Software Engineering Final Presentation

Team XRAYS (G7T9)

Table of contents

The agenda for today

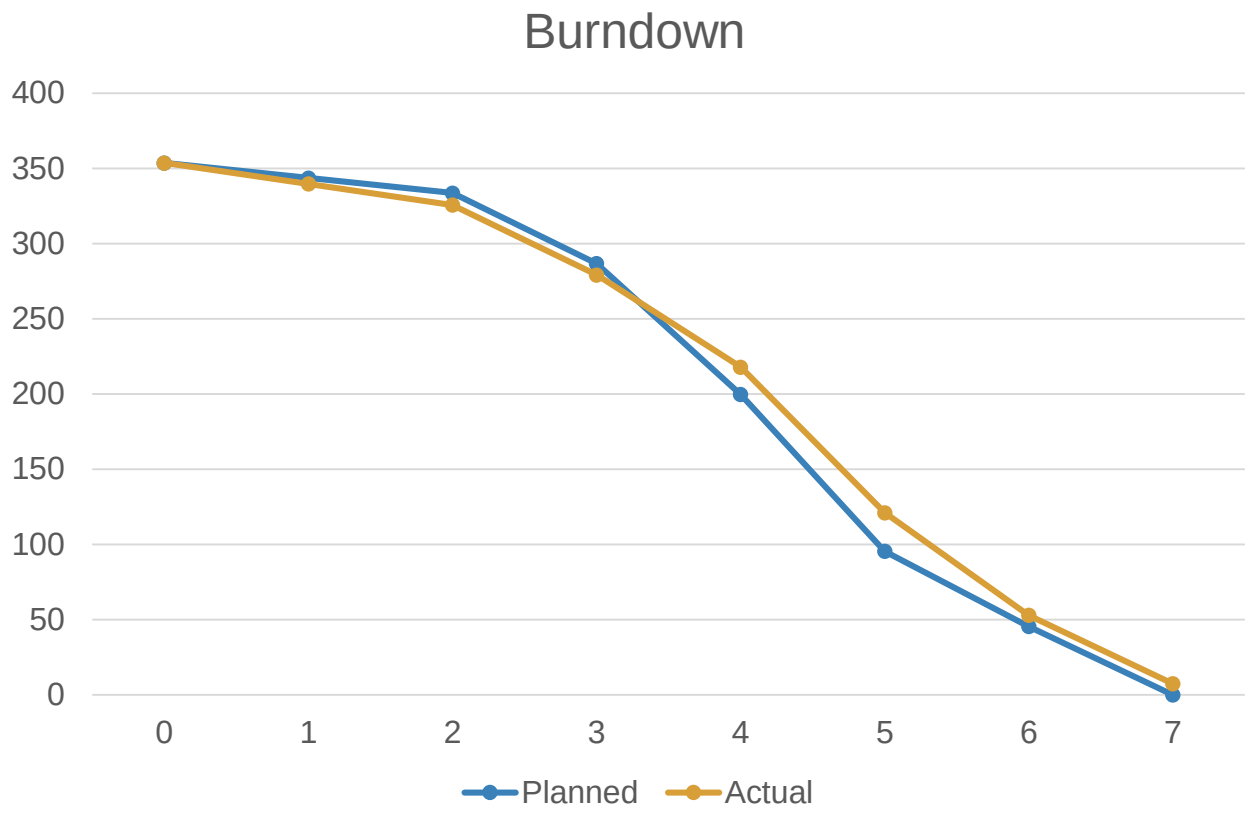
1.

SCHEDULE

To code or not to code

Actual vs Planned

#	Planned	Actual
0	354	354
1	344	340
2	334	326
3	287	279
4	200	218
5	96	121
6	46	53
7	0	7



Planned

- Basic Location Report
- Heatmap
- In Class Demo
- Auto Group Detection
- Update Sequence Diagram

5

6

- Auto Group Detection
- Deployment
- UAT
- 5 x Debugging

- 4 x Debugging
- 2 x Integration Testing
- 2 x Regression Testing
- Final Presentation Preparation

7

Actual

- Basic Location Report
- Heatmap
- In Class Demo
- 4 x Auto Group Detection
- 1 x Update Sequence Diagram

5

6

- Auto Group Detection
- Deployment
- UAT
- 5 x Debugging

\+ 2 x Debugging

7

- 4 x Debugging
- 2 x Integration Testing
- 2 x Regression Testing
- Final Presentation Preparation

Functionalities

Implemented Functionalities	× Login & Logout × Bootstrap & Additional Data Upload × Heatmap × Basic Location Report - Top K Popular Places × Basic Location Report – Breakdown × Basic Location Report – Top K Companions × Basic Location Report – Top K Next Place × Auto Group Detection
Add On Functionalities	× Dashboard (Allow admin to truncate each single database table)

Frameworks/ External Libraries

Frontend Framework:

Bootstrap
SB Admin 2

Frontend Library

jQuery
metisMenu

Frameworks Used:

Model View Controller
(JSP Model 2)

Backend Libraries:

MySQL JDBC Driver
gson-2.8.2.jar
is203-jwt-v2.jar
json-smart-1.2.jar
nimbus-jose-jwt-2.26.1.jar
opencsv-4.1.jar

Challenges Faced

Project Manager/Leader

Being new to both the Software Engineering process and taking on the role of a project manager, there were many uncertainties and differing viewpoints. Many times the leader is just as lost as the other members but as the project manager, the leader has to come up with a game plan albeit having little to no experience and banking on a whole lot of Google and EvaLive. Given that there is no sample/model answer, we often sought other groups as point of reference and also comfort in knowing that we are on the right track.

More often than not, the programmers are waddling neck deep in codes and tend to forget to update the schedule. Therefore the Project Manager has to constantly remind the programming teams to update the schedule with the commit ID and the pair programming photos (which we didn't realise wasn't a requirement until Week 13). In order to ensure that the Project Manager is kept abreast of the pair programming teams' schedule, the aid of Google Calendar was enlisted which proved to be a much needed and necessary companion in this quest to ensure that the project schedule is up to date.

2.

IMPROVEMENT

Improvement begins with I

Comments We Received

Plain UI

Unbalanced Hours

What We Did

Plain UI

Added in preloading screen when awaiting results

Unbalanced Hours

PM did better job allocations this time

	Xinyi	Rainean	Amos	Yigang	Samantha
Prog Hrs	71	67	74	71	67
Non-prog Hrs	105	108	105	114	113

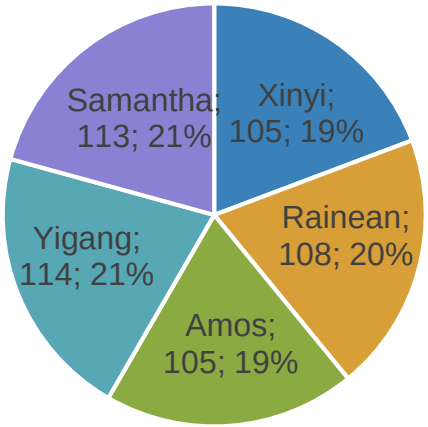
- Added a [Resolved On] column in bug logs for better accountability
- Project Managers took a more active role by checking in with Pair Programmers
- Spending more time and effort during project kickstart to iron out any possible planning issues

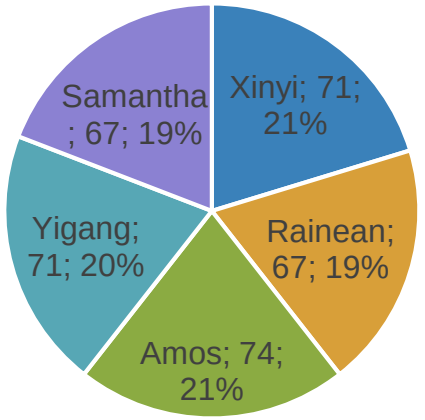
3.

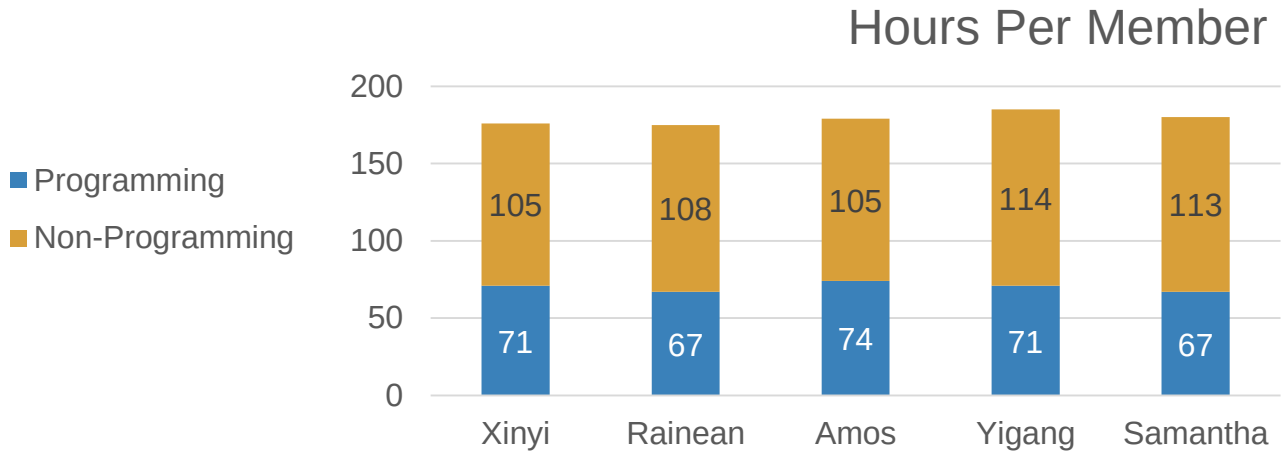
BREAKDOWN OF WORK

Lets get down to business

Overall Work Breakdown







Member: Xinyi

Programming (71 Hours)	Non-Programming (105 Hours)
× Bootstrap (Import & Validation) × Basic Location Reports (Breakdown, Top K Companions, Top K Next Places, Top K Popular Place) × System Integration × Debugging	× Review system design × Review technical documents × Preparation for Milestones (PM Review, Online Demo) × Testing (Unit, System Integration, Regression) × AWS Deployment

Member: Rainean

Programming (67 Hours)	Non-Programming (108 Hours)
× Login Page × Bootstrap (Import) × Basic Location Reports (Breakdown, Top K Companions, Top K Next Places, Top K Popular Place) × Integration (Top K Companions, Top K Next Places, AGD) × Debugging	× Review system design × Review technical documents × Preparation for Milestones (PM Review, Online Demo) × Testing (Unit, System Integration) × AWS Deployment

Member: Amos

Programming (74 Hours)	Non-Programming (105 Hours)
× Bootstrap (Import, Validate, Logic) × Login Page × Basic Location Reports (Breakdown, Top K Popular Place) × AGD × Heatmap × Debugging	× Review system design × Review technical documents × Preparation for Milestones (PM Review, Online Demo) × System Integration & Regression Testing × AWS Deployment × Testing (Unit, System Integration)

Member: Yigang

Programming (71 Hours)	Non-Programming (114 Hours)
× Login Page × Bootstrap (Import, Validate) × Basic Location Reports (Top K Next Places, Breakdown, Top K Popular Place) × Debugging	× Review system design × Review technical documents × Preparation for Milestones (PM Review, Online Demo) × AWS Deployment × Create Test Cases (Bootstrap, Top K Companions, Top K Next Places) × Testing (Unit, System Integration, Regression)

Member: Samantha

Programming (67 Hours)	Non-Programming (113 Hours)
× Bootstrap (Import, Validate, Logic) × Login Page × AGD × Heatmap × Debugging	× Review system design × Review technical documents × Create Test Case (Bootstrap) × Preparation for Milestones (PM Review, Online Demo) × AWS Deployment × Testing (Unit, System Integration, Regression)

4.

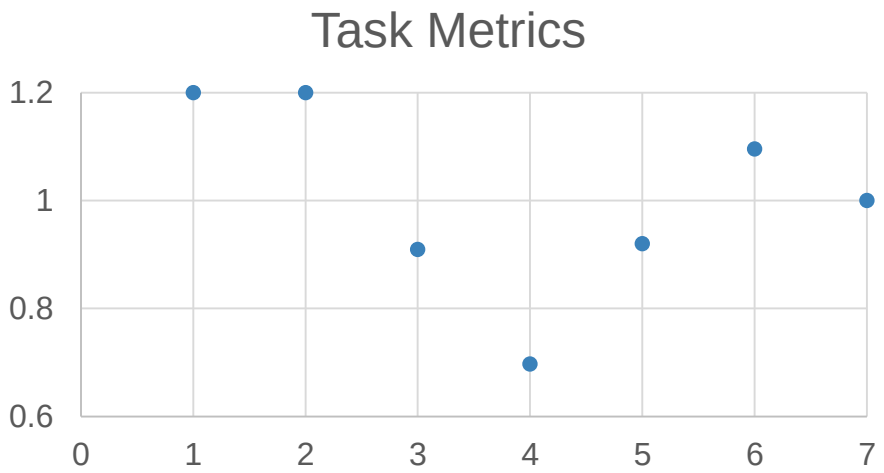
TASK AND BUG METRICS

Squish Squash Squeeze

Task Metrics

#	No. of Planned tasks	No. of Actual Tasks	Score (%)
1	5	6	120
2	5	6	120
3	22	20	91
4	33	23	69. 7
5	50	46	92
6	21	23	109
7	16	16	1

Score(%)	Action
TM < 50	1. Immediately, inform your supervisor about the slip within 24 hours. 2. Then use the same mitigation as the category 50 < TM < 90 and seriously consider dropping tasks.
50 < TM <= 90	1. Re-estimate the tasks for the future iterations. 2. Deduct the number of days behind schedule from buffer days. 3. If there is no more buffer day, decide the functionalities to drop.
90 < TM <= 110	Our estimates are fairly accurate, and we are roughly on track. 1.Add/Deduct the number of days behind schedule from buffer days. 2.If there is no more buffer day, decide the functionalities to drop
110 < TM <= 150	Gross over-estimated the effort required. 1.Re-estimate the tasks for the future iterations. 2.Add the number of days gained to buffer days.
150 < TM	1. Immediately, inform your supervisor about the slip within 24 hours. 2. Then use the same mitigation as the category 110 < SM < 150

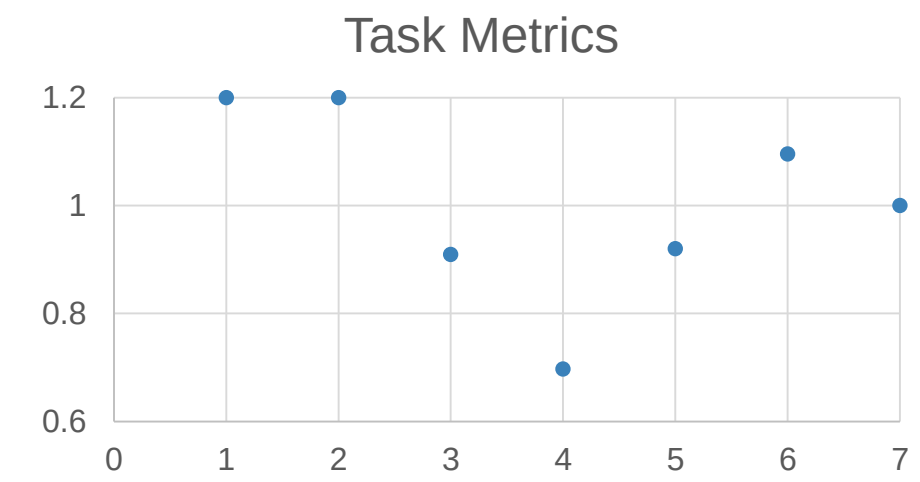


For Iterations 1 & 2, there was gross over estimation of effort. From this, we knew we were over allocating needed time for technical documents. Hence, in future iterations, we reduced the number of time allocated for creating technical documents.

Task Metrics

#	No. of Planned tasks	No. of Actual Tasks	Score (%)
1	5	6	120
2	5	6	120
3	22	20	91
4	33	23	69. 7
5	50	46	92
6	21	23	109
7	16	16	1

Score(%)	Action
TM < 50	1. Immediately, inform your supervisor about the slip within 24 hours. 2. Then use the same mitigation as the category 50 < TM < 90 and seriously consider dropping tasks.
50 < TM <= 90	1. Re-estimate the tasks for the future iterations. 2. Deduct the number of days behind schedule from buffer days. 3. If there is no more buffer day, decide the functionalities to drop.
90 < TM <= 110	Our estimates are fairly accurate, and we are roughly on track. 1.Add/Deduct the number of days behind schedule from buffer days. 2.If there is no more buffer day, decide the functionalities to drop
110 < TM <= 150	Gross over-estimated the effort required. 1.Re-estimate the tasks for the future iterations. 2.Add the number of days gained to buffer days.
150 < TM	1. Immediately, inform your supervisor about the slip within 24 hours. 2. Then use the same mitigation as the category 110 < SM < 150

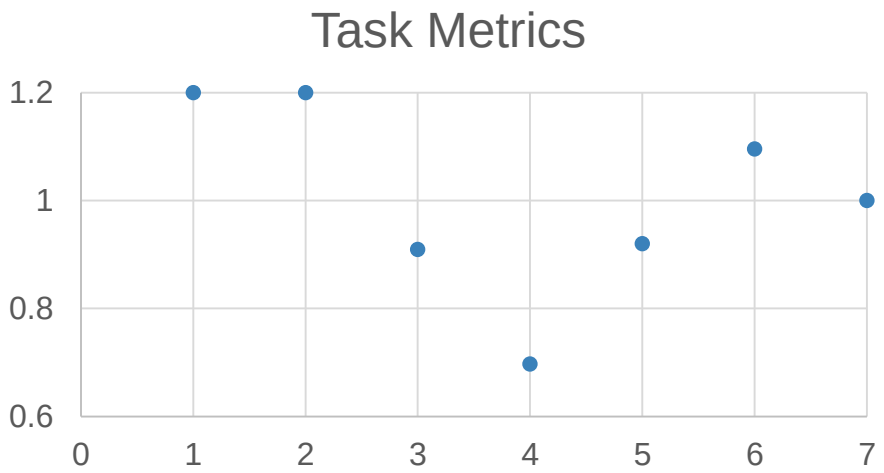


For Iteration 4, there was gross under estimation of effort.
We were complacent and underestimated the difficulty of AGD and BLR because we thought we could allocate the same time we did for login and bootstrap. Codes were then not completed on time and a lot of incomplete tasks. Hence in future iterations, we discussed the major parts of the business logic during our project kickstart and from that, we were able to better derive the estimated time taken. Furthermore, we decided to allot longer sessions for AGD and BLR as well as more sessions for the two.

Task Metrics

#	No. of Planned tasks	No. of Actual Tasks	Score (%)
1	5	6	120
2	5	6	120
3	22	20	91
4	33	23	69. 7
5	50	46	92
6	21	23	109
7	16	16	1

Score(%)	Action
TM < 50	1. Immediately, inform your supervisor about the slip within 24 hours. 2. Then use the same mitigation as the category 50 < TM < 90 and seriously consider dropping tasks.
50 < TM <= 90	1. Re-estimate the tasks for the future iterations. 2. Deduct the number of days behind schedule from buffer days. 3. If there is no more buffer day, decide the functionalities to drop.
90 < TM <= 110	Our estimates are fairly accurate, and we are roughly on track. 1.Add/Deduct the number of days behind schedule from buffer days. 2.If there is no more buffer day, decide the functionalities to drop
110 < TM <= 150	Gross over-estimated the effort required. 1.Re-estimate the tasks for the future iterations. 2.Add the number of days gained to buffer days.
150 < TM	1. Immediately, inform your supervisor about the slip within 24 hours. 2. Then use the same mitigation as the category 110 < SM < 150



For Iterations 3,5,6 & 7, our estimates are fairly accurate and we are roughly on track. This goes to show that our analysis of prior iterations and discussing how we will tackle the logic of each feature at the start of each iteration paid off and that we are more in control now.

Task Metrics

#	No. of Planned tasks	No. of Actual Tasks	Score (%)
1	5	6	120
2	5	6	120
3	22	20	91
4	33	23	69. 7
5	50	46	92
6	21	23	109
7	16	16	1

Score(%)	Action
TM < 50	1. Immediately, inform your supervisor about the slip within 24 hours. 2. Then use the same mitigation as the category 50 < TM < 90 and seriously consider dropping tasks.
50 < TM <= 90	1. Re-estimate the tasks for the future iterations. 2. Deduct the number of days behind schedule from buffer days. 3. If there is no more buffer day, decide the functionalities to drop.
90 < TM <= 110	Our estimates are fairly accurate, and we are roughly on track. 1.Add/Deduct the number of days behind schedule from buffer days. 2.If there is no more buffer day, decide the functionalities to drop
110 < TM <= 150	Gross over-estimated the effort required. 1.Re-estimate the tasks for the future iterations. 2.Add the number of days gained to buffer days.
150 < TM	1. Immediately, inform your supervisor about the slip within 24 hours. 2. Then use the same mitigation as the category 110 < SM < 150

Follow Up Actions for Most Severe Cases:

The most severe case for our team would be that of Iteration 4. Iteration 4 was the iteration where we began tackling the complicated business logic of the project. As our group kept a close eye on the Project Schedule, we saw signs of trouble when we couldn't think of the logic for the Auto Group Detection despite our best efforts.

Halfway through the week, we decided to inform the Project Manager about this and requested to work on Heatmap instead. Thus, we added in unplanned tasks so that we could focus on working on the slightly less complicated Heatmap instead of the Auto Group Detection which was proving to be very troublesome.

Challenges Faced:

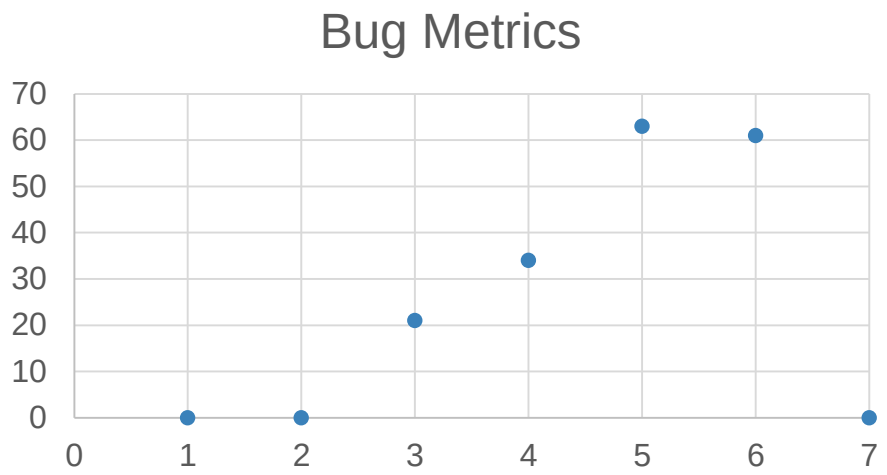
The format of the Project Schedule was not the most intuitive of tools to be used. Therefore, we took time during the initial phases of the project to take ownership of the Project Schedule by adjusting things that we could adjust.

This included adding thick black lines to demarcate boundaries between each iteration as well as changing the datetime to DD-MMM HH:mm [4-Sep 13:00] which was much easier to read and also reduces the likelihood of us jumbling up the date and the month.

Bug Metrics

#	Score
1	0
2	0
3	21
4	34
5	63
6	61
7	0

Points in Iteration	Action
Points=<5	Fix during buffer time only.
5<Points<10	Use the planned debugging time.
Points>=10	Stop current development and resolve the bug immediately. Project Manager reschedules the project.



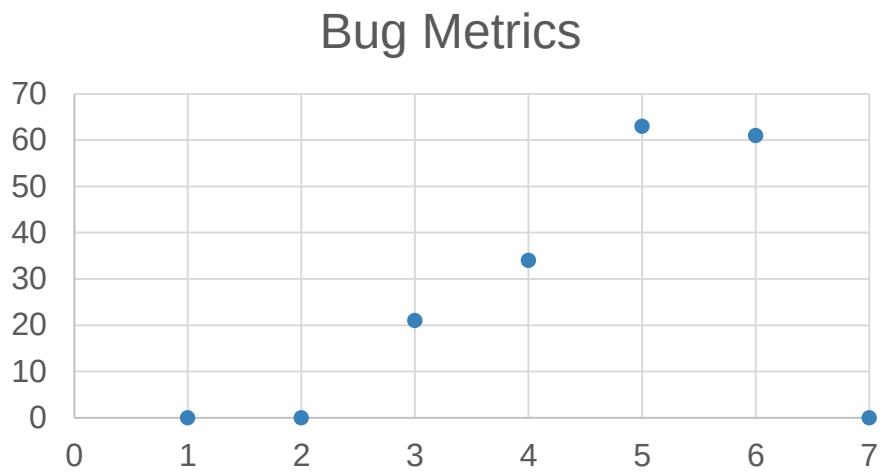
For Iterations 1 & 2, there were no bugs due to these 2 iterations being mainly creating of technical documents and programming was done in silos without nothing substantial to test. Therefore the Scores were 0 and there was nothing to be done.

For Iteration 7, we resolved all bugs that were identified. This being the last iteration of the project, there was nothing to be done.

Bug Metrics

#	Score
1	0
2	0
3	21
4	34
5	63
6	61
7	0

Points in Iteration	Action
Points=<5	Fix during buffer time only.
5<Points<10	Use the planned debugging time.
Points>=10	Stop current development and resolve the bug immediately. Project Manager reschedules the project.



For Iterations 3,4,5,6 & 7 there were far more bugs being found. This resulted in us using our scheduled buffer time to take care of them. However for those which overflowed, the subsequent Project Manager planned for a debugging session to take care of these bugs before planning other more tasks.

The goal here was to take care of the critical bugs before carrying on to other parts of the project. We made use of the Bug Severity score to prioritize which bugs to take care of first which made it easier to move on from there.

Bug Metrics

#	Score
1	0
2	0
3	21
4	34
5	63
6	61
7	0

Points in Iteration	Action
Points=<5	Fix during buffer time only.
5<Points<10	Use the planned debugging time.
Points>=10	Stop current development and resolve the bug immediately. Project Manager reschedules the project.

Follow Up Actions for Most Severe Cases:

The most severe cases for our team would be that of Iteration 5 and 6. This is due to the bulk of the complex codes being implemented during these 2 iterations which resulted in more cleverly hidden bugs.

Having monitored the bug log closely and seeing the number of bugs piled up and the number of debugging sessions decreasing. We quickly informed the Project Manager and requested for even more debugging sessions in the following iterations after having exhausted the buffer we had.

Challenges Faced:

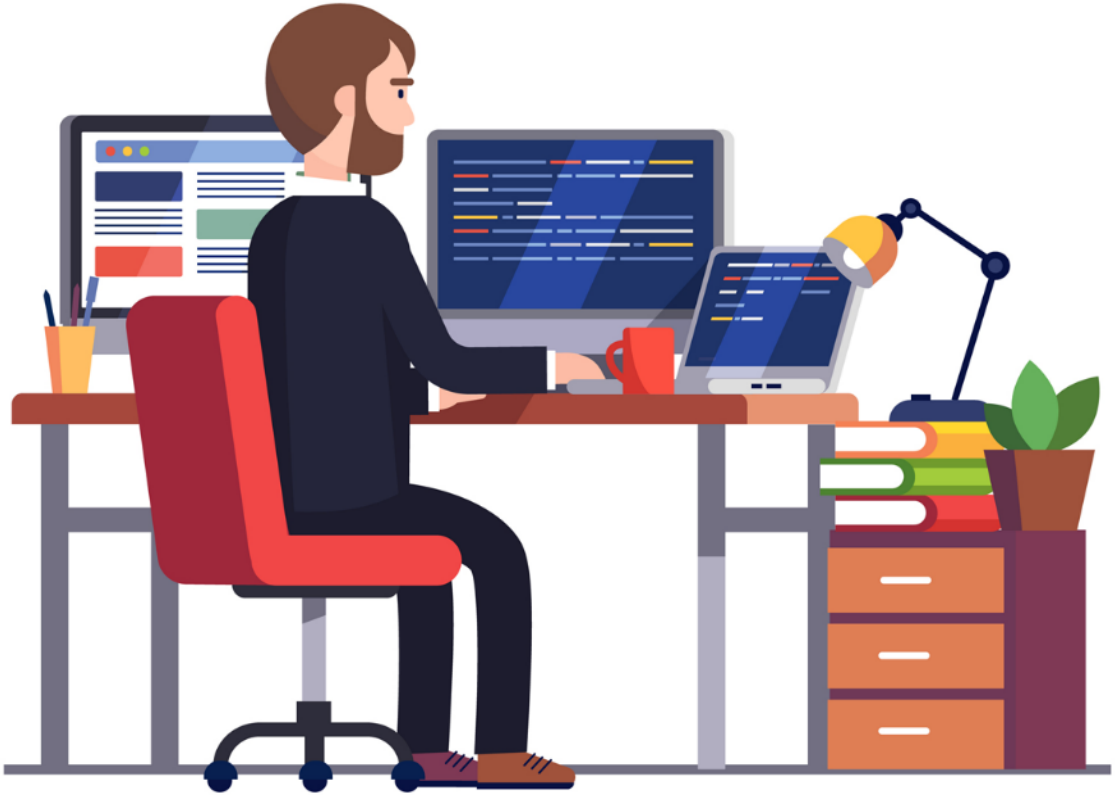
As beginners to the whole Software Engineering process, almost every member of the team were not used to the whole process of logging bugs. For those who do log it down properly, most of it were in different formats and decipherable only by its creator.

Therefore, we had to hold a meeting to bring this up and standardize a couple of things before we could move on. This saved a lot of trouble in the later parts of the project where we had to come up with the bug metrics.

5.

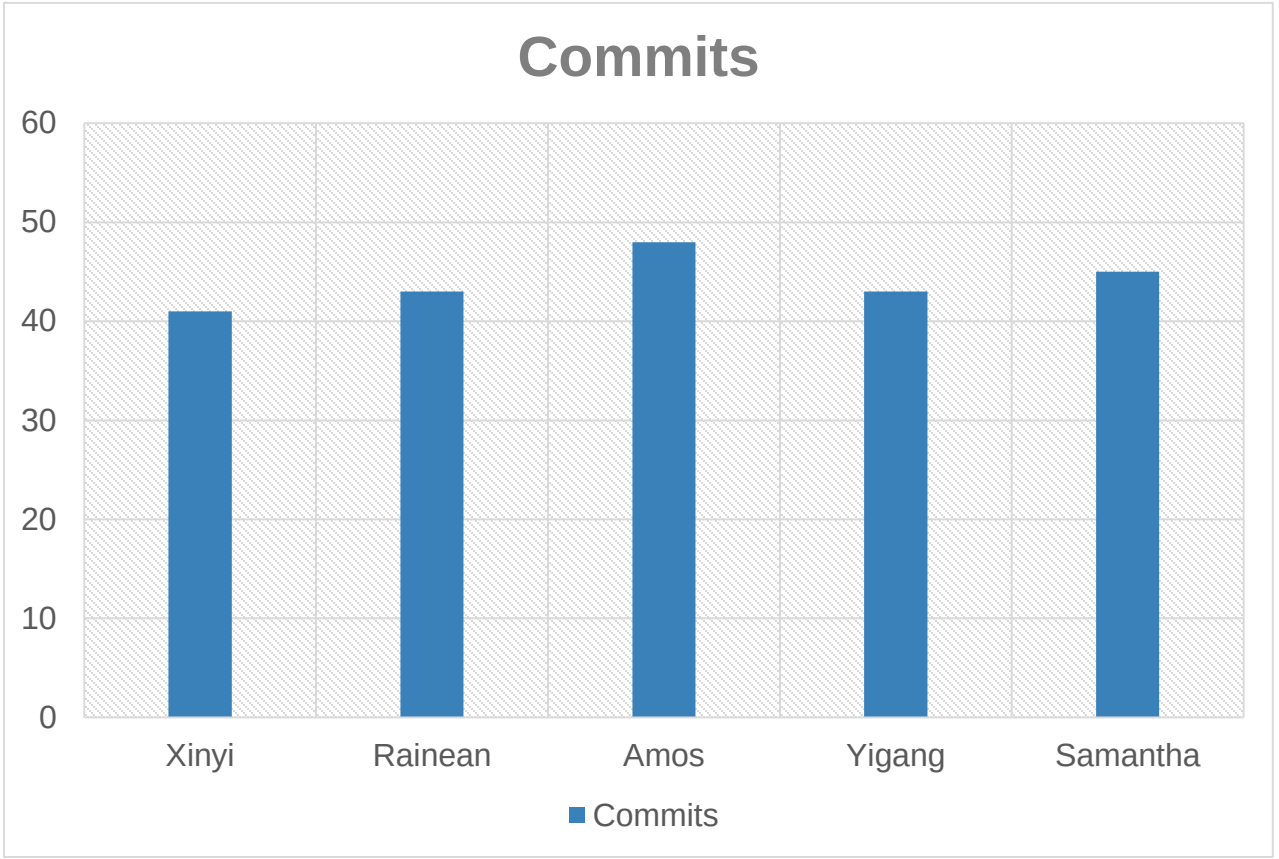
USE OF GIT

In case of fire:
Git Commit
Git Push
Leave Building



Pull Commit Push

Author	Commits
Xinyi	41
Rainean	43
Amos	48
Yigang	43
Samantha	45



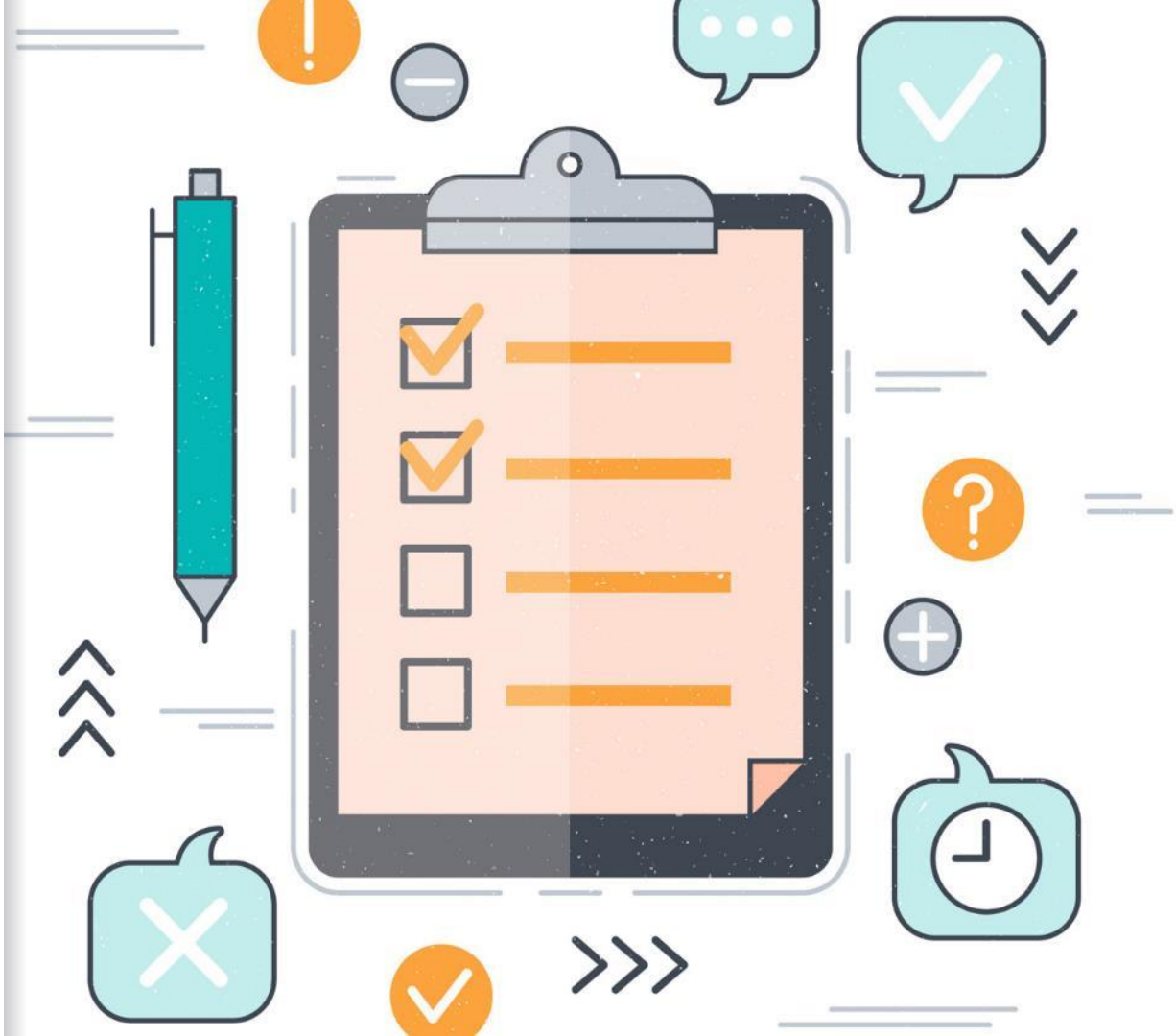
Team XRAYS Git 101

Step 1	Ensure that work is completed before committing
Step 2	Test before you commit
Step 3	Commit regularly
Step 4	Write good commit message
Step 5	Copy latest Commit ID and paste on Project Schedule

6.

TEST SCORE

42: The answer to life



Our Results

```
C:\Users\adept\Downloads\jsonchecker
Test Case 0 passed
Test Case 1 passed
executing request POST http://localh
Test Case 2 passed
Test Case 3 passed
Test Case 4 passed
Test Case 5 passed
Test Case 6 passed
Test Case 7 passed
Test Case 8 passed
Test Case 9 passed
Test Case 10 passed
Test Case 11 passed
Test Case 12 passed
Test Case 13 passed
Test Case 14 passed
Test Case 15 passed
Test Case 16 passed
Test Case 17 passed
Test Case 18 passed
Test Case 19 passed
Test Case 20 passed
executing request POST http://localh
Test Case 21 passed
Total: 22/22
```

Manual UAT Testing

JSON Automated UAT Testing

6.

SERVER INFO

Bitnami

SLOCA App

Site Details

IP Address	**http://52.221.242.151**
Web URL	**http://52.221.242.151/app/login.jsp**

Login Credentials

Username	**admin**
Password	**pxraysw**

7.

OTHERS

Any other business

Main Takeaways

If you want to go fast, go alone.
If you want to go far, go together.

Communication
A concept that is always brought up but easily forgotten. Communication is key to any successful relationship and our team has learnt the importance of saying what is on our mind; in a polite way of course. This has allowed us to trust each and made our teamwork stronger.

Assumptions
In a project with many uncertainty, the path of least resistance will be to assume and think about it later. However, this strategy has cost us dearly and taught us to never assume. Always double check.

Conflict Management

Peace is not absence of conflict, it is the ability to handle conflicts by peaceful means ~ Ronald Reagan

Listening

Conflicts are necessary in order to produce innovative solutions. We have learnt to first listen before providing an objective response. Being open minded and willing to listen to others opinions and consider the fact that you may be wrong has allowed us to spend on time on quality work rather than senseless argument

Swear Jar

Having a swear jar in placed has made the environment a much more candid one. Albeit annoying at times, the jar has helped to further bond the team together. It is rather amusing that more attention is paid to the Swear jar rather than the Bug logs

Something Interesting

Profanities
One of our dear members has contributed up to $20 to the Swear jar. Yet one member has yet to swear/come late during this project. Surely someone is gaining from this exchange.

The number of people in your house
It is unthinkable that you will not know the number of people in your house. However that doesn't seem to be the case for one of our foreign friends. Either it is a trend not to know or he really doesn't care.

Red
Also a colour found in our food establishment of choice, Tea Party is the ideal energy booster Team XRAYS partakes in almost religiously.

Thank you very much for your time

If you have any questions about this document please don’t hesitate to contact us at:

- EvaLive :D

Images referenced from Freepik.com