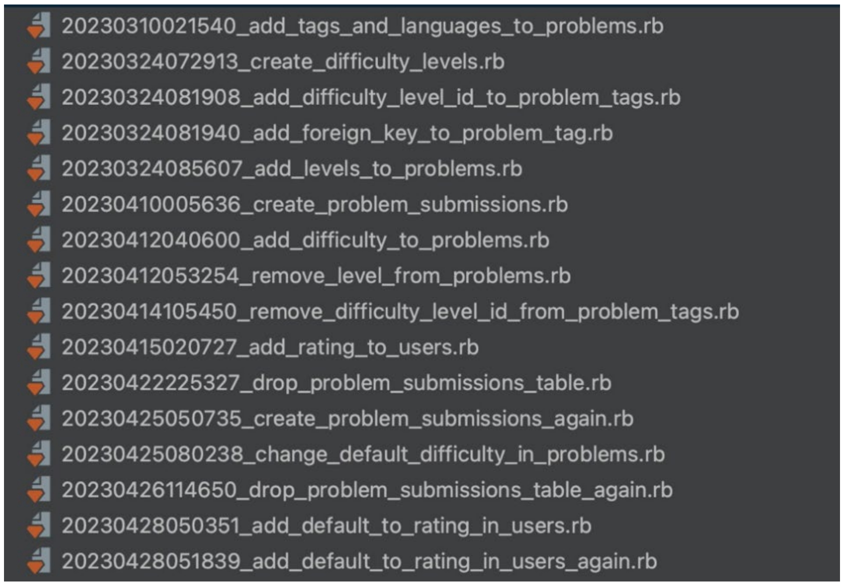
Relationships added

- One to One relationship between problems and languages tables.
- One to One relationship between problems and difficult_level tables.
- One to Many relationships between problems and problem_tags tables.

Migration changes

Added new migrations to support new schema changes described above. Below is the list of new migrations that were added.

Migrations added



```
20230310021540_add_tags_and_languages_to_problems.rb
20230324072913_create_difficulty_levels.rb
20230324081908_add_difficulty_level_id_to_problem_tags.rb
20230324081940_add_foreign_key_to_problem_tag.rb
20230324085607_add_levels_to_problems.rb
20230410005636_create_problem_submissions.rb
20230412040600_add_difficulty_to_problems.rb
20230412053254_remove_level_from_problems.rb
20230414105450_remove_difficulty_level_id_from_problem_tags.rb
20230415020727_add_rating_to_users.rb
20230422225327_drop_problem_submissions_table.rb
20230425050735_create_problem_submissions_again.rb
20230425080238_change_default_difficulty_in_problems.rb
20230426114650_drop_problem_submissions_table_again.rb
20230428050351_add_default_to_rating_in_users.rb
20230428051839_add_default_to_rating_in_users_again.rb
```

Views for Different Roles/Users:

We added a new admin role and a view to control the privileges of other users on the platform. Previously Instructors could edit problems that they did not create. We have added restrictions that prohibit them from editing problems that they did not create.

Views differ based on the privilege assigned to the user in case of problem creation/edition, class creation, and overall user management on the portal. Admin and Instructor have a different set of headers and components to support their roles. Some of the UI and general flow changes requested by the client are as follows:

- Admin can remove any user from the platform
- Instructors can now add Multiple Tags to a problem
- Students can see multiple tags and filter based on them
- User Rating page is available for both Instructor and Students
- Comprehensive error message display for failed test cases for student view.

Roles of team members

Each team member had taken on the responsibilities of both scrum master and product owner at least once. Our team utilized a handover process, whereby the previous product owner would discuss the current sprint status and the backlog tasks that were ready for the next sprint with the new product owner. This helped establish the context and expectations for the upcoming sprint. Any necessary changes to the sprint stories would be made after consulting with the client.

Iterations & Customer Meetings

The following section describes the meeting summaries as well as a breakdown of what was completed in each iteration of the project.

Iteration 0 + Meeting 0

We met our client Dr. Ritchey on Feb 06, 2023, for about 30 mins to discuss the requirements and to understand the Code Judge application. The professor described the requirement as a sort of online coding platform like Kattis and Leet code with some additional functionalities to them. In this meeting, the professor explained the deliverables he was expecting, such as his need for this project, including the ability to upload batch test cases, a multi-tag system for problem categorization, a rating score for students, and an instructor-tracking feature for student progress. These additions and improvements are requested to enhance the existing project and provide a better user experience.

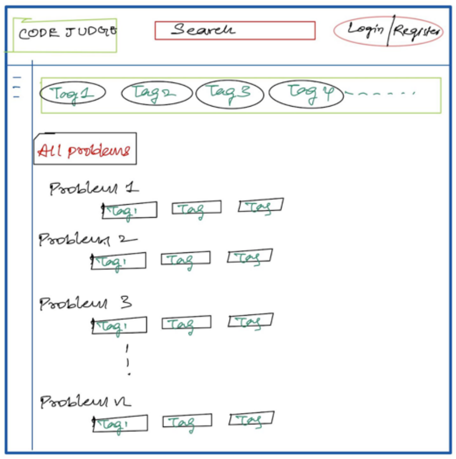
To meet the requirements of Dr. Philip Ritchey, the project will add several new features to enhance the user experience. These features include:

1. Batch test case upload functionality.
2. An icon to indicate the problem's difficulty level.
3. Improved filtering options based on problem status,
4. Topic tag
5. Difficulty level to each problem,
6. Student Score - 0 or 1 based on if all test cases are passed or not.
7. Group and global level rank list
8. A better understanding of the existing schema and code flow.

Hence, created eight major stories depending on client requirements, each epic represents a major goal in achieving a better experience for the end user. These are created as a feature that needs to be developed upon the existing code judge. We worked on some of the UI mockups. Some of them are attached as images below.

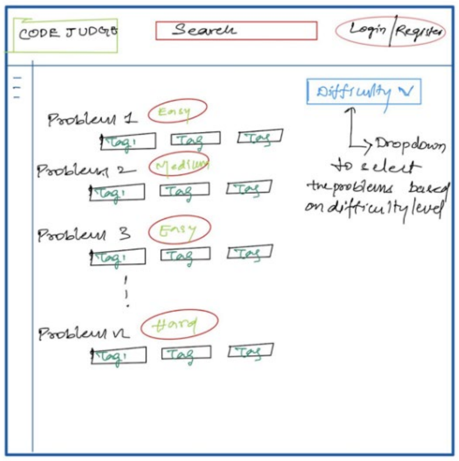
For students

Adding topic wise tags to problems to filterout.



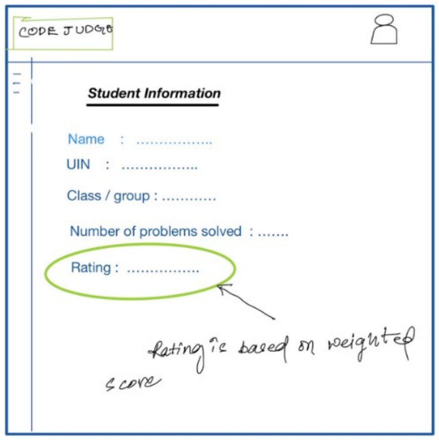
For students

Filter based on difficulty level to the problems



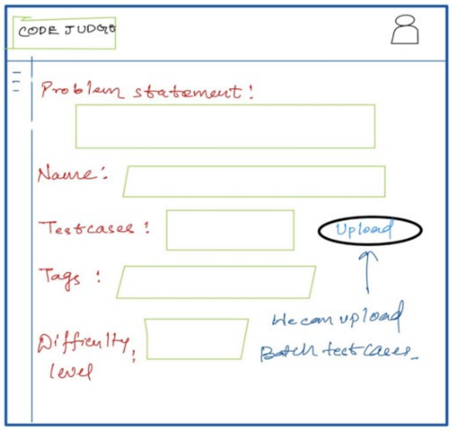
For students

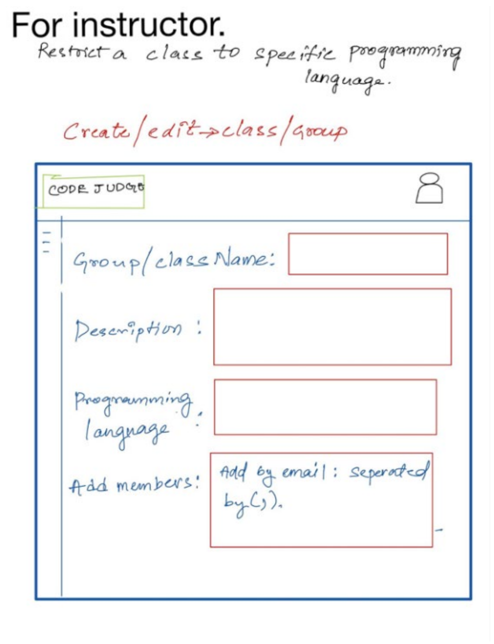
Student rating based on weighted score

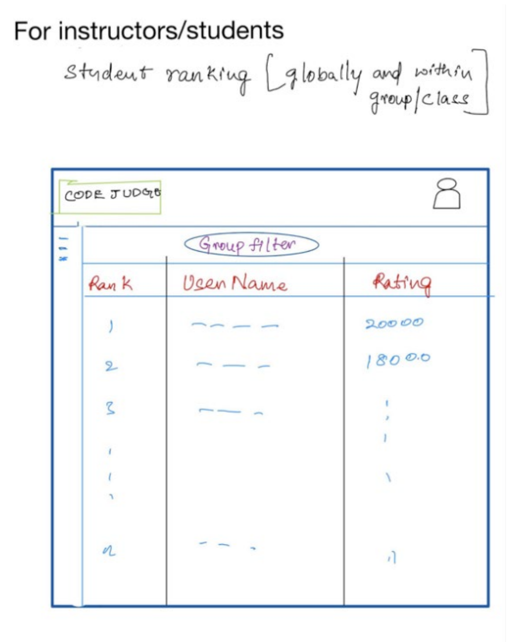


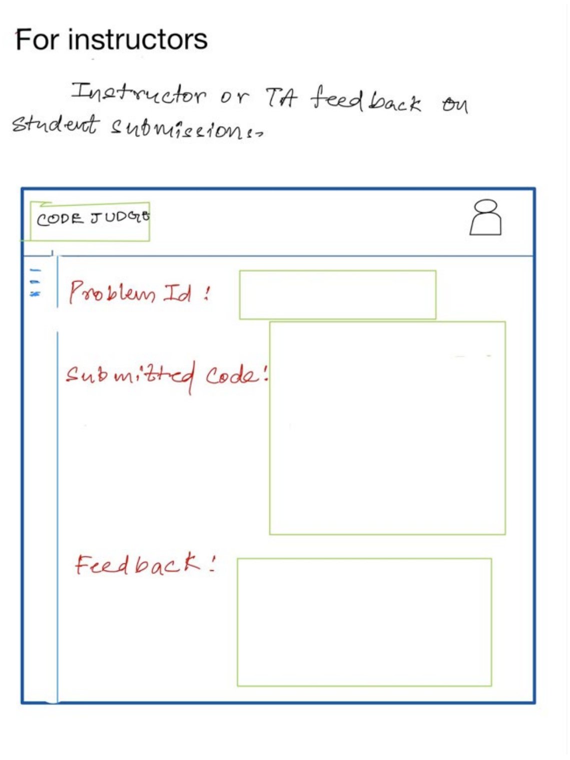
CODE JUDGE
Student Information
Name : ....
UIN : ....
Class / group : ....
Number of problems solved : ....
Rating : ....
Rating is based on weighted score

For instructors : Enhancement for problem Creation flow :-
1) Batch Test Case upload
2) Add multi-tag functionality
3) Difficulty level









Iteration 1 + Meeting 1

Completed Stories:

- Create a document for local setup (184554326)
- Project Deployment in the local environment (184554317)
- Understand the existing schema and code flow to enhance problem creation (184452848)

The entire team was focused on preparing the local development environment, which involved cloning the existing git repository onto their local machines. Following that, they set up the necessary configurations and finally deployed the application to the Heroku platform. Since this was a legacy project, the development team dedicated the first iteration to installing the current version on their machines, setting up the working environment, and comprehending the code and its various components. During this iteration, the group held multiple discussions where everyone exchanged thoughts on the application's functionality and gained a deeper understanding of the code's different parts.

Iteration 2

Completed stories:

- understand existing code base, functionality and tools associated (184554341)
- deploy project to heroku (184554331)
- Adding language dropdown menu while setting a problem (184562729)
- populate tags and language restrictions in the problems table (184562837)
- Fix error for empty test case creation (184674828)
- Restrict a problem creation multiple times a in class (184674914)
- Fix search in instructor and student views (184674756)
- Tags are not displayed in problem-list for instructor and student views (184674811)
- Add test cases for language restriction functionality (184668643)
- In Student view, Appropriate message should be given for wrong class code (184674766)
- Deleting class group throws an error (184674844)
- Throw an user message once a problem is added to a class in instructor view (184674897)
- In the student view, language restriction is not shown while attempting a problem which has a restriction (184674795)
- Restrict creation of problems without title (184674855)
- In instructor view, edit problem throws an error (184674721)

Following the successful deployment of the application to both the local and Heroku environments last week, the entire team is now ready to begin working on development stories. To ensure optimal progress, I prioritized all the epics and identified the one that's currently blocking most of the remaining epics. Furthermore, our first meeting with the client highlighted the crucial need to implement support for filtering problems based on tags and difficulty. As a result, we selected this epic as our top priority, as it will provide us with valuable insights into the Code Judge application's core functionalities while potentially introducing new discoveries. Given that no one in the team had experience in working on a Ruby project, it took some time to fully grasp the application's flow, syntax, and other components.

Iteration 3 + Meeting 2

Completed stories:

- add difficulty level to the problem based on the number of attempts (184676523)
- Filtering problems based on difficulty level (184452819)
- set a default selection in case no tag is selected (184676528)
- resolve why spec is not working (184676524)
- story creation, acceptance, deployment to Heroku(184816870)
- Report (184816806)
- write three test cases (184676526)

During this iteration, the team focused on fulfilling new functionalities and a dependency story that was required to be completed for future stories. We successfully completed these tasks and investigated why Rspec was not functioning correctly. As the previous team had not written many Rspec or Cucumber tests, we had to develop new test cases to increase code coverage.

Additionally, we met with Professor Philip Ritchey to demonstrate the new UI screens and explain why the changes were critical in delivering the user profiles feature. Based on the client's feedback, we created some new stories in the backlog that are necessary to deliver the feature.

Iteration 4 + Meeting 3

Completed stories:

- Add a range filter for problem ratings on the problems page (184893203)
- apply language restriction for code evaluation (184893186)
- Multiple tags (be able to add multiple tags and apply a filter on problems page using multiple tags) (184893205)
- Compiler evaluates and records submissions correct/wrong - (Scope work) (184893184)
- Calculate student rating based on number of correct attempts (184893197)
- Heroku deployment and documentation report (184893188)

This iteration involved features include a range filter for problem ratings, language restrictions for code evaluation, the ability to add and filter problems by multiple tags, compiler evaluation and recording of submission correctness, calculation of a student rating based on correct attempts, and documentation for deployment on Heroku. The rating range filter would make it easier for find users with a particular level, while language restrictions would ensure that code is evaluated only in the intended language. The ability to add and filter problems by multiple tags would facilitate better organization and searching for relevant problems. Compiler evaluation and recording would provide valuable feedback to users, and student rating calculation would help track progress. Finally, documentation for Heroku deployment would make the application more accessible.

Iteration 5 + Meeting 5

Completed stories:

- write cucumber test for modify class details to increase code coverage (185057125)

- Write rspec tests (185057110)
- created the user submissions page and ratings page (185057112)
- Display problem tags in class groups problem list (185057115)
- add class description to class details (185057119)
- dynamic difficulty and scoring based on number of student submissions (185057123)
- write cucumber test for delete class details to increase code coverage (185057134)
- document, heroku, story management (185057136)

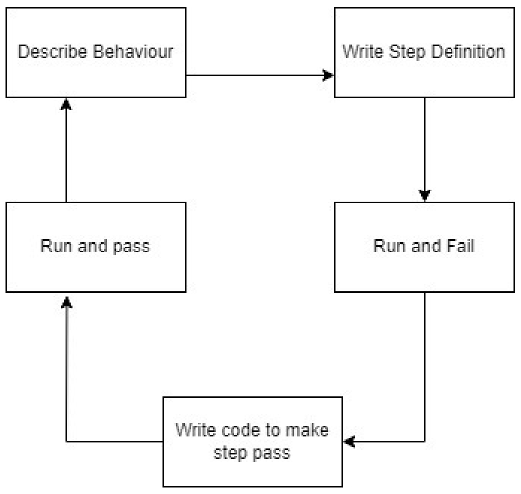
One of the features added was the ability to modify class details, and Cucumber tests were written to increase code coverage. In addition, RSpec tests were written to ensure that the application was functioning as intended. The team also created a user submissions page and ratings page to provide feedback to students. They also added problem tags to the class groups problem list, which would make it easier for students to find relevant problems. Another feature that was added was the ability to add a class description to the class details page. The platform's difficulty and scoring system were made more dynamic based on the number of student submissions. Lastly, the team documented their work, deployed the platform on Heroku, and used story management to track their progress.

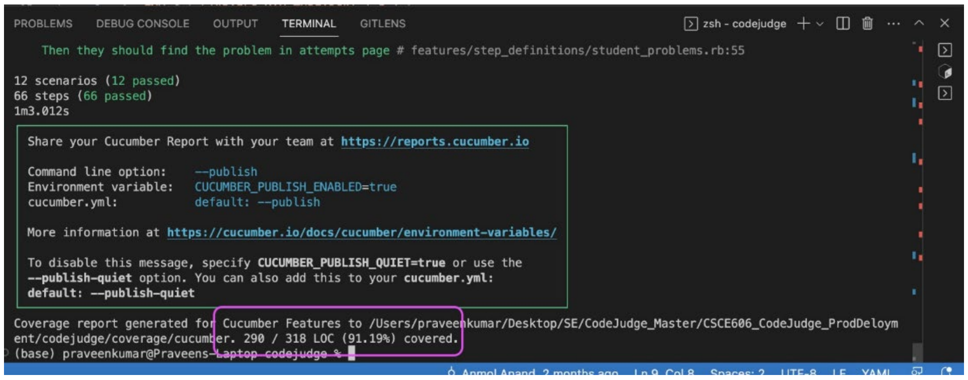
<u>**BDD/TDD testing:**</u>

While working on our project, we used the BDD approach to create user stories (the features we wish we had). We used Cucumber to write acceptance and integration tests.

<u>**Cucumber:**</u>

We have used Selenium to write cucumber tests. We have used Gherkin syntax to define feature files for the scenarios. Once we define feature files, we have implemented respective step definitions for them with the help of automated web interactions.





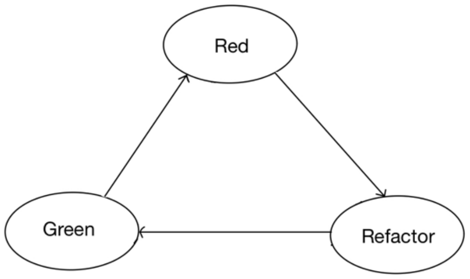
```
   Then they should find the problem in attempts page # features/step_definitions/student_problems.rb:55

12 scenarios (12 passed)
66 steps (66 passed)
1m3.012s
 Share your Cucumber Report with your team at https://reports.cucumber.io

 Command line option:    --publish
 Environment variable:   CUCUMBER_PUBLISH_ENABLED=true
 cucumber.yml:           default: --publish

 More information at https://cucumber.io/docs/cucumber/environment-variables/

 To disable this message, specify CUCUMBER_PUBLISH_QUIET=true or use the
 --publish-quiet option. You can also add this to your cucumber.yml:
 default: --publish-quiet
Coverage report generated for Cucumber Features to /Users/praveenkumar/Desktop/SE/CodeJudge_Master/CSCE606_CodeJudge_ProdDeloym
ent/codejudge/coverage/cucumber. 290 / 318 LOC (91.19%) covered.
(base) praveenkumar@Praveens-Laptop codejudge %
```

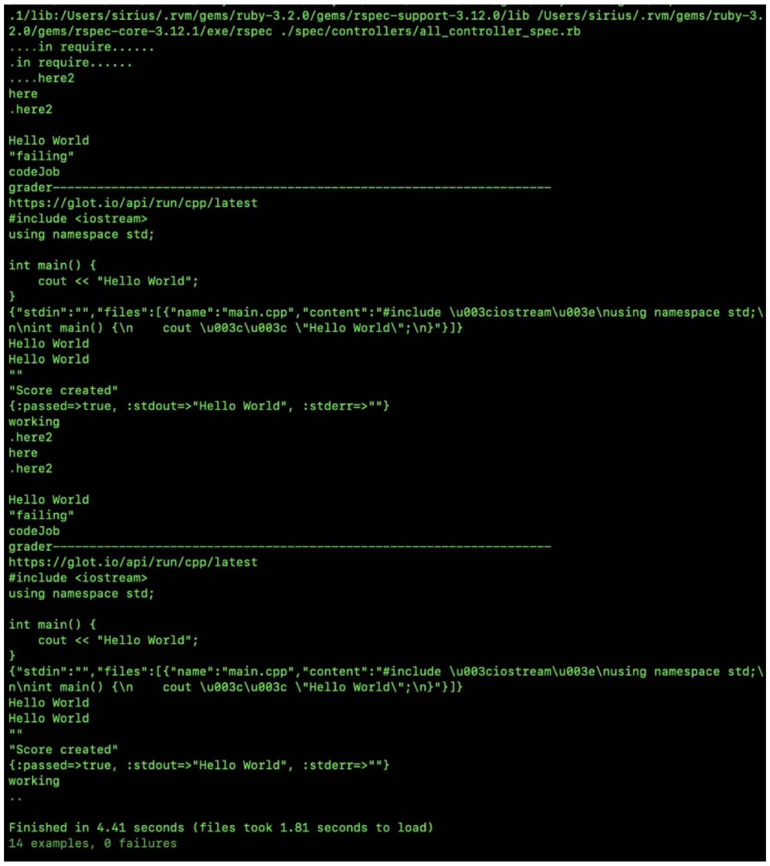
Test-Driven Development (TDD):

Along with BDD, we also used the TDD approach to develop user stories (the code we wish we had). We followed the standard methodology while using the TDD approach, i.e. FIRST (Fast, Independent, Repeatable, Self-Checking, Timely).

Rspec:

We have used Rspec to implement unit tests for the newly developed features.

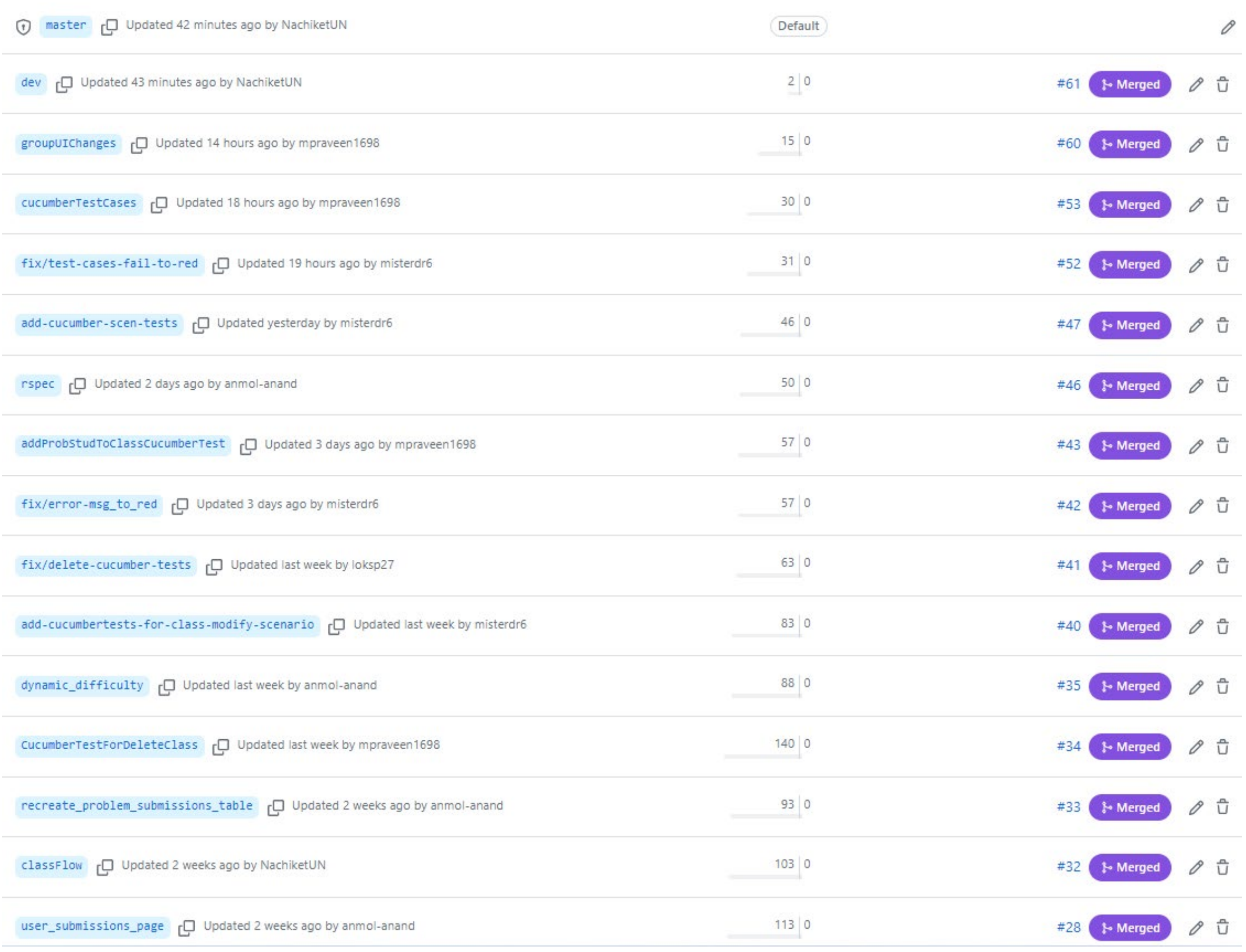




```
.1/lib:/Users/sirius/.rvm/gems/ruby-3.2.0/gems/rspec-support-3.12.0/lib /Users/sirius/.rvm/gems/ruby-3.
2.0/gems/rspec-core-3.12.1/exe/rspec ./spec/controllers/all_controller_spec.rb
....in require......
.in require......
....here2
here
.here2

Hello World
"failing"
codeJob
grader----------------------------------------------------------------------
https://glot.io/api/run/cpp/latest
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World";
}
{"stdin":"","files":[{"name":"main.cpp","content":"#include \u003ciostream\u003e\nusing namespace std;\
n\nint main() {\n    cout \u003c\u003c \"Hello World\";\n}"}]}
Hello World
Hello World
""
"Score created"
{:passed=>true, :stdout=>"Hello World", :stderr=>""}
working
.here2
here
.here2

Hello World
"failing"
codeJob
grader----------------------------------------------------------------------
https://glot.io/api/run/cpp/latest
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World";
}
{"stdin":"","files":[{"name":"main.cpp","content":"#include \u003ciostream\u003e\nusing namespace std;\
n\nint main() {\n    cout \u003c\u003c \"Hello World\";\n}"}]}
Hello World
Hello World
""
"Score created"
{:passed=>true, :stdout=>"Hello World", :stderr=>""}
working
..

Finished in 4.41 seconds (files took 1.81 seconds to load)
14 examples, 0 failures
```

<u>Git Overview - Branches and Releases</u>

We have used Github (Git) for the version control and configuration management system. So, Whenever a user story is assigned to a developer, the respective developer understands all the necessary requirements and will make the changes to a local branch. After that, the developer will raise a pull request and request a review from other team members. Once it is approved the pull request will be merged into the master.



master	Updated 42 minutes ago by NachiketUN	Default			
dev	Updated 43 minutes ago by NachiketUN	2	0	#61	Merged
groupUIChanges	Updated 14 hours ago by mpraveen1698	15	0	#60	Merged
cucumberTestCases	Updated 18 hours ago by mpraveen1698	30	0	#53	Merged
fix/test-cases-fail-to-red	Updated 19 hours ago by misterdr6	31	0	#52	Merged
add-cucumber-scen-tests	Updated yesterday by misterdr6	46	0	#47	Merged
rspec	Updated 2 days ago by anmol-anand	50	0	#46	Merged
addProbStudToClassCucumberTest	Updated 3 days ago by mpraveen1698	57	0	#43	Merged
fix/error-msg_to_red	Updated 3 days ago by misterdr6	57	0	#42	Merged
fix/delete-cucumber-tests	Updated last week by loksp27	63	0	#41	Merged
add-cucumbertests-for-class-modify-scenario	Updated last week by misterdr6	83	0	#40	Merged
dynamic_difficulty	Updated last week by anmol-anand	88	0	#35	Merged
CucumberTestForDeleteClass	Updated last week by mpraveen1698	140	0	#34	Merged
recreate_problem_submissions_table	Updated 2 weeks ago by anmol-anand	93	0	#33	Merged
classFlow	Updated 2 weeks ago by NachiketUN	103	0	#32	Merged
user_submissions_page	Updated 2 weeks ago by anmol-anand	113	0	#28	Merged

<u>Local Development (Without Docker Container):</u>

- This guide is primarily for Mac/Linux Machines to set up local development.
- After cloning this project on your local machine, install Ruby, Rails, and Postgresql with Brew.

```
brew install ruby
gem install rails
brew install postgresql@14
```

- Create a master. key document in ./config/ path and paste the key text (contact one of the members
- Now we need to configure the database and configure our app to connect to it
  1. Start PostgreSQL server

  brew services start Postgresql

  2. By default, PostgreSQL creates a role with your machine username and password.
  3. Install Datagrip from Jetbrains and connect to psql server with the machine username and password.
  4. You should be able to see Postgres as a database in datagrip.
  5. If you are unable to do the above step create a new role and database as follows

```
CREATE ROLE <username> LOGIN PASSWORD 'securePass1';
CREATE DATABASE <dbname>;
grant all privileges on database <dbname> to <username>;
```

6. In ./config/database.yml comment out the database key-value pair in the development section. Add the edit database value in the default section to the database name created in the above step. It should be something like this

```
database: postgres
 # username: <%= ENV['DATABASE_USER'] %>.  #Commented
 # password: <%= ENV['DATABASE_PASSWORD'] %> #Commented
 development:
 <<: *default
 # database: codejudge_dev
```

- Install Redis and configure it in ./config/cable.yml

1. Install Redis and start the service

```
brew install redis
brew services start redis
```

2. Configure the following in cable.yml

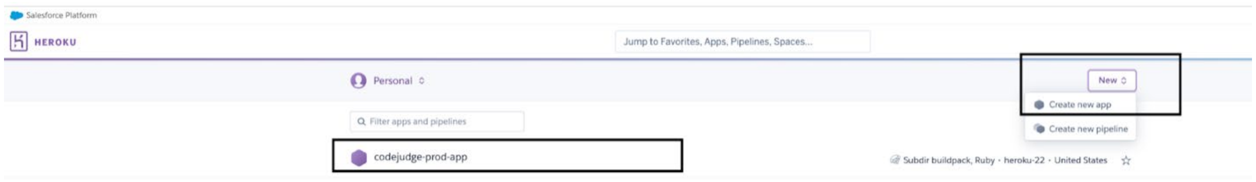
```
 # url: <%= ENV.fetch("REDIS_URL") { "redis://redis:6379/1" } %> # Comment original config line
 url: redis://localhost:6379/1
```

- Make sure the ruby version in the gem file matches your local ruby version.
- Now install the gems and start the server.
  - bundle install
  - rails server
- Using Datagrip you can add users and assign roles to use the credentials for login.
- Note => In some Apple Sillicon machines ruby might point to a native executable.

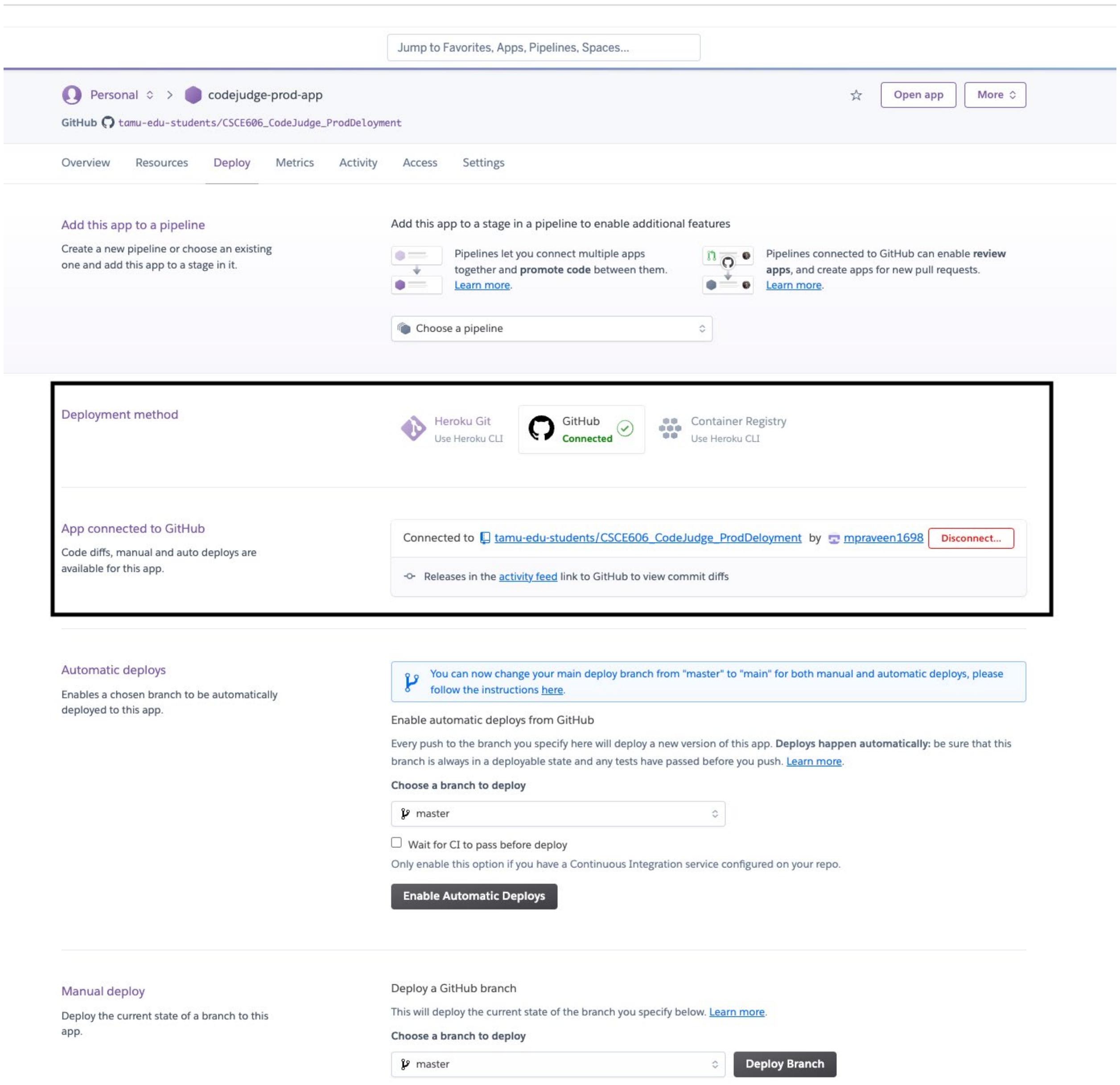
1. Use which ruby to find the path
2. To update the path if required
3. one of the brew command or gem installs won't work, one workaround is to use append sudo to the brew command or use sudo su mod

Heroku release process

- Create a new application on the Heroku platform.


Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Personal
New
Create new app
Create new pipeline
Filter apps and pipelines
codejudge-prod-app
Subdir buildpack, Ruby · heroku-22 · United States

- Connect the Git repository in the deployment method under the Deploy tab.


Jump to Favorites, Apps, Pipelines, Spaces...
Personal
codejudge-prod-app
Open app
More
GitHub tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment
Overview
Resources
Deploy
Metrics
Activity
Access
Settings

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

Deployment method

Heroku Git — Use Heroku CLI | GitHub — Connected | Container Registry — Use Heroku CLI

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment by mpraveen1698 Disconnect...

Releases in the activity feed link to GitHub to view commit diffs

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy

master

[ ] Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

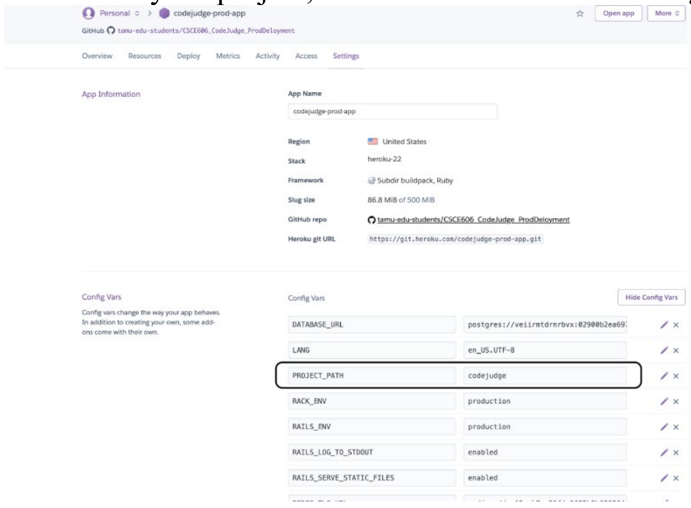
Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

master **Deploy Branch**

- To configure the root folder for deployment, you can add the PROJECT_PATH variable in the Config Vars and build packs under the Settings tab. This will allow you to specify the path to the root folder of your project, which Heroku will use during deployment



Personal > codejudge-prod-app

GitHub tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment

Overview Resources Deploy Metrics Activity Access Settings

App Information

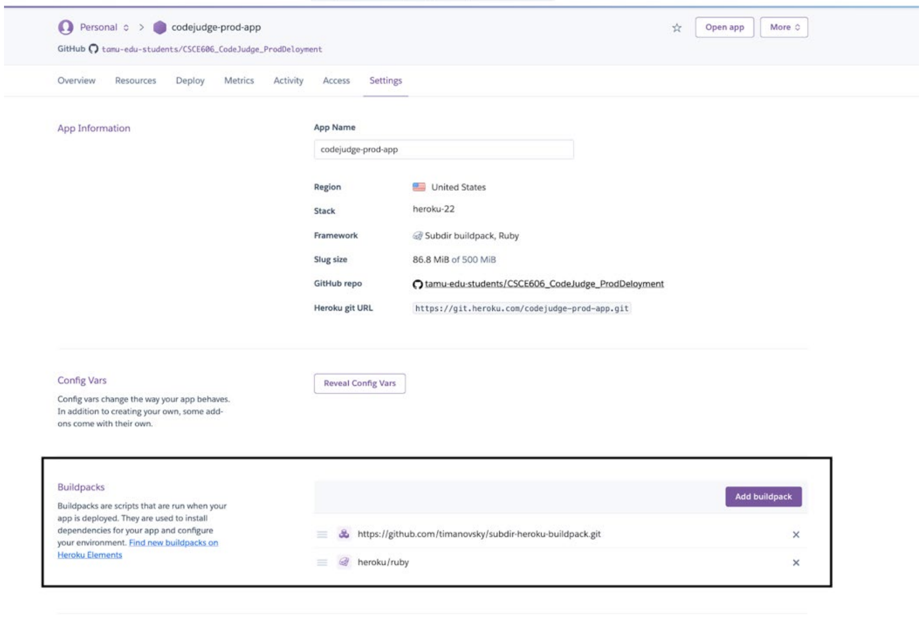
App Name: codejudge-prod-app

Region	United States
Stack	heroku-22
Framework	Subdir buildpack, Ruby
Slug size	86.8 MiB of 500 MiB
GitHub repo	tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment
Heroku git URL	https://git.heroku.com/codejudge-prod-app.git

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

DATABASE_URL	postgres://veiirmtdrnrbvx:02900b2ea69
LANG	en_US.UTF-8
PROJECT_PATH	codejudge
RACK_ENV	production
RAILS_ENV	production
RAILS_LOG_TO_STDOUT	enabled
RAILS_SERVE_STATIC_FILES	enabled



Personal > codejudge-prod-app

GitHub tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment

Overview Resources Deploy Metrics Activity Access Settings

App Information

App Name: codejudge-prod-app

Region	United States
Stack	heroku-22
Framework	Subdir buildpack, Ruby
Slug size	86.8 MiB of 500 MiB
GitHub repo	tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment
Heroku git URL	https://git.heroku.com/codejudge-prod-app.git

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

Reveal Config Vars

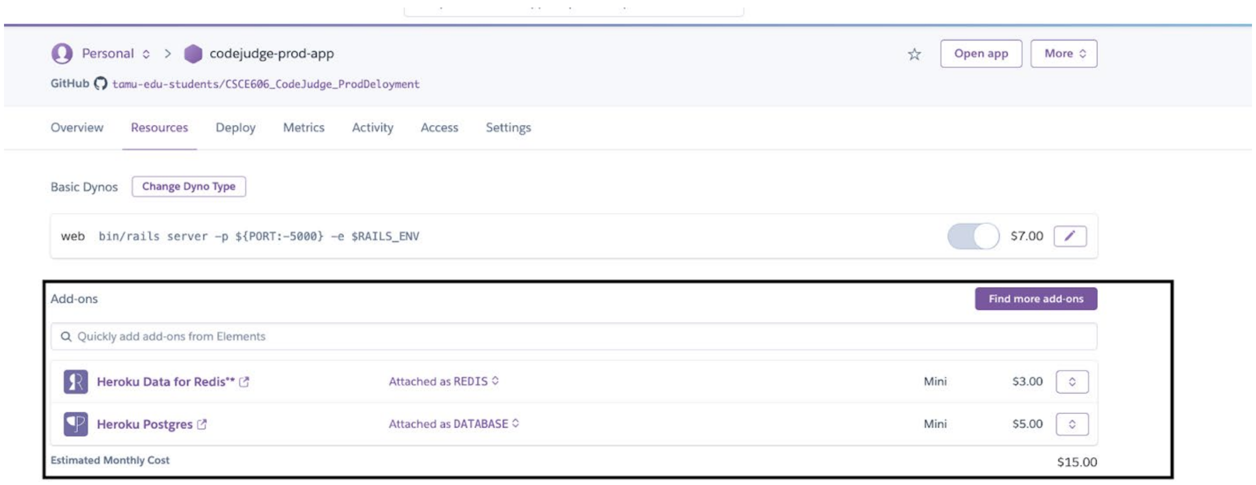
Buildpacks

Buildpacks are scripts that are run when your app is deployed. They are used to install dependencies for your app and configure your environment. Find new buildpacks on Heroku Elements

Add buildpack

- https://github.com/timanovsky/subdir-heroku-buildpack.git
- heroku/ruby

- Add the following PostgreSQL and Redis to resources.


Personal > codejudge-prod-app

GitHub tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment

Basic Dynos

```
web  bin/rails server -p ${PORT:-5000} -e $RAILS_ENV
```
$7.00

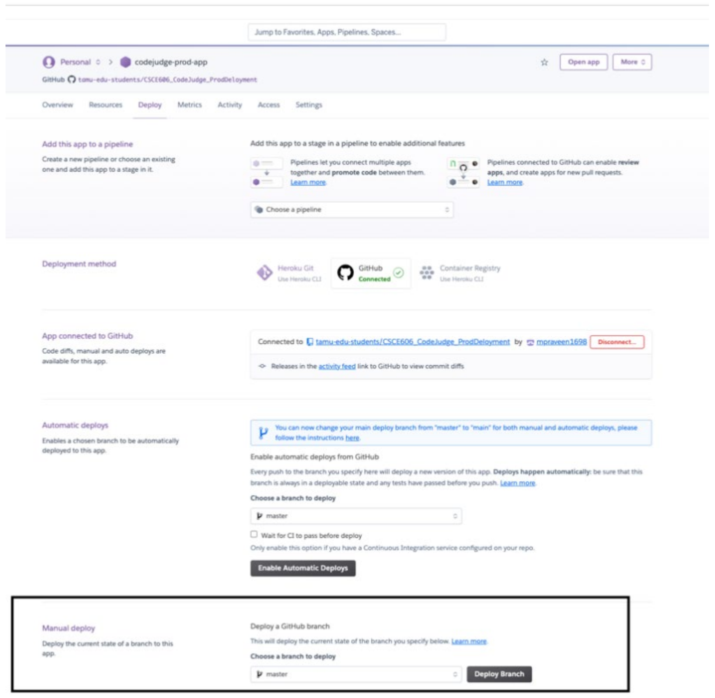
Add-ons

Quickly add add-ons from Elements

Heroku Data for Redis**	Attached as REDIS	Mini	$3.00
Heroku Postgres	Attached as DATABASE	Mini	$5.00

Estimated Monthly Cost $15.00

- The configuration has been completed. Now, choose the branch to deploy the application on Heroku.


Jump to Favorites, Apps, Pipelines, Spaces...

Personal > codejudge-prod-app

GitHub tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

Deployment method

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment by mpraveen1698

Releases in the activity feed link to GitHub to view commit diffs

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy

master

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Manual deploy

Deploy the current state of a branch to this app.

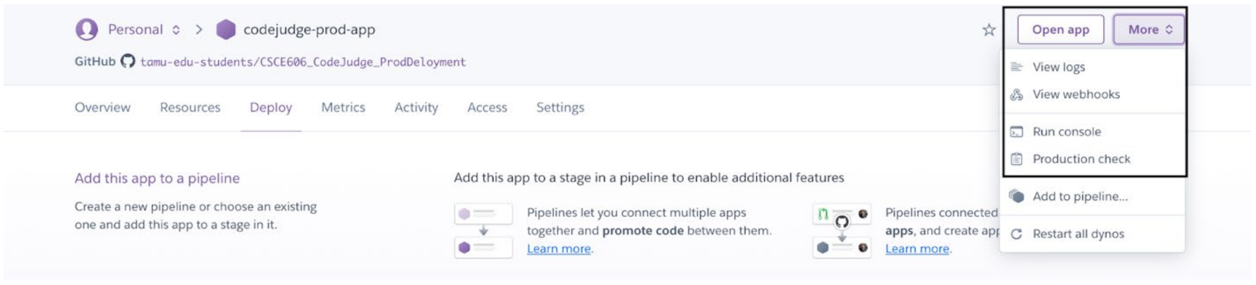
Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

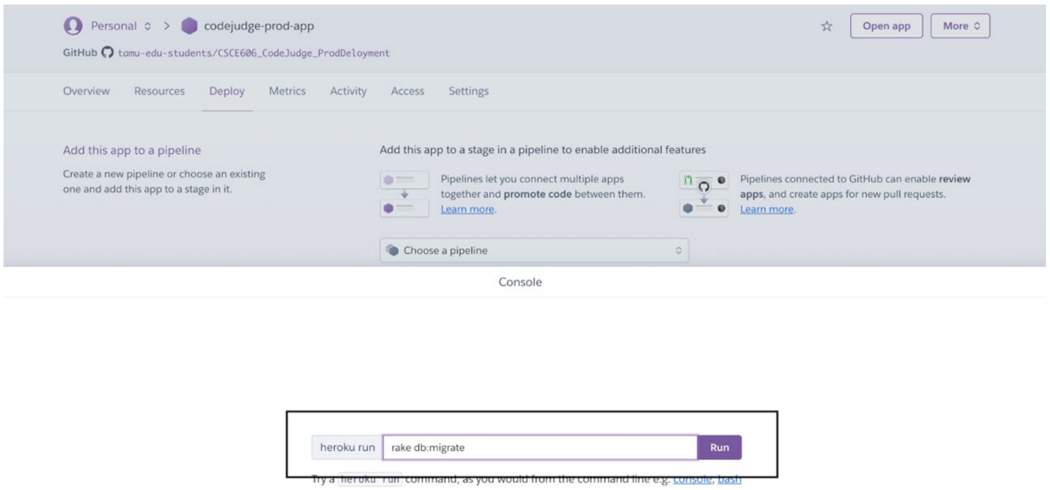
Choose a branch to deploy

master

- Now, navigate to 'More', and we can check logs and use the run console for executing commands for database migrations and seeding.

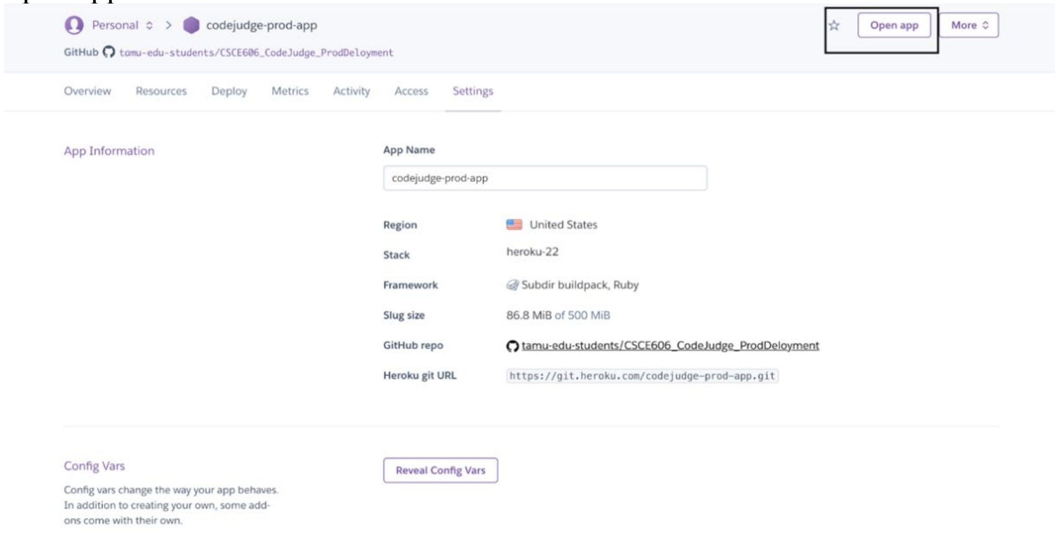

Personal
codejudge-prod-app
GitHub tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment
Overview
Resources
Deploy
Metrics
Activity
Access
Settings
Open app
More
View logs
View webhooks
Run console
Production check
Add to pipeline...
Restart all dynos
Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.
Add this app to a stage in a pipeline to enable additional features
Pipelines let you connect multiple apps together and promote code between them. Learn more.
Pipelines connected
apps, and create app
Learn more.

- The screenshot below shows the execution of the migration command (rake db:migrate). Similarly, we can execute the seed command (rake db:seed). These commands are required if there are any changes in the schema or if you want to seed data into the database.


Personal
codejudge-prod-app
GitHub tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment
Open app
More
Overview
Resources
Deploy
Metrics
Activity
Access
Settings
Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.
Add this app to a stage in a pipeline to enable additional features
Pipelines let you connect multiple apps together and promote code between them. Learn more.
Pipelines connected to GitHub can enable review apps, and create apps for new pull requests. Learn more.
Choose a pipeline
Console
Run
Try a heroku run command, as you would from the command line e.g. console, bash

```
heroku run rake db:migrate
```

- You are ready to use the CodeJudge application. You can navigate to the homepage by clicking on 'Open app'.


Personal
codejudge-prod-app
GitHub tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment
Open app
More
Overview
Resources
Deploy
Metrics
Activity
Access
Settings
App Information
Config Vars
Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.
Reveal Config Vars

App Name	codejudge-prod-app
Region	United States
Stack	heroku-22
Framework	Subdir buildpack, Ruby
Slug size	86.8 MiB of 500 MiB
GitHub repo	tamu-edu-students/CSCE606_CodeJudge_ProdDeloyment
Heroku git URL	https://git.heroku.com/codejudge-prod-app.git

Issues Faced

- During the initial deployment of CodeJudge on Heroku, it requires the root directory of the codejudge. However, when the connected Git repository has a different directory structure, Heroku is unable to find the root path for the code. To mitigate this issue, we initially tried to merge only the code folder to another repository and link that repository to Heroku. This allowed us to complete the deployment successfully.Later on, we discovered that we can specify the ProjectPath in the Config vars so that Heroku considers that path as the root.

PROJECT_PATH : codejudge

Software tools

- RubyMine & Datagrip: It is a multi-database IDE at is very efficient when it comes to schema navigation and query console.
- Redis: It is an open-source memory data structure and is often used as a cache.
- Postman: A collaboration tool that can be used to test endpoints of an API
- Postgres-SQL: It is a relational database management system that supports structured and unstructured data.

Issues

As the above sources were widely used and have high compatibility with development environments. We haven't had any issues with these tools.

Benefits

- To visualize the data in our production environment (heroku), Datagrip was very handy, and it made debugging very easy for issues related to databases (constraints violation, duplicate data, etc).
- Postman is also a great tool for debugging API's. It supports all types of HTTP requests with customized support to various authentication mechanisms. We have created a new token in the Glot API and tested some of the GET and POST requests to evaluate a code with the new token.

Git repo contents

The git repo consists of codejudge (source code for the project) and documentation (folder containing the documents of legacy teams). Apart from these folders, we have README.md where we have included the steps for local setup development.

Project links

- Pivotal tracker
- Heroku deployment
- Github Repository
- Presentation and Demo Video