

Assignment 1 – Innovation & Project Management

Ben Roberts (2205382)

Changemakers: Creativity and Value Creation (ACCA5033)

University of Wales Trinity Saint David – School of Applied Computing

Introduction

Tuesday, March 10, 2026 1:31 PM

Introduction

The purpose of this document is to evaluate the concept of innovation and to discuss a number of potential project ideas, after which one of those project ideas will be brought forth and built upon as a potential project.

Innovation

Tuesday, March 10, 2026 1:32 PM

Innovation

Innovation is defined as “new knowledge incorporated in products, processes, and services. [Afuah] classifies innovations according to technological, market, and administrative/organizational characteristics [1]”.

In 2016, the World Economic Forum identified six technologies which will shape the world, those being computing power, big data, digital health, digitisation of matter, the Internet of Things and the blockchain [2]. Within the area of healthcare specifically, technologies such as artificial intelligence, 3D printed joint replacements, gene editing, virtual and augmented reality, and bandages with integrated sensors [3]. NHS England states that innovation is vital in order to maintain efficiency and scalability whilst also maintaining a positive patient experience in addition to improving healthcare [4].

The fourth industrial revolution is defined as a period wherein further importance is placed on cyber-physical systems, resulting in a more globalised society. The fourth industrial revolution will involve the proliferation of technologies such as the Internet of Things, virtual reality, robotics and artificial intelligence. [5]

Kondratieff waves are a term given to a trend in economics wherein economic activity cyclically sees a significant upswing before an eventual downswing. These upswings often coincide with major technological innovations [6]. As an example, the Internet of Things has been cited as a potential major upswing as it can provide significant advances in the efficiency of the medical field, and it will be able to do so because of advances in internet connection speeds [7].

Innovation is heavily linked with creativity. When creative individuals such as scientists, engineers, artists, writers, etc. mix, they will drive the development of new innovations [8]. Scientists and engineers may come up with new ideas that they might not have come up with on their own, while other creatives such as artists and musicians might receive inspiration from one and other, potentially driving them to further innovate. Similarly, being in an environment that rewards creativity may encourage innovation even more [8].

Ideation

Tuesday, March 10, 2026 1:32 PM

Ideation

Ideally these projects would be produced using the Rust programming language as it includes a fully integrated toolchain, unlike other popular languages such as C++ which are generally fragmented. This integrated toolchain (named cargo) includes a compiler and library manager which will reduce the time taken in prototyping. It also removes the need for a cumbersome Integrated Development Environment.

The first project idea is a repository library designed to easily install mods for video games. In most instances, users are required to access websites to download mods and manually insert them into an external program which will add the necessary mods to the game. This would work by loading a definition file which will contain a variety of information, including a HTTP URL which points to the necessary files that will be downloaded. This library could then be used to produce game-specific frontends or even integrated directly into a mod loader.

Another project idea would be to create a game system for text-based role-playing adventure games. Similarly to the mod downloader, it would be implemented as a library and would have frontends written for it. This would allow it to be ported to any platform supported by Rust's standard library. It could also use the Mozilla Fluent library to support multiple languages.

As an alternative to writing projects associated with video games, making a project that compiles to WebAssembly utilising an extension of the cargo compiler. This would allow the program to run inside of a web browser instead of needing to be installed.

Since the first idea is the most developed, that one will be taken forward. The working project title will be "Mod Downloader" and will aim to produce an easily implementable framework for installing video game mods. It is to be written in Rust and should be designed for simplicity and ease of use with potential space for extensibility.

Research

Tuesday, March 10, 2026 1:32 PM

Research

The primary resource for every Rust project will be the official Rust documentation, specifically an online book source named *The Rust Programming Language*, which is frequently referred to by Rust programmers as “the book”. It is commonly used as it thoroughly explains the use of the cargo toolchain in addition to full explanations of the language’s standard library [9].

To better aid with project organisation and overall structure, a course on LinkedIn Learning that teaches how to use Cargo’s built-in project management features can be followed. This would provide any information necessary regarding the use of Cargo workspaces and library management [10].

Another source that could be used is the official documentation for the Serde library. This library can be used to serialise and deserialise data files (such as JSON), so following the documentation can provide necessary information on using the library to interpret repository definition files [11].

Project Scope

Tuesday, March 10, 2026 1:33 PM

Project Scope

In many cases the downloading and installing of video game mods can be difficult for novice users. In a lot of cases manual installation is required and complex guides will need to be followed. While there are many programs to automate this process, many of them are only designed to function with a single provider.

This project aims to solve this to create a utility library for downloading and installing video game modifications. This would handle repositories and downloading files. Then a game-specific frontend can be written to configure the correct file structure.

Required Technologies / Resources

Tuesday, March 10, 2026 1:33 PM

Required Technologies / Resources

- Cargo toolchain (includes Rust compiler, linter and code formatter)
- Rust documentation
- Serde library for formatting RON and JSON data into structs
- HTTP library for file downloading
- Set of mod files for testing and demonstration
- Example program using the library
- Desktop computer with Linux operating system
- Virtual machine with Windows 11 for testing
- Code editor
- Git version control system for uploading to a remote repository

Project Timeline

Tuesday, March 10, 2026 1:34 PM

Project Timeline

The project will ideally be conducted over the course of one week.

The first day of the project should be focused on researching for the project and designing the specification. The second day will be used to produce a proof of concept.

Preferably, the project should be finished before the end of May at the latest.

Task Schedule

Tuesday, March 10, 2026 1:34 PM

Task Schedule

This task schedule was written to approximate the amount of time each task would take for the purposes of writing a Gantt chart.

Task Name	Duration
Initialise cargo project	2 hour
Research libraries	4 hour
Install dependencies	2 hour
Create basic specification for repository format	6 hour
Choose sample files	1 hour
Produce proof-of-concept	4 hour
Create online Git repository	1 hour
Convert proof-of-concept binary into library	8 hour
Produce an example tool using the library	4 hour
Test for issues	4 hour
Optimise library	4 hour
Do a final testing pass before release	5 hour
Ensure Git repository is up to date	1 hour
Add a Git release containing the minimum viable product	1 hour
Publish first version	1 hour

Gantt Chart

Tuesday, March 10, 2026 1:35 PM

Gantt Chart

A Gantt chart was produced to plan tentative timings for each phase of the project. The main intention for the project timeline is to take place over the course of a working week (excluding weekends) for a maximum of eight hours a day between 9AM and 5 PM. Each part is sectioned to prevent overlap as the Waterfall development model follows a strict step-by-step process.

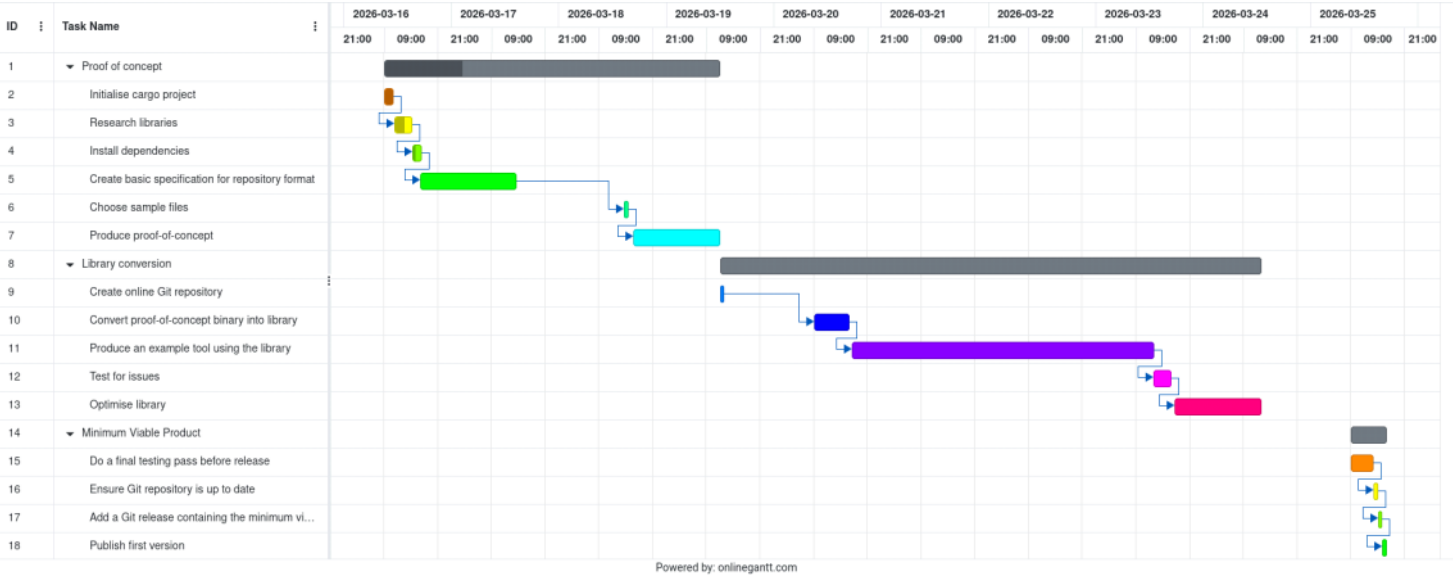


Figure 1: Gantt chart containing the current project timeline

Risk Register

Tuesday, March 10, 2026 1:36 PM

Risk Register

A risk register template was used to quickly produce a list of potential risks that are imposed on the project, as well as mitigation and contingency plans in case those risks come up.

Risk Register																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
---------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Figure 2: Risk register

Development Methodology

Tuesday, March 10, 2026 1:35 PM

Development Methodology

This program will be designed using the Waterfall method. This is because the project only has a single developer, so complicated methodologies such as Agile may be unfeasible to follow within a short timeframe.

The Waterfall method involves following a step-by-step process in which steps flow sequentially into each other [12]. Every specific phase will allow set points to get feedback on the project. While Waterfall is less flexible than other methodologies such as Agile, its rigidity makes it easier to understand and will also allow for a more consistent product.

Design Work

Tuesday, March 10, 2026 1:40 PM

Design Work

The following lines are a MoSCoW prioritisation specification, which is used to categorise features based on importance with “Must Have” being essential for the project’s success, “Should Have” being ideal to try and include if possible, “Could Have” being nonessential additions that can be included if there is enough time or budget left and “Will Not Have” (often shortened to “Won’t Have”) being deemed out of scope for the release [13].

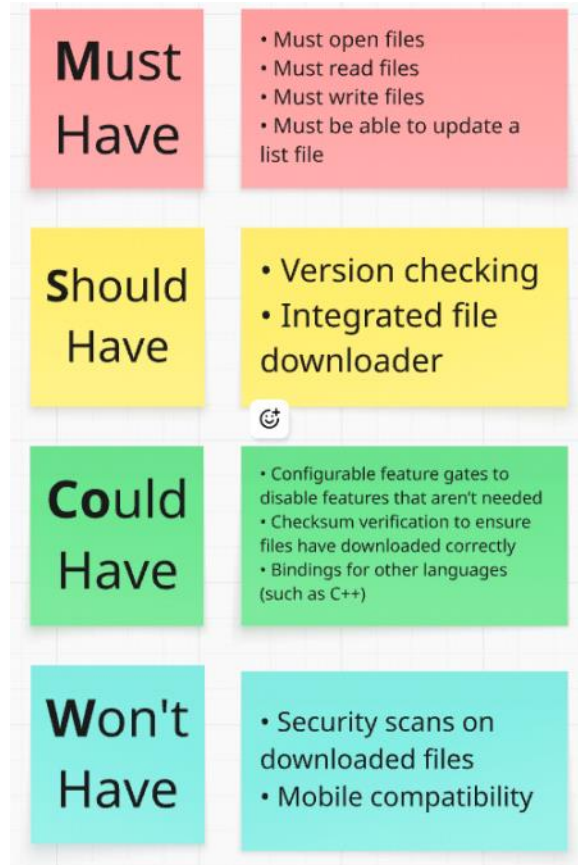


Figure 3: Simple MoSCoW requirements list for the project

Additionally, some user stories were formulated in order to make it easier to come up with ideas for the project. These user stories are intended to describe a potential function and its use case [14]. Each user story has been separated into either a first or second release based on importance, with release one being the Minimum Viable Product.

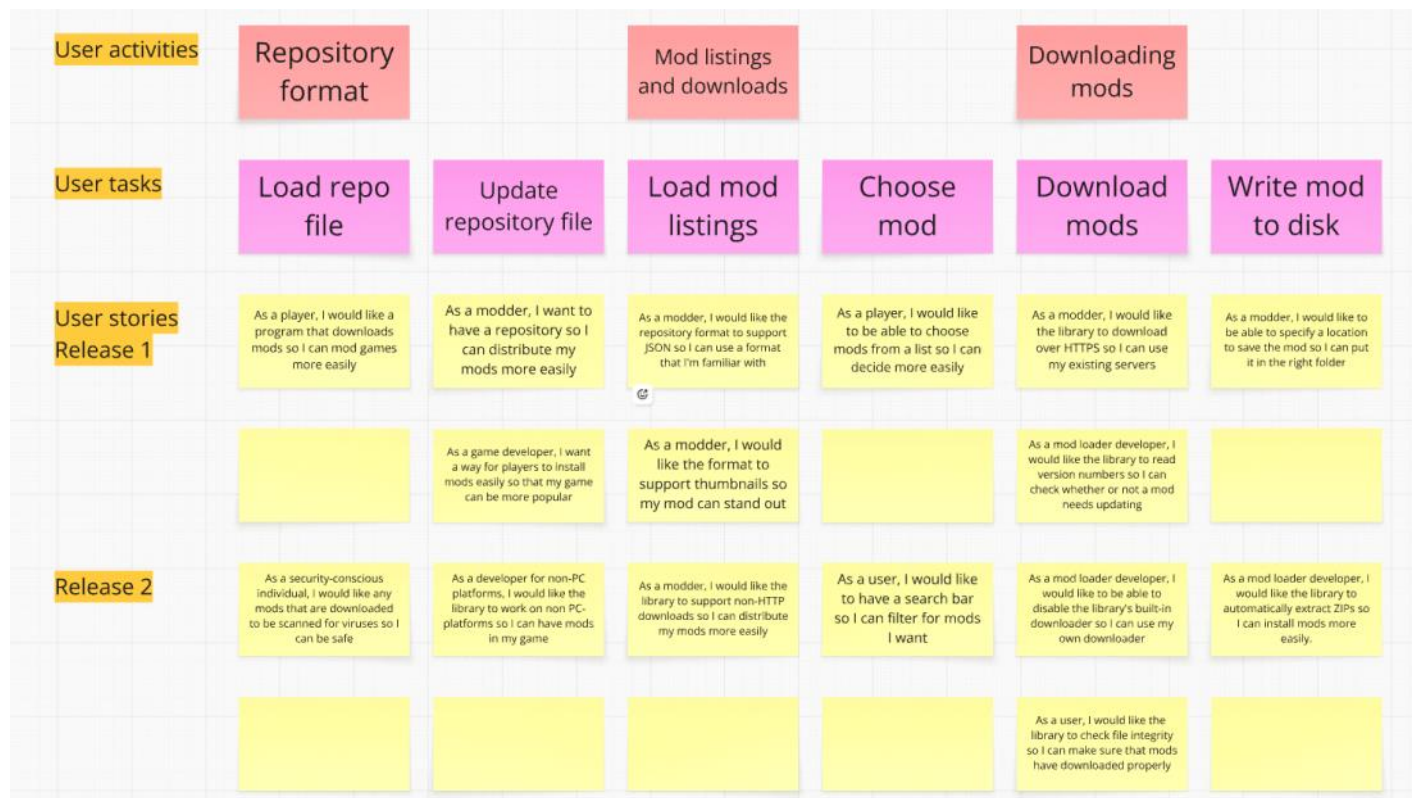


Figure 4: User stories for two potential releases of the mod downloader project

Initial Practical Implementation

Tuesday, March 10, 2026 1:41 PM

Initial Practical Implementation

A simple proof-of-concept prototype for the project was produced to check its feasibility.

Cargo can automatically generate documentation for a program using comments in Rust code, so the first thing done for the

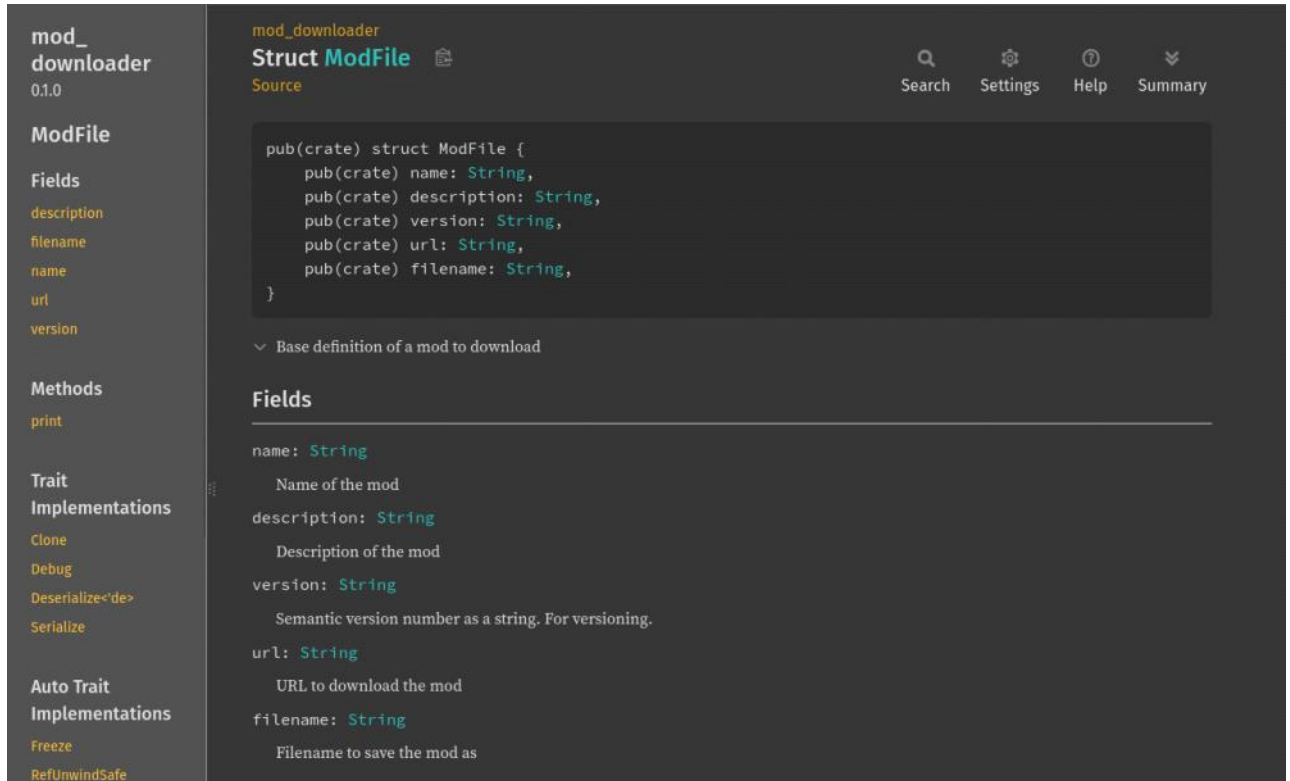


Figure 5: Rustdoc screenshot showing automatically generated documentation for the prototype's *ModFile* struct

The prototype was written in Rust and uses the reqwest library to handle downloading. Unlike the intention for the final product, it functions as a normal program and is executed on its own.

```
// Mod Downloader (working title)
// An example implementation of a module installer for video games that uses HTTP
use serde::{Deserialize, Serialize};
use std::env;
use std::fs;
use std::{fs::File, io::Write};
//use semver::{BuildMetadata, Prerelease, Version, VersionReq};

/// Definition for a mod repository and its listings
#[derive(Debug, Clone, Deserialize, Serialize)]
struct ModRepo {
    /// Name of the modrepo that appears on the list
    name: String,
    /// URL used to pull modfile refreshes from
    url: String,
    /// Contents of the modrepo as a RON
    contents: Vec<ModFile>,
}
```

```

    /// Contents of the modrepo as a RON
    contents: Vec<ModFile>,
}

/// Base definition of a mod to download
#[derive(Debug, Clone, Deserialize, Serialize)]
struct ModFile {
    /// Name of the mod
    name: String,
    /// Description of the mod
    description: String,
    /// Semantic version number as a string. For versioning.
    version: String,
    /// URL to download the mod
    url: String,
    /// Filename to save the mod as
    filename: String,
}

impl ModFile {
    fn print(&self) {
        println!(
            "Name: {} \nDescription: {} \nVersion: {} \nURL: {} \nFilename: {}",
            self.name, self.description, self.version, self.url, self.filename
        )
    }
}

// Might need from_ron and from_json methods in the future but it's fine for now.

fn download_file(url: &str, filename: &str) -> Result<(), Box<dyn std::error::Error>> {
    println!("Downloading file, please wait patiently");

    let download = request::blocking::get(url)?.bytes();
    if download.is_err() {
        println!("Download failed!");
        Ok(()) // Placeholder behaviour
    } else {
        println!("Download finished. Writing to file.");

        let mut file = File::create(filename).expect("Could not create save file!");
        let _ = file.write_all(&download.unwrap());
        println!("Done!");

        Ok(())
    }
}

fn main() -> Result<(), Box<dyn std::error::Error>> {
    println!("The Mod Downloader!\n");

    let args: Vec<String> = env::args().collect();
    //dbg!(&args);
    println!("{}", args.len());
    if args.len() <= 1 {
        println!("No arguments passed. Please point this program to a file.");
        Ok(())
    } else {
        let path = args[1].as_str();

        let mut load = fs::read_to_string(path).unwrap();
        let mut repo: ModRepo =
            ron::from_str(load.as_str()).expect("Couldn't interpret the passed RON file!");

        // Update load
        println!("Refreshing repository file");
        let mut repo_file: File = File::options().read(true).write(true).open(path)?;
        let write = request::blocking::get(repo.url)?.text()?;
        repo_file
            .write_all(write.as_bytes())
            .expect("Could not write refreshed repository listing to filesystem");
        println!("Repo refreshed! Reopening...");

        load = fs::read_to_string(path).unwrap();
    }
}

```

```
println!("Repo refreshed! Reopening...");

load = fs::read_to_string(path).unwrap();
repo = ron::from_str(load.as_str()).expect("Couldn't interpret the passed RON file!");

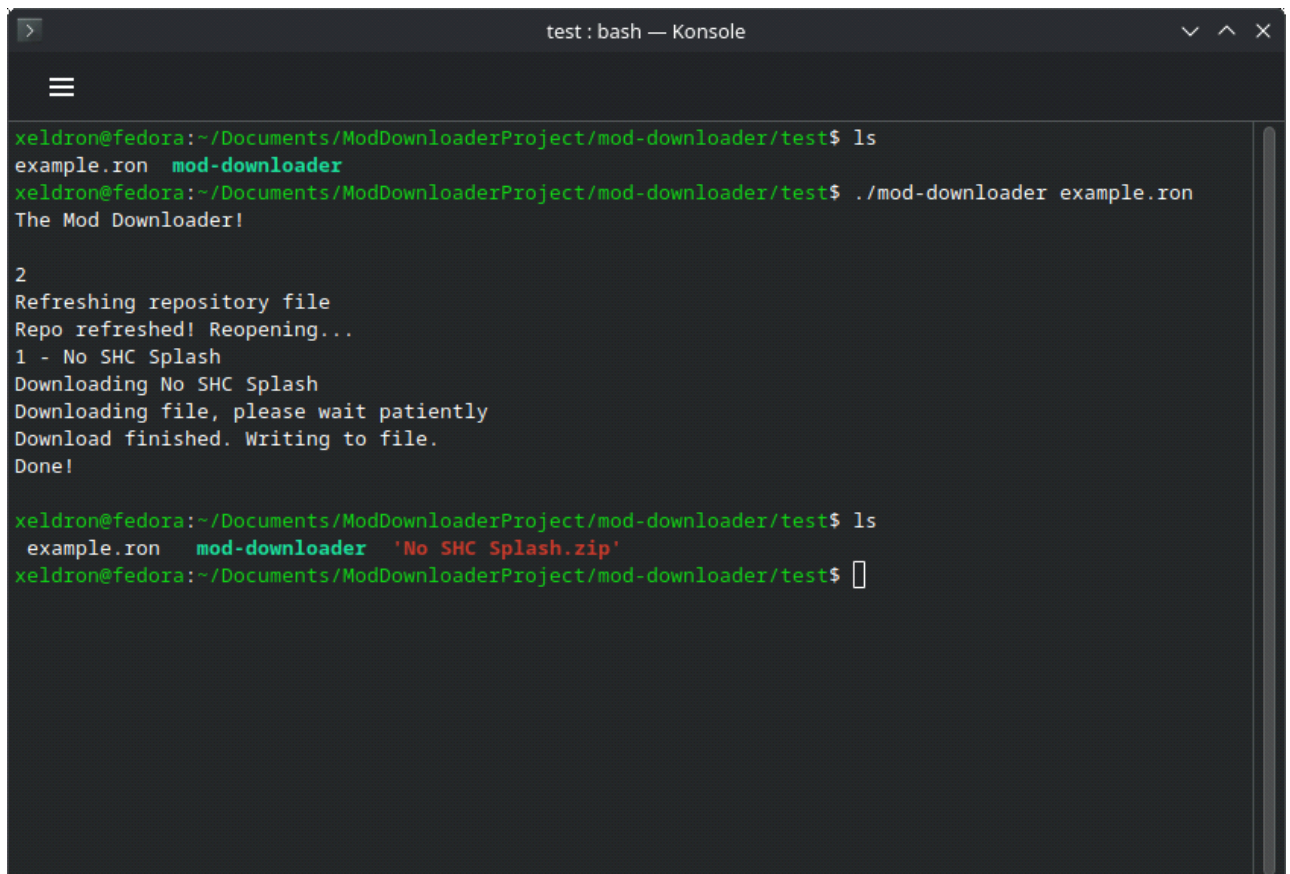
for counter in 0..repo.contents.len() {
    println!("{}", counter + 1, repo.contents[counter].name);

    println!("Downloading {}", repo.contents[counter].name);
    download_file(
        &repo.contents[counter].url,
        &repo.contents[counter].filename,
    )
    .expect("Failed to download mod!");
    println!()
}

Ok(())
}
```

Figure 6: The code for the proof-of-concept prototype

File output from the prototype, using a sample file which downloads a small ZIP file from an online source.



```
test : bash — Konsole

xeldron@fedora:~/Documents/ModDownloaderProject/mod-downloader/test$ ls
example.ron  mod-downloader
xeldron@fedora:~/Documents/ModDownloaderProject/mod-downloader/test$ ./mod-downloader example.ron
The Mod Downloader!

2
Refreshing repository file
Repo refreshed! Reopening...
1 - No SHC Splash
Downloading No SHC Splash
Downloading file, please wait patiently
Download finished. Writing to file.
Done!

xeldron@fedora:~/Documents/ModDownloaderProject/mod-downloader/test$ ls
example.ron  mod-downloader  'No SHC Splash.zip'
xeldron@fedora:~/Documents/ModDownloaderProject/mod-downloader/test$
```

Figure 7: File output

References

Thursday, March 12, 2026 11:30 AM

References

- [1] '(PDF) The definition and classification of innovation', *ResearchGate*, doi: 10.1515/hjbpa-2017-0005.
- [2] World Economic Forum, 'WEF Annual Report 2016-2017', World Economic Forum, 2016. Accessed: May 19, 2025. [Online]. Available: https://www3.weforum.org/docs/WEF_Annual_Report_2016_17.pdf
- [3] '5 innovations that are revolutionizing global healthcare', World Economic Forum. Accessed: May 19, 2025. [Online]. Available: <https://www.weforum.org/stories/2023/02/health-future-innovation-technology/>
- [4] N. H. S. England, 'NHS England » Innovation'. Accessed: May 19, 2025. [Online]. Available: <https://www.england.nhs.uk/ourwork/innovation/>
- [5] J. Naidoo, 'Teaching and Learning in the 21st Century: Embracing the Fourth Industrial Revolution', in *Teaching and Learning in the 21st Century*, Brill, 2021. Accessed: Jun. 16, 2025. [Online]. Available: <https://brill.com/display/title/60268>
- [6] A. Korotayev, J. Zinkina, and J. Bogevoľnov, 'Kondratieff waves in global invention activity (1900–2008)', *Technol. Forecast. Soc. Change*, vol. 78, no. 7, pp. 1280–1284, Sep. 2011, doi: 10.1016/j.techfore.2011.02.011.
- [7] M.-C. Hu, F. Phillips, and C.-Y. Wu, 'Social innovations mediate technology and economy in the Kondratieff waves', *Technol. Forecast. Soc. Change*, vol. 201, p. 123262, Apr. 2024, doi: 10.1016/j.techfore.2024.123262.
- [8] K. Stolarick and R. Florida, 'Creativity, Connections and Innovation: A Study of Linkages in the Montréal Region', *Environ. Plan. A*, vol. 38, no. 10, pp. 1799–1817, 2006, doi: 10.1068/a3874.
- [9] S. Klabnik, C. Nichols, and C. Krycho, 'The Rust Programming Language - The Rust Programming Language'. Accessed: May 19, 2025. [Online]. Available: <https://doc.rust-lang.org/stable/book/#the-rust-programming-language>
- [10] 'Manage and test Rust projects | LinkedIn Learning'. Accessed: Feb. 26, 2026. [Online]. Available: <https://www.linkedin.com/learning/advanced-rust-managing-projects/manage-and-test-rust-projects?resume=false&u=77966930>
- [11] 'Overview · Serde'. Accessed: May 20, 2025. [Online]. Available: <https://serde.rs/>
- [12] A. Saravanos and M. X. Curinga, 'Simulating the Software Development Lifecycle: The Waterfall Model', *Appl. Syst. Innov.*, vol. 6, no. 6, Nov. 2023, doi: 10.3390/asi6060108.
- [13] V. Stray, K.-J. Stol, M. Paasivaara, and P. Kruchten, *Agile Processes in Software Engineering and Extreme Programming*, vol. 00445. in *Lecture Notes in Business Information Processing*, vol. 00445. Cham, Switzerland: Springer International Publishing, 2022. Accessed: Mar. 05, 2026. [Online]. Available: <https://research.ebsco.com/linkprocessor/plink?id=2a31cc6b-1d15-34a5-b0c6-6930b275b7f7>
- [14] 'Writing user stories', GOV.UK. Accessed: Mar. 10, 2026. [Online]. Available: <https://www.gov.uk/service-manual/agile-delivery/writing-user-stories>

LinkedIn Learning

Thursday, March 12, 2026 11:31 AM

LinkedIn Learning

The course “Advanced Rust: Managing Projects” was done on LinkedIn learning. The course URL is <https://www.linkedin.com/learning/advanced-rust-managing-projects>.

The course description was “Are you looking for the next step to advance from the initial Rust essential training to manage growing Rust projects and writing automated test routines? In this course, instructor Barron Stone has exactly what you need. He explains how to manage projects using the Rust module system and explores topics such as using the “mod” keyword to define modules, navigating paths in Rust, and identifying parts of a program as public or private. Barron shows you how to automate tests, using test functions, assert! macros, custom failure messages, unit tests, integration tests, and more. This course can help you take the next step in your learning journey for mastering Rust.”

This course involves testing and managing Rust projects using Cargo, which will be essential for developing the project.

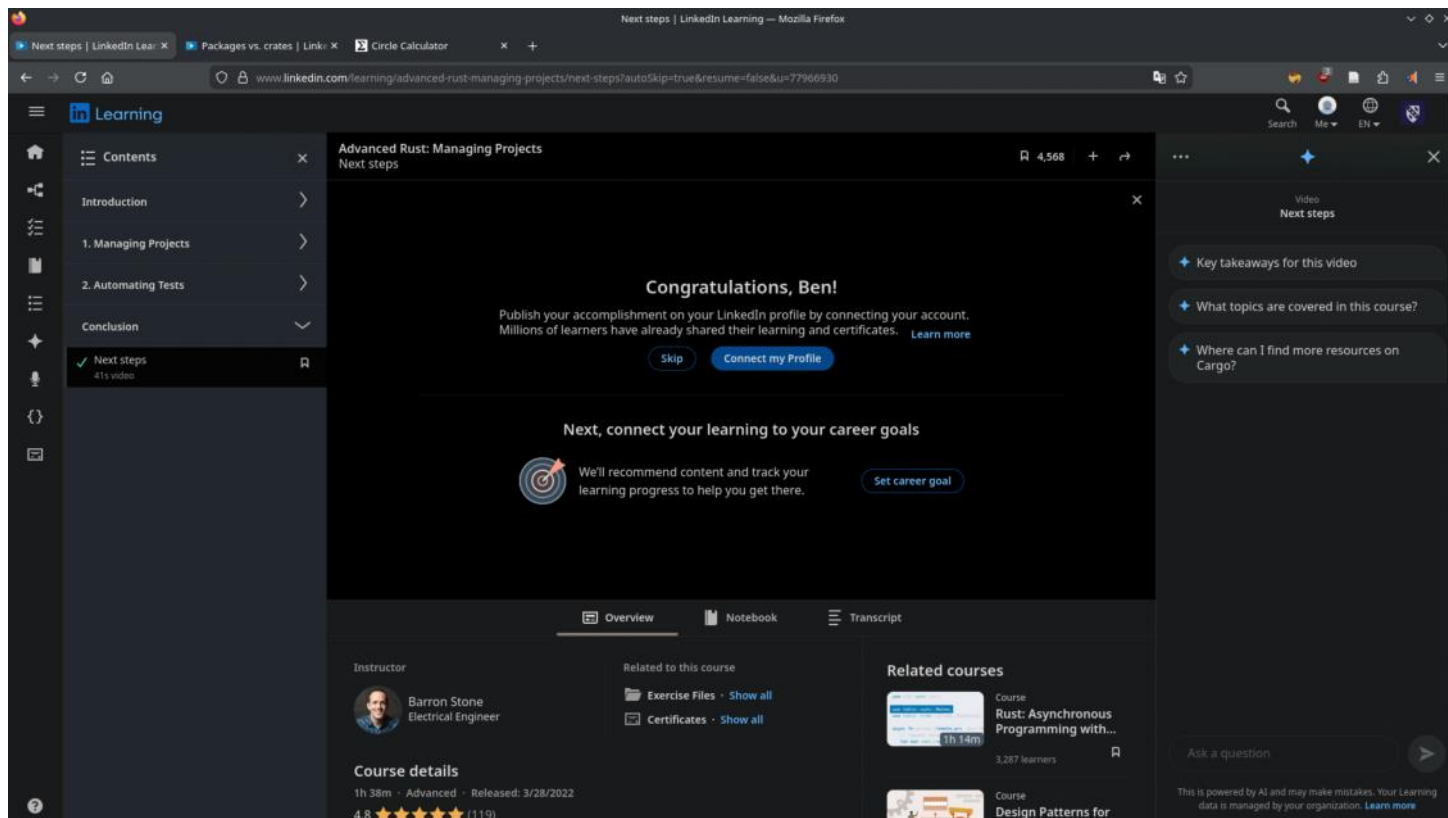


Figure 5: Screenshot of the completed "Advanced Rust: Managing Projects" course



Advanced Rust: Managing Projects

Course completed by Ben Roberts
Feb 11, 2026 at 02:23PM UTC • 1 hour 38 minutes

Top skills covered

Rust (Programming Language)

A stylized, handwritten signature in blue ink, reading "Shea Hanson".

Shea Hanson, Head of Learning Content Strategy

Certificate ID: eacfl43c383052cc2e9e8a682e47595eb94af9b6b8d918d062ed7cbead0facd7



Figure 6: Certificate for completing the course