Postmortem Report: Shooter Game Project

# Project Management

## What Went Right

We did manage to create the game regardless of all the obstacles that stopped us. Even though much of the work came together at the last minute, we still pulled it off and delivered a working shooter game. The ability to push through and finish despite everything working against us is what we are most proud of.

## What Went Wrong

Two members of the team contributed nothing meaningful to the project. Instead of helping, they created more issues by trying to break the game and then passing along work that did not function while pretending it did. This wasted valuable time and added stress. Their lack of genuine contribution left the real workload to the remaining two of us.

## Mitigation

We adapted by reassigning tasks between the two of us who were committed. We stopped relying on unhelpful teammates and focused on completing the game ourselves. This shift in focus allowed us to recover time, prioritize features, and push the project to completion.

## Key Decisions That Worked

The most important decision we made was simply to handle the work between the two of us as best as we could. By cutting out distractions and staying focused, we were able to stay productive and deliver results.

## Design Documentation Impact

The design documents were useful in setting the core direction for the project, but with unreliable teammates, following them exactly became less practical. We had to make on-the-fly adjustments and keep only what was realistic in the timeframe. Ultimately, the documentation gave us a starting point, but the actual execution came down to what we could realistically finish.

# Development

## Successes

A lot of changes happened during development, and many adjustments had to be made along the way. Despite that, everything ended up working in the final version. The shooter game functioned as intended, with mechanics and features that fit together in the final product.

## Problems

The biggest problem we faced was trusting our third classmate to actually help. That trust slowed us down, brought frustration, and ultimately forced us to re-do or replace their work.

## Resolution

We resolved this by cutting our reliance on them altogether and focusing only on what the two of us could control. Moving forward, we will set stricter expectations for accountability, regularly verify progress, and not let non-contributors hold back the project. This experience taught us the importance of confronting problems directly rather than hoping they resolve themselves.

# Quality Assurance and Testing

## What Worked

Testing identified critical bugs early in core systems such as shooting mechanics and collision. Iterative playtesting ensured that the game was playable and balanced enough to be fun.

## What Did Not Work

Late stage testing revealed problems that could have been caught earlier if we had implemented formal testing sooner. Without structured test plans, some issues slipped through until the very end.

## Mitigation

We focused on fixing the most important issues first and simplified some mechanics to guarantee stability. Quick feedback loops helped us push out last-minute fixes.

## Future Improvement

For future projects, we need to apply formal test plans and regression testing from the beginning. This would prevent wasted time near deadlines and ensure consistent quality throughout development.

# Tools and Practice

## What Worked

Version control and our chosen engine tools were critical in keeping progress safe and organized. Communication tools like Discord made collaboration efficient for the two of us who were actually contributing.

## What Did Not Work

Some project management tools went unused. Since part of the team did not participate, systems meant to track tasks ended up being irrelevant. This failure was tied to lack of accountability rather than the tools themselves.

## Documentation Contribution

The design documents helped early on but became less useful once we realized we had to cut features and simplify scope. They served as a guide, but the reality of development required more flexibility.

# Communications

## Strengths

The two of us who contributed worked extremely well together. Our communication was clear and honest, and that teamwork is the reason the game was completed.

## Weaknesses

The weakness was trusting and relying on team members who did not contribute. This miscommunication of expectations caused unnecessary setbacks and wasted effort.

## Mitigation

We ultimately solved this by keeping communication between the two of us who were committed, which created a more efficient workflow. In future projects, stronger accountability measures will help avoid this problem.

# Conclusions

## Lessons Learned

Future development teams should focus on holding all members accountable from the start. It is better to confront lack of contribution early rather than waste time correcting broken or fake work. Another lesson is to implement structured testing much earlier in the process to avoid last-minute bug fixing.

## Suggestions for Future Teams

Set clear expectations and accountability measures for every team member.  
Update planning documents regularly to reflect actual progress and scope changes.  
Use formal testing throughout development, not just at the end.  
Do not hesitate to simplify scope when team resources are limited.  
Build around the strengths of reliable teammates rather than waiting on those who do not deliver.