

# Title Section

15 February 2026 11:17

Name: Harry Keinzley

Student Number: 2401099

Course Name: Computing (Games Development)

# Task 1: Ideation & Innovation

15 February 2026 11:22

## Q1: Innovation

Innovation is a key part of society as humans and can be categorised in a variety of ways whether this be innovation through the development something new, or the improvement of an already existing process such as increased processing, greater throughput or lower cost [1]. Almost everything that people use on a daily basis are a product of innovation, whether this being major innovation such as the creation of mobile phones or computers, to the less interesting things such as the standardisation of USB-C cables in recent years.

Throughout the last few hundred years there have been multiple major waves of innovation that have been named "Industrial Revolution's" these being the innovation of steam water and mechanical production equipment in 1784, the division of labour, electricity and mass production from 1870 and the electronics, IT and automated production from 1969.

One way in which industrial revolutions are often predicted is using Kondratiev Waves, this is a cycle that was first studied by Nikolai Kondratiev where he found that the economy has waves which relate to innovation periods, where major breakthroughs heavily increase the world economy before eventually beginning to drop back to normal levels, this has been seen multiple times within the last couple hundred years, some of the more recent ones are the innovations in cars after the great depression, then the major innovation in IT after the oil crisis, and currently after the financial crisis it is possible that another innovation such as AI becomes the next peak of the wave. [2]

## Q2: Ideation

### Project Ideas:

- Mod Development Toolkit for Deadlock
- Adaptive Game Difficulty System In Unreal Engine 5
- Simple Factory Game
- In Game Overlay for Deadlock
- Physics Based Puzzle System

### Idea of Choice:

API Driven In-Game Overlay for Deadlock

### Aim of Project:

The aim of this project is to create a comprehensive API driven overlay for the game Deadlock by Valve

### Project Objectives:

- Tracking of players active match showing all players ranks / leaderboard position and a range of information about their account such as total games played, win rate, smurf detection, what their best hero is, whether they perform well in lane etc. (if the game is within the top 200 games being played at the time due to game scraping capabilities.)
- Shows Information about recently played matches (that have been ingested by the third party API's as Valve does not currently provide a large first party API)
- Shows progress to next rank

- Post game stats showing what mistakes were made in the game, such as a lack of participation in fights, below average farming stats, more deaths than average

## Project Description:

This project aims to create an easy to access, in depth overlay for deadlock which will contain a large variety of information that usually the player can only usually get access to by opening websites after the game has been played, as well as new features exclusive to this overlay. This project will make use of 2 different third party API's the first of which is the public deadlock API used for general game data (<https://deadlock-api.com/> [2]) the second of these will be for accessing a ranking system and a large range of post-match AI and statistical models for player and game performance (<https://statlocker.gg/> [3])

## Q3: Research

Some of the key sources that will be used for this project will be as mentioned before, the API's from both <https://deadlock-api.com/> [2] as well as <https://statlocker.gg/> [3]. These API's will be used as the backbone of the project are necessary for the functionality of the project. The next courses that will need to be looked into are for the backend side of the overlay that will need to be written to make the overlay function as intended, I intent to use TypeScript and Electron[4] for the project so will need to learn how to use those effectively to ensure that the overlay is visually appealing as well as very light on system usage as this overlay will be running at the same time as the game so ensuring in game frame rate isn't affected is highly important.

## References

- [1] K. B. Kahn, "Understanding innovation," *Bus.Horiz.* vol. 61, no. 3, pp.453–460, 2018, doi: 10.1016/j.bushor.2018.01.011. Accessed 15/02/2026 Available at: <https://www.sciencedirect.com/science/article/pii/S0007681318300119>.
- [2] O'Hara, P.A. (2025). Capitalism's Long Waves: Kondratiev, Schumpeter, *et al.*. In: Long Waves of Growth, Hegemonic Power, and Climate Change in the World Economy. Springer, Singapore. [https://doi.org/10.1007/978-981-96-4132-1\\_2](https://doi.org/10.1007/978-981-96-4132-1_2)
- [3] Deadlock API, "Deadlock Game Data API Documentation," Deadlock API. [Online]. Available: <https://deadlock-api.com/>. [Accessed: 15 Feb. 2026].
- [4] Statlocker, Statlocker API Documentation. [Online]. Available: <https://statlocker.gg/>. [Accessed: 15 Feb. 2026].
- [5] "Introduction | Electron." Accessed: Feb 15 2026. [Online]. Available: <https://electronjs.org/docs/latest/>

# Task 2

21 February 2026 13:13

## Q1: Project Scope

### Proposed Solution:

This project will aim to be ran as an external desktop app that will recognise when the game is open and act as an overlay to that game window only, with a configurable key bind to open the overlay, inside this overlay there will be multiple tabs, the first of these will be the default tab that will be opened on, this tab will contain information about the players current game, this tab will not feature any vert time sensitive information as the API can only get data 1 minute after it happens in game, however it will show useful information about players such as ranks that is not as time sensitive and is the same throughout the game.

### Resources Needed:

#### Software:

- Typescript
- Electron
- Deadlock-API
- Statlocker API
- JSON Parsing
- UI Framework
- Deadlock

#### Hardware:

- Development PC

#### External Resources:

- Documentation
- Online Coursesww
- API Providers

### Project Timeline and Schedule:

#### Phase 1 - Research and Planning - Week 1:

- API Testing for viable features
- UI Design Ideas
- Setup Required Software and Dependencies

#### Phase 2 - Core Development - Week 2/3:

- API Calling Systems
- Data Parsing System
- Overlay Window

#### Phase 3 - Feature Development - Week 4/5/6:

- Live Match Stats
- Match History
- Player Hero Stats
- Game Leaderboards

#### Phase 4 - Testing - Week 7:

- Performance Testing
- Bug Fixing
- Optimisation

#### Phase 5 - Finalisation - Week 8:

- Documentation
- Finishing UI Touches
- Release Project

### Q2: Gantt Chart

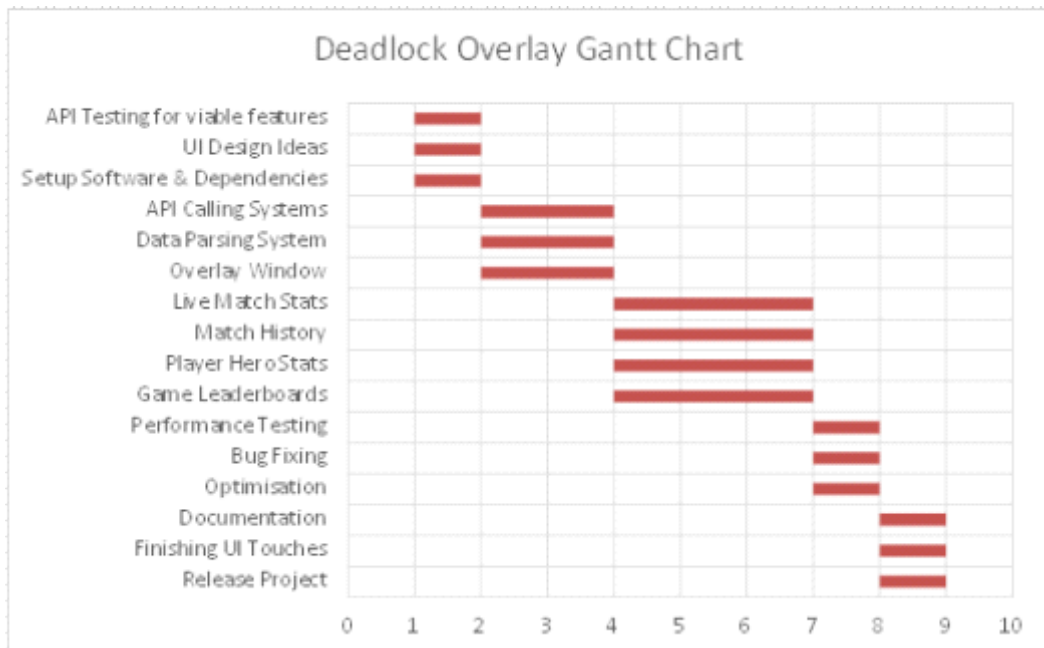


Figure 1: Project Gantt Chart

This Gantt chart shows what parts of the project will be worked on during which weeks, this matches the project timeframe shown above showing the progression plan of the project in a better way visually

### Q3: Risks and Challenges

#### API Constraints:

The biggest challenge with this project will be the restrictions on the current API, this will be an issue due to that fact that Valve does not provide their own API for Deadlock yet, this means that this project will have to use third party community API's, these being Deadlock-API [1] for raw game related information such as in game stats, and then a secondary API, Statlocker.gg [2] for additional information about games and players which is inferred from the raw data by themselves, these 2 API's should allow for good enough data to create a feature full overlay despite the lack of a first party API by Valve themselves, these API's also provide generous rate limits, with Deadlock-API providing 100 requests per second, and Statlocker.gg providing unlimited requests.

#### Performance Requirements:

As this is an overlay for a game, it will need to be always ran while the game is open, this means that it adds overhead to the game that needs to be reduced as much as possible to ensure that frame rate in game is not heavily affected by the app, to solve this I intend to use Typescript [3] and Electron [4] in order to create a well

optimised yet functional overlay.

## Data Availability:

One of the issues caused by having to make use of a third party API for data is that it cannot get all of the data from every game, in this case, data can only be retrieved from the top 200 matches at any given time, which is roughly equal to the top 50% of games, this means that live game features such as player ranks will not be available to players within these lower ranks, this is intended to be mitigated by ensuring that there are other features available on the overlay that do not rely on live game data and can be accessed by everyone, however this issue somewhat mitigates itself due to players within lower ranks generally being more casual at the game and having less need for features such as player ranks and other live statistics, as well as being less likely to install the overlay in general.

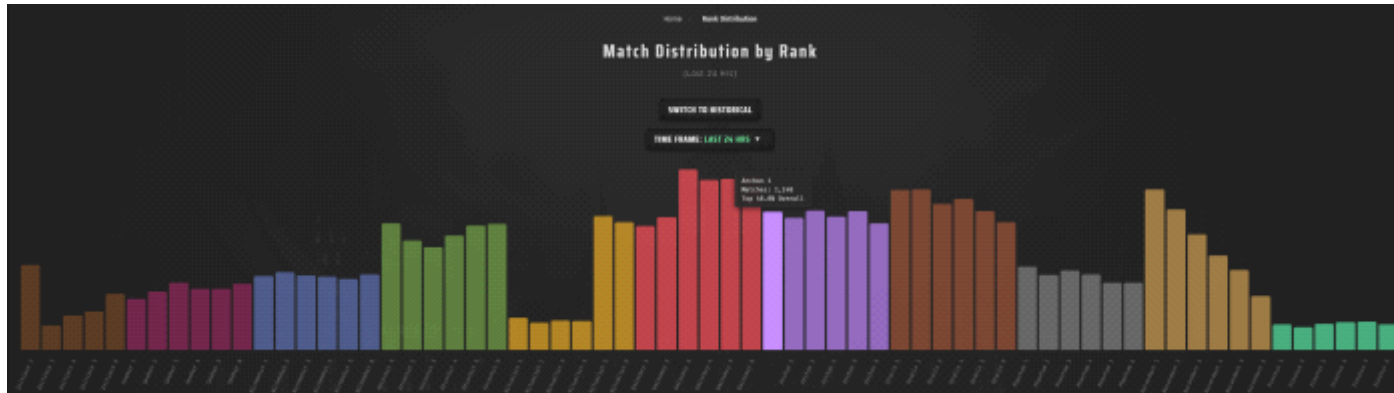


Figure 2: Match Rank Distribution [2]

## References

- [1] Deadlock API, "Deadlock Game Data API Documentation," Deadlock API. [Online]. Available: <https://deadlock-api.com/>. [Accessed: 21 Feb. 2026].
- [2] Statlocker, Statlocker API Documentation. [Online]. Available: <https://statlocker.gg/>. [Accessed: 21 Feb. 2026].
- [3] "The starting point for learning TypeScript." Accessed: Feb 21 2026. [Online]. Available: <https://www.typescriptlang.org/docs/>
- [4] "Introduction | Electron." Accessed: Feb 21 2026. [Online]. Available: <https://electronjs.org/docs/latest/>

# Task 3

22 February 2026 12:29

## Q1: Development Method

For this project I will be using an agile development method, this development method focuses on iterative development where smaller parts of the software are created in small stages instead of 1 large development cycle, this method fits the best with the given project as each part of the overlay does not really need to function with anything other than itself, this allows for different features to be developed and tested independently of anything else within the overlay. Then once each feature is functionally added and works, a couple passes can be done on it for UI design as well as merging the few things that may overlap with the rest of the project, such as specific API calls that may be used in multiple features.

This method will be applied by making sure that each feature / tab within the overlay is as separated from the rest of the project as best as possible allow for clear boundary's to be set on the scope of each feature.

The main reason agile development will be used for this project is due to the fact that the API's and main functionality of this overlay will be constantly changing, whether Valve provides more information in match salts that can be turned into more endpoints for the API, or if Valve was to create an API of their own, this evolving landscape means that one strict development plan from beginning to end would likely not work long term without making modifications at one point or another.

## Q2: Early Design Work

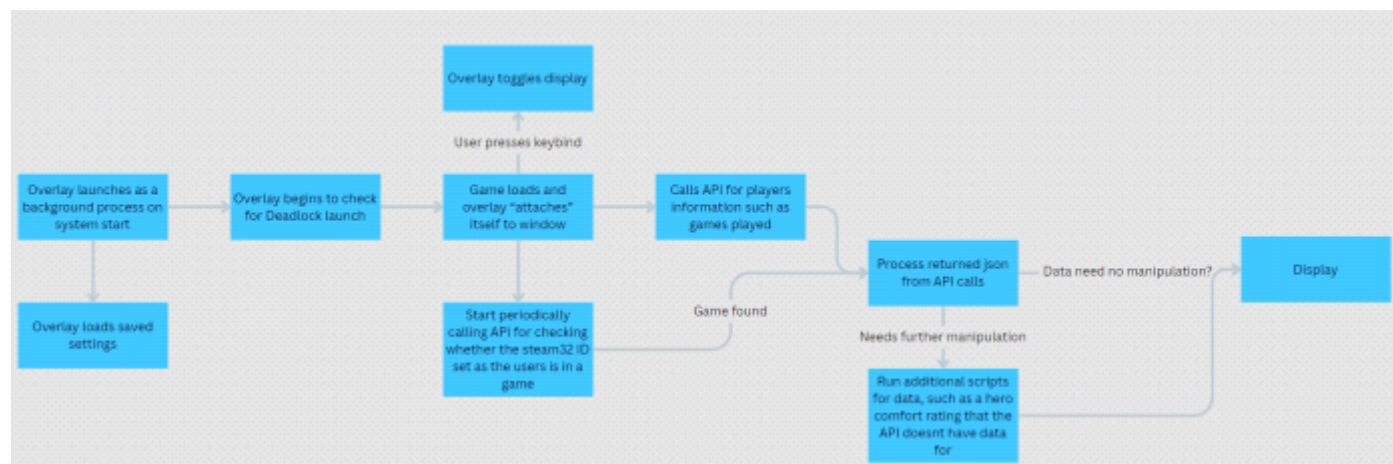


Figure 1: Overlay Flowchart [1]

This flowchart shows the general process of how the overlay is meant to work, showing each step of the process of making it work, detailing the process that data goes through before being displayed to the player as part of the overlay. A simple explanation of how it will work is first the overlay launches as a background process on the PC, it will then begin checking for currently running processes called "Deadlock", once it finds the deadlock process, it will start checking for an active live game, as well as begin populating information that doesn't need live game information such as player game history, winrate, their pp, and other general information, once a live game is found, it will then begin grabbing and processing all the information for that game, such as all the players in the lobby, their ranks, heroes, and their comfort on that character, once all processing is complete, it will then be displayed to the user.

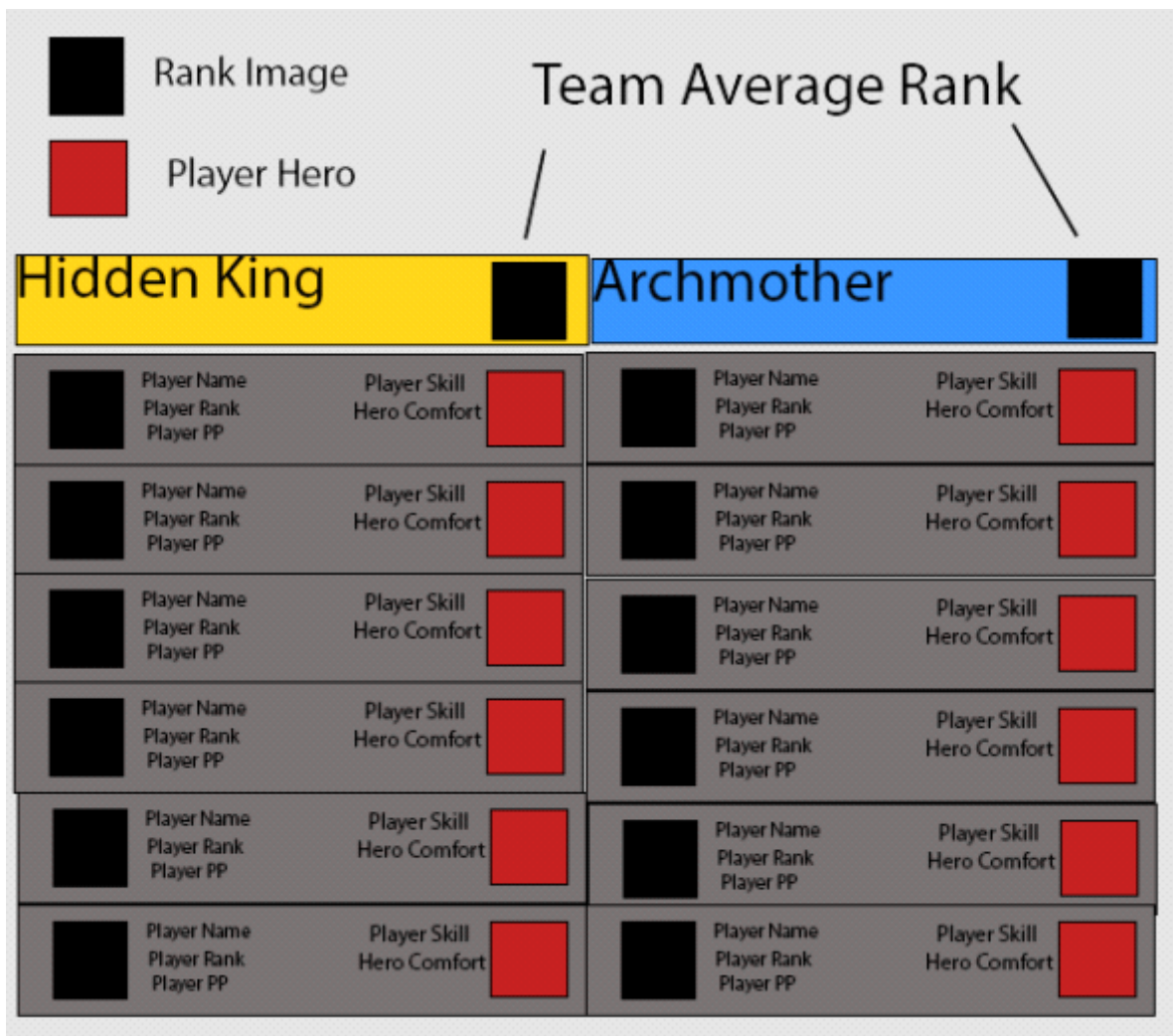


Figure 2: Drawing of live game tab [2]

This is a quick sketch up of the design that the live game tab will follow, showing player ranks, names, the heroes they are playing and how good they are at those heroes.

### Q3: Initial Implementation Work

#### Live Game Tracking:

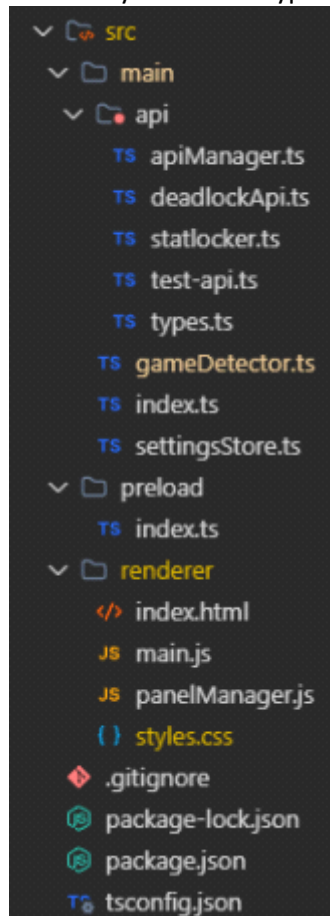
Below is a screenshot of a WIP version of the live game tracking part of the overlay, it features each players name, rank, and pp score with a pretty basic UI interface.

| TEAM 1                                                                                                                            | TEAM 2                                                                                                                          |
|-----------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
|  <b>#1 isamellon fan</b><br>Eternus 4<br>6495 PP |  <b>mutt meat</b><br>Eternus 6<br>6606 PP      |
|  <b>starrk</b><br>Eternus 5<br>6508 PP           |  <b>LovingSnek</b><br>Eternus 4<br>6418 PP     |
|  <b>ImBullet</b><br>Eternus 6<br>6872 PP         |  <b>Shadowfax</b><br>Eternus 6<br>6744 PP      |
|  <b>AShires</b><br>Eternus 6<br>6652 PP          |  <b>the_lavabolt</b><br>Eternus 6<br>6640 PP   |
|  <b>gwen gaming</b><br>Eternus 6<br>6732 PP      |  <b>hoc</b><br>Eternus 3<br>6352 PP            |
|  <b>Curse</b><br>Eternus 6<br>7132 PP            |  <b>twitch.tv/Xera</b><br>Eternus 6<br>7018 PP |

Figure 3: Deadlock Overlay Live Game Tab

## File Structure:

Here is also the current file structure for the very simple stating implementation, while this project was intended to be fully written in typescript, it ended up being easier to also have some things be done via js.



## References

[1] Canva Pty Ltd., "Canva – Visual communication platform," Canva. [Online]. Available: <https://www.canva.com>. [Accessed: Mar. 4, 2026].

[2] Adobe Inc., "Adobe Photoshop," Adobe. [Online]. Available: <https://www.adobe.com/products/photoshop.html>. [Accessed: Mar. 4, 2026].