

# Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-milestone-2-chatroom-2024/grade/owe>

IT114-006-S2024 - [IT114] Milestone 2 Chatroom 2024

## Submissions:

Submission Selection

1 Submission [active] 4/28/2024 5:44:26 PM

## Instructions

^ COLLAPSE ^

Implement the Milestone 2 features from the project's proposal document:

<https://docs.google.com/document/d/1ONmvEvel97GTfPGfVwwQC96xSsobbSbk56145XizQG4/view>

Make sure you add your ucid/date as code comments where code changes are done

All code changes should reach the Milestone2 branch

Create a pull request from Milestone2 to main and keep it open until you get the output PDF from this assignment.

Gather the evidence of feature completion based on the below tasks.

Once finished, get the output PDF and copy/move it to your repository folder on your local machine.

Run the necessary git add, commit, and push steps to move it to GitHub

Complete the pull request that was opened earlier

Upload the same output PDF to Canvas

Branch name: Milestone2

Tasks: 12 Points: 10.00



Demonstrate Usage of Payloads (2 pts.)

^ COLLAPSE ^



Task #1 - Points: 1

Text: Screenshots of your Payload class and subclasses and PayloadType

## Checklist

\*The checkboxes are for your own tracking

#

Points

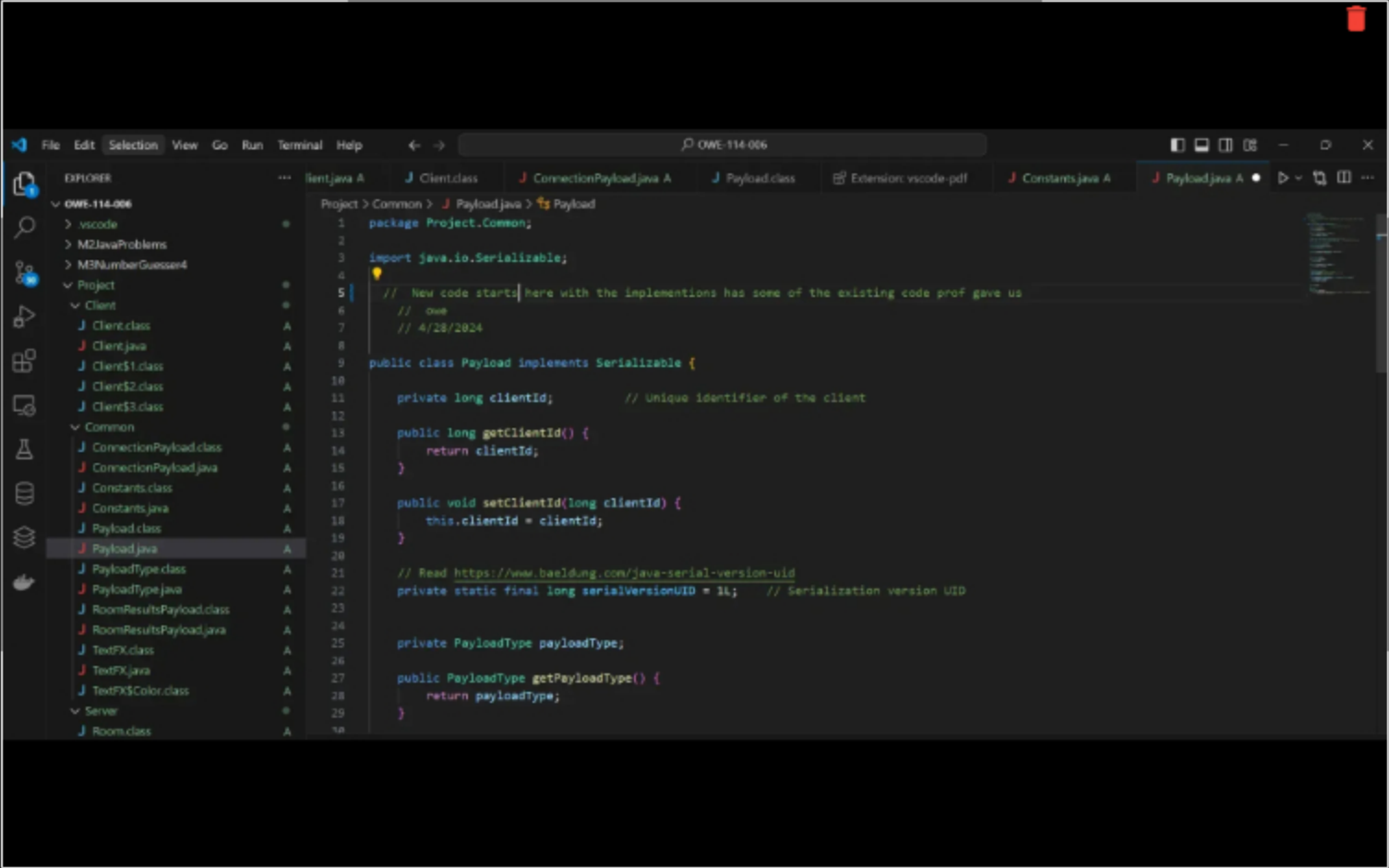
Details

#	Points	Details
#1	1	Payload, equivalent of RollPayload, and any others
#2	1	Screenshots should include ucid and date comment
#3	1	Each screenshot should be clearly captioned

Task Screenshots:

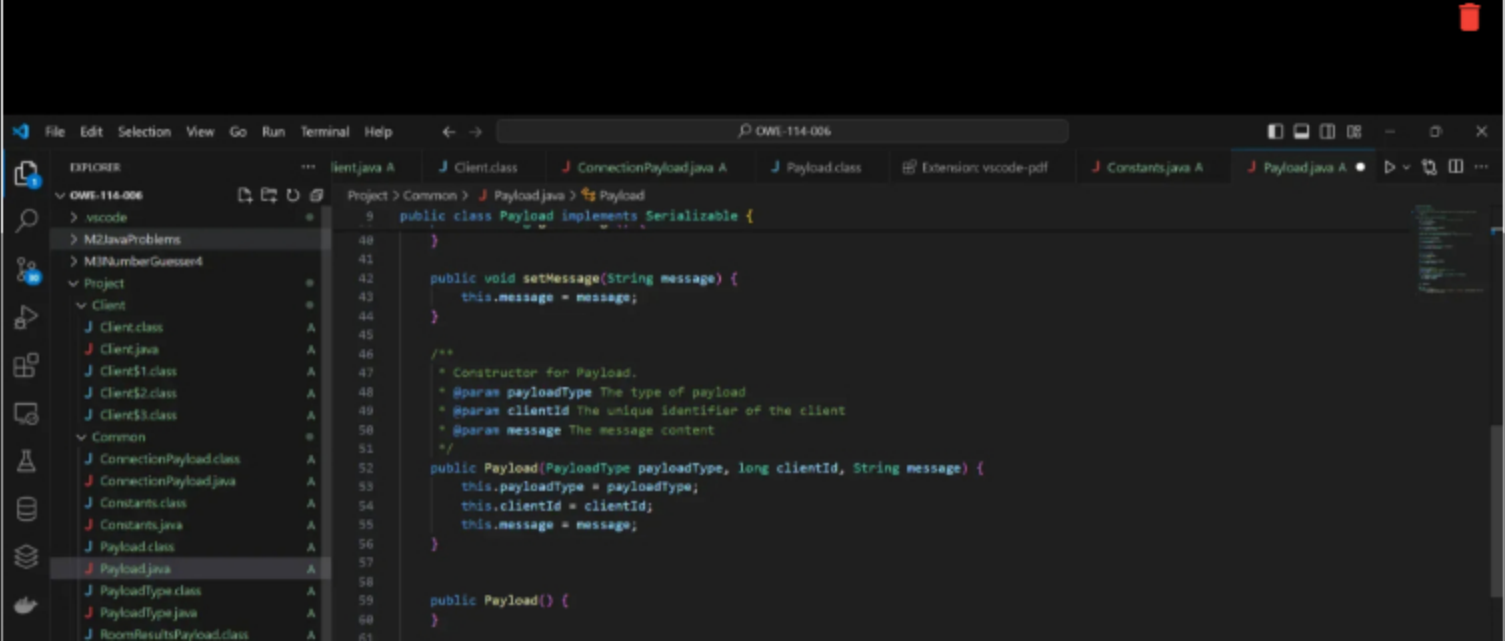
Gallery Style: Large View

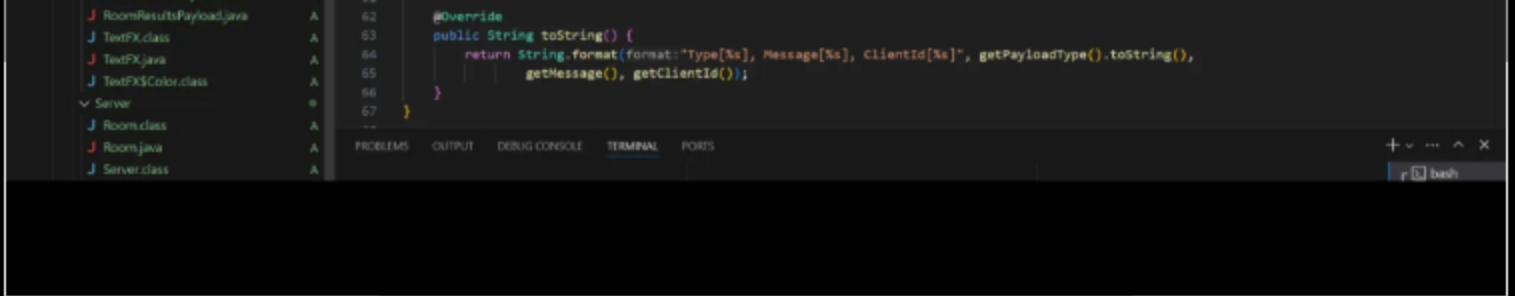
Small Medium Large



payload class with implications

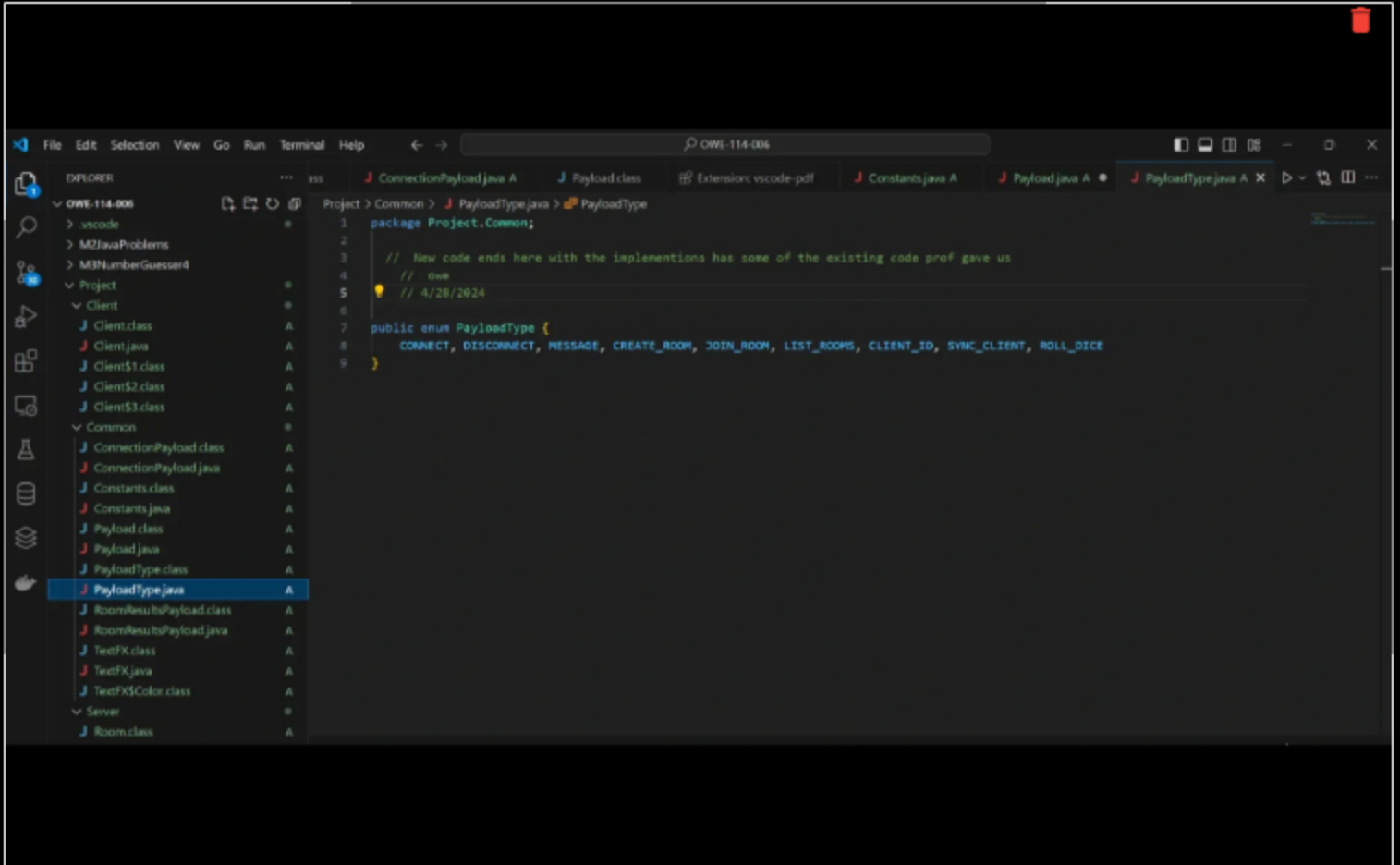
Checklist Items (0)





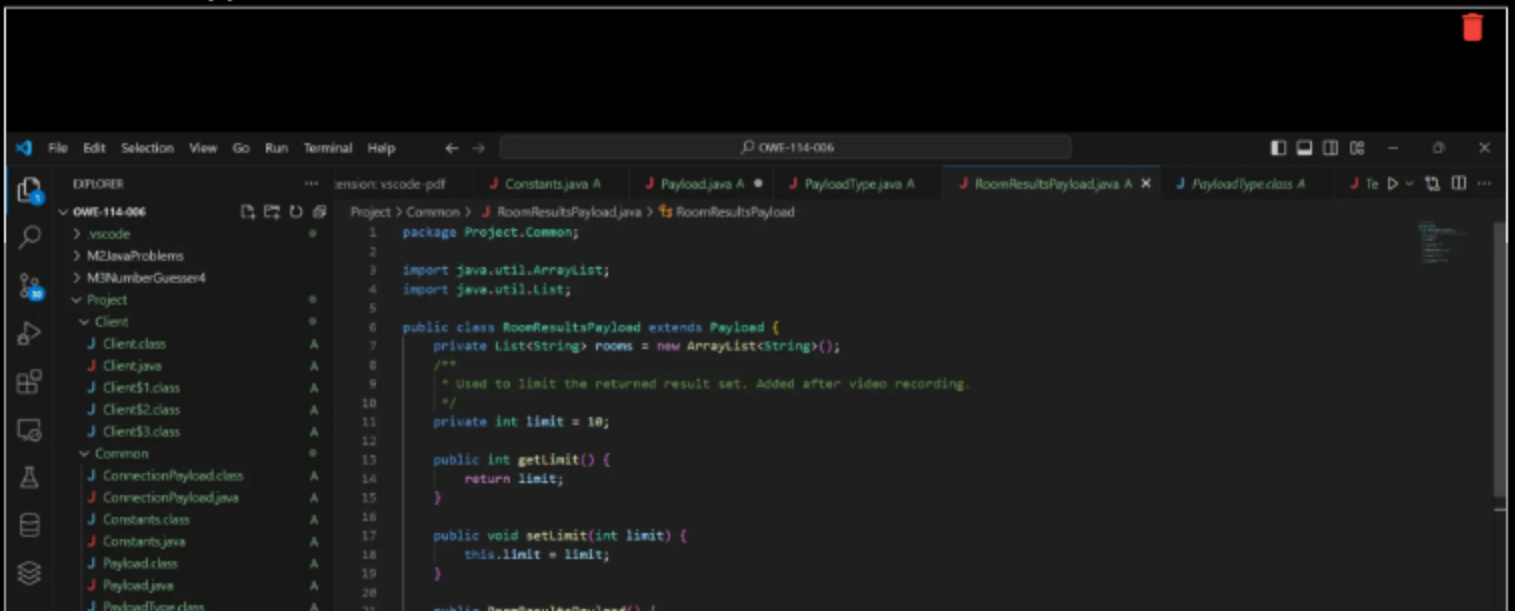
payload continued

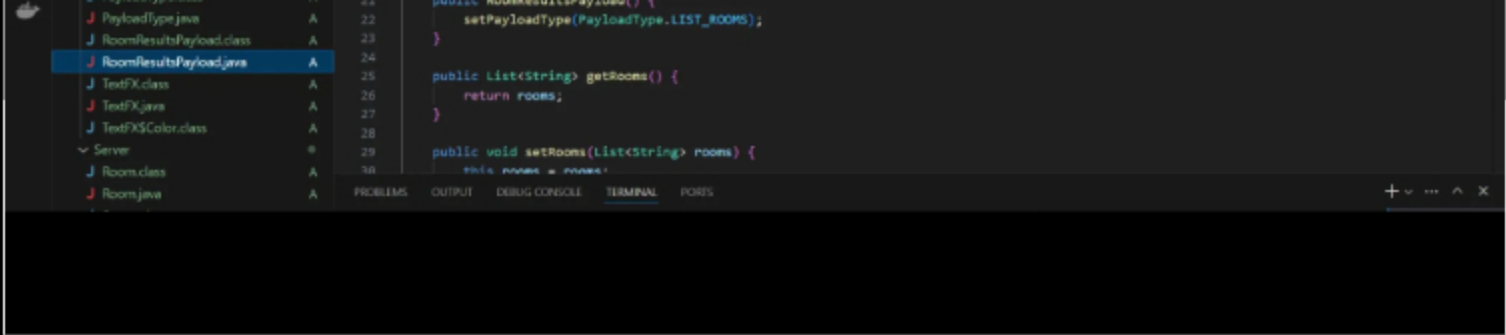
## Checklist Items (0)



payload type class

## Checklist Items (0)





subclasses for payload

Checklist Items (0)



^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots of the payloads being debugged/output to the terminal

### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Demonstrate flip
<input type="checkbox"/> #2	1	Demonstrate roll (both versions)
<input type="checkbox"/> #3	1	Demonstrate formatted message along with any others
<input type="checkbox"/> #4	1	Each screenshot should be clearly captioned

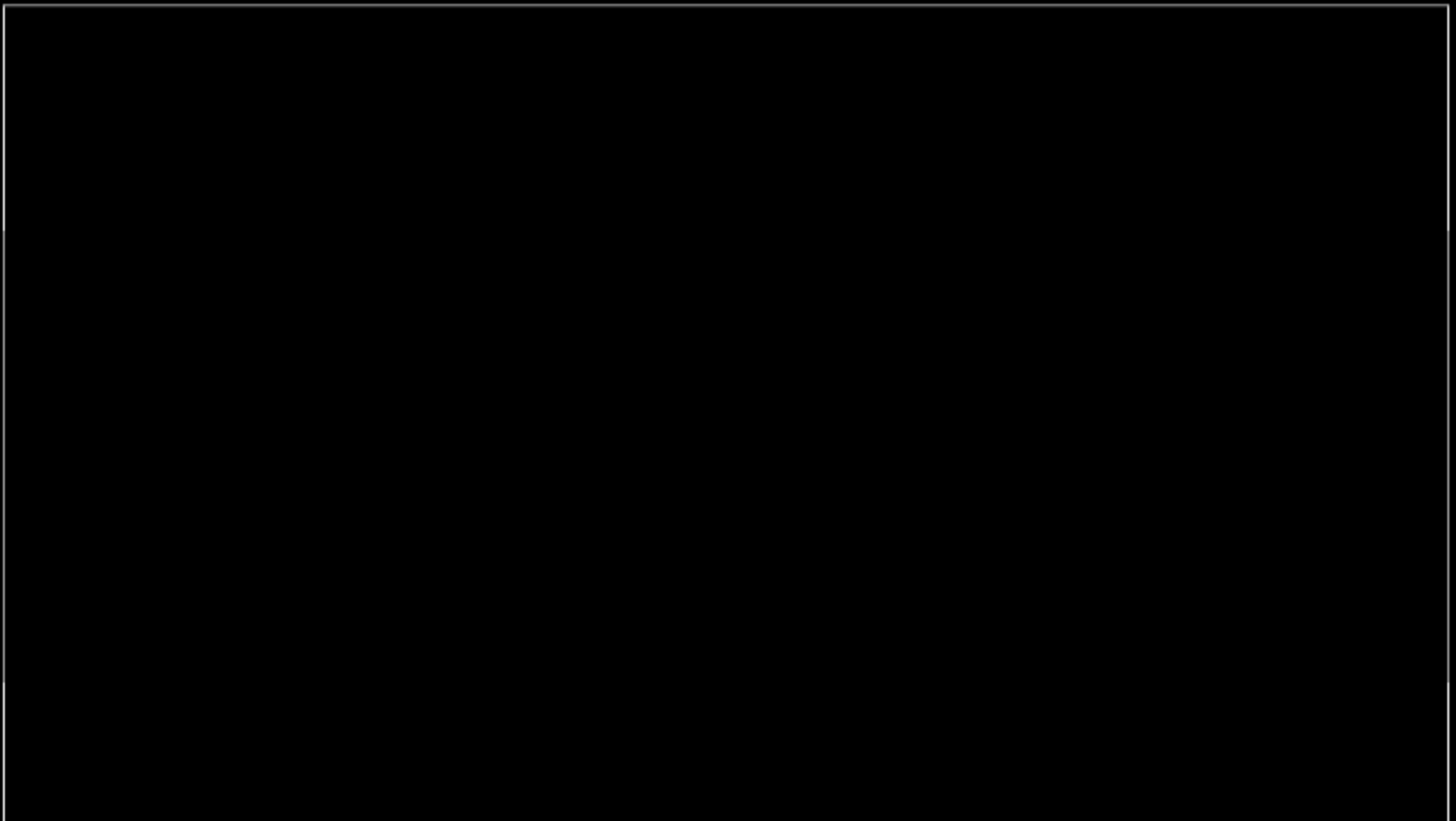
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Missing Caption

### Task #3 - Points: 1

Text: Explain the purpose of payloads and how your flip/roll payloads were made

Response:

In our system, payloads are structured data packages that help clients and servers communicate with each other. They contain a variety of commands and messages, including CREATE\_ROOM, JOIN\_ROOM, MESSAGE, CONNECT, DISCONNECT, and LIST\_ROOMS. We have added a new payload type called ROLL, which includes the data needed to simulate dice rolls. The system's functionality is improved by this new feature, which lets users participate in interactive games like dice.

Demonstrate Roll Command (2 pts.)

### Task #1 - Points: 1

Text: Screenshot of the following items

#### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Client code that captures the command and converts it to a RollPayload (or equivalent) for both scenarios /roll # and /roll #d#
<input type="checkbox"/> #2	1	ServerThread code receiving the payload and passing it to the Room
<input type="checkbox"/> #3	1	Room handling the roll action correctly for both scenarios (/roll # and /roll #d#) including the message going back out to all clients
<input type="checkbox"/> #4	1	Code screenshots should include ucid and date comment
<input type="checkbox"/> #5	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
9         e.printStackTrace();
10     }
11     return true;
12 } else if (text.equalsIgnoreCase(FLIP_COMMAND)) {
13     try {
14         sendFlip();
```

```

4         } catch (IOException e) {
5             e.printStackTrace();
6         }
7         return true;
8     } else if (text.startsWith(ROLL_COMMAND)) {
9         String rollString = text.replace(target: "/roll", replacement: "").trim();
10        try {
11            int result = roll(rollString);
12            sendRoll(result);
13        } catch (IOException e) {
14            sendMessage(message: "Wrong Format. Use : '/roll 123' or '/roll 456'.");
15        }
16        return true;
17    }
18    return false;
19 }

```

the client code for the dice for roll

## Checklist Items (0)

The screenshot shows the IDE with the following files open: `RoomResultsPayload.java`, `PayloadType.class`, `TextX.java`, `Room.java`, `Server.java`, and `ServerThread.java`. The `ServerThread.java` file is the active editor, showing the following code:

```

21 public class ServerThread extends Thread {
22     ...
23     @SuppressWarnings("unused")
24     private void sendFlip() throws IOException {
25         Payload p = new Payload();
26         p.setPayloadType(PayloadType.MESSAGE);
27         p.setMessage(message: "/flip or you shall be flipped");
28         out.writeObject(p);
29     }
30     /**
31      * @param rollString
32      * @return
33      */
34     @SuppressWarnings("unused")
35     private int roll() {
36         return (int) (Math.random() * 6) + 1;
37     }
38     @SuppressWarnings("unused")
39     private void sendRoll(int result) throws IOException {
40         Payload p = new Payload();
41         p.setPayloadType(PayloadType.MESSAGE);
42         p.setMessage("Roll result: " + result);
43         out.writeObject(p);
44     }
45 }

```

The bottom of the IDE shows the `PROBLEMS`, `OUTPUT`, `DEBUG CONSOLE`, and `TERMINAL` tabs. The `TERMINAL` tab is active, showing the command `git status` and its output.

more server thread code for roll

## Checklist Items (0)

### Task #2 - Points: 1

Text: Explain the logic in how the two different roll formats are handled and how the message flows from the client, to the Room, and shared with all other users



Response:

In our system, the Room class receives a roll message sent from the client, which is then processed to extract the roll details. All other users in the room receive this roll message after it is broadcast by the Room class. Real-time communication is made possible during gameplay as each user receives the message and can see the outcome of the roll. This logic improves the user experience by keeping all users informed and synchronized about the current dice rolls.

Demonstrate Flip Command (1 pt.)

^COLLAPSE ^

Task #1 - Points: 1

^COLLAPSE ^

Text: Screenshot of the following items

Checklist			*The checkboxes are for your own tracking
#	Points	Details	
<input type="checkbox"/> #1	1	Client code that captures the command and converts it to a payload	
<input type="checkbox"/> #2	1	ServerThread receiving the payload and passing it to the Room	
<input type="checkbox"/> #3	1	Room handling the flip action correctly	
<input type="checkbox"/> #4	1	Code screenshots should include ucid and date comment	
<input type="checkbox"/> #5	1	Each screenshot should be clearly captioned	

Task Screenshots:

Gallery Style: Large View

SmallMediumLarge

OWE-114-006

vscode

M2JavaProblems

M3NumberGuesser4

Project

Client

Client.class

**Client.java**

Client\$1.class

Client\$2.class

Client\$3.class

Common

ConnectionPayload.class

ConnectionPayload.java

Constants.class

Constants.java

Payload.class

Payload.java

PayloadType.class

PayloadType.java

RoomResultsPayload.class

RoomResultsPayload.java

TextFX.class

TextFX.java

TextFX\$Color.class

Server

Room.class

Room.java

Server.class

Project > Class > Client.java > Client > getClassName()

21 public enum Client {  
132 private boolean processClientCommand(String text) throws IOException {  
140 return true;  
189 }  
190 // New Code Begins where i added everything asked in the instructions such as the flip, roll, and hello command.  
191 // OWE  
192 // 4-28-24  
193  
194 else if (text.equalsIgnoreCase(HELLO\_COMMAND)) {  
195 try {  
196 sendHello();  
197 } catch (IOException e) {  
198 e.printStackTrace();  
199 }  
200 return true;  
201 } else if (text.equalsIgnoreCase(FLIP\_COMMAND)) {  
202 try {  
203 sendFlip();  
204 } catch (IOException e) {  
205 e.printStackTrace();  
206 }  
207 return true;  
208 } else if (text.startsWith(ROLL\_COMMAND)) {  
209 String rollString = text.replace(target: "/roll", replacement: "").trim();  
210 try {  
211 int result = roll(rollString);  
212 sendRoll(result);  
213 } catch (IOException e) {  
214 e.printStackTrace();  
215 }  
216 }  
217 }  
218 }  
219 }  
220 }  
221 }  
222 }  
223 }  
224 }  
225 }  
226 }  
227 }  
228 }  
229 }  
230 }  
231 }  
232 }  
233 }  
234 }  
235 }  
236 }  
237 }  
238 }  
239 }  
240 }  
241 }  
242 }  
243 }  
244 }  
245 }  
246 }  
247 }  
248 }  
249 }  
250 }  
251 }  
252 }  
253 }  
254 }  
255 }  
256 }  
257 }  
258 }  
259 }  
260 }  
261 }  
262 }  
263 }  
264 }  
265 }  
266 }  
267 }  
268 }  
269 }  
270 }  
271 }  
272 }  
273 }  
274 }  
275 }  
276 }  
277 }  
278 }  
279 }  
280 }  
281 }  
282 }  
283 }  
284 }  
285 }  
286 }  
287 }  
288 }  
289 }  
290 }  
291 }  
292 }  
293 }  
294 }  
295 }  
296 }  
297 }  
298 }  
299 }  
300 }  
301 }  
302 }  
303 }  
304 }  
305 }  
306 }  
307 }  
308 }  
309 }  
310 }  
311 }  
312 }  
313 }  
314 }  
315 }  
316 }  
317 }  
318 }  
319 }  
320 }  
321 }  
322 }  
323 }  
324 }  
325 }  
326 }  
327 }  
328 }  
329 }  
330 }  
331 }  
332 }  
333 }  
334 }  
335 }  
336 }  
337 }  
338 }  
339 }  
340 }  
341 }  
342 }  
343 }  
344 }  
345 }  
346 }  
347 }  
348 }  
349 }  
350 }  
351 }  
352 }  
353 }  
354 }  
355 }  
356 }  
357 }  
358 }  
359 }  
360 }  
361 }  
362 }  
363 }  
364 }  
365 }  
366 }  
367 }  
368 }  
369 }  
370 }  
371 }  
372 }  
373 }  
374 }  
375 }  
376 }  
377 }  
378 }  
379 }  
380 }  
381 }  
382 }  
383 }  
384 }  
385 }  
386 }  
387 }  
388 }  
389 }  
390 }  
391 }  
392 }  
393 }  
394 }  
395 }  
396 }  
397 }  
398 }  
399 }  
400 }  
401 }  
402 }  
403 }  
404 }  
405 }  
406 }  
407 }  
408 }  
409 }  
410 }  
411 }  
412 }  
413 }  
414 }  
415 }  
416 }  
417 }  
418 }  
419 }  
420 }  
421 }  
422 }  
423 }  
424 }  
425 }  
426 }  
427 }  
428 }  
429 }  
430 }  
431 }  
432 }  
433 }  
434 }  
435 }  
436 }  
437 }  
438 }  
439 }  
440 }  
441 }  
442 }  
443 }  
444 }  
445 }  
446 }  
447 }  
448 }  
449 }  
450 }  
451 }  
452 }  
453 }  
454 }  
455 }  
456 }  
457 }  
458 }  
459 }  
460 }  
461 }  
462 }  
463 }  
464 }  
465 }  
466 }  
467 }  
468 }  
469 }  
470 }  
471 }  
472 }  
473 }  
474 }  
475 }  
476 }  
477 }  
478 }  
479 }  
480 }  
481 }  
482 }  
483 }  
484 }  
485 }  
486 }  
487 }  
488 }  
489 }  
490 }  
491 }  
492 }  
493 }  
494 }  
495 }  
496 }  
497 }  
498 }  
499 }  
500 }  
501 }  
502 }  
503 }  
504 }  
505 }  
506 }  
507 }  
508 }  
509 }  
510 }  
511 }  
512 }  
513 }  
514 }  
515 }  
516 }  
517 }  
518 }  
519 }  
520 }  
521 }  
522 }  
523 }  
524 }  
525 }  
526 }  
527 }  
528 }  
529 }  
530 }  
531 }  
532 }  
533 }  
534 }  
535 }  
536 }  
537 }  
538 }  
539 }  
540 }  
541 }  
542 }  
543 }  
544 }  
545 }  
546 }  
547 }  
548 }  
549 }  
550 }  
551 }  
552 }  
553 }  
554 }  
555 }  
556 }  
557 }  
558 }  
559 }  
560 }  
561 }  
562 }  
563 }  
564 }  
565 }  
566 }  
567 }  
568 }  
569 }  
570 }  
571 }  
572 }  
573 }  
574 }  
575 }  
576 }  
577 }  
578 }  
579 }  
580 }  
581 }  
582 }  
583 }  
584 }  
585 }  
586 }  
587 }  
588 }  
589 }  
590 }  
591 }  
592 }  
593 }  
594 }  
595 }  
596 }  
597 }  
598 }  
599 }  
600 }  
601 }  
602 }  
603 }  
604 }  
605 }  
606 }  
607 }  
608 }  
609 }  
610 }  
611 }  
612 }  
613 }  
614 }  
615 }  
616 }  
617 }  
618 }  
619 }  
620 }  
621 }  
622 }  
623 }  
624 }  
625 }  
626 }  
627 }  
628 }  
629 }  
630 }  
631 }  
632 }  
633 }  
634 }  
635 }  
636 }  
637 }  
638 }  
639 }  
640 }  
641 }  
642 }  
643 }  
644 }  
645 }  
646 }  
647 }  
648 }  
649 }  
650 }  
651 }  
652 }  
653 }  
654 }  
655 }  
656 }  
657 }  
658 }  
659 }  
660 }  
661 }  
662 }  
663 }  
664 }  
665 }  
666 }  
667 }  
668 }  
669 }  
670 }  
671 }  
672 }  
673 }  
674 }  
675 }  
676 }  
677 }  
678 }  
679 }  
680 }  
681 }  
682 }  
683 }  
684 }  
685 }  
686 }  
687 }  
688 }  
689 }  
690 }  
691 }  
692 }  
693 }  
694 }  
695 }  
696 }  
697 }  
698 }  
699 }  
700 }  
701 }  
702 }  
703 }  
704 }  
705 }  
706 }  
707 }  
708 }  
709 }  
710 }  
711 }  
712 }  
713 }  
714 }  
715 }  
716 }  
717 }  
718 }  
719 }  
720 }  
721 }  
722 }  
723 }  
724 }  
725 }  
726 }  
727 }  
728 }  
729 }  
730 }  
731 }  
732 }  
733 }  
734 }  
735 }  
736 }  
737 }  
738 }  
739 }  
740 }  
741 }  
742 }  
743 }  
744 }  
745 }  
746 }  
747 }  
748 }  
749 }  
750 }  
751 }  
752 }  
753 }  
754 }  
755 }  
756 }  
757 }  
758 }  
759 }  
760 }  
761 }  
762 }  
763 }  
764 }  
765 }  
766 }  
767 }  
768 }  
769 }  
770 }  
771 }  
772 }  
773 }  
774 }  
775 }  
776 }  
777 }  
778 }  
779 }  
780 }  
781 }  
782 }  
783 }  
784 }  
785 }  
786 }  
787 }  
788 }  
789 }  
790 }  
791 }  
792 }  
793 }  
794 }  
795 }  
796 }  
797 }  
798 }  
799 }  
800 }  
801 }  
802 }  
803 }  
804 }  
805 }  
806 }  
807 }  
808 }  
809 }  
810 }  
811 }  
812 }  
813 }  
814 }  
815 }  
816 }  
817 }  
818 }  
819 }  
820 }  
821 }  
822 }  
823 }  
824 }  
825 }  
826 }  
827 }  
828 }  
829 }  
830 }  
831 }  
832 }  
833 }  
834 }  
835 }  
836 }  
837 }  
838 }  
839 }  
840 }  
841 }  
842 }  
843 }  
844 }  
845 }  
846 }  
847 }  
848 }  
849 }  
850 }  
851 }  
852 }  
853 }  
854 }  
855 }  
856 }  
857 }  
858 }  
859 }  
860 }  
861 }  
862 }  
863 }  
864 }  
865 }  
866 }  
867 }  
868 }  
869 }  
870 }  
871 }  
872 }  
873 }  
874 }  
875 }  
876 }  
877 }  
878 }  
879 }  
880 }  
881 }  
882 }  
883 }  
884 }  
885 }  
886 }  
887 }  
888 }  
889 }  
890 }  
891 }  
892 }  
893 }  
894 }  
895 }  
896 }  
897 }  
898 }  
899 }  
900 }  
901 }  
902 }  
903 }  
904 }  
905 }  
906 }  
907 }  
908 }  
909 }  
910 }  
911 }  
912 }  
913 }  
914 }  
915 }  
916 }  
917 }  
918 }  
919 }  
920 }  
921 }  
922 }  
923 }  
924 }  
925 }  
926 }  
927 }  
928 }  
929 }  
930 }  
931 }  
932 }  
933 }  
934 }  
935 }  
936 }  
937 }  
938 }  
939 }  
940 }  
941 }  
942 }  
943 }  
944 }  
945 }  
946 }  
947 }  
948 }  
949 }  
950 }  
951 }  
952 }  
953 }  
954 }  
955 }  
956 }  
957 }  
958 }  
959 }  
960 }  
961 }  
962 }  
963 }  
964 }  
965 }  
966 }  
967 }  
968 }  
969 }  
970 }  
971 }  
972 }  
973 }  
974 }  
975 }  
976 }  
977 }  
978 }  
979 }  
980 }  
981 }  
982 }  
983 }  
984 }  
985 }  
986 }  
987 }  
988 }  
989 }  
990 }  
991 }  
992 }  
993 }  
994 }  
995 }  
996 }  
997 }  
998 }  
999 }  
1000 }

## client code for flip command

### Checklist Items (0)

The screenshot shows an IDE with a project named 'OWE-114-006'. The file explorer on the left lists various Java files, including 'ServerThread.java'. The main editor displays the code for 'ServerThread.java', which is a class extending 'Thread'. It includes methods for 'getClientId()', 'sendHello()', and 'sendFlip()'. The 'sendFlip()' method sets a message to '/flip or you shall be flipped' and writes it to the output. The terminal at the bottom shows the command prompt for a user named 'omar elhossary' in a directory related to 'OWE-114-006'. The terminal output shows the command 'git status' and the resulting output.

## serverthread code for flip

### Checklist Items (0)

#### Task #2 - Points: 1

Text: Explain the logic in how the flip command is handled and processed and how the message flows from the client, to the Room, and shared with all other users

### Response:

In order to handle the flip command, it is sent from the client to the Room class, where it is processed to ascertain the flip's outcome. The message is then broadcast to all other users in the room with this outcome. Real-time communication and coordinated gameplay are made possible by the message that each user receives and the ability to view the flip's result. This logic makes sure that everyone is aware of the outcome of the flip, which makes it easier for people to interact and communicate in the shared virtual environment.



## Demonstrate Formatted Messages (4 pts.)

^COLLAPSE ^

### Task #1 - Points: 1

Text: Screenshot of Room how the following formatting is processed from a message

#### Details:

Note: this processing is server-side

Slash commands are not valid solutions for this and will receive 0 credit

#### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Room code processing for bold
<input type="checkbox"/> #2	1	Room code processing for italic
<input type="checkbox"/> #3	1	Room code processing for underline
<input type="checkbox"/> #4	1	Room code processing for color (at least R, G, B or support for hex codes)
<input type="checkbox"/> #5	1	Show each one working individually and one showing a combination of all of the formats and 1 color from the terminal
<input type="checkbox"/> #6	1	Must not rely on the user typing html characters, but the output can be html characters
<input type="checkbox"/> #7	1	Code screenshots should include ucid and date comment
<input type="checkbox"/> #8	1	Each screenshot should be clearly captioned

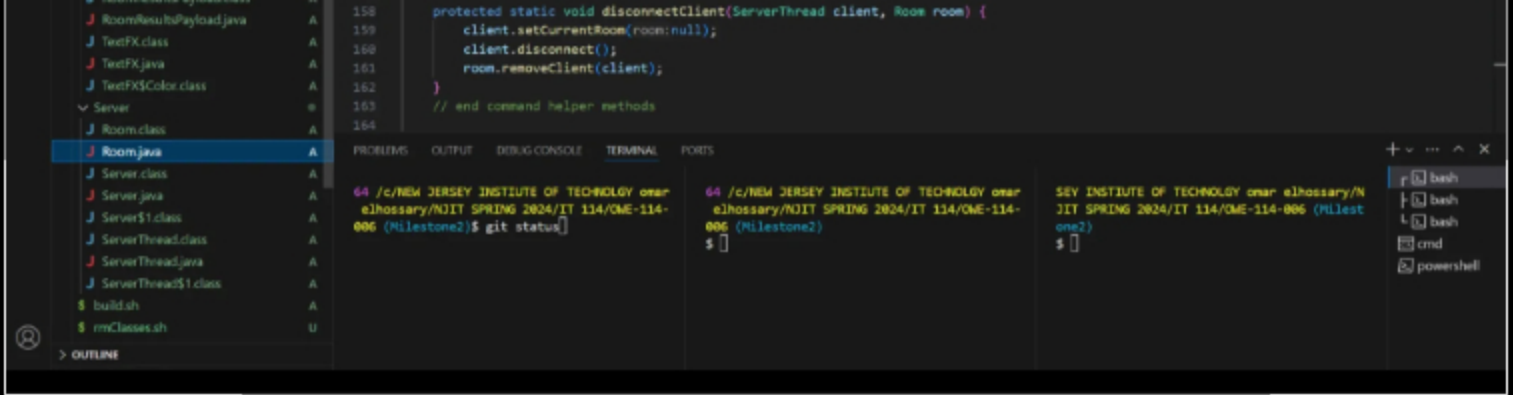
#### Task Screenshots:

#### Gallery Style: Large View

Small

Medium

Large



code for room was not finished

## Checklist Items (0)



^COLLAPSE ^

Task #2 - Points: 1

Text: Explain the following

### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Which special characters translate to the desired effect
<input type="checkbox"/> #2	1	How the logic works that converts the message to its final format

Response:

i did not get to do this



Misc (1 pt.)

^COLLAPSE ^



^COLLAPSE ^

Task #1 - Points: 1

Text: Add the pull request link for the branch

**i** Details:

Note: the link should end with /pull/#

URL #1

<https://github.com/WayguBeef5/OWE-114-006/pull/9>



^COLLAPSE ^

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Response.

Missing Response



^COLLAPSE ^

Task #3 - Points: 1

Text: WakaTime Screenshot

### Details:

Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

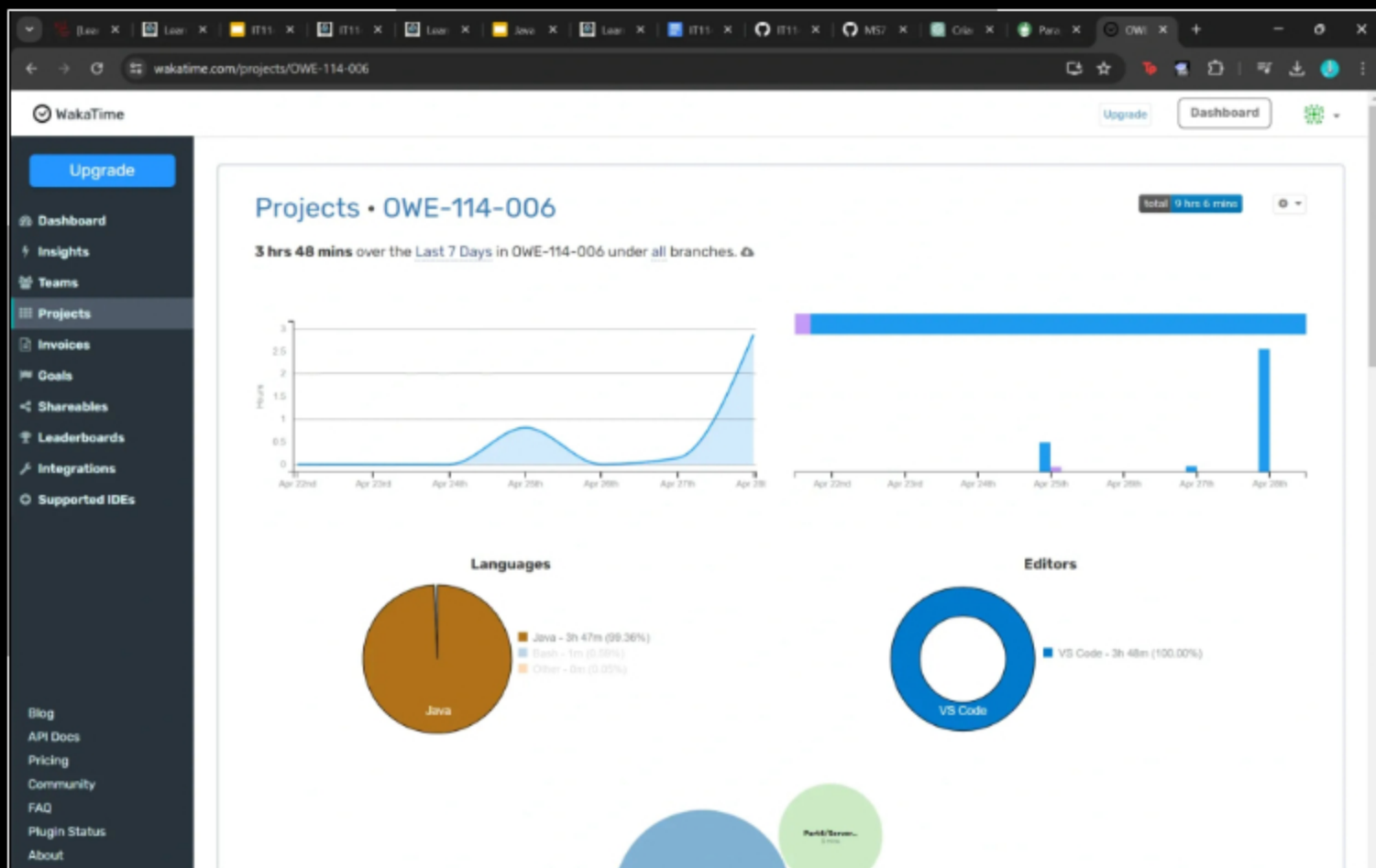
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



screen shot for the waka time

End of Assignment