Team Contributions: Rev 0
CXR

Team 27, Neuralyzers
Ayman Akhras
Nathan Luong
Patrick Zhou
Kelly Deng
Reza Jodeiri

This document summarizes the contributions of each team member for the Rev 0 Demo. The time period of interest is the time between the POC demo and the Rev 0 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	10
Ayman Akhras	6
Nathan Luong	8
Patrick Zhou	8
Kelly Deng	7
Reza Jodeiri	8

All students had their own difficult problems to solve. People with higher meeting counts tended to meet with each other more often.

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	2
Ayman Akhras	2
Nathan Luong	2
Patrick Zhou	2
Kelly Deng	2
Reza Jodeiri	2

[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	15
Ayman Akhras	1
Nathan Luong	1
Patrick Zhou	2
Kelly Deng	2
Reza Jodeiri	9

[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	24
Ayman Akhras	8
Nathan Luong	1
Patrick Zhou	2
Kelly Deng	2
Reza Jodeiri	6

For the purpose of this report, any form of communication with the TA, including both formal meetings and informal messaging, is considered a "meeting." This includes questions asked and answered via email, messaging platforms, or any other medium of communication. The following table summarizes the number of such communications for each team member over the time period of interest.

6 Commits

Student	Commits	Percent
Total	327	100%
Ayman Akhras	59	18.04%
Nathan Luong	53	16.21%
Patrick Zhou	24	7.34%
Kelly Deng	45	13.76%
Reza Jodeiri	146	44.65%

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)
Ayman Akhras	0	3
Nathan Luong	3	3
Patrick Zhou	1	3
Kelly Deng	0	3
Reza Jodeiri	25	10

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

8 CICD

Our CI/CD pipeline ensures our project's code quality and automates essential processes like testing and deployment. It begins with setting up the environment, where the latest code is checked out from our repository, Python is configured, and dependencies are installed while utilizing cached pip packages for efficiency. Next, the pipeline performs linting and type checks using tools such as flake8 and mypy to maintain code consistency and catch potential issues early. Finally, it runs unit and integration tests using a framework like pytest or unittest to verify that changes do not break existing functionality.

To maintain code coverage, the pipeline integrates tools like coverage.py to measure the extent of code exercised by tests. The coverage reports are generated and analyzed to ensure that a high percentage of the codebase is tested. If the code coverage falls below a predefined threshold, the pipeline can be configured to fail, prompting developers to add necessary tests. This automated workflow is triggered whenever code is pushed or merged, streamlining our development process, reducing human errors, and ensuring a stable and reliable codebase.