MIRC Gradebook Final Report

CSCE 606 Software Engineering

Team Name: Agile Sprinters.
Team Members: Satish, Pranav, Ashutosh, Vivek, Sagar and Harsha

1. Summary:

MIRC Gradebook is a web-based application for teachers and students, the primary goal of this project is to provide real-time performance information to students throughout the semester, not just at the end. So, that it can help to increase student success and transparency in the grading process. Most of the other gradebooks are like online databases, they just show the grades on different performance measures (quizzes, assignments midterms and endterms), but they don't tell the final grade and Gradebook tries to do this. Gradebook asks professors to add grading criterias and weightage of each assignment for a particular course before course begins, so that Gradebook can evaluate students' performance implicitly and show them the real-time grade. Moreover, Gradebook provides a statistics dashboard for each course for both professor and student so that they can get a sense of how the whole class is performing in real-time.

Gradebook features three major stakeholders, an Admin who can add teachers and students so that they get access to the gradebook and also can add course details, a Professor who can enter or edit students grades for each course and a Student who can view his/her grades. As a future step, Gradebook aims to provide feedback for students based on his/her current standing in the class and grading criteria set by the professor, So that they can improve their performance incrementally.

2. User Stories:

2.1 Iteration 0:

Total Story Points - 18 (6 stories * 3 points each)
Implementation Status: Finished
NOTE: These stories are not added in pivotal tracker because, we had an issue with pivotal tracker free trail in this iteration.
User Stories:

1. Feature: Login for Professor. The professor logs in with admin access. We need a login UI.
2. Feature: Login for Student. Students log in with view access.
3. Feature: Add student marks Professor needs to add student marks.
4. Feature: View for professor. Professor wants to look at all student marks for an assessment. So, we need a UI to list them.

5. Feature: View for student. Students should be able to view their grades and marks. So, we need a UI to show all his/her grades.
6. Feature: Professor dashboard to view class performance as a whole.

Summary:
In this Iteration, We first met with our client and he introduced us about the motivation and goal of the MIRC Gradebook project. And as an initial iteration we finished setting up Github, Pivotal tracker and we have also prepared a roadmap for what we have to do for further iterations. As part of initial roadmap, we realized there would be two types of users i.e. professors and students, and they would be granted access through google authentication. Additionally, professors would have functionality to add each student's marks and students would have the ability to see their marks and grades.

2.2 Iteration 1:
Total Story Points - 18 (6 stories * 3 points each)
Implementation Status: Finished
User Stories:

1. Feature: Database setup for Gradebook
    a. Description: sqlite3 setup in Local Env and Postgres setup in heroku Environment.
2. Feature: Google Authentication for user's login
    a. Description: Enable Google Authentication for users and based on email id, show separate login pages based on user type.
3. Feature: After Login Student Landing UI Page.
    a. Description: As a Student, I need to see my landing page after logging in.
4. Feature: After login Teaching staff login page
    a. Description: As a Professor, I need to see my landing page after logging in. So, I can view my dashboard.
5. Feature: Heroku Deployment of our Application
    a. Description: Researching about configuration needed for heroku deployment.
6. Feature: Front-end setup and UI for Login
    a. Description: As a User, I should be able to see a login page, where I can login So, that I can see my dashboard.

Summary:
In this Iteration, We have implemented Google Authentication & Authorization, Database setup and Heroku deployment. Additionally, we designed UIs for login, professor and student landing pages, we further discussed with client and understood about some key project requirements.

2.3 Iteration 2:

Total Story Points - 18 (6 stories * 3 points each)
Implementation Status: Finished
User Stories:

1. Feature: Backend logic to add professors and students by admin
    a. Description: As an Admin, I want to add professor and students data to the database. So that professors and students could get access to gradebook.
2. Feature: UI for admin to add professor and student
    a. Description: As an Admin, I need a UI where I want to add professor and students details. So that both of them could get access to gradebook.
3. Feature: Feature: Backend logic to add courses by admin
    a. Description: As an Admin, I want to add course details. So that I can view the courses and can assign courses to students and professors.
4. Feature: UI for admin to able to add courses
    a. Description: As an Admin, I need a user interface to add course information. So that I can assign courses to students and professors.
5. Feature: Course to professor mapping by admin
    a. Description: As an Admin, I want to assign professors to a particular course. So that professor can see his own courses when he logs in.
6. Feature: Add students to particular course
    a. Description: As a Professor, I want to add students to my course. So that I can add their grades.

Summary:
In this iteration our main objective was to implement the functionalities of admin, who takes care of addition, deletion and updation of profiles (students and professors) who can access the application. Moreover admin is also responsible for course lifecycle, i.e. entering details for each course and assigning students and professors to a course.

2.4 Iteration 3:

Total Story Points: 18 (6 stories * 3 points each)
Implementation Status: Finished
User Stories:

1. Feature: Professor should be able to see all the courses allocated (Dashboard on the landing page) (UI and backend logic).
    a. Description: As a professor, I want to see courses allocated to me in my dashboard with all the course details.
2. Feature: UI Revamp: Creating common modular UI and theme for all features.

    a. Description: Make all views, templates, static files and other project files modular and consistent.
3. Feature: Admin should be able to add professors and students to a course
    a. Description: As an Admin, I want to add professors and students to a course. So that the Professor and students can plan a class.
4. Feature: Professor should be able to see all the exam details in a course
    a. Description: As a professor, I want to see all the exam details in my course. So that I have a record of all exams.
5. Feature: Create data model/ database structure and implement corresponding models in Django (in both local & Heroku environments)
    a. Description: Create all database models i.e Marks table, Evaluation table, and ProfileCourse table in both environments.
6. Feature: Professor should be able to see all the student details in a course. A list view without edit/delete features.
    a. Description: As a professor, I want to see all students details in my course. So that I have a record of them.

Summary:
At the start of this Iteration, we realized that there is a need to expand our tables with some more information. So, we first started with finalizing our database ER diagram so that it could fit all our use cases, then we worked on professor functionalities such as viewing all the courses allocated to him/her and being able to view all students registered for a particular course. Additionally, we have completed all the remaining features of admin. Finally, Story 5 has undergone changes in between iteration, because our use cases keep changing and we have to accommodate changes so that our tables have enough and non redundant information to fit all our use cases.

2.5 Iteration 4:
Total Story Points: 17 (5 stories * 3 points each + 2 point story)
User Stories:

1. Feature: Professor should be able to add all performance measures along with weights.
    a. Description: As a Professor, I should be able to enter all the performance measures (Quizzes, assignments, etc) and assign weight to each. This will help me have the flexibility to give different weights to different measures.
2. Feature: Professor should be able to define a grade function by entering threshold values for different marks.

    a. Description: As a professor, I want to have the flexibility to define threshold values for different grades (A, B, C, D, F). So that I could customize the different course grading parameters.
3. Feature: Function for calculating current predicted score and current predicted grade
    a. Description: Accessing all stored marks of students in the database and performance criteria by the professor and using them to calculate predicted scores and grade of students.
4. Feature: Creating alert popups for all use cases (2 point story)
    a. Description: Creating a common alert toast interface to display all warnings and messages.
5. Feature: Professor should be able to see students in class and their grading as per the pre-defined model
    a. Description: As a Professor, I should see the student's scores across created evaluations (if available) in a tabular format in the student's tab.
6. Feature: Professor dashboard to see all the evaluations in the course
    a. Description: As a Professor, To create a dashboard for the professor to view the performance measure and evaluation metrics for the respective courses

Summary:
In this iteration, we have implemented the key functionality of the application that is providing an ability for professors to add all the customized performance measures and grading criteria for each course. Additionally, we also implemented score and grade calculation functionalities for students based on criteria setup by the professor for each course.

2.6 Iteration 5:
Total Story Points - 17 (5 stories * 3 points each)
User Stories:

1. Feature: Exception Handling and Display Error Messages for All form validations.
    a. Description: As a professor, student and admin, should see error messages whenever anything goes wrong or invalid data entered in the form.
2. Feature: Student complete dashboard view and detail course view.
    a. Description: As a student, I want to see all my graded assignments and course statistics such as mean and max score and I also want to see my Grade.
3. Feature: Unit Test cases and improve the code coverage
    a. Description: Improving Unit test cases and improve code coverage
4. Feature: Add CSS (style sheets) for aesthetic improvement (2 point story)
    a. Description: As a professor, student and admin, I want to see fluent beautiful UI with all functional features.

5. Feature: Professor should be able to add, update or delete student marks for a particular course. Calculating and storing projected score and grade.
    a. Description: As a Professor, I should be able to add grades for one assignment at a time and I should be able to edit the particular student grades.
6. Feature: Course Dashboard to display all the performance metrics for professor
    a. Description: As a professor, I want to see my course metrics in a dashboard with grade distribution of the course.

Summary:
As part of last Iteration, the primary objective was to make our application stable and robust, there by we used ajax to for all form validations and reporting errors in the form without actually reloading the page. The second most important feature is the student dashboard, basically a view only dashboard of all his/her allocated courses and course details. Moreover, we also worked on displaying course statistics for professors such as top5 and bottom5 students and grade distribution among students. Finally, we also aimed to strengthen code coverage by implementing Unit test cases perfectly and Improve our UI aesthetically by adding CSS.

Total points: 115 (i0-18 , i1-18, i2-18, i3-18, i4-17, i5-17, report-3, demo-3, presentation-3)

3. Team roles:

As discussed in lecture and iteration report submission details, we have exchanged our roles as scrum master and product owner for every iteration on a rotational basis.

Iteration	**Scrum master**	**Product owner**
0	Vivek Vamsi	Ashutosh Chauhan
1	Pranav Taukari	Satish kumar Reddy
2	Harshavardhan	Sagar Adhikari
3	Sagar Adhikari	Harshavardhan
4	Ashutosh chauhan	Vivek vamsi
5	Satish Kumar Reddy	Pranav Taukari

4. Customer meeting dates and discussions:

Date	Discussion
09/21/2022	Meeting the client and getting to know each other and introduction to the project.
09/29/2022	Discussion with the client and understanding project requirements.
10/14/2022	Given demo of features executed in iteration 1 and discussed plan for iteration 2..
10/28/2022	Demo and Feedback of Iteration 2 and plan for iteration 3.
11/11/2022	Demo and Feedback of Iteration 3.
11/18/2022	Demo/Feedback of Iteration 4 and plan for iteration 5.
12/06/2022	Final Iteration demo

5. BDD/TDD process benefits:
Test Driven Development has helped us create robust functions, where we often miss edge cases and realize later, TDD helped us to avoid such scenarios and helped us create resilient functions. BDD helped us to test our application as a whole and helped us identify problems with the real world environment before even deploying.

6. Configuration Management Approach:
In Iteration 3, we did a spike story and we did not flag a story as spike, but underlying effort was about estimating effort needed for change in database model. We have created 24 branches and 5 releases for this project till today.

7. Deployment Process:

7.1 Release:
At the end of each iteration, we have to release a new version of our application. In order to have a seamless deployment process, we have integrated our Github repo with Heroku. And, we have also enabled Automatic Deployment from our repo’s main branch, so that whenever we make a new release it will be updated and deployed automatically in Heroku without any

intervention. Moreover, Our repo's main branch is protected, that means, in order to push to our main branch, we should raise a pull request with atleast 2 approvals.

7.2 Development:

Release occurs at the end of each iteration, but development takes place throughout the iteration. So, At the start of each iteration, after creating our story board, we create our own feature branches and combine all changes into develop branch. From there on we raise a pull request to the main branch.

7.3 Each Iteration Development and Deployment process:

Individual Feature branches → develop branch → main branch → Heroku.

8. Deployment Scripts:

To Deploy into Heroku, we need a Procfile, which specifies the commands for heroku to run inorder to deploy.

Commands in Procfile:

```
web: gunicorn gradebook_project.wsgi --log-file -
release: bash release.sh
```

Commands in release.sh:

```
python manage.py makemigrations
python manage.py migrate
```

The first command in Procfile specifies Heroku to start gunicorn, which is a python HTTP server for WSGI applications, the second command specifies to migrate all new database changes made before starting the server.

9. Issues faced with Heroku and Github:

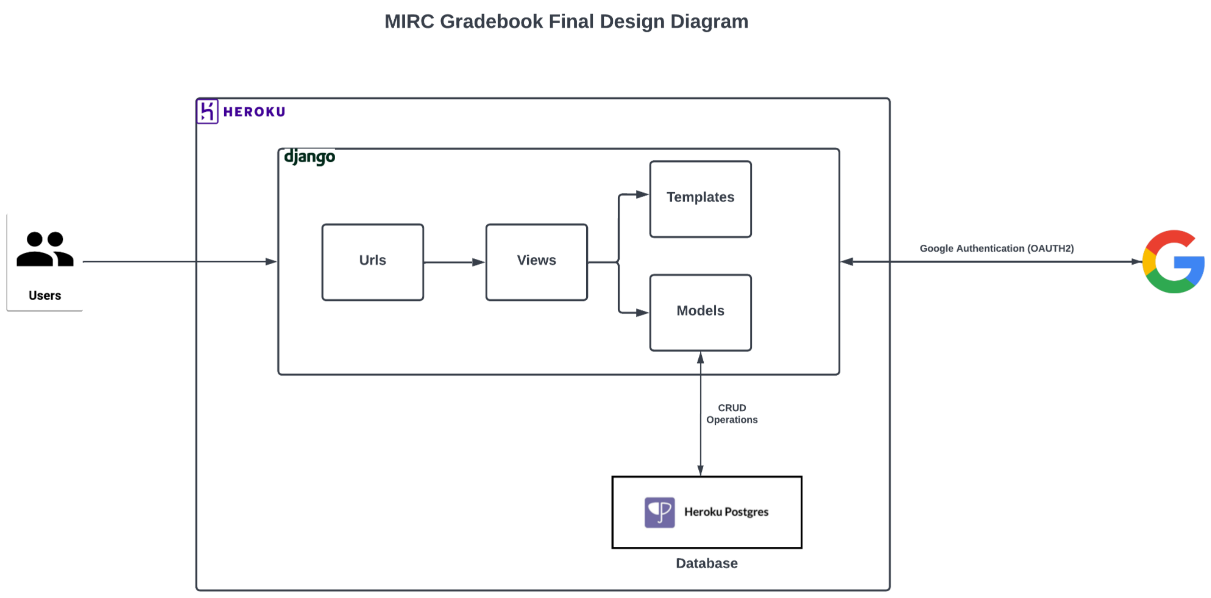
We faced two major issues/challenges with heroku and django:

1. The first challenge was when we tried to deploy to heroku for the first time, the database configured did not work and after researching in the internet and debugging our code, we realized that heroku does not support sqlite3 and we have to configure separate database only for the heroku environment, which is different from local database, then we solved that using heroku postgres database.
2. The second challenge was also related to database, where for the first three iterations we created preliminary database tables, then we realized we needed some more new tables and needed to make changes to existing tables. Due to the nature of changes we made, we had to destroy and re-configure the entire database both in local and heroku environments.

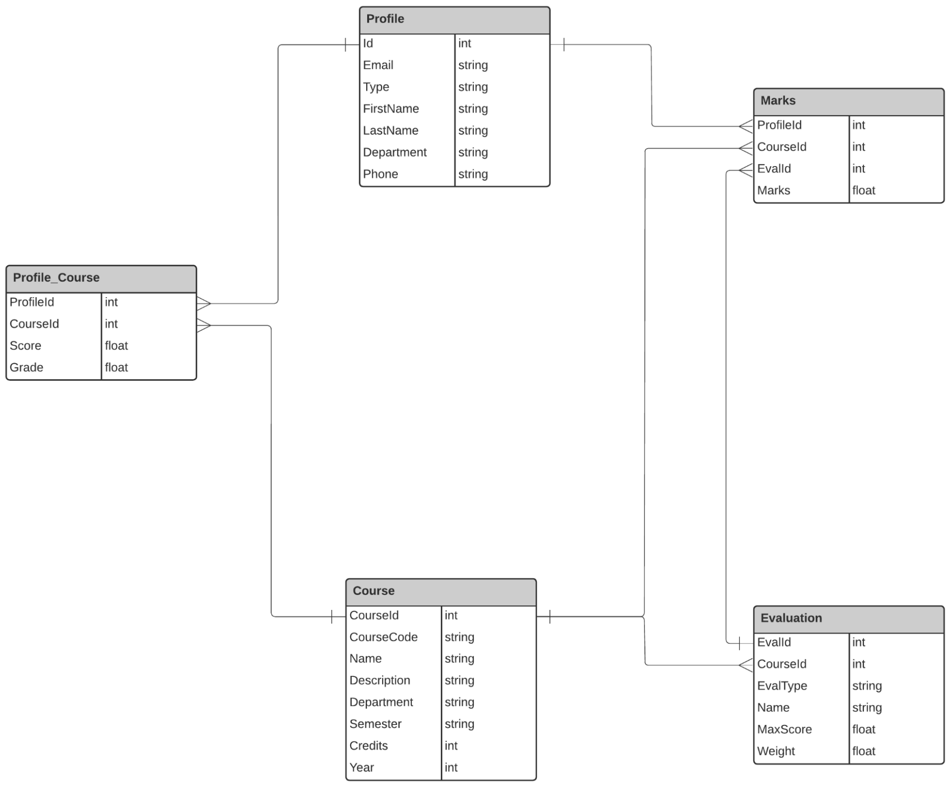
10. Links:

1. Gradebook Pivotal Tracker
2. Gradebook GitHub Repository
3. Gradebook Heroku Application
4. Software Engineering Presentation
5. Graadebook Demo
6. Gradebook Demo and Presentation

11. Design Diagram and ER diagram:



FInal ER Diagram



Profile
Id int
Email string
Type string
FirstName string
LastName string
Department string
Phone string
Marks
ProfileId int
CourseId int
EvalId int
Marks float
Profile_Course
ProfileId int
CourseId int
Score float
Grade float
Course
CourseId int
CourseCode string
Name string
Description string
Department string
Semester string
Credits int
Year int
Evaluation
EvalId int
CourseId int
EvalType string
Name string
MaxScore float
Weight float