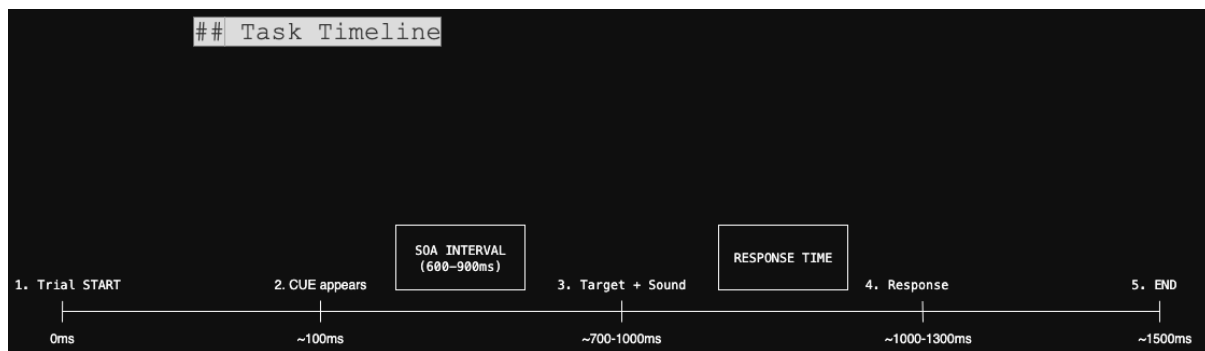


## ## Task Timeline



link to graph in google drive -> <https://shorturl.at/MuEMI>

### **\*\*What happens:\*\***

- **\*\*0ms\*\***: Trial starts (empty screen)
- **\*\*~100ms\*\***: CUE appears (arrows ←, →, or ⇕)
- **\*\*600-900ms\*\***: Target appears (left/right) + Sound (if bimodal)
- **\*\*~1200ms\*\***: User presses UP or DOWN button
- **\*\*END\*\***: Data saved

### **\*\*Cue Types:\*\***

- **\*\*Valid\*\***: ← points left, target appears left = Fast response
- **\*\*Invalid\*\***: → points right, target appears left = Slow response
- **\*\*Neutral\*\***: ⇕ no direction = Medium response

## ## CSV Columns

## ## Encodings Overview

**\*\*Modal\*\*** defines the sensory modality of stimulus presentation:

- **Unimodal (1)**: Visual-only stimulus
- **Bimodal (2)**: Visual + auditory stimulus (3500 Hz beep)

Half of trials are unimodal, half are bimodal. Determined by server during trial generation.

The screenshot shows a VS Code editor with a file explorer on the left and a Python file named 'api.py' on the right. The file explorer shows a project structure for 'HAPPY-AGAIN-BACKEND-NEW' with folders like 'bin', 'data', 'docs', and 'happy\_again'. The 'api.py' file is open in the editor, showing a function 'generate\_test()' that generates test trials. A red circle highlights the 'new\_trial' dictionary creation and the 'modal' assignment logic. The trial is generated based on a random number and the current counts of left, right, and up/down responses.

This line randomly shuffles the list of generated trials so their order is unpredictable when sent to the frontend. As a result, unimodal and bimodal trials are mixed rather than appearing in fixed blocks.

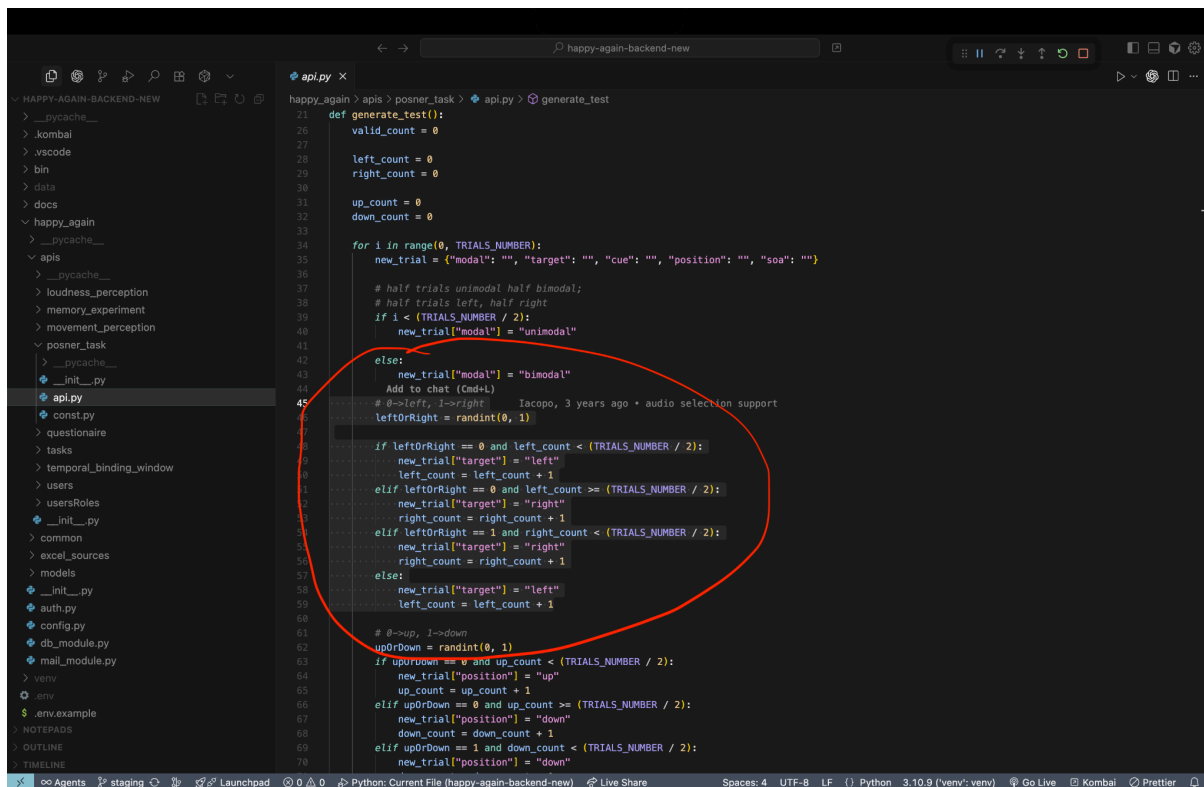
```
TS target-detection.component.ts TS admin-area.component.ts
src > app > views > target-detection > TS target-detection.component.ts > TS targetDetectionComponent > completeRound
227 export class TargetDetectionComponent implements OnInit, OnDestroy, AfterViewInit {
286   @ViewChild('audio') audio: HTMLAudioElement;
333   async completeRound() {
334     .subscribe(() => {
335       this.round = 0 //reset rounds:
341       this.correctAnswersCount = 0
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   }

```

**\*\*Target\*\*** defines the spatial location of the visual stimulus:

- **\*\*Left (0)\*\***: Stimulus appears on the left side of screen
- **\*\*Right (1)\*\***: Stimulus appears on the right side of screen

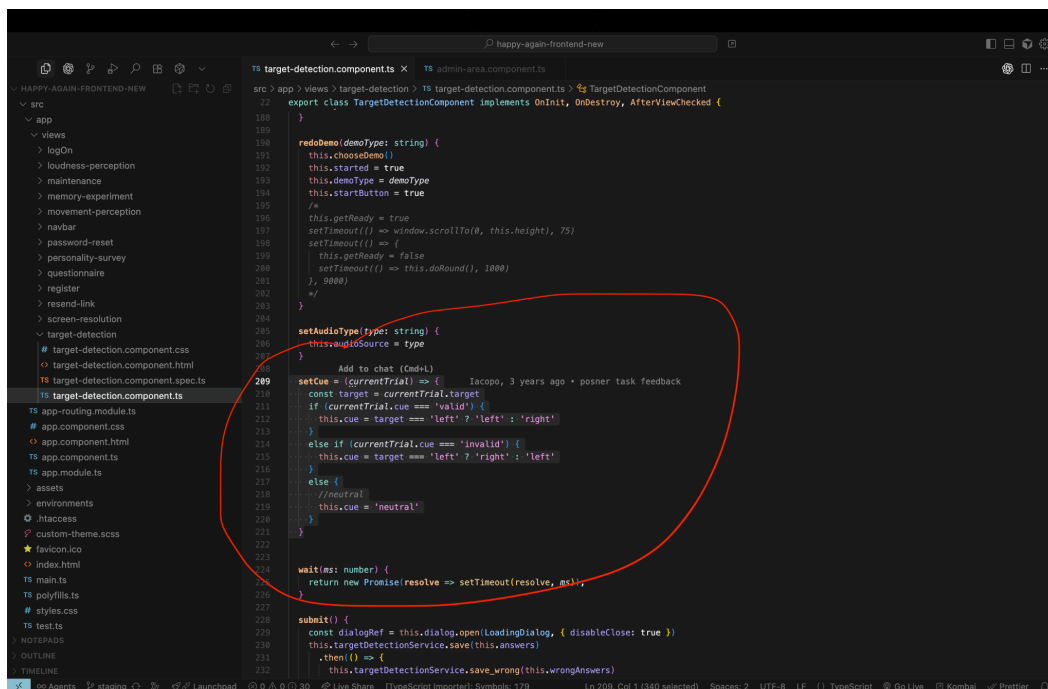
Half of trials have left target, half have right target. Determined by server during trial generation.



```
21 def generate_test():
22     valid_count = 0
23
24     left_count = 0
25     right_count = 0
26
27     up_count = 0
28     down_count = 0
29
30     for i in range(0, TRIALS_NUMBER):
31         new_trial = {"modal": "", "target": "", "cue": "", "position": "", "soa": ""}
32
33         # half trials unimodal half bimodal;
34         # half trials left, half right
35         if i < (TRIALS_NUMBER / 2):
36             new_trial["modal"] = "unimodal"
37
38         else:
39             new_trial["modal"] = "bimodal"
40
41         # 0->left, 1->right
42         leftOrRight = randint(0, 1)
43
44         if leftOrRight == 0 and left_count < (TRIALS_NUMBER / 2):
45             new_trial["target"] = "left"
46             left_count = left_count + 1
47
48         elif leftOrRight == 0 and left_count >= (TRIALS_NUMBER / 2):
49             new_trial["target"] = "right"
50             right_count = right_count + 1
51
52         elif leftOrRight == 1 and right_count < (TRIALS_NUMBER / 2):
53             new_trial["target"] = "right"
54             right_count = right_count + 1
55
56         else:
57             new_trial["target"] = "left"
58             left_count = left_count + 1
59
60         # 0->up, 1->down
61         upOrDown = randint(0, 1)
62
63         if upOrDown == 0 and up_count < (TRIALS_NUMBER / 2):
64             new_trial["position"] = "up"
65             up_count = up_count + 1
66
67         elif upOrDown == 0 and up_count >= (TRIALS_NUMBER / 2):
68             new_trial["position"] = "down"
69             down_count = down_count + 1
70
71         elif upOrDown == 1 and down_count < (TRIALS_NUMBER / 2):
72             new_trial["position"] = "down"
```

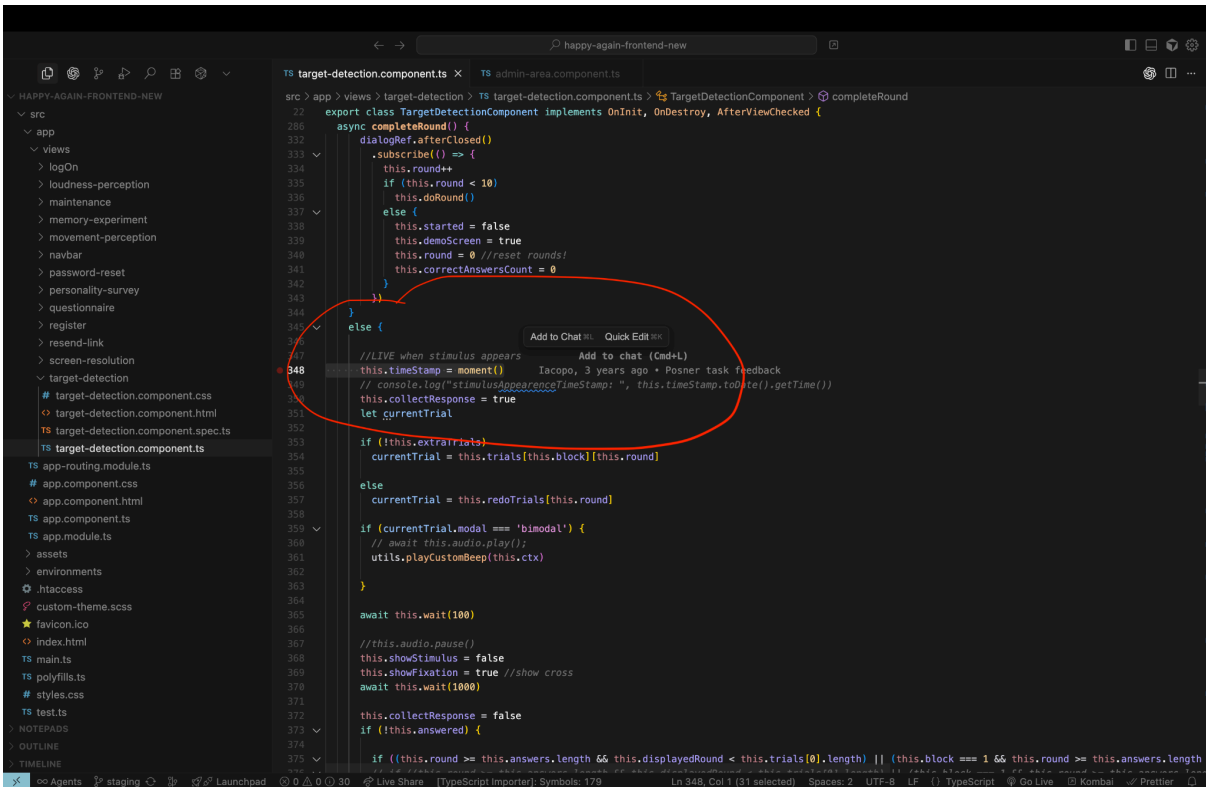
**How set:** the server randomly assigns left/right while enforcing equal counts; afterwards trials are shuffled, so left/right appear in random sequence.

**Purpose:** used to determine cue validity (valid/invalid/neutral) and analyze reaction times by side.



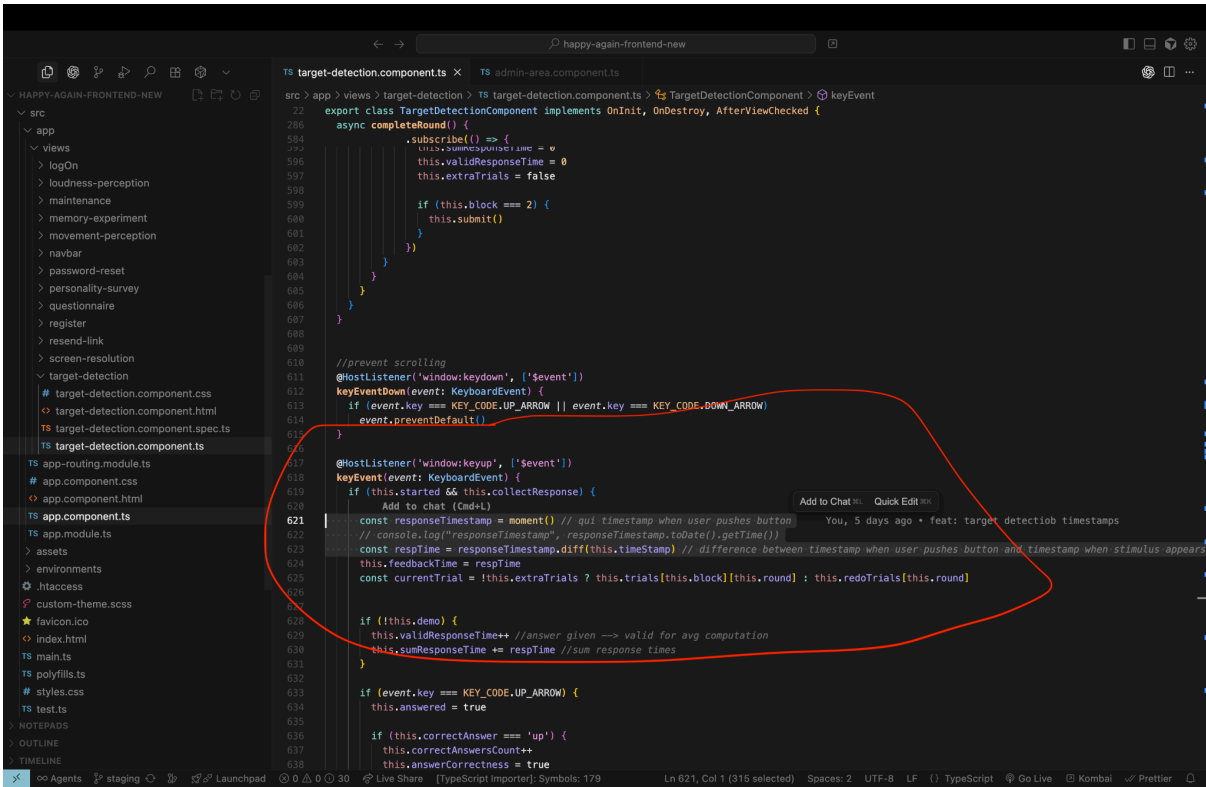
```
209 setAudioType(type: string) {
210     this.audioSource = type
211 }
212
213 // Add to chat (Cmd+L)
214 // Iacopo, 3 years ago • posner task feedback
215 setCue = (currentTrial) => {
216     const target = currentTrial.target
217     if (currentTrial.cue === 'valid') {
218         this.cue = target === 'left' ? 'left' : 'right'
219     }
220     else if (currentTrial.cue === 'invalid') {
221         this.cue = target === 'left' ? 'right' : 'left'
222     }
223     else {
224         //neutral
225         this.cue = 'neutral'
226     }
227 }
228
229 wait(ms: number) {
230     return new Promise(resolve => setTimeout(resolve, ms))
231 }
232
233 submit() {
234     const dialogRef = this.dialog.open(LoadingDialog, { disableClose: true })
235     this.targetDetectionService.save(this.answers)
236     .then(() => {
237         this.targetDetectionService.save_wrong(this.wrongAnswers)
238     })
239 }
```

**\*\*Response time\*\*** - time in milliseconds between stimulus appearance and user button press. Measures reaction speed and attention performance.



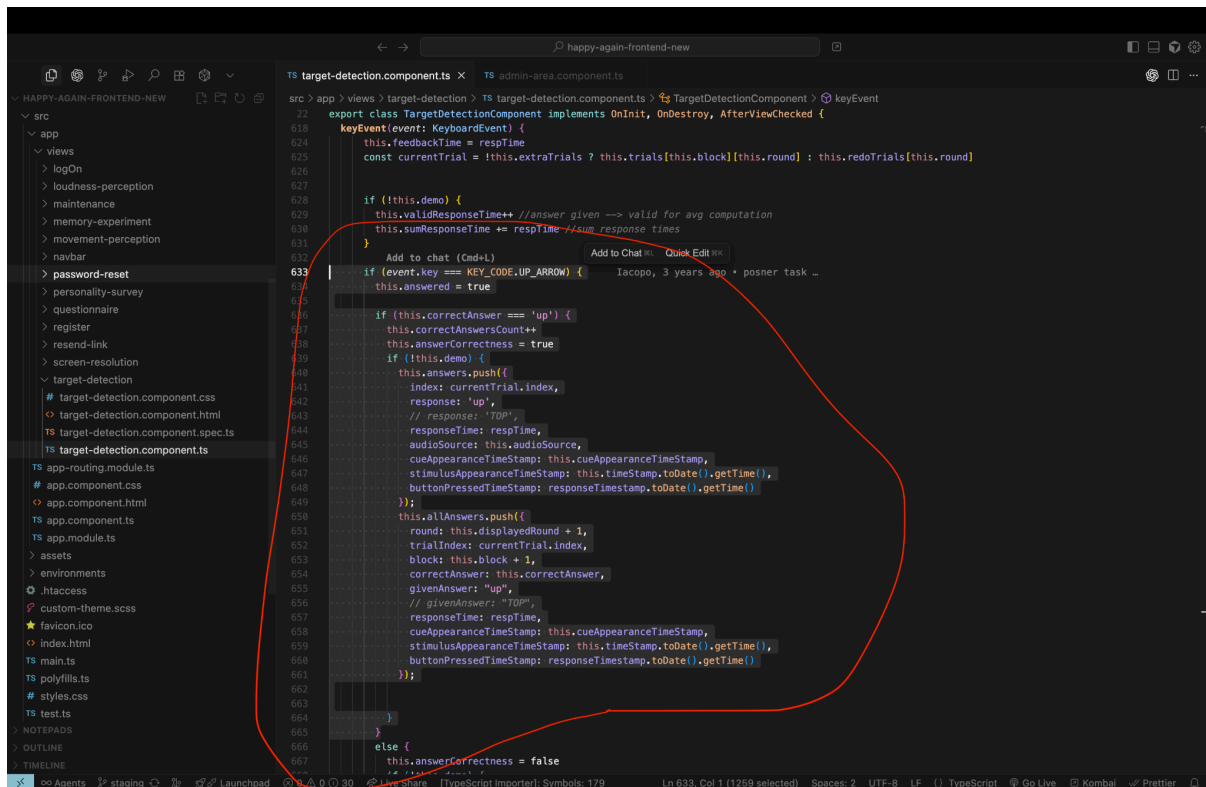
**\*\*Process:\*\***

1. **\*\*`this.timestamp`\*\*** - recorded when stimulus appears (line 348)
  2. **\*\*`responseTimestamp`\*\*** - recorded when user releases key (line 621)
  3. **\*\*`respTime`\*\*** - calculated difference in milliseconds
- \*\*Result:\*\*** Reaction time in milliseconds measuring attention performance.



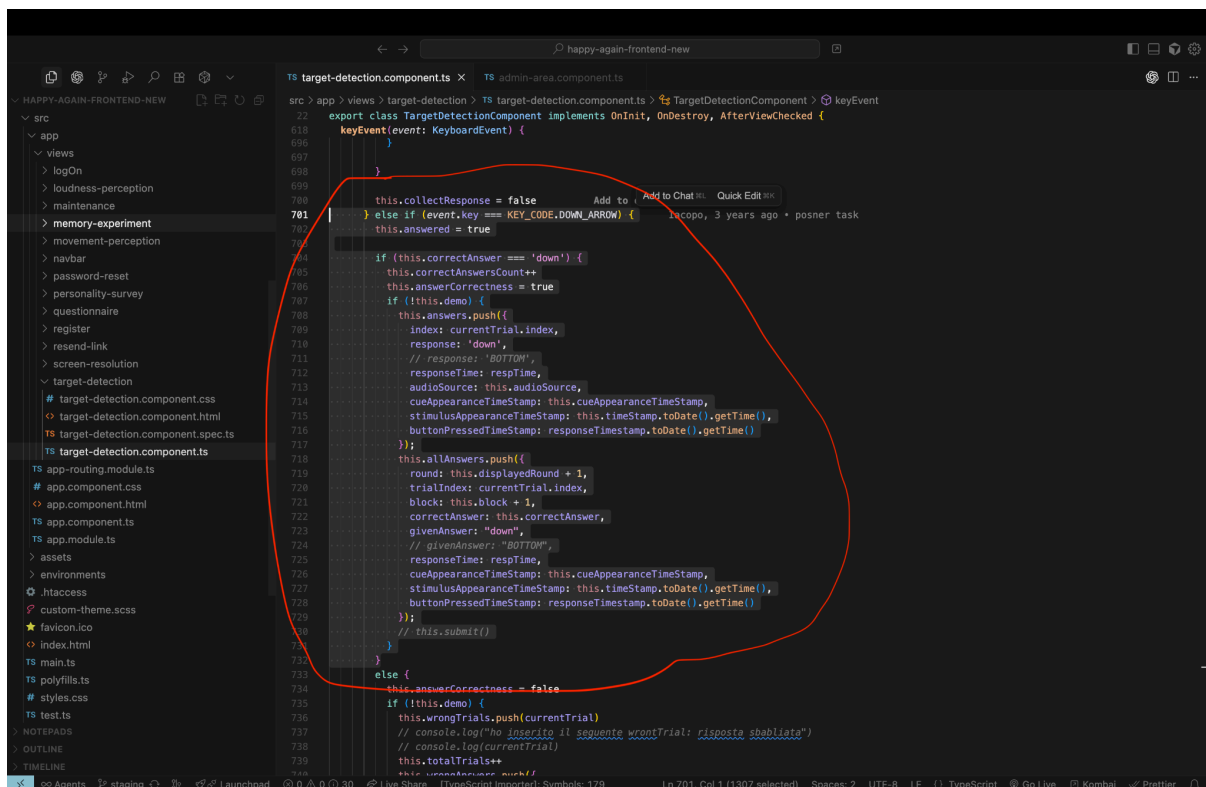
**\*\*Response\*\*** - user button press:

- Up (1): UP arrow key
- Down (0): DOWN arrow key



```
src > app > views > target-detection > TS target-detection.component.ts > TargetDetectionComponent > keyEvent
22 export class TargetDetectionComponent implements OnInit, OnDestroy, AfterViewInit {
618   keyEvent(event: KeyboardEvent) {
624     this.feedbackTime = respTime
625     const currentTrial = !this.extraTrials ? this.trials[this.block][this.round] : this.redoTrials[this.round]
626
627     if (!this.demo) {
628       this.validResponseTime++ //answer given => valid for avg computation
629       this.sumResponseTime += respTime //sum response times
630     }
631
632     Add to chat (Cmd+I) Add to Chat Quick Edit
633     if (event.key === KEY_CODE.UP_ARROW) {
634       this.answared = true
635
636       if (this.correctAnswer === 'up') {
637         this.correctAnswersCount++
638         this.answerCorrectness = true
639         if (!this.demo) {
640           this.answers.push({
641             index: currentTrial.index,
642             response: 'up',
643             // response: 'TOP',
644             responseTime: respTime,
645             audioSource: this.audioSource,
646             cueAppearanceTimeStamp: this.cueAppearanceTimeStamp,
647             stimulusAppearanceTimeStamp: this.timeStamp.toDate().getTime(),
648             buttonPressedTimeStamp: responseTimeStamp.toDate().getTime()
649           });
650           this.allAnswers.push({
651             round: this.displayedRound + 1,
652             trialIndex: currentTrial.index,
653             block: this.block + 1,
654             correctAnswer: this.correctAnswer,
655             givenAnswer: "up",
656             // givenAnswer: "TOP",
657             responseTime: respTime,
658             cueAppearanceTimeStamp: this.cueAppearanceTimeStamp,
659             stimulusAppearanceTimeStamp: this.timeStamp.toDate().getTime(),
660             buttonPressedTimeStamp: responseTimeStamp.toDate().getTime()
661           });
662         }
663       }
664     }
665     else {
666       this.answerCorrectness = false
667     }
668   }
669 }
```

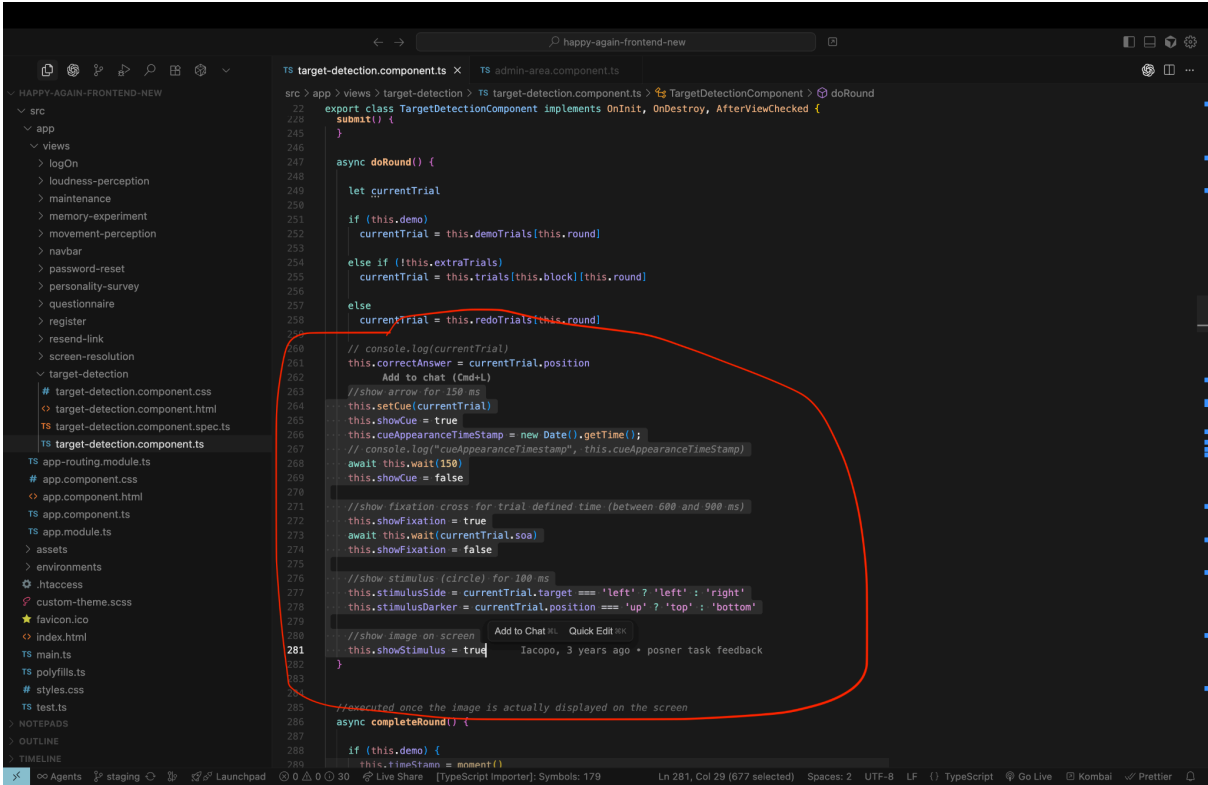
Process: Directly captures which arrow key the user pressed in response to the stimulus. Value comes from keyboard event and is stored in trial data.



```
src > app > views > target-detection > TS target-detection.component.ts > TargetDetectionComponent > keyEvent
22 export class TargetDetectionComponent implements OnInit, OnDestroy, AfterViewInit {
618   keyEvent(event: KeyboardEvent) {
624     this.feedbackTime = respTime
625     const currentTrial = !this.extraTrials ? this.trials[this.block][this.round] : this.redoTrials[this.round]
626
627     if (!this.demo) {
628       this.validResponseTime++ //answer given => valid for avg computation
629       this.sumResponseTime += respTime //sum response times
630     }
631
632     Add to chat Add to Chat Quick Edit
633     if (event.key === KEY_CODE.UP_ARROW) {
634       this.answared = true
635
636       if (this.correctAnswer === 'up') {
637         this.correctAnswersCount++
638         this.answerCorrectness = true
639         if (!this.demo) {
640           this.answers.push({
641             index: currentTrial.index,
642             response: 'up',
643             // response: 'TOP',
644             responseTime: respTime,
645             audioSource: this.audioSource,
646             cueAppearanceTimeStamp: this.cueAppearanceTimeStamp,
647             stimulusAppearanceTimeStamp: this.timeStamp.toDate().getTime(),
648             buttonPressedTimeStamp: responseTimeStamp.toDate().getTime()
649           });
650           this.allAnswers.push({
651             round: this.displayedRound + 1,
652             trialIndex: currentTrial.index,
653             block: this.block + 1,
654             correctAnswer: this.correctAnswer,
655             givenAnswer: "up",
656             // givenAnswer: "TOP",
657             responseTime: respTime,
658             cueAppearanceTimeStamp: this.cueAppearanceTimeStamp,
659             stimulusAppearanceTimeStamp: this.timeStamp.toDate().getTime(),
660             buttonPressedTimeStamp: responseTimeStamp.toDate().getTime()
661           });
662         }
663       }
664     }
665     else if (event.key === KEY_CODE.DOWN_ARROW) {
666       this.answared = true
667
668       if (this.correctAnswer === 'down') {
669         this.correctAnswersCount++
670         this.answerCorrectness = true
671         if (!this.demo) {
672           this.answers.push({
673             index: currentTrial.index,
674             response: 'down',
675             // response: 'BOTTOM',
676             responseTime: respTime,
677             audioSource: this.audioSource,
678             cueAppearanceTimeStamp: this.cueAppearanceTimeStamp,
679             stimulusAppearanceTimeStamp: this.timeStamp.toDate().getTime(),
680             buttonPressedTimeStamp: responseTimeStamp.toDate().getTime()
681           });
682           this.allAnswers.push({
683             round: this.displayedRound + 1,
684             trialIndex: currentTrial.index,
685             block: this.block + 1,
686             correctAnswer: this.correctAnswer,
687             givenAnswer: "down",
688             // givenAnswer: "BOTTOM",
689             responseTime: respTime,
690             cueAppearanceTimeStamp: this.cueAppearanceTimeStamp,
691             stimulusAppearanceTimeStamp: this.timeStamp.toDate().getTime(),
692             buttonPressedTimeStamp: responseTimeStamp.toDate().getTime()
693           });
694           // this.submit()
695         }
696       }
697     }
698     else {
699       this.answerCorrectness = false
700       if (!this.demo) {
701         this.wrongTrials.push(currentTrial)
702         // console.log("ho inserito il seguente wrongTrial: risposta sbagliata")
703         // console.log(currentTrial)
704         this.totalTrials++
705       }
706     }
707   }
708 }
```

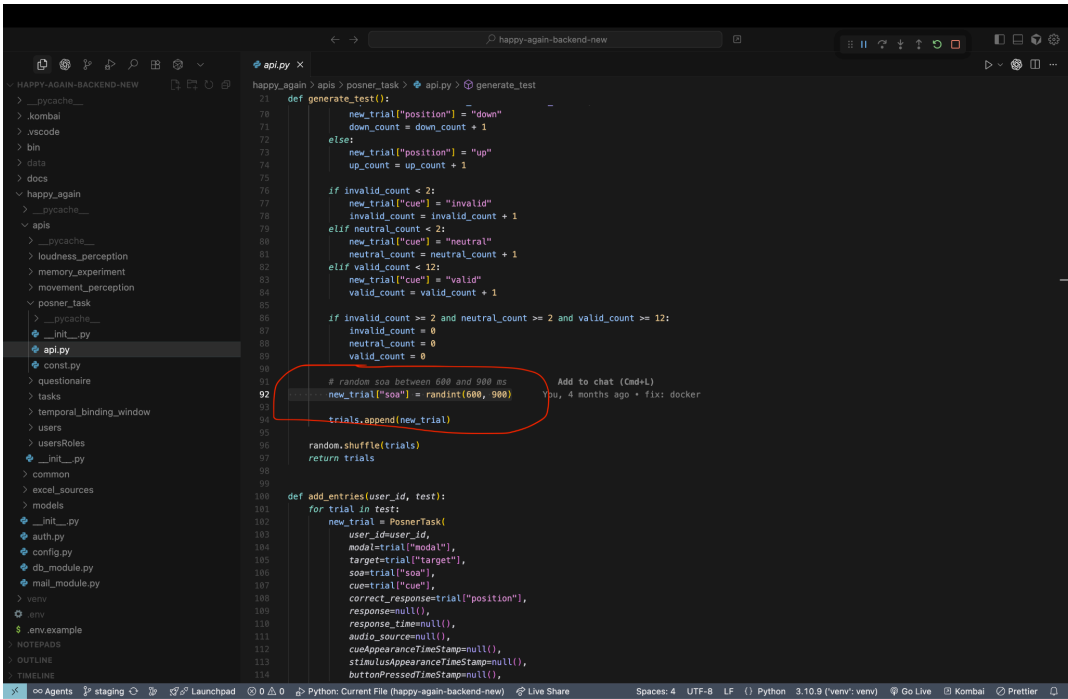
- **Range**: 600-900 milliseconds
- **Purpose**: Prevents timing predictability
- **Generated**: Randomly by server for each trial
- **Usage**: Controls fixation cross display duration

Variable timing ensures participants cannot predict stimulus appearance.



Process: Server generates random SOA (600-900ms) for each trial. Frontend uses this value to control how long the fixation cross is displayed before showing the stimulus.

Purpose: Prevents timing predictability, ensuring participants cannot anticipate stimulus appearance.



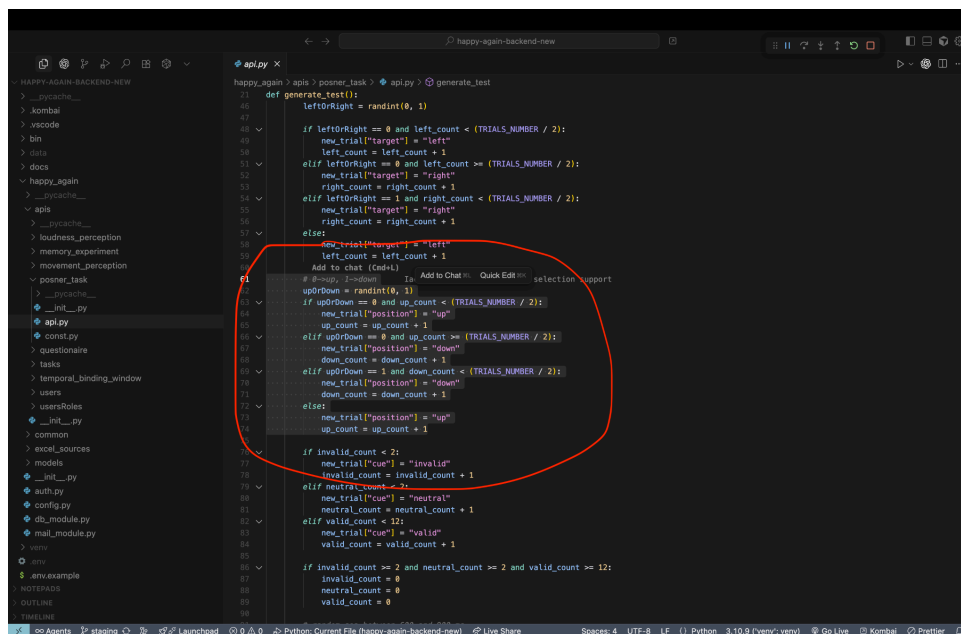
**\*\*Correct response\*\*** - expected answer based on stimulus position:

**Up (1):** Stimulus circle darker on top

**Down (0):** Stimulus circle darker on bottom

What it means: the expected key the participant should press on this trial - "up" or "down".

How it's set: generated on the server during test generation and balanced 50/50 across the session.



```
def generate_test():
    leftOrRight = randint(0, 1)

    if leftOrRight == 0 and left_count < (TRIALS_NUMBER / 2):
        new_trial["target"] = "left"
        left_count = left_count + 1
    elif leftOrRight == 0 and left_count >= (TRIALS_NUMBER / 2):
        new_trial["target"] = "right"
        right_count = right_count + 1
    elif leftOrRight == 1 and right_count < (TRIALS_NUMBER / 2):
        new_trial["target"] = "right"
        right_count = right_count + 1
    else:
        new_trial["target"] = "left"
        left_count = left_count + 1

    # 0=up, 1=down
    upOrDown = randint(0, 1)
    if upOrDown == 0 and up_count < (TRIALS_NUMBER / 2):
        new_trial["position"] = "up"
        up_count = up_count + 1
    elif upOrDown == 0 and up_count >= (TRIALS_NUMBER / 2):
        new_trial["position"] = "down"
        down_count = down_count + 1
    elif upOrDown == 1 and down_count < (TRIALS_NUMBER / 2):
        new_trial["position"] = "down"
        down_count = down_count + 1
    else:
        new_trial["position"] = "up"
        up_count = up_count + 1

    if invalid_count < 2:
        new_trial["cue"] = "invalid"
        invalid_count = invalid_count + 1
    elif neutral_count < 2:
        new_trial["cue"] = "neutral"
        neutral_count = neutral_count + 1
    elif valid_count < 12:
        new_trial["cue"] = "valid"
        valid_count = valid_count + 1

    if invalid_count >= 2 and neutral_count >= 2 and valid_count >= 12:
        invalid_count = 0
        neutral_count = 0
        valid_count = 0
```

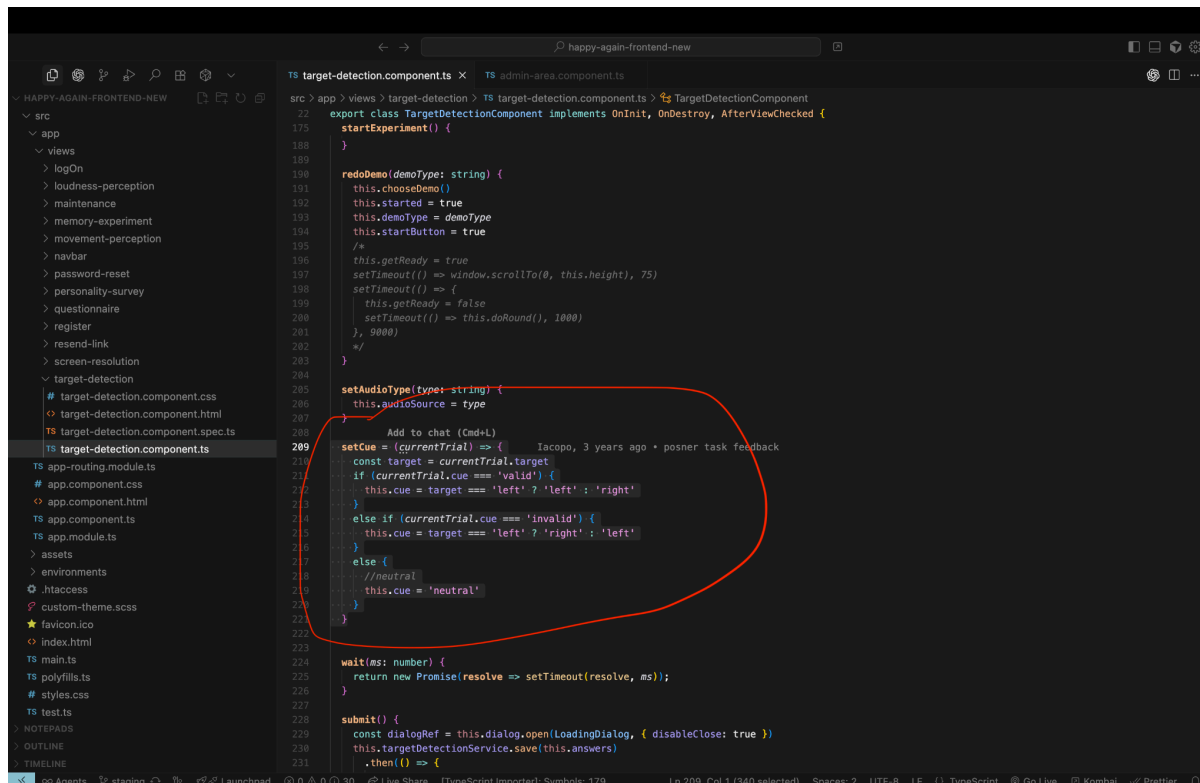
**\*\*Cue\*\*** - directional arrow hint before stimulus:

**Valid (0):** Arrow points toward stimulus location

**Invalid (1):** Arrow points away from stimulus location

**Neutral (2):** Both arrows shown (no directional hint)

**Purpose:** Tests attention and cue processing effects on reaction time.



```
export class TargetDetectionComponent implements OnInit, OnDestroy, AfterViewInit {
    startExperiment() {
    }

    redoDemo(demoType: string) {
        this.chooseDemo();
        this.started = true;
        this.demoType = demoType;
        this.startButton = true;
        /*
        this.getReady = true
        setTimeout(() => window.scrollTo(0, this.height), 75)
        setTimeout(() => {
            this.getReady = false
            setTimeout(() => this.doRound(), 1000)
        }, 9000)
        */
    }

    setAudioType(type: string) {
        this.audioSource = type
    }

    setCue = (currentTrial) => {
        const target = currentTrial.target
        if (currentTrial.cue === 'valid') {
            this.cue = target === 'left' ? 'left' : 'right'
        }
        else if (currentTrial.cue === 'invalid') {
            this.cue = target === 'left' ? 'right' : 'left'
        }
        else {
            //neutral
            this.cue = 'neutral'
        }
    }

    wait(ms: number) {
        return new Promise(resolve => setTimeout(resolve, ms));
    }

    submit() {
        const dialogRef = this.dialog.open(LoadingDialog, { disableClose: true })
        this.targetDetectionService.save(this.answers)
        .then(() => {

```

```
api.py x
happy-again-backend-new
> __pycache__
> .kombai
> .vscode
> bin
> data
> docs
> happy_again
> __pycache__
> apis
> __pycache__
> loudness_perception
> memory_experiment
> movement_perception
> posner_task
> __pycache__
> __init__.py
api.py
const.py
questionnaire
tasks
temporal_binding_window
users
usersRoles
__init__.py
common
excel_sources
models
__init__.py
auth.py
config.py
db_module.py
mail_module.py
venv
.env
.env.example
NOTEPADS
OUTLINE
TIMELINE

21 def generate_test():
22     elif leftOrRight == 1 and right_count < (TRIALS_NUMBER / 2):
23         new_trial["target"] = "right"
24         right_count = right_count + 1
25     else:
26         new_trial["target"] = "left"
27         left_count = left_count + 1
28
29     # 0=up, 1=down
30     upOrDown = randint(0, 1)
31     if upOrDown == 0 and up_count < (TRIALS_NUMBER / 2):
32         new_trial["position"] = "up"
33         up_count = up_count + 1
34     elif upOrDown == 0 and up_count >= (TRIALS_NUMBER / 2):
35         new_trial["position"] = "down"
36         down_count = down_count + 1
37     elif upOrDown == 1 and down_count < (TRIALS_NUMBER / 2):
38         new_trial["position"] = "down"
39         down_count = down_count + 1
40     else:
41         new_trial["position"] = "up"
42         up_count = up_count + 1
43
44     Add to chat (Cmd+I)
45     if invalid_count < 2:
46         new_trial["cue"] = "invalid"
47         invalid_count = invalid_count + 1
48     elif neutral_count < 2:
49         new_trial["cue"] = "neutral"
50         neutral_count = neutral_count + 1
51     elif valid_count < 12:
52         new_trial["cue"] = "valid"
53         valid_count = valid_count + 1
54
55     if invalid_count >= 2 and neutral_count >= 2 and valid_count >= 12:
56         invalid_count = 0
57         neutral_count = 0
58         valid_count = 0
59
60     # random soa between 600 and 900 ms
61     new_trial["soa"] = randint(600, 900)
62
63     trials.append(new_trial)
64
65     random.shuffle(trials)
66     return trials
67
68 Python: Current File (happy-again-backend-new) Live Share
Spaces: 4 UTF-8 LF () Python 3.10.9 (venv: venv) Go Live Kombai Prettier
```

**\*\*Audio source\*\*** - user's audio device choice:

**Speakers (0):** External speakers selected

**Headphones (1):** Headphones selected

Selected by user before experiment starts via radio buttons.

```
TS target-detection.component.ts TS admin-area.component.ts x
happy-again-frontend-new
> .idea
> .kombai
> dist
> e2e
> node_modules
> src
> app
> service
> utils
TS utils.ts
views
admin-area
# admin-area.component.css
admin-area.component.html
TS admin-area.component.ts
captcha
dialogs
experimentData
flash-beep
home
info
invite-register
logOn
loudness-perception
maintenance
memory-experiment
movement-perception
navbar
password-reset
personality-survey
questionnaire
register
resend-link
screen-resolution
target-detection
NOTEPADS
OUTLINE
TIMELINE

19 export class AdminAreaComponent implements OnInit {
20 }
21
22 formatTargetDetectionTimestamps(data: any[]): any[] {
23     const encodeUpDown = (val: any) => {
24         if (val === null || val === undefined) return val;
25         if (typeof val === 'number') return val; // assume already encoded 0/1
26         const str = String(val).toLowerCase();
27         if (str === 'up') return 1;
28         if (str === 'down') return 0;
29         if (str === '1') return 1;
30         if (str === '0') return 0;
31         return val;
32     };
33     const encodeCue = (val: any) => {
34         if (val === null || val === undefined) return val;
35         if (typeof val === 'number') return val; // already encoded 0/1/2
36         const str = String(val).toLowerCase();
37         if (str === 'valid') return 0;
38         if (str === 'invalid') return 1;
39         if (str === 'neutral') return 2;
40         return val;
41     };
42     // Add to chat (Cmd+I)
43     const encodeAudioSource = (val: any) => {
44         if (val === null || val === undefined) return val;
45         if (typeof val === 'number') return val; // already encoded 0/1
46         const str = String(val).toLowerCase();
47         if (str === 'speakers') return 0;
48         if (str === 'headphones') return 1;
49         return val;
50     };
51     const encodeModal = (val: any) => {
52         if (val === null || val === undefined) return val;
53         if (typeof val === 'number') return val; // already encoded 1/2
54         const str = String(val).toLowerCase();
55         if (str === 'unimodal') return 1;
56         if (str === 'bimodal') return 2;
57         return val;
58     };
59     const encodeTarget = (val: any) => {
60         if (val === null || val === undefined) return val;
61         if (typeof val === 'number') return val; // already encoded 0/1
62         const str = String(val).toLowerCase();
63         if (str === 'left') return 0;
64         if (str === 'right') return 1;
65         return val;
66     };
67 }
68
69 TypeScript Importer: Symbols: 179 Ln 3307, Col 1 (324 selected) Spaces: 2 UTF-8 LF () TypeScript Go Live Kombai Prettier
```



**\*\*Cue appearance timestamp\*\*** - Exact moment when directional cue (arrow) appears on screen

**Format:** Unix timestamp in milliseconds

**Recorded:** Immediately after cue display starts (line 266)

**Purpose:** Measures cue presentation timing for SOA calculations

**Captured** `1703123456789` → 12/21/2023, 14:30:56

```
TS target-detection.component.ts X
src > app > views > target-detection > TS target-detection.component.ts > TargetDetectionComponent > doRound
22 export class TargetDetectionComponent implements OnInit, OnDestroy, AfterViewChecked {
229 submit() {
244   }
245 }
246
247 async doRound() {
248   let currentTrial
249   if (this.demo)
250     currentTrial = this.demoTrials[this.round]
251   else if (!this.extraTrials)
252     currentTrial = this.trials[this.block][this.round]
253   else
254     currentTrial = this.redoTrials[this.round]
255   // console.log(currentTrial)
256   this.correctAnswer = currentTrial.position
257   //show arrow for 150 ms
258   this.setCue(currentTrial)
259   this.showCue = true
260   this.cueAppearanceTimestamp = new Date().getTime()
261   // console.log("cueAppearanceTimestamp", this.cueAppearanceTimestamp)
262   await this.wait(150)
263   this.showCue = false
264   //show fixation cross for trial defined time (between 600 and 900 ms)
265   this.showFixation = true
266   await this.wait(currentTrial.soa)
267   this.showFixation = false
268   //show stimulus (circle) for 100 ms
269   this.stimulusSide = currentTrial.target === 'left' ? 'left' : 'right'
270   this.stimulusBarker = currentTrial.position === 'up' ? 'top' : 'bottom'
271   //show image on screen
272   this.showStimulus = true
273 }
274 //executed once the image is actually displayed on the screen
275 async completeRound() {
276 }
```

**## \*\*Stimulus Appearance Timestamp\*\***

**\*\*Records:\*\*** Exact moment when target stimulus (arrow) appears on screen

**\*\*Format:\*\*** Unix timestamp in milliseconds

**\*\*When:\*\*** Immediately after stimulus display starts (lines 289, 348)

**\*\*Purpose:\*\*** Marks stimulus onset for response time calculations

**\*\*Example:\*\*** `1703123456789` → 12/21/2023, 14:30:56

```
TS target-detection.component.ts X
src > app > views > target-detection > TS target-detection.component.ts > TargetDetectionComponent > completeRound
22 export class TargetDetectionComponent implements OnInit, OnDestroy, AfterViewChecked {
229 async completeRound() {
230   //close feedback after 1700 ms
231   this.timeOut = setTimeout(() => dialogRef.close(), 1700)
232
233   dialogRef.afterClosed()
234   .subscribe() => {
235     this.round++
236     if (this.round < 10)
237       this.doRound()
238     else {
239       this.started = false
240       this.demoScreen = true
241       this.round = 0 //reset rounds!
242       this.correctAnswersCount = 0
243     }
244   }
245   //LIVE when stimulus appears
246   this.timeStamp = moment()
247   // console.log("stimulusAppearanceTimestamp", this.timeStamp.toDate().getTime())
248   this.collectResponse = true
249   let currentTrial
250   if (!this.extraTrials)
251     currentTrial = this.trials[this.block][this.round]
252   else
253     currentTrial = this.redoTrials[this.round]
254   if (currentTrial.modal === 'bimodal') {
255     // await this.audio.play()
256     utils.playCustomBeep(this.ctx)
257   }
258   await this.wait(100)
259   //this.audio.pause()
260   this.showStimulus = false
261   this.showFixation = true //show cross
262   await this.wait(1000)
263   this.collectResponse = false
264 }
```

## ## \*\*Button Pressed Timestamp\*\*

**\*\*Records:\*\*** Exact moment when user presses keyboard key (↑ or ↓)

**\*\*Format:\*\*** Unix timestamp in milliseconds

**\*\*When:\*\*** Immediately when key is pressed during response collection (line 621)

**\*\*Purpose:\*\*** Measures response timing for reaction time analysis

**\*\*Example:\*\*** `1703123456789` → 12/21/2023, 14:30:56

```
target-detection.component.ts X
src > app > views > target-detection > ts target-detection.component.ts > TargetDetectionComponent
22 export class TargetDetectionComponent implements OnInit, OnDestroy, AfterViewInit {
23   async completeRound() {
24     // ...
25   }
26 }
27
28 //prevent scrolling
29 @HostListener('window:keydown', ['$event'])
30 keyEvent(event: KeyboardEvent) {
31   if (event.key === KEY_CODE.UP_ARROW || event.key === KEY_CODE.DOWN_ARROW)
32     event.preventDefault()
33 }
34
35 Add to chat (Ctrl+)
36 @HostListener('window:keyup', ['$event'])
37 keyEvent(event: KeyboardEvent) {
38   if (this.started && this.collectResponse) {
39     const responseTimestamp = moment() // qui timestamp when user pushes button
40     // console.log('responseTimestamp', responseTimestamp.toDate().getTime())
41     const respTime = responseTimestamp.diff(this.timeStamp) // difference between timestamp when user pushes button and timestamp when stimulus appears
42     this.feedbackTime = respTime
43     const currentTrial = this.extraTrials ? this.trials[this.block][this.round] : this.redoTrials[this.round]
44
45     if (this.demo) {
46       this.validResponseTime += answer given → valid for avg computation
47       this.sumResponseTime += respTime //sum response times
48     }
49
50     if (event.key === KEY_CODE.UP_ARROW) {
51       this.answared = true
52
53       if (this.correctAnswer === 'up') {
54         this.correctAnswersCount++
55         this.answerCorrectness = true
56         if (this.demo) {
57           this.answers.push({
58             index: currentTrial.index,
59             response: 'up',
60             // response: 'TOP',
61             responseTime: respTime,
62             audioSource: this.audioSource,
63           })
64         }
65       }
66     }
67   }
68 }
69
70 In 617, Col 7 (443 selected). Spaces: 2 UTF-8 LF (1) TypeScript Go Live Kombat Prettier
```

## ## \*\*interTrialTime\*\*

**\*\*Records:\*\*** Time gap between consecutive trials for the same user.

**\*\*Format:\*\*** Milliseconds (number).

**\*\*When:\*\*** For each record after the first per user; first record per user is 0.

**\*\*Formula:\*\*** current cueAppearanceTimeStamp – previous cueAppearanceTimeStamp

```
ts admin-area.components.ts X
src > app > views > admin-area > ts admin-area.components.ts > AdminAreaComponent > formatTargetDetectionTimestamps > interTrialByIndex
13 export class AdminAreaComponent implements OnInit {
14   // ...
15 }
16
17 formatTargetDetectionTimestamps(data: any[] | any[]) {
18   // Compute inter-trial time per user; difference between consecutive cueAppearanceTimeStamp
19   const interTrialByIndex: Record-number, number = {} // You, 2 hours ago • feat: interTrialTime
20   const kristinaByIndex: Record-number, number = {}
21   try {
22     const sorted = [...data].sort((a, b) => {
23       if (a.user_id === b.user_id) {
24         const at = a.cueAppearanceTimeStamp ?? 0;
25         const bt = b.cueAppearanceTimeStamp ?? 0;
26         return at - bt;
27       }
28       return String(a.user_id).localeCompare(String(b.user_id));
29     });
30     let lastUser: any = null;
31     let prevCueTs: number | null = null;
32     let prevSOA: number | null = null;
33     let prevRT: number | null = null;
34     for (const item of sorted) {
35       const user = item.user_id;
36       const cueTs: number | null =
37         item && typeof item.cueAppearanceTimeStamp === 'number'
38         ? item.cueAppearanceTimeStamp
39         : (item && item.cueAppearanceTimeStamp != null && !isNaN(Number(item.cueAppearanceTimeStamp))
40           ? Number(item.cueAppearanceTimeStamp)
41           : null);
42       if (user !== lastUser) {
43         // First record for this user
44         if (typeof item.index === 'number') {
45           interTrialByIndex[item.index] = 0;
46           kristinaByIndex[item.index] = 0;
47         }
48         lastUser = user;
49         prevCueTs = cueTs;
50         prevSOA = Number(item7.soa);
51         prevRT = Number(item7.response_time);
52         continue;
53       }
54       if (typeof item.index === 'number') {
55         if (prevCueTs != null && cueTs != null) {
56           const inter = cueTs - prevCueTs;
57           interTrialByIndex[item.index] = inter;
58           const soaVal = prevSOA != null && !isNaN(prevSOA) ? prevSOA : 0;
59           const rVal = prevRT != null && !isNaN(prevRT) ? prevRT : 0;
60           kristinaByIndex[item.index] = inter - soaVal - rVal;
61         }
62       }
63     }
64   } catch {
65     // ...
66   }
67 }
68
69 In 9289, Col 11 Spaces: 2 UTF-8 LF (1) TypeScript Go Live Kombat Prettier
```

## ## \*\*kristina\*\*

**\*\*Records:\*\*** Inter-trial time adjusted by previous trial's SOA and response\_time.

**\*\*Format:\*\*** Milliseconds (number).

**\*\*When:\*\*** For each record after the first per user; first record per user is 0.

**\*\*Formula:\*\*** interTrialTime(current) – SOA(previous) – response\_time(previous).

```

src > app > views > admin-area > TS admin-area.components.ts > AdminAreaComponent > formatTargetDetectionTimestamps
131 export class AdminAreaComponent implements OnInit {
132   formatTargetDetectionTimestamps(data: any[]): any[] {
133     const cueTs: number | null =
134       ? Number(item.cueAppearanceTimeStamp)
135       : null;
136     if (user != lastUser) {
137       // First record for this user
138       if (typeof item.index === 'number') {
139         interTrialByIndex[item.index] = 0;
140         kristinaByIndex[item.index] = 0;
141       }
142       lastUser = user;
143       prevCueTs = cueTs;
144       prevSOA = Number(item7.soa);
145       prevRT = Number(item7.response_time);
146       continue;
147     }
148     // Add to chat (Cmd+L)
149     if (typeof item.index === 'number') {
150       if (prevCueTs != null && cueTs != null) {
151         const inter = cueTs - prevCueTs;
152         interTrialByIndex[item.index] = inter;
153         const soaVal = prevSOA != null && !isNaN(prevSOA) ? prevSOA : 0;
154         const rtVal = prevRT != null && !isNaN(prevRT) ? prevRT : 0;
155         kristinaByIndex[item.index] = inter - soaVal - rtVal;
156       } else {
157         interTrialByIndex[item.index] = 0;
158         kristinaByIndex[item.index] = 0;
159       }
160     }
161     if (cueTs != null) prevCueTs = cueTs;
162     // Update prev SOA and RT for next record of same user
163     prevSOA = Number(item7.soa);
164     prevRT = Number(item7.response_time);
165   }
166   catch (_) {
167     // In case of unexpected shapes, fall back to zeros
168     for (const item of data) {
169       if (typeof item.index === 'number') {
170         interTrialByIndex[item.index] = 0;
171         kristinaByIndex[item.index] = 0;
172       }
173     }
174   }
175   const encodeUpDown = (val: any) => {
176     if (val === null || val === undefined) return val;
177     if (typeof val === 'number') return val; // assume already encoded 0/1
178   }

```

## \*\*round\*\*

**\*\*Meaning:\*\*** Trial attempt number

**\*\*Example:\*\*** 3, 5, 7, 9, 25, 43, 50

**\*\*Logic:\*\*** Tracks how many times user tried to complete the task. Each new session or restart increases the round number.

```

src > app > views > target-detection > TS target-detection.components.ts > TargetDetectionComponent > keyEvent
718 keyEvent(event: KeyboardEvent) {
719   correctAnswer = this.correctAnswer;
720   givenAnswer = "down";
721   // correctAnswer: "BOTTON"
722   responseTime: respTime;
723   cueAppearanceTimeStamp: this.cueAppearanceTimeStamp,
724   stimulusAppearanceTimeStamp: this.timeStamp.toDate().getTime(),
725   buttonPressedTimeStamp: responseTimeStamp.toDate().getTime()
726 }
727 // this.submit()
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 }

```

## \*\*trial\_index\*\*

**\*\*Meaning:\*\*** Original trial ID from main posner\_task table

**\*\*Example:\*\*** 1027, 1029, 1031, 1033

**\*\*Logic:\*\*** Links wrong answer back to the original trial in target\_detection\_correct.csv. When user gives wrong answer, system saves the index of that specific trial.

```

happy_again > apis > posner_task > api.py > add_entries
97     return trials
98
99
100
101 def add_entries(user_id, test):
102     for trial in test:
103         new_trial = PosnerTask(
104             user_id=user_id,
105             modal=trial["modal"],
106             target=trial["target"],
107             soa=trial["soa"],
108             cue=trial["cue"],
109             correct_response=trial["position"],
110             response=None,
111             response_time=None,
112             audio_source=None,
113             cueAppearanceTimestamp=None,
114             stimulusAppearanceTimestamp=None,
115             buttonPressedTimestamp=None,
116         )
117         db.session.add(new_trial)
118         db.session.commit()
119
120     # need this to return the index of trials that is available only after insertion
121     test_index = {
122         db.session.query(PosnerTask).filter_by(user_id=user_id, response=None).all()
123     }
124
125     returnable_test = []
126     for trial in test_index:
127         new_trial = {
128             "index": trial.index,
129             "modal": trial.modal,
130             "target": trial.target,
131             "cue": trial.cue,
132             "position": trial.correct_response,
133             "soa": trial.soa,
134         }
135         returnable_test.append(new_trial)
136
137     return returnable_test
138
139
140 # checks if there is already a generated test for user, that has NOT been answered
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
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

```

## \*\*block\*\*

**Meaning:** Trial block number

**Example:** 1

**Logic:** Groups trials into blocks for analysis. Most records show value 1, meaning all trials belong to one block. Used for performance analysis by blocks, 1 block = 128 trials

```

TS admin-area.component.ts TS target-detection.component.ts
src > app > views > target-detection > TS target-detection.component.ts > TargetDetectionComponent > completeRound
72 export class TargetDetectionComponent implements OnInit, OnDestroy, AfterViewInit {
73     async completeRound() {
74         const dialogRef = this.dialog.open(FeedbackCountDialog, {
75             data: {
76                 correctAnswersCount: this.correctAnswersCount, //this.totalTrials,
77                 totalAnswersCount: this.totalTrials,
78             },
79             disableClose: true
80         });
81
82         dialogRef.afterClosed().subscribe() => {
83
84             this.started = false
85             this.round = 0 //reset rounds!
86             this.displayedRound = 0
87             this.totalTrials = 128
88             // this.totalTrials = 5 //remove test only
89
90             this.block++ //move to next block
91             this.correctAnswersCount = 0
92             this.sumResponseTime = 0
93             this.validResponseTime = 0
94             this.extraTrials = false
95
96             if (this.block === 2) {
97                 this.submit()
98             }
99         })
100     }
101
102     //prevent scrolling
103     @HostListener('window:keydown', ['$event'])
104     keyEventDown(event: KeyboardEvent) {
105         if (event.key === KEY_CODE.UP_ARROW || event.key === KEY_CODE.DOWN_ARROW)
106             event.preventDefault()
107     }
108
109     @HostListener('window:keyup', ['$event'])
110     keyEventUp(event: KeyboardEvent) {
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
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

```