Course Project

The course project represents a more substantial understaking than the course assignments, and involves a team of 2-3 team members. Deliverables of the course project includes a *project proposal* (due on **October 6, 2025**, Monday), as well as a *final set of project deliverables* that includes:

- The *source code*, including a `README.md` file, and submitted as a public or private git repository on GitHub. Needless to say, the source code should be completely written in Rust.
- A *video slide presentation* with a duration of 5-10 minutes. Its URL should be included in the `# Video Slide Presentation` section in your `README.md` file.
- A *video demo* with a duration of 3-10 minutes. Its URL should be included in the `# Video Demo` section of your `README.md` file.
- A *final report* of the project to ensure full reproducibility, included in the `# Final Report` section in your `README.md` file.

Project Ideas

The course project represents a more innovative, demanding, and time-consuming piece of work than the course assignments. There are no prescribed requirements for the nature of your course project: it can be a new software framework similar to any of the frameworks on crates.io, a new standalone command-line utility, a new web application that involves a web backend, or even a new pull request that adds a feature to — or fixes a critical issue in — an existing open-source project. The only requirement is that the source code of your course project should be 100% written in Rust. It would also be desirable for most third-party frameworks that the project uses to be mostly written in Rust as well. With the prevalence of large language models designed specifically for coding assistance, you may be able to build a project that is quite ambitious and novel within a matter of weeks.

The course project involves three stages: building a team, writing a project proposal, and completing the project by the due date. Here are some inspiring project ideas that may help you define your own project. However, your mileage may vary as teams will have very different backgrounds and skill sets, and these project ideas only serve as sources of inspiration. Your team's project may be substantially less (or more) challenging than these project ideas.

1 Personal Finance Tracker

Design and build a new command-line utility that implements a personal finance tracker, assisting users to manage their income and expenses in various categories.

Key Features

- Database stored in a HTTPS back-end server
- Transaction logging (income and expenses)
- Categories
- Different types of accounts, including checking and credit cards
- Ability to enter complex transactions, such as a transaction with expenses in different categories
- Reconciliation of accounts
- Optionally, Budgeting tools and financial reports

Recommended Tech Stack

Ratatui for text user interface, Axum, Rocket, or Actix Web for the backend, Diesel or SQLx for database interactions.

2 Web Crawler with Data Analysis

Design and build a new web crawler that collects data from various websites and performs basic analysis.

Key Features

- Asynchronous crawling using stackless coroutines to speed up data collection
- Data parsing and extraction using HTML parsers
- Basic data analysis (*e.g.*, word frequency, sentiment analysis)
- Text user interface to display results

Recommended Tech Stack

Ratatui for text user interface, Reqwest for HTTP requests, Html5ever or Scraper for HTML parsing, and Tokio for asynchronous operations using stackless coroutines.

3 Simple Game Engine

Design and build a new and simple game engine that can be used to create basic 2D games.

Key Features

- Scene management (loading and switching scenes)
- Entity component system (ECS) for game objects

Recommended Tech Stack

Bevy for the foundation of the game engine, and SDL2 for rendering.

4 Large Language Model Inference Service

Design and build a new back-end inference service to serve large language models with support for streaming.

Key Features

- Load and manage multiple large language models
- API endpoints for clients
- Streaming support for inference results

- Basic chat interface to interact with the back-end inference service

Recommended Tech Stack

Candle, Burn, or Mistral.rs for model inference, Axum, Rocket or Actix Web for the backend.

5 Large Language Model Fine-Tuning Engine

Design and build a Rust-powered training engine that fine-tunes large language models using a user's private data.

Key Features

- Fine-tunes any large language model from Hugging Face
- Compatible with at least two mainstream GPU backend types, such as Nvidia/CUDA and Apple/Metal.
- Supports a terminal-based text user interface
- Supports inference after the model is fine-tuned

Recommended Tech Stack

Burn or Candle for model training and inference, and Ratatui for the text user interface.

6 Simple LLM-Powered CLI

Design and build a simple CLI (command-line interface) that is powered by large language models, similar to (but much less ambitious than) existing open-source CLIs, such as Codex CLI and AIChat.

Key Features

- Maintains context-aware sessions

- Integrates available tools, such as MCP servers or editors supporting the Agent Client Protocol (ACP), to automate tasks
- Supports agentic workflows

Recommended Tech Stack

Local model inference such as Ollama, and Ratatui for the text user interface.

7 Simple Terminal Emulator

Rust-powered terminal emulators, such as Alacritty, WezTerm, Rio, and Warp, are blazingly fast due to Rust's performance benefits. The objective of this project idea is to build a simple terminal emulator in Rust.

Key Features

- Supports a single window
- Supports basic commands
- Supports colours

Recommended Tech Stack

Third-party core frameworks in crates.io, such as Rayon and Serde. In particular, gl for more sophisticated GPU-accelerated rendering with OpenGL, Termion or Crossterm for simpler text-based rendering.

8 New Pull Request in Open-Source Projects

If you are considering the possibility of contributing a Pull Request to an existing open-source project, you may wish to choose a project that is actively being maintained and updated. Here are some of the instructor's favourites:

- Zed, a next-generation text editor written from scratch in Rust (GitHub)
- Codex CLI, a new Rust-powered coding CLI from OpenAI

- GitButler, a new and different git client (GitHub)
- Tauri, a UI framework for building tiny and fast binaries for all major desktop (macOS, linux, windows) and mobile (iOS, Android) platforms (GitHub)
- Ratatui, a terminal-based Rust-powered text UI framework
- Pagefind, a fully static search library that aims to perform well on large sites (GitHub)
- Burn, a Rust-powered framework for training machine learning models (GitHub)
- uv, an extremely fast Python package and project manager, written in Rust. Think of it as Cargo for Python. (GitHub)
- Ruff, an extremely fast Python linter and code formatter, written in Rust. (GitHub)

Project Proposal

The project proposal should be written as a single file in the form of a Markdown document, named `README.md`, of no more than 2000 words in total length. Other formats (such as Microsoft Word or Adobe PDF) will not be accepted, because, at the end of the day, software developers *should* use Markdown as their documentation format. This file should be included in the top-level directory of your GitHub repository, and your submission will be a web URL pointing to your GitHub repository. The project proposal should include the following aspects of the project, described clearly and concisely:

1 Motivation

What motivated your team to spend time on this project? An excellent project idea is satisfying and fun to work on, and fills a gap that may not be easily found in the Rust ecosystem.

2 Objective and key features

What is the objective of this project? What are the key features to be built in the project to achieve this objective? In other words, what is the idea that the completed

project is trying to implement? It would be excellent if the idea has some novelty, but it is also important that it is feasible to be implemented within the timeframe of this course. Novelty is represented by the fact that the project, while small in scale, may be filling a gap in the current Rust ecosystem.

3 Tentative plan

Briefly and concisely, describe how your team plans to achieve the project objective in a matter of weeks, with clear descriptions of responsibilities for each team member in the team. As the duration of the project is quite short, there is no need to include milestones and tentative dates.

4 Marking rubrics

- **Motivation**: 30% (out of 10 Points)
    - The motivation is sufficiently convincing, showing that the team has thought about the project thoroughly (10 Points)
    - The motivation is lackluster and not convincing. (6 Points)
    - The motivation is not mentioned in the proposal. (0 Point)
- **Objective and key features**: 30% (out of 10 Points)
    - The objective and key features of the proposal may be filling a gap in the current Rust ecosystem. (3 Bonus Points)
    - The objective and key features of the proposal are clearly defined, with a reasonable amount of work for each team member. (10 Points)
    - The objective and key features of the proposal, are not clearly defined; or the amount of work for each team member is not well defined or insufficient. (6 Points)
    - The objective of the proposal is not mentioned in the proposal. (0 Point)
- **Tentative plan**: 40% (out of 10 Points)
    - The proposed plan is concise and clear, includes responsibilities for each team member, and a casual reader can be convinced that the project can be

reasonably completed by the project due date. (10 Points)

- The proposed plan has been included, but not clear to a casual reader. (6 Points)
- The proposed plan is not comprehensible. (0 Point)

5 Submission

Submit a single URL — the URL to your team's GitHub repository — to the assignment labeled **Project Proposal** in the Quercus course website. Each member of the team should make his/her own submission, but obviously all members in the same team should submit the same URL. If your GitHub repository is private, add the instructor (GitHub username `baochunli`), to your repository so that it can be accessed.

> ⚠️ The deadline for the project proposal is **Monday, October 6, 2025**, at **11:59pm** Eastern time, and late submissions will **not** be accepted. Do remember to add the instructor (username `baochunli`) to your GitHub repository, if it is private, **before** the deadline.

Final Project Deliverables

The final set of course project deliverables should be submitted as a URL to a public or private GitHub repository[1]. If your repository is private, add the instructor as a collaborator so that it can be read. As previously stated at the beginning of this document, your GitHub repository needs to contain all four pieces of information: a *final report* in a Markdown document `README.md`, your *source code*, your *video slide presentation*, and your *video demo*.

1 Final Report

A `README.md` file that contains the final report, in the form of a Markdown document, of no more than 5000 words in total length[2][3]. If you wish to include images (such as screenshots) in the final report, make sure that it can be visible when the instructor

visits your GitHub repository with a web browser. The final report should include the following logistical and technical aspects of the project, described clearly and concisely:

- The **names**, **student numbers**, and the **preferred email addresses** of all team members. Email messages requesting clarification may be sent to the team, so make sure that these emails can be successfully delivered and read.
- **Motivation:** What motivated your team to spend time on this project? An excellent project idea is satisfying and fun to work on, and fills a gap that may not be easily found in the Rust ecosystem.
- **Objectives:** What are the objectives of this project?
- **Features:** What are the main features offered by the final project deliverable?
- **User's (or Developer's) Guide:** How does a user — or developer, if the project is a crate — use each of the main features in the project deliverable?
- **Reproducibility Guide:** What are the commands needed to set up the runtime environment, if any, and to build the project, so that its features can be used by a user or a developer? *Note:* The instructor will follow the steps you have included in this section, step-by-step, with no deviation. The instructor has access to a Ubuntu Linux server and a macOS Sonoma laptop computer.
- **Contributions by each team member:** What were the individual contributions by each member in the team?
- **Lessons learned and concluding remarks:** Write about any lessons the team has learned throughout the project and concluding remarks, if any.

2 Source Code

You need to include all the source code, in Rust, required to build the project. The number of lines in your source code, counted by using the `cloc` utility and excludes blank lines and comments, must be more than 500 lines per team member. For example, if your team has 2 members, your source code needs to have more than 1000 lines of code. Do not include binary artifacts, such as builds in the `target/` directory, within your GitHub repository.

3 Video Slide Presentation

You need to include a video slide presentation, of no shorter than 5 minute and no longer than 10 minutes. Its URL should be included in the `# Video Slide Presentation` section of your `README.md` file. You can feel free to place the video at any website, such as YouTube, Dropbox, or Google Drive. If the video file is under 100 MB, you can even include it directly in the GitHub repository.

4 Video Demo

You need to include a video demo, of no shorter than 3 minutes and no longer than 10 minutes. Its URL should be included in the `# Video Demo` section of your `README.md` file. You can feel free to place the video at any website, such as YouTube, Dropbox, or Google Drive. If the video file is under 100 MB, you can even include it directly in the GitHub repository.

5 Marking Rubrics

- **Motivation and Objectives**: 20% (out of 10 Points), to be marked by reading the final report `README.md`
    - The project is well motivated and the objectives are very clear. (10 Points)
    - The motivation is lackluster *or* the objectives are not clear. (8 Points)
    - The motivation is lackluster *and* the objectives are not clear. (6 Points)
    - The project is not motivated *or* the objectives are not discussed. (4 Points)
    - The project is not motivated *and* the objectives are not discussed. (0 Points)
- **Features**: 30% (out of 10 Points), to be marked by reading the final report `README.md` and watching the video demo
    - The main features of the project have been clearly introduced, and *all* of them add value towards achieving the project objectives. (10 Points)
    - The main features of the project have been clearly introduced, and *most* of them add value towards achieving the project objectives. (8 Points)

    - Some of the features in the project have *not* been clearly introduced, *or* more than half of them do not add value towards achieving the project objectives. (6 Points)
    - Some of the features in the project have *not* been clearly introduced, *and* more than half of them do not add value towards achieving the project objectives. (4 Points)
    - The main features of the project have *not* been introduced. (0 Points)
- **Reproducibility**: 30% (out of 10 Points), to be marked by following the reproducibility guide and user's (developer's) guide described in the final report `README.md`, working with the source code directly, and watching the video demo

    - The runtime environment can be properly set up, the project can be successfullu built, and all features can be tested, without sending any emails to the team requesting clarification. (10 Points)
    - One email to the team requesting clarification is needed to properly set up the runtime environment, build the project successfully, and test all the features. (8 Points)
    - Two emails to the team requesting clarification are needed to properly set up the runtime environment, build the project successfully, and test all the features. (6 Points)
    - Three emails to the team requesting clarification are needed to properly set up the runtime environment, build the project successfully, and test all the features. (4 Points)
    - Even after three emails to the team requesting clarification, the runtime environment cannot be properly set up, or the project cannot be built successfully, or some features cannot be tested. (0 Point)
- **Individual Contributions**: 20% (out of 10 Points), individual marks will be assigned to each team member. This category will be marked by reading the final report `README.md`, and reading commit messages in the commit history in the GitHub repository.

    - The team member has made a fair amount of contributions to the project, without these contributions the project cannot be successfully completed on time. (10 Points)
    - The team member has made less than a fair amount of contributions to the project, *or* without these contributions the project can still be successfully

completed on time. (6 Points)

- The team member has made less than a fair amount of contributions to the project, *and* without these contributions the project can still be successfully completed on time. (4 Points)
- The team member has not made any contributions to the project. (0 Point)

- **Bonus Points**

  - The objective and key features of the project are filling a gap in the current Rust ecosystem, in that a crate's features have not been previously available, or a similar project that has not been previously implemented. (10% Overall Bonus Points)
  - If offered as open source, the design and features of the project have an educational value to those who learn Rust. (10% Overall Bonus Points)

6 Submission

Submit a single URL — the URL to your team's GitHub repository — to the assignment labeled **Course Project** in the Quercus course website. Each member of the team should make his/her own submission, but obviously all members in the same team should submit the same URL.

> ⚠️ The deadline for the course project is **Monday, December 15, 2025**, at **11:59pm** Eastern time, and late submissions will **not** be accepted. Do remember to add the instructor (username `baochunli`) to your GitHub repository, if it is private, **before** the deadline.

1. You can feel free to add new commits to your GitHub repository after the deadline. If you choose to do so, please add a tag or create a branch to the version that you wish to submit, and add a note to your `README.md`. ↩
2. There is no minimum length requirement for the final report in `README.md`. Other formats (such as Microsoft Word or Adobe PDF) will not be accepted, because, at the

end of the day, software developers *should* use Markdown as their documentation format. ↩

3. When there is a need to do so, it is fine to copy and paste sentences from your course project proposal. ↩

Last updated on September 29, 2025
