Team Contributions: POC
Software Engineering

Team 8 – Rhythm Rangers

Ansel Chen
Muhammad Jawad
Mohamad-Hassan Bahsoun
Matthew Baleanu
Ahmed Al-Hayali

This document summarizes the contributions of each team member up to the POC Demo. The time period of interest is the time between the beginning of the term and the POC demo.

1 Demo Plans

[What will you be demonstrating —SS]

2 Team Meeting Attendance

[For each team member how many team meetings have they attended over the time period of interest. This number should be determined from the meeting issues in the team's repo. The first entry in the table should be the total number of team meetings held by the team. —SS]

Student	**Meetings**
Total	Num
Name 1	Num
Name 2	Num
Name 3	Num
Name 4	Num
Name 5	Num

[If needed, an explanation for the counts can be provided here. —SS]

3 Supervisor/Stakeholder Meeting Attendance

[For each team member how many supervisor/stakeholder team meetings have they attended over the time period of interest. This number should be determined from the supervisor meeting issues in the team's repo. The first entry in the table should be the total number of supervisor and team meetings held by the team. If there is no supervisor, there will usually be meetings with stakeholders (potential users) that can serve a similar purpose. —SS]

Student	**Meetings**
Total	Num
Name 1	Num
Name 2	Num
Name 3	Num
Name 4	Num
Name 5	Num

[If needed, an explanation for the counts can be provided here. —SS]

4 Lecture Attendance

[For each team member how many lectures have they attended over the time period of interest. This number should be determined from the lecture issues in the team's repo. The first entry in the table should be the total number of lectures since the beginning of the term. —SS]

Student	**Lectures**
Total	Num
Name 1	Num
Name 2	Num
Name 3	Num
Name 4	Num
Name 5	Num

[If needed, an explanation for the lecture attendance can be provided here. —SS]

5 TA Document Discussion Attendance

[For each team member how many of the informal document discussion meetings with the TA were attended over the time period of interest. —SS]

Student	Lectures
Total	3
Ansel Chen	Num
Muhammad Jawad	3
Mohamed-Hassan Bahsoun	3
Matthew Baleanu	3
Ahmed Al-Hayali	Num

[If needed, an explanation for the attendance can be provided here. —SS]

6 Commits

[For each team member how many commits to the main branch have been made over the time period of interest. The total is the total number of commits for the entire team since the beginning of the term. The percentage is the percentage of the total commits made by each team member. —SS]

Student	Commits	Percent
Total	318	100%
Ansel Chen	Num	%
Muhammad Jawad	35	11%
Mohamed-Hassan Bahsoun	38	12%
Matthew Baleanu	60	18.9%
Ahmed Al-Hayali	Num	%

[If needed, an explanation for the counts can be provided here. For instance, if a team member has more commits to unmerged branches, these numbers can be provided here. If multiple people contribute to a commit, git allows for multi-author commits. —SS]

7 Issue Tracker

[For each team member how many issues have they authored (including open and closed issues (O+C)) and how many have they been assigned (only counting closed issues (C only)) over the time period of interest. —SS]

Student	**Authored (O+C)**	**Assigned (C only)**
Ansel Chen	Num	Num
Muhammad Jawad	10	19
Mohamed-Hassan Bahsoun	22	28
Matthew Baleanu	34	25
Ahmed Al-Hayali	Num	Num

[If needed, an explanation for the counts can be provided here. —SS]

8 CICD

For this project, CICD will be implemented through GitHub Actions. Each time a pull request is created (or when commits are made to an open pull request), a GitHub Actions workflow is triggered. This workflow will run on a virtual machine and use `Tox` to orchestrate the building, linting, formatting, type checking and unit testing of the project. The GitHub Actions page will show the progress of each action and display a green check mark for each action that completes without errors. Once all actions pass, the pull request will be allowed to be merged. This CICD pipeline ensures that code is written to a high standard by only allowing code to be merged if it meets the rigorous style and functionality requirements enforced by the static analyzers and unit tests.