

# Advanced OOP Project Report

## Real-Time Multiplayer Jet Game using Spring Boot

**Md Ali Maruf** (112410221)  
**Maimun Hossain** (112410242)  
**Jannatul Ferdous** (112410357)  
**Sabbir Hossain** (112310210)

*Department of Computer Science*

January 27, 2026

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>System Overview and Functionality</b>	<b>3</b>
2.1	Game Features and Functionality . . . . .	3
2.2	System Architecture . . . . .	3
<b>3</b>	<b>Implementation Details</b>	<b>4</b>
3.1	WebSocket Configuration . . . . .	4
3.2	Game Controller and Thread Management . . . . .	4
3.3	Domain Models (Encapsulation) . . . . .	5
3.4	Data Persistence . . . . .	5
3.5	Core Logic: Game Service . . . . .	6
<b>4</b>	<b>OOP Concepts Analysis</b>	<b>7</b>
4.1	Encapsulation . . . . .	7
4.2	Single Responsibility Principle (SRP) . . . . .	7
4.3	Dependency Injection (DI) . . . . .	7
4.4	Concurrency . . . . .	7
<b>5</b>	<b>Conclusion</b>	<b>8</b>

# 1 Introduction

This report details the design and implementation of a real-time multiplayer 2D shooter game ("JetGame"). The backend is built using **Java Spring Boot**, leveraging **WebSockets** for low-latency communication and **JPA/Hibernate** for persistent data storage.

The primary objective of this project is to demonstrate **Advanced Object-Oriented Programming (OOP)** concepts in a practical, distributed system scenario. The system handles game physics, collision detection, and state synchronization on the server side to prevent cheating and ensure consistency.

## 2 System Overview and Functionality

### 2.1 Game Features and Functionality

The "JetGame" is designed to provide a seamless competitive experience for two players. The core functional requirements and gameplay features include:

- **Real-Time Multiplayer Synchronization:** The system supports two concurrent players connected via WebSockets. Player movements and actions are synchronized in real-time (approx. 60 updates per second) to ensure a lag-free experience.
- **Combat Mechanics:** Players can fire projectiles (bullets) that travel across the game canvas. The server calculates trajectory, speed, and positioning to maintain consistency across all clients.
- **Collision Detection System:** A server-side physics engine detects intersections between entities (Bullet-to-Player). Upon collision, the system registers a "hit," updates health/score, and removes the projectile from the game state.
- **Score Tracking and Leaderboard:** The game tracks successful hits in real-time. Upon game completion (time expiry or health depletion), the final scores are persisted to a PostgreSQL database and displayed on a global leaderboard.
- **Session Management:** The application manages game sessions, handling player connection ('/join'), game start triggers, and automated game termination logic.

### 2.2 System Architecture

The application follows a standard **Model-View-Controller (MVC)** architecture, enhanced with a Service layer for business logic.

- **Configuration Layer:** Handles WebSocket message broker setup and endpoint registration.
- **Controller Layer:** Manages incoming HTTP requests (REST API) and WebSocket messages (STOMP protocol).
- **Service Layer:** Contains the core game loop, physics engine, and collision logic.
- **Model Layer:** Represents game entities (Players, Bullets) and database entities (Leaderboard).
- **Persistence Layer:** Uses Spring Data JPA repositories for database interactions.

### 3 Implementation Details

#### 3.1 WebSocket Configuration

To enable real-time bidirectional communication, we configure the WebSocket endpoint and message broker. We use the STOMP protocol over SockJS.

```
1 @Configuration
2 @EnableWebSocketMessageBroker
3 public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
4
5     @Override
6     public void configureMessageBroker(MessageBrokerRegistry config) {
7         config.enableSimpleBroker("/topic");
8         config.setApplicationDestinationPrefixes("/app");
9     }
10
11    @Override
12    public void registerStompEndpoints(StompEndpointRegistry registry) {
13        registry.addEndpoint("/jet-game").withSockJS();
14    }
15}
```

Listing 1: WebSocketConfig.java

**Explanation:** This class decouples the connection logic from the business logic. By implementing the `WebSocketMessageBrokerConfigurer` interface, we override specific methods to define our messaging protocol, adhering to the Open/Closed Principle.

#### 3.2 Game Controller and Thread Management

The Controller handles player actions and manages the game loop using a separate thread executor.

```
1 @Controller
2 public class GameController {
3
4     private final GameService gameService;
5     private final SimpMessagingTemplate messagingTemplate;
6     private ScheduledExecutorService executorService;
7
8     @Autowired
9     public GameController(GameService gameService, SimpMessagingTemplate
10                         messagingTemplate) {
11         this.gameService = gameService;
12         this.messagingTemplate = messagingTemplate;
13     }
14
15     @MessageMapping("/join")
16     public void join(String playerName) {
17         gameService.addPlayer(playerName);
18         if (gameService.getGameState().getPlayers().size() == 2) {
19             startGameLoop();
20             messagingTemplate.convertAndSend("/topic/game", new GameMessage(
21                 MessageType.GAME_START, gameService.getGameState()));
22         }
23     }
24
25     private void startGameLoop() {
```

```

24     executorService = Executors.newSingleThreadScheduledExecutor();
25     executorService.scheduleAtFixedRate(() -> {
26         if (gameService.isGameOver()) {
27             messagingTemplate.convertAndSend("/topic/game", new
28                 GameMessage(MessageType.GAME_OVER, gameService.
29                     getGameState()));
30             executorService.shutdown();
31         } else {
32             gameService.update();
33             messagingTemplate.convertAndSend("/topic/game", new
34                 GameMessage(MessageType.SCORE_UPDATE, gameService.
35                     getGameState()));
36         }
37     }, 0, 16, TimeUnit.MILLISECONDS);
38 }

```

Listing 2: GameController.java (Snippet)

**Explanation:** The controller acts as an orchestrator. It uses Dependency Injection (via the constructor) to access the `GameService`. The `startGameLoop` method utilizes a `ScheduledExecutorService` to run the game physics at approximately 60 frames per second (every 16ms), ensuring the main server thread remains unblocked.

### 3.3 Domain Models (Encapsulation)

The `Bullet` class demonstrates encapsulation. All properties are private and accessed via public getters/setters.

```

1  public class Bullet {
2      private double x;
3      private double y;
4      private double speed;
5      private double width;
6      private double height;
7      private String shooterName;
8
9      public Bullet(double x, double y, double speed, double width, String
10         shooterName) {
11         this.x = x;
12         this.y = y;
13         this.speed = speed;
14         this.width = width;
15         this.height = 5;
16         this.shooterName = shooterName;
17     }

```

Listing 3: Bullet.java

### 3.4 Data Persistence

The `Leaderboard` class is an Entity mapped to a database table.

```

1  @Entity
2  @Table(name = "leaderboard")
3  public class Leaderboard {
4      @Id
5      @GeneratedValue(strategy = GenerationType.IDENTITY)

```

```

6   private Long id;
7
8   @Column(name = "player_name")
9   private String playerName;
10
11  @Column(name = "score")
12  private int score;
13
14  @Column(name = "match_date")
15  private Timestamp matchDate;
16
17  public Leaderboard(String playerName, int score, Timestamp matchDate) {
18      this.playerName = playerName;
19      this.score = score;
20      this.matchDate = matchDate;
21  }
22}

```

Listing 4: Leaderboard.java

The LeaderboardController exposes a REST API to retrieve these records.

```

1 @RestController
2 @RequestMapping("/api/leaderboard")
3 public class LeaderboardController {
4     @GetMapping
5     public List<Leaderboard> getLeaderboard() {
6         return leaderboardRepository.findTop10ByOrderByScoreDesc();
7     }
8 }

```

Listing 5: LeaderboardController.java

### 3.5 Core Logic: Game Service

This is the most complex class, handling collision detection, power-up spawning, and state management.

```

1 @Service
2 public class GameService {
3     public static final double GAME_WIDTH = 800;
4
5     public void update() {
6         if (System.currentTimeMillis() - gameStartTime > 30000) {
7             gameOver = true;
8             saveGameResults();
9             return;
10        }
11
12        List<Bullet> bullets = gameState.getBullets();
13        List<Bullet> bulletsToRemove = new ArrayList<>();
14
15        for (Bullet bullet : bullets) {
16            bullet.setX(bullet.getX() + bullet.getSpeed());
17
18            for (Player player : gameState.getPlayers()) {
19                if (bullet.getShooterName().equals(player.getName()))
20                    continue;
21
22                if (checkCollision(bullet, player)) {
23                    updateScore(bullet.getShooterName());
24                }
25            }
26        }
27    }
28
29    private void updateScore(String shooterName) {
30        // Implementation
31    }
32
33    private boolean checkCollision(Bullet bullet, Player player) {
34        // Implementation
35    }
36
37    private void saveGameResults() {
38        // Implementation
39    }
40
41    private void gameOver() {
42        // Implementation
43    }
44
45    private void spawnPowerUp() {
46        // Implementation
47    }
48
49    private void handlePlayerInput(Player player) {
50        // Implementation
51    }
52
53    private void handleCollision() {
54        // Implementation
55    }
56
57    private void handlePowerUp() {
58        // Implementation
59    }
60
61    private void handleStateManagement() {
62        // Implementation
63    }
64
65    private void handlePlayerMovement() {
66        // Implementation
67    }
68
69    private void handleBulletMovement() {
70        // Implementation
71    }
72
73    private void handlePowerUpSpawning() {
74        // Implementation
75    }
76
77    private void handlePlayerCollision() {
78        // Implementation
79    }
80
81    private void handlePlayerPowerUp() {
82        // Implementation
83    }
84
85    private void handlePlayerScore() {
86        // Implementation
87    }
88
89    private void handlePlayerHealth() {
90        // Implementation
91    }
92
93    private void handlePlayerMovement() {
94        // Implementation
95    }
96
97    private void handleBulletCollision() {
98        // Implementation
99    }
100
101    private void handleBulletScore() {
102        // Implementation
103    }
104
105    private void handleBulletHealth() {
106        // Implementation
107    }
108
109    private void handleBulletMovement() {
110        // Implementation
111    }
112
113    private void handleBulletPowerUp() {
114        // Implementation
115    }
116
117    private void handleBulletScore() {
118        // Implementation
119    }
120
121    private void handleBulletHealth() {
122        // Implementation
123    }
124
125    private void handleBulletMovement() {
126        // Implementation
127    }
128
129    private void handleBulletPowerUp() {
130        // Implementation
131    }
132
133    private void handleBulletScore() {
134        // Implementation
135    }
136
137    private void handleBulletHealth() {
138        // Implementation
139    }
140
141    private void handleBulletMovement() {
142        // Implementation
143    }
144
145    private void handleBulletPowerUp() {
146        // Implementation
147    }
148
149    private void handleBulletScore() {
150        // Implementation
151    }
152
153    private void handleBulletHealth() {
154        // Implementation
155    }
156
157    private void handleBulletMovement() {
158        // Implementation
159    }
160
161    private void handleBulletPowerUp() {
162        // Implementation
163    }
164
165    private void handleBulletScore() {
166        // Implementation
167    }
168
169    private void handleBulletHealth() {
170        // Implementation
171    }
172
173    private void handleBulletMovement() {
174        // Implementation
175    }
176
177    private void handleBulletPowerUp() {
178        // Implementation
179    }
180
181    private void handleBulletScore() {
182        // Implementation
183    }
184
185    private void handleBulletHealth() {
186        // Implementation
187    }
188
189    private void handleBulletMovement() {
190        // Implementation
191    }
192
193    private void handleBulletPowerUp() {
194        // Implementation
195    }
196
197    private void handleBulletScore() {
198        // Implementation
199    }
200
201    private void handleBulletHealth() {
202        // Implementation
203    }
204
205    private void handleBulletMovement() {
206        // Implementation
207    }
208
209    private void handleBulletPowerUp() {
210        // Implementation
211    }
212
213    private void handleBulletScore() {
214        // Implementation
215    }
216
217    private void handleBulletHealth() {
218        // Implementation
219    }
220
221    private void handleBulletMovement() {
222        // Implementation
223    }
224
225    private void handleBulletPowerUp() {
226        // Implementation
227    }
228
229    private void handleBulletScore() {
230        // Implementation
231    }
232
233    private void handleBulletHealth() {
234        // Implementation
235    }
236
237    private void handleBulletMovement() {
238        // Implementation
239    }
240
241    private void handleBulletPowerUp() {
242        // Implementation
243    }
244
245    private void handleBulletScore() {
246        // Implementation
247    }
248
249    private void handleBulletHealth() {
250        // Implementation
251    }
252
253    private void handleBulletMovement() {
254        // Implementation
255    }
256
257    private void handleBulletPowerUp() {
258        // Implementation
259    }
260
261    private void handleBulletScore() {
262        // Implementation
263    }
264
265    private void handleBulletHealth() {
266        // Implementation
267    }
268
269    private void handleBulletMovement() {
270        // Implementation
271    }
272
273    private void handleBulletPowerUp() {
274        // Implementation
275    }
276
277    private void handleBulletScore() {
278        // Implementation
279    }
280
281    private void handleBulletHealth() {
282        // Implementation
283    }
284
285    private void handleBulletMovement() {
286        // Implementation
287    }
288
289    private void handleBulletPowerUp() {
290        // Implementation
291    }
292
293    private void handleBulletScore() {
294        // Implementation
295    }
296
297    private void handleBulletHealth() {
298        // Implementation
299    }
300
301    private void handleBulletMovement() {
302        // Implementation
303    }
304
305    private void handleBulletPowerUp() {
306        // Implementation
307    }
308
309    private void handleBulletScore() {
310        // Implementation
311    }
312
313    private void handleBulletHealth() {
314        // Implementation
315    }
316
317    private void handleBulletMovement() {
318        // Implementation
319    }
320
321    private void handleBulletPowerUp() {
322        // Implementation
323    }
324
325    private void handleBulletScore() {
326        // Implementation
327    }
328
329    private void handleBulletHealth() {
330        // Implementation
331    }
332
333    private void handleBulletMovement() {
334        // Implementation
335    }
336
337    private void handleBulletPowerUp() {
338        // Implementation
339    }
340
341    private void handleBulletScore() {
342        // Implementation
343    }
344
345    private void handleBulletHealth() {
346        // Implementation
347    }
348
349    private void handleBulletMovement() {
350        // Implementation
351    }
352
353    private void handleBulletPowerUp() {
354        // Implementation
355    }
356
357    private void handleBulletScore() {
358        // Implementation
359    }
360
361    private void handleBulletHealth() {
362        // Implementation
363    }
364
365    private void handleBulletMovement() {
366        // Implementation
367    }
368
369    private void handleBulletPowerUp() {
370        // Implementation
371    }
372
373    private void handleBulletScore() {
374        // Implementation
375    }
376
377    private void handleBulletHealth() {
378        // Implementation
379    }
380
381    private void handleBulletMovement() {
382        // Implementation
383    }
384
385    private void handleBulletPowerUp() {
386        // Implementation
387    }
388
389    private void handleBulletScore() {
390        // Implementation
391    }
392
393    private void handleBulletHealth() {
394        // Implementation
395    }
396
397    private void handleBulletMovement() {
398        // Implementation
399    }
400
401    private void handleBulletPowerUp() {
402        // Implementation
403    }
404
405    private void handleBulletScore() {
406        // Implementation
407    }
408
409    private void handleBulletHealth() {
410        // Implementation
411    }
412
413    private void handleBulletMovement() {
414        // Implementation
415    }
416
417    private void handleBulletPowerUp() {
418        // Implementation
419    }
420
421    private void handleBulletScore() {
422        // Implementation
423    }
424
425    private void handleBulletHealth() {
426        // Implementation
427    }
428
429    private void handleBulletMovement() {
430        // Implementation
431    }
432
433    private void handleBulletPowerUp() {
434        // Implementation
435    }
436
437    private void handleBulletScore() {
438        // Implementation
439    }
440
441    private void handleBulletHealth() {
442        // Implementation
443    }
444
445    private void handleBulletMovement() {
446        // Implementation
447    }
448
449    private void handleBulletPowerUp() {
450        // Implementation
451    }
452
453    private void handleBulletScore() {
454        // Implementation
455    }
456
457    private void handleBulletHealth() {
458        // Implementation
459    }
460
461    private void handleBulletMovement() {
462        // Implementation
463    }
464
465    private void handleBulletPowerUp() {
466        // Implementation
467    }
468
469    private void handleBulletScore() {
470        // Implementation
471    }
472
473    private void handleBulletHealth() {
474        // Implementation
475    }
476
477    private void handleBulletMovement() {
478        // Implementation
479    }
480
481    private void handleBulletPowerUp() {
482        // Implementation
483    }
484
485    private void handleBulletScore() {
486        // Implementation
487    }
488
489    private void handleBulletHealth() {
490        // Implementation
491    }
492
493    private void handleBulletMovement() {
494        // Implementation
495    }
496
497    private void handleBulletPowerUp() {
498        // Implementation
499    }
500
501    private void handleBulletScore() {
502        // Implementation
503    }
504
505    private void handleBulletHealth() {
506        // Implementation
507    }
508
509    private void handleBulletMovement() {
510        // Implementation
511    }
512
513    private void handleBulletPowerUp() {
514        // Implementation
515    }
516
517    private void handleBulletScore() {
518        // Implementation
519    }
520
521    private void handleBulletHealth() {
522        // Implementation
523    }
524
525    private void handleBulletMovement() {
526        // Implementation
527    }
528
529    private void handleBulletPowerUp() {
530        // Implementation
531    }
532
533    private void handleBulletScore() {
534        // Implementation
535    }
536
537    private void handleBulletHealth() {
538        // Implementation
539    }
540
541    private void handleBulletMovement() {
542        // Implementation
543    }
544
545    private void handleBulletPowerUp() {
546        // Implementation
547    }
548
549    private void handleBulletScore() {
550        // Implementation
551    }
552
553    private void handleBulletHealth() {
554        // Implementation
555    }
556
557    private void handleBulletMovement() {
558        // Implementation
559    }
560
561    private void handleBulletPowerUp() {
562        // Implementation
563    }
564
565    private void handleBulletScore() {
566        // Implementation
567    }
568
569    private void handleBulletHealth() {
570        // Implementation
571    }
572
573    private void handleBulletMovement() {
574        // Implementation
575    }
576
577    private void handleBulletPowerUp() {
578        // Implementation
579    }
580
581    private void handleBulletScore() {
582        // Implementation
583    }
584
585    private void handleBulletHealth() {
586        // Implementation
587    }
588
589    private void handleBulletMovement() {
590        // Implementation
591    }
592
593    private void handleBulletPowerUp() {
594        // Implementation
595    }
596
597    private void handleBulletScore() {
598        // Implementation
599    }
600
601    private void handleBulletHealth() {
602        // Implementation
603    }
604
605    private void handleBulletMovement() {
606        // Implementation
607    }
608
609    private void handleBulletPowerUp() {
610        // Implementation
611    }
612
613    private void handleBulletScore() {
614        // Implementation
615    }
616
617    private void handleBulletHealth() {
618        // Implementation
619    }
620
621    private void handleBulletMovement() {
622        // Implementation
623    }
624
625    private void handleBulletPowerUp() {
626        // Implementation
627    }
628
629    private void handleBulletScore() {
630        // Implementation
631    }
632
633    private void handleBulletHealth() {
634        // Implementation
635    }
636
637    private void handleBulletMovement() {
638        // Implementation
639    }
640
641    private void handleBulletPowerUp() {
642        // Implementation
643    }
644
645    private void handleBulletScore() {
646        // Implementation
647    }
648
649    private void handleBulletHealth() {
650        // Implementation
651    }
652
653    private void handleBulletMovement() {
654        // Implementation
655    }
656
657    private void handleBulletPowerUp() {
658        // Implementation
659    }
660
661    private void handleBulletScore() {
662        // Implementation
663    }
664
665    private void handleBulletHealth() {
666        // Implementation
667    }
668
669    private void handleBulletMovement() {
670        // Implementation
671    }
672
673    private void handleBulletPowerUp() {
674        // Implementation
675    }
676
677    private void handleBulletScore() {
678        // Implementation
679    }
680
681    private void handleBulletHealth() {
682        // Implementation
683    }
684
685    private void handleBulletMovement() {
686        // Implementation
687    }
688
689    private void handleBulletPowerUp() {
690        // Implementation
691    }
692
693    private void handleBulletScore() {
694        // Implementation
695    }
696
697    private void handleBulletHealth() {
698        // Implementation
699    }
700
701    private void handleBulletMovement() {
702        // Implementation
703    }
704
705    private void handleBulletPowerUp() {
706        // Implementation
707    }
708
709    private void handleBulletScore() {
710        // Implementation
711    }
712
713    private void handleBulletHealth() {
714        // Implementation
715    }
716
717    private void handleBulletMovement() {
718        // Implementation
719    }
720
721    private void handleBulletPowerUp() {
722        // Implementation
723    }
724
725    private void handleBulletScore() {
726        // Implementation
727    }
728
729    private void handleBulletHealth() {
730        // Implementation
731    }
732
733    private void handleBulletMovement() {
734        // Implementation
735    }
736
737    private void handleBulletPowerUp() {
738        // Implementation
739    }
740
741    private void handleBulletScore() {
742        // Implementation
743    }
744
745    private void handleBulletHealth() {
746        // Implementation
747    }
748
749    private void handleBulletMovement() {
750        // Implementation
751    }
752
753    private void handleBulletPowerUp() {
754        // Implementation
755    }
756
757    private void handleBulletScore() {
758        // Implementation
759    }
760
761    private void handleBulletHealth() {
762        // Implementation
763    }
764
765    private void handleBulletMovement() {
766        // Implementation
767    }
768
769    private void handleBulletPowerUp() {
770        // Implementation
771    }
772
773    private void handleBulletScore() {
774        // Implementation
775    }
776
777    private void handleBulletHealth() {
778        // Implementation
779    }
780
781    private void handleBulletMovement() {
782        // Implementation
783    }
784
785    private void handleBulletPowerUp() {
786        // Implementation
787    }
788
789    private void handleBulletScore() {
790        // Implementation
791    }
792
793    private void handleBulletHealth() {
794        // Implementation
795    }
796
797    private void handleBulletMovement() {
798        // Implementation
799    }
800
801    private void handleBulletPowerUp() {
802        // Implementation
803    }
804
805    private void handleBulletScore() {
806        // Implementation
807    }
808
809    private void handleBulletHealth() {
810        // Implementation
811    }
812
813    private void handleBulletMovement() {
814        // Implementation
815    }
816
817    private void handleBulletPowerUp() {
818        // Implementation
819    }
820
821    private void handleBulletScore() {
822        // Implementation
823    }
824
825    private void handleBulletHealth() {
826        // Implementation
827    }
828
829    private void handleBulletMovement() {
830        // Implementation
831    }
832
833    private void handleBulletPowerUp() {
834        // Implementation
835    }
836
837    private void handleBulletScore() {
838        // Implementation
839    }
840
841    private void handleBulletHealth() {
842        // Implementation
843    }
844
845    private void handleBulletMovement() {
846        // Implementation
847    }
848
849    private void handleBulletPowerUp() {
850        // Implementation
851    }
852
853    private void handleBulletScore() {
854        // Implementation
855    }
856
857    private void handleBulletHealth() {
858        // Implementation
859    }
860
861    private void handleBulletMovement() {
862        // Implementation
863    }
864
865    private void handleBulletPowerUp() {
866        // Implementation
867    }
868
869    private void handleBulletScore() {
870        // Implementation
871    }
872
873    private void handleBulletHealth() {
874        // Implementation
875    }
876
877    private void handleBulletMovement() {
878        // Implementation
879    }
880
881    private void handleBulletPowerUp() {
882        // Implementation
883    }
884
885    private void handleBulletScore() {
886        // Implementation
887    }
888
889    private void handleBulletHealth() {
890        // Implementation
891    }
892
893    private void handleBulletMovement() {
894        // Implementation
895    }
896
897    private void handleBulletPowerUp() {
898        // Implementation
899    }
900
901    private void handleBulletScore() {
902        // Implementation
903    }
904
905    private void handleBulletHealth() {
906        // Implementation
907    }
908
909    private void handleBulletMovement() {
910        // Implementation
911    }
912
913    private void handleBulletPowerUp() {
914        // Implementation
915    }
916
917    private void handleBulletScore() {
918        // Implementation
919    }
920
921    private void handleBulletHealth() {
922        // Implementation
923    }
924
925    private void handleBulletMovement() {
926        // Implementation
927    }
928
929    private void handleBulletPowerUp() {
930        // Implementation
931    }
932
933    private void handleBulletScore() {
934        // Implementation
935    }
936
937    private void handleBulletHealth() {
938        // Implementation
939    }
940
941    private void handleBulletMovement() {
942        // Implementation
943    }
944
945    private void handleBulletPowerUp() {
946        // Implementation
947    }
948
949    private void handleBulletScore() {
950        // Implementation
951    }
952
953    private void handleBulletHealth() {
954        // Implementation
955    }
956
957    private void handleBulletMovement() {
958        // Implementation
959    }
960
961    private void handleBulletPowerUp() {
962        // Implementation
963    }
964
965    private void handleBulletScore() {
966        // Implementation
967    }
968
969    private void handleBulletHealth() {
970        // Implementation
971    }
972
973    private void handleBulletMovement() {
974        // Implementation
975    }
976
977    private void handleBulletPowerUp() {
978        // Implementation
979    }
980
981    private void handleBulletScore() {
982        // Implementation
983    }
984
985    private void handleBulletHealth() {
986        // Implementation
987    }
988
989    private void handleBulletMovement() {
990        // Implementation
991    }
992
993    private void handleBulletPowerUp() {
994        // Implementation
995    }
996
997    private void handleBulletScore() {
998        // Implementation
999    }
1000
1001    private void handleBulletHealth() {
1002        // Implementation
1003    }
1004
1005    private void handleBulletMovement() {
1006        // Implementation
1007    }
1008
1009    private void handleBulletPowerUp() {
1010        // Implementation
1011    }
1012
1013    private void handleBulletScore() {
1014        // Implementation
1015    }
1016
1017    private void handleBulletHealth() {
1018        // Implementation
1019    }
1020
1021    private void handleBulletMovement() {
1022        // Implementation
1023    }
1024
1025    private void handleBulletPowerUp() {
1026        // Implementation
1027    }
1028
1029    private void handleBulletScore() {
1030        // Implementation
1031    }
1032
1033    private void handleBulletHealth() {
1034        // Implementation
1035    }
1036
1037    private void handleBulletMovement() {
1038        // Implementation
1039    }
1040
1041    private void handleBulletPowerUp() {
1042        // Implementation
1043    }
1044
1045    private void handleBulletScore() {
1046        // Implementation
1047    }
1048
1049    private void handleBulletHealth() {
1050        // Implementation
1051    }
1052
1053    private void handleBulletMovement() {
1054        // Implementation
1055    }
1056
1057    private void handleBulletPowerUp() {
1058        // Implementation
1059    }
1060
1061    private void handleBulletScore() {
1062        // Implementation
1063    }
1064
1065    private void handleBulletHealth() {
1066        // Implementation
1067    }
1068
1069    private void handleBulletMovement() {
1070        // Implementation
1071    }
1072
1073    private void handleBulletPowerUp() {
1074        // Implementation
1075    }
1076
1077    private void handleBulletScore() {
1078        // Implementation
1079    }
1080
1081    private void handleBulletHealth() {
1082        // Implementation
1083    }
1084
1085    private void handleBulletMovement() {
1086        // Implementation
1087    }
1088
1089    private void handleBulletPowerUp() {
1090        // Implementation
1091    }
1092
1093    private void handleBulletScore() {
1094        // Implementation
1095    }
1096
1097    private void handleBulletHealth() {
1098        // Implementation
1099    }
1100
1101    private void handleBulletMovement() {
1102        // Implementation
1103    }
1104
1105    private void handleBulletPowerUp() {
1106        // Implementation
1107    }
1108
1109    private void handleBulletScore() {
1110        // Implementation
1111    }
1112
1113    private void handleBulletHealth() {
1114        // Implementation
1115    }
1116
1117    private void handleBulletMovement() {
1118        // Implementation
1119    }
1120
1121    private void handleBulletPowerUp() {
1122        // Implementation
1123    }
1124
1125    private void handleBulletScore() {
1126        // Implementation
1127    }
1128
1129    private void handleBulletHealth() {
1130        // Implementation
1131    }
1132
1133    private void handleBulletMovement() {
1134        // Implementation
1135    }
1136
1137    private void handleBulletPowerUp() {
1138        // Implementation
1139    }
1140
1141    private void handleBulletScore() {
1142        // Implementation
1143    }
1144
1145    private void handleBulletHealth() {
1146        // Implementation
1147    }
1148
1149    private void handleBulletMovement() {
1150        // Implementation
1151    }
1152
1153    private void handleBulletPowerUp() {
1154        // Implementation
1155    }
1156
1157    private void handleBulletScore() {
1158        // Implementation
1159    }
1160
1161    private void handleBulletHealth() {
1162        // Implementation
1163    }
1164
1165    private void handleBulletMovement() {
1166        // Implementation
1167    }
1168
1169    private void handleBulletPowerUp() {
1170        // Implementation
1171    }
1172
1173    private void handleBulletScore() {
1174        // Implementation
1175    }
1176
1177    private void handleBulletHealth() {
1178        // Implementation
1179    }
1180
1181    private void handleBulletMovement() {
1182        // Implementation
1183    }
1184
1185    private void handleBulletPowerUp() {
1186        // Implementation
1187    }
1188
1189    private void handleBulletScore() {
1190        // Implementation
1191    }
1192
1193    private void handleBulletHealth() {
1194        // Implementation
1195    }
1196
1197    private void handleBulletMovement() {
1198        // Implementation
1199    }
1200
1201    private void handleBulletPowerUp() {
1202        // Implementation
1203    }
1204
1205    private void handleBulletScore() {
1206        // Implementation
1207    }
1208
1209    private void handleBulletHealth() {
1210        // Implementation
1211    }
1212
1213    private void handleBulletMovement() {
1214        // Implementation
1215    }
1216
1217    private void handleBulletPowerUp() {
1218        // Implementation
1219    }
1220
1221    private void handleBulletScore() {
1222        // Implementation
1223    }
1224
1225    private void handleBulletHealth() {
1226        // Implementation
1227    }
1228
1229    private void handleBulletMovement() {
1230        // Implementation
1231    }
1232
1233    private void handleBulletPowerUp() {
1234        // Implementation
1235    }
1236
1237    private void handleBulletScore() {
1238        // Implementation
1239    }
1240
1241    private void handleBulletHealth() {
1242        // Implementation
1243    }
1244
1245    private void handleBulletMovement() {
1246        // Implementation
1247    }
1248
1249    private void handleBulletPowerUp() {
1250        // Implementation
1251    }
1252
1253    private void handleBulletScore() {
1254        // Implementation
1255    }
1256
1257    private void handleBulletHealth() {
1258        // Implementation
1259    }
1260
1261    private void handleBulletMovement() {
1262        // Implementation
1263    }
1264
1265    private void handleBulletPowerUp() {
1266        // Implementation
1267    }
1268
1269    private void handleBulletScore() {
1270        // Implementation
1271    }
1272
1273    private void handleBulletHealth() {
1274        // Implementation
1275    }
1276
1277    private void handleBulletMovement() {
1278        // Implementation
1279    }
1280
1281    private void handleBulletPowerUp() {
1282        // Implementation
1283    }
1284
1285    private void handleBulletScore() {
1286        // Implementation
1287    }
1288
1289    private void handleBulletHealth() {
1290        // Implementation
1291    }
1292
1293    private void handleBulletMovement() {
1294        // Implementation
1295    }
1296
1297    private void handleBulletPowerUp() {
1298        // Implementation
1299    }
1300
1301    private void handleBulletScore() {
1302        // Implementation
1303    }
1304
1305    private void handleBulletHealth() {
1306        // Implementation
1307    }
1308
1309    private void handleBulletMovement() {
1310        // Implementation
1311    }
1312
1313    private void handleBulletPowerUp() {
1314        // Implementation
1315    }
1316
1317    private void handleBulletScore() {
1318        // Implementation
1319    }
1320
1321    private void handleBulletHealth() {
1322        // Implementation
1323    }
1324
1325    private void handleBulletMovement() {
1326        // Implementation
1327    }
1328
1329    private void handleBulletPowerUp() {
1330        // Implementation
1331    }
1332
1333    private void handleBulletScore() {
1334        // Implementation
1335    }
1336
1337    private void handleBulletHealth() {
1338        // Implementation
1339    }
1340
1341    private void handleBulletMovement() {
1342        // Implementation
1343    }
1344
1345    private void handleBulletPowerUp() {
1346        // Implementation
1347    }
1348
1349    private void handleBulletScore() {
1350        // Implementation
1351    }
1352
1353    private void handleBulletHealth() {
1354        // Implementation
1355    }
1356
1357    private void handleBulletMovement() {
1358        // Implementation
1359    }
1360
1361    private void handleBulletPowerUp() {
1362        // Implementation
1363    }
1364
1365    private void handleBulletScore() {
1366        // Implementation
1367    }
1368
1369    private void handleBulletHealth() {
1370        // Implementation
1371    }
1372
1373    private void handleBulletMovement() {
1374        // Implementation
1375    }
1376
1377    private void handleBulletPowerUp() {
1378        // Implementation
1379    }
1380
1381    private void handleBulletScore() {
1382        // Implementation
1383    }
1384
1385    private void handleBulletHealth() {
1386        // Implementation
1387    }
1388
1389    private void handleBulletMovement() {
1390        // Implementation
1391    }
1392
1393    private void handleBulletPowerUp() {
1394        // Implementation
1395    }
1396
1397    private void handleBulletScore() {
1398        // Implementation
1399    }
1400
1401    private void handleBulletHealth() {
1402        // Implementation
1403    }
1404
1405    private void handleBulletMovement() {
1406        // Implementation
1407    }
1408
1409    private void handleBulletPowerUp() {
1410        // Implementation
1411    }
1412
1413    private void handleBulletScore() {
1414        // Implementation
1415    }
1416
1417    private void handleBulletHealth() {
1418        // Implementation
1419    }
1420
1421    private void handleBulletMovement() {
1422        // Implementation
1423    }
1424
1425    private void handleBulletPowerUp() {
1426        // Implementation
1427    }
1428
1429    private void handleBulletScore() {
1430        // Implementation
1431    }
1432
1433    private void handleBulletHealth() {
1434        // Implementation
1435    }
1436
1437    private void handleBulletMovement() {
1438        // Implementation
1439    }
1440
1441    private void handleBulletPowerUp() {
1442        // Implementation
1443    }
1444
1445    private void handleBulletScore() {
1446        // Implementation
1447    }
1448
1449    private void handleBulletHealth() {
1450        // Implementation
1451    }
1452
1453    private void handleBulletMovement() {
1454        // Implementation
1455    }
1456
1457    private void handleBulletPowerUp() {
1458        // Implementation
1459    }
1460
1461    private void handleBulletScore() {
1462        // Implementation
1463    }
1464
1465    private void handleBulletHealth() {
1466        // Implementation
1467    }
1468
1469    private void handleBulletMovement() {
1470        // Implementation
1471    }
1472
1473    private void handleBulletPowerUp() {
1474        // Implementation
1475    }
1476
1477    private void handleBulletScore() {
1478        // Implementation
1479    }
1480
1481    private void handleBulletHealth() {
1482        // Implementation
1483    }
1484
1485    private void handleBulletMovement() {
1486        // Implementation
1487    }
1488
1489    private void handleBulletPowerUp() {
1490        // Implementation
1491    }
1492
1493    private void handleBulletScore() {
1494        // Implementation
1495    }
1496
1497    private void handleBulletHealth() {
1498        // Implementation
1499    }
1500
150
```

```

23             bulletsToRemove.add(bullet);
24         }
25     }
26 }
27 bullets.removeAll(bulletsToRemove);
28
29 }
30
31 private boolean checkCollision(Bullet b, Player p) {
32     return b.getX() < p.getX() + p.getWidth() &&
33         b.getX() + b.getWidth() > p.getX() &&
34         b.getY() < p.getY() + p.getHeight() &&
35         b.getY() + b.getHeight() > p.getY();
36 }
37 }
```

Listing 6: GameService.java (Logic Snippet)

## 4 OOP Concepts Analysis

### 4.1 Encapsulation

The game state is heavily encapsulated. Direct access to player coordinates or scores is restricted. For instance, the `Bullet` and `Player` classes maintain their internal state (x, y coordinates) and only allow modification through specific methods (e.g., `movePlayer` in `GameService`), ensuring data integrity.

### 4.2 Single Responsibility Principle (SRP)

The application adheres to SRP by separating concerns:

- **GameService:** Handles physics and rules. It does not know about HTTP or WebSockets.
- **GameController:** Handles communication. It does not calculate physics.
- **LeaderboardRepository:** Handles database SQL operations.

### 4.3 Dependency Injection (DI)

Spring's Inversion of Control (IoC) container is used throughout. For example, `GameController` requests a `GameService` in its constructor. This makes the code testable and loosely coupled; we could easily swap `GameService` for a `MockGameService` during unit testing without changing the controller code.

### 4.4 Concurrency

While not strictly a standard OOP concept, the object-oriented handling of threads via `ScheduledExecutor` allows us to treat the game loop as an object-managed task, rather than raw procedural thread management.

## 5 Conclusion

This project successfully implements a real-time multiplayer game using Spring Boot. By adhering to Object-Oriented principles and the MVC design pattern, the system is modular, scalable, and easy to maintain. The separation of the communication layer (WebSockets) from the physics engine (Service layer) demonstrates a robust architectural approach suitable for complex interactive applications.