**­­­­Team project #2: A Trustworthy Module Registry**

*As a Boilermaker pursuing academic excellence, we pledge to be honest and true in all that we do. Accountable together – We are Purdue.*

*(On group submissions, have each team member type their name).*

Type or sign your names: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Write today’s date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **Assignment Goal**

This assignment provides an opportunity for you to work as a team on a still-small-but-more-challenging software engineering project. It will introduce you to several new challenges: more careful planning; refactoring and extending an existing system; deploying and monitoring on a modern Cloud platform (“DevOps skills”); and performance and cybersecurity considerations.

WARNING: This project spec is **designed** to be far more work than I think you can reasonably do in the time allowed. In your design, your team must identify and justify a reduced scope. However, all teams should complete the requirements marked “*baseline*”. This set of requirements is relatively short; proposals to complete *only* this set will need to be justified (e.g. with time estimates) to persuade me that you are not trying to inflate your contract.

In this project you have slightly more constraints.

* You must use multiple components from Google Cloud Platform (GCP). Details below.
* You must start with an existing implementation of Project 1 – albeit not your own. You are allowed to maintain, refactor, and evolve this implementation, but not change the programming language.
* New components can be implemented in a programming language of your choice.
* One mid-project milestone has been defined for you.
* Beyond a few infrastructural restrictions, your team has freedom of choice in terms of the programming language(s), design decisions, etc. Make good choices.

This project has a longer timeline than Project 1. It will run for a total of 9 weeks.

# **Relevant Course Outcomes**

A student who successfully completes this assignment will have demonstrated skill in all learning outcomes of the course.

# **Resources**

The following resources and links will help you understand and complete this assignment.

* **REST APIs**
  + Fielding’s 2000 dissertation: You can start at [Chapter 5](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm) (“REST”) but the [whole thing](https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm) is eminently readable and edifiying.
  + 20 years later, [brief commentary](https://twobithistory.org/2020/06/28/rest.html) on what Fielding meant vs. what REST means in practice (and conjectures about why).
  + [REST vs CRUD](https://nordicapis.com/crud-vs-rest-whats-the-difference/)
  + GitHub’s [REST API documentation](https://docs.github.com/en/rest).
  + Specifying REST APIs: <https://swagger.io/specification/>
  + Testing REST APIs: <https://github.com/microsoft/restler-fuzzer>
* **Google Cloud Platform**
  + Here’s the starting point: <https://cloud.google.com/docs/overview>

# **Assignment**

## Introduction

**The bad news**: ACME Corporation rejected your company’s deliverable. They decided to proceed with another company’s offering.

**The worse news**: Your company was gambling on a lucrative contract with ACME Corporation. Your company is now out of business, and you are out of a job.

**Good news**: Your competitor, Beta Software Solutions (BSS), won the longer-term ACME Corporation contract. They are awash in cash and hiring software engineers to work on the next phase of the contract. Thanks to your domain expertise, you and your team have been hired for this phase. You will need to extend BSS’s winning implementation to ACME Corporation’s new requirements.[[1]](#footnote-1)

ACME Corporation is continuing to expand its Node.js footprint. They are really benefiting from (your erstwhile competitor’s, now employer’s) tool for identifying trustworthy npm modules. However, they find themselves dissatisfied with aspects of the npm registry and their process for interacting with it:

* npm contains many untrustworthy or irrelevant modules, which pollute the search results.
* ACME Corporation would like to put its own modules – which are trusted, of course – into the same place that third-party modules are stored, but it does not want to share those modules publicly. It seems reputationally hazardous to do so without careful consideration.
* Some of ACME Corporation’s component re-use review processes involve a time-consuming manual inspection, and ACME Corporation doesn’t have a good system in place for tracking which modules have been vetted.
* ACME Corporation’s engineers download many modules, and npm can take awhile to serve the desired content.

They would therefore like your team at BSS to develop a custom registry for their npm modules.[[2]](#footnote-2)

Sarah is still your contact person at ACME Corporation. Here are ACME Corporation’s desired requirements, in her words.

## Sarah’s desired requirements

­­­

### Functional requirements

Your system must support the following general behaviors. Some behaviors are detailed later on, and others are introduced under an appropriate heading.

* Upload, update, rate, and download packages represented as zipped files.
  + Should work for individual packages
  + Should have support for doing so on package groups, using a three-stage transaction – (a) initiate an empty request; (b) add one or more commands to the request; and execute the request.
  + These queries should support package versions and version ranges. Notation:
    - Exact: “1.2.3”
    - Bounded range: “1.2.3-2.1.0”
    - Tilde and Carat ranges following the definition [here](https://github.com/npm/node-semver). “~1.2.0” or “^1.2.0”.
  + The “rate” option should provide the net score and sub-scores from Project 1, to inform prospective users of the quality of the module. It should also include **one new metric**: The fraction of its dependencies that are pinned to at least a specific major+minor version, e.g. version 2.3.X of a package. (If there are zero dependencies, they should receive a 1.0 rating. If there are two dependencies, one pinned to this degree, then they should receive a ½ = 0.5 rating).
* Fetch history for an individual package or a package group.
* Fetch a listing of all packages. This should be paginated (10 packages per page) so that a very large registry does not become a denial-of-service vector for your system.
* Request the ingestion of a public npm package. To be ingestible, the package must score at least 0.5 on each of the metrics from Project 1. If ingestible, the command should proceed to a package upload.
* Request the update of a public npm package. To be updated, the new version must be “ingestible” as described previously.
* To reduce storage costs, the “upload” and “update” features should support a “debloat” option that removes unnecessary “bloat” from the package.
* Request a security audit of one package, a group of packages, or the entire registry. This should use the “audit” mechanism provided by npm.
* Reset to the default system state (an empty registry with the default user)

Your system should be accessible via a REST-ful API. (HTTP verbs with conventional meanings.)

Your system should be accessible via a pleasant and ADA-compliant[[3]](#footnote-3) web browser interface. Use a front-end framework of your choice, and automate your tests with Selenium. Use some Purdue branding.

#### Baseline requirement

You must support, via a REST-ful API:

* Upload, update, rate, and download individual packages, including the new metric
* Ingestion of a public npm package as described
* Paginated list of all packages
* Reset to default system state

### Non-functional requirements

#### Performance requirements

Your system should support concurrent interactions with up to 10,000 clients.

ACME Corporation wants to know the mean, median, and 99th percentile latency when 10,000 clients are all trying to download the lodash package, from a registry containing 50,000 distinct packages. Provide measurements.

#### Traceability requirements

ACME Corporation is concerned about untrustworthy modules entering the registry. They would like to be able to trace the history of a module within the registry – who uploaded each version when?

ACME Corporation is concerned about the downloading of sensitive modules.

* They would like to execute an arbitrary JavaScript program prior to the download of any sensitive modules. Each such sensitive module can have zero or one JavaScript program associated with it. Any user can upload and delete sensitive modules and provide or query the associated JavaScript monitoring program. This program may need to communicate with ACME Corporation’s audit servers.
* This JavaScript program expects to run under Node.js v13, and accepts five command line arguments: “MODULE\_NAME MODULE\_VERSION UPLOADER\_USERNAME DOWNLOADER\_USERNAME ZIP\_FILE”. If the program exits with a non-zero code, the download of the module should be rejected with an appropriate error message that includes the stdout from the program.
* They would also like your system to be able to return the download history of the sensitive modules – which accounts downloaded them at what times?

#### Security requirements

ACME Corporation would like to restrict access. Your system should support:

* Register, authenticate, and remove users with distinct “upload”, “search”, and “download” permissions.
* The system should permit authentication using a combination of username + secure password, and yield a token to be supplied to the other APIs as a payload parameter.
* This token should remain valid for 1000 API interactions or 10 hours.
* Some users should have an “admin” permission. Only administrators can register users. Users can delete their own accounts.
* Some teams at ACME Corporation have experimental code that should not be accessible by other members of the company. There should be a “user group” mechanism that can be specified at user creation time. During upload, a package can be marked as “secret”. Such a package can only be queried by a member of the group that uploaded it.

You must persuade ACME Corporation that your system does not expose them to substantial security risks. To this end, you should:

* *Justify:* Prepare a security casebased on a [STRIDE](https://en.wikipedia.org/wiki/STRIDE_(security)) analysis of your system. See the document <DOCUMENT>.
* *Demonstrate:* Carry out automated security probing of your system using the RESTler tool.

##### Baseline requirement

All teams must conduct a STRIDE analysis.

#### System deployment and costs

ACME Corporation is a big company, so for scalability reasons your system should be deployed on Google Cloud Platform (GCP). You must expose a single uniform API, but under the hood you may use multiple GCP components. These should be indicated in your project design.

Here are some suggestions depending on your need:

|  |  |
| --- | --- |
| **Need** | **Some relevant GCP products** |
| Compute | Compute Engine (VMs); Kubernetes Engine (containers) |
| Storage | Storage (blobs in buckets); Bigtable (key-value); Firestore (NoSQL); MySQL (relational) |
| Auto-scaling | “Managed group instance” within compute engine; Pods in Kubernetes |

ACME Corporation has provided GCP credits for you to develop and test your system. If you exhaust the credits, please contact Professor Davis.

Your system source code must reside in a single GitHub repository within GitHub Classroom.

ACME Corporation places a high value on CI/CD. By Deliverable 1, your system should be using GitHub Actions to perform:

* Automated tests (CI/Continuous Integration) on any pull request
  + You might conduct some tests (e.g. end-to-end performance tests with many clients) outside of your automated test pipeline.
* Automated service deployment to GCP on successful merge (CD/Continuous Deployment)

For consistent quality, ACME Corporation requests that every pull request receive a code review from at least three independent evaluators.

###### Baseline requirement

* All teams must use at least one GCP component – e.g. a VM with a lot of storage attached, hosted on Google Compute Engine.[[4]](#footnote-4)
* In their mid-project milestone, all teams should include screenshots demonstrating the use of CI/CD via GitHub Actions.

## Internal (Baseline) Requirements (To make the course staff’s lives easier!)

### Auto-grader API

To facilitate automatic grading, the project material includes an OpenAPI (“Swagger”) specification for the “baseline” components. You can also use this specification to test your system; see the *RESTler* link under the Resources heading.

We will not attempt to deploy your system ourselves. Instead, you will provide us with a URI (hostname + port) that we can use to interact with a clean version of your system.

In its initial and “Reset” state, your system must have a default user. If your team chooses to support authentication, the username and password are given next. If you support user creation/deletion, groups, etc., then this user should have admin privileges.

* Username: “*ece461defaultadminuser*”
* Password “*correcthorsebatterystaple123(!\_\_+@\*\*(A*”

### Source code hosting

Your project should be implemented on GitHub Classroom. We have created an assignment for it; here is the link to click: [GitHub Classroom](https://classroom.github.com/a/AyDwkkg0).

**Per the academic honesty requirements of this course, you are not permitted to compare software implementations with other teams.** I encourage you, however, to discuss “Operations” issues such as GCP configuration with one another.

### Project management

I recommend but do not require that you use GitHub Project Boards for progress tracking. You must, however, use some project management software (e.g. Trello, Monday, GitHub Projects, etc.) and use this to report status in your weekly milestones.

### Software re-use

You are allowed to re-use existing software to support your implementation, either as tools (e.g. VSCode; git; GitHub; TravisCI) or as components in your implementation (e.g. a module to help you parse command-line arguments). You should include a justification of any components you choose to re-use – how will you decide whether they are trustworthy? (discuss in the Project Plan) and how did that assessment work out in practice? (discuss in the Project Postmortem)

You are allowed to re-use code snippets from software engineering resources such as Stack Overflow. You must provide a citation (web URL is fine) to the relevant post. [Also, please review Prof. Davis’s general perspective on Stack Overflow](https://davisjam.medium.com/what-is-stack-overflow-for-1960b794a471). Technically, Stack Overflow snippets are themselves governed by a software license, but you are not trying to distribute your project code and I do not think anyone would seriously pursue litigation along these lines.

You are **not** allowed to copy-paste code snippets out of an open-source project – this is a great way to expose ACME Corporation (and your future employer in the real-world) to lawsuits. Any re-use of this nature must use existing module APIs and/or extend those APIs so that you can access the logic you want.

## Timeline and Deliverables

The project will be completed over an 8-week period:

* Weeks 1-2: Assessment of BSS’s codebase; learning, planning, estimating, negotiation, and initial design.
* Weeks 2-5: Internal milestones culminating in *Delivery 1:* *Demo of some functionality* (specific features are up to you; must demonstrate automated tests and deployment by now)
* Weeks 6-8: Internal milestones culminating in *Delivery 2: Deliver the rest of the functionality*
* Week 9: Postmortem

### Weeks 1-2

By Monday 10/25, teams should submit a Project Plan Document (Word Doc or PDF) including the following:

* Tool selection and preparation
  + Programming language, toolset, component selection [linter? Git-hooks? CI? Testing framework? Logging library?]
  + Communication mechanism [Slack? Teams? Email?]
* Team contract
  + For example, your team might agree: do the work they take on, to document their code, testing rules, style guide, timeliness expectations
* Team synchronous meeting times
  + I recommend at least one (short) mid-week sync to discuss issues, and one end-of-week sync to put together your weekly reports.
* Requirements
  + A refined and organized list of requirements, based on Sarah’s description and specification.
* Preliminary design
  + Diagrams to support planning
    - Activity diagrams to depict the various activities performed by your system.
    - Data flow diagrams to depict the flow of information (including who can access which information)
    - (Photos of a whiteboard are fine, or online UML tools like those provided by LucidChart)
* Timeline and planned milestones for weeks 3-8
  + Each milestone should list the features, sub-tasks, the owners of those tasks, the estimated time to complete it, and how success will be measured.
  + Any communication requirements between tasks should be noted, e.g. "Jason and Tahani need to discuss the interface involved between task A and task B."
* Validation and Assessment plan
  + What is your plan to assess whether the delivered software satisfies Sarah’s requirements? What behaviors will you check? What performance metrics (if any) will you apply?
* Starting project analysis
  + You **must** use another team’s Project 1 implementation. This implementation may (a) not work correctly; and/or (b) may use languages and tools with which your team is unfamiliar. You should review this implementation and indicate the cost (if any) of integrating it into your design; the interface you plan to use to talk to it (e.g. continuing in the same language vs. wrapping it with an interface in your preferred language); the shortcomings and changes you plan to make to it; and so on.
  + You should conduct an independent assessment of how trustworthy the project is, and list this in your document.

### Weeks 3-7: Complete your milestones

Each week, submit a report with your updated list of milestones, tasks, etc. representing completion and the actual time spent by each team member on the project.

This report should be self-contained, e.g. including the relevant information from the original plan. For example, you should use a table for each feature and track the status of the relevant tasks.

If you *deviate substantially* from your timeline, consider attending one of the course staff office hours to discuss the deviation.

### Week 8: Deliver

Submit the software itself, along with a report describing the status of the software in relation to Sarah’s requirements and specification.

### Week 9: Postmortem

Deliver a project postmortem report. This report should reflect on each aspect of your Plan (from week 1) compared to your Execution. What went well? What went poorly? Where did your time estimates fail? When and why did you deviate from your Plan? For all of these questions, try to answer the question “Why?”

## Grading rubric

Points breakdown:

* 35% Design & Planning document + Milestone documents.
* 50% Working delivery, broken down as:
  + 25% "Our auto-grader can interact with it in the “baseline” features"
  + 15% "The additional promised features are delivered with evidence that they are working correctly" (details to follow, in the project report template)
  + 10% "Per our manual inspection, the software follows reasonable-looking engineering practices, e.g. good file/class/variable names, consistent style, choice of data structures, use of patterns to isolate what is changing, appropriate pipeline for automated deployment, etc."
* 15% Post-mortem.

Provided that the teammates complete the tasks they were assigned as part of the project plan, all team members will receive the same grades. If there is an issue with teamwork, please raise it with Prof. Davis as early as possible.

* Your team’s milestones should allow you to observe problems with forward progress.
* For personality clashes etc., use your judgment to determine if you want to speak with Prof. Davis.

## ACME Corporation’s Budget is not Bottomless

Sarah reminds you that your team members are from an independent contracting firm. She says the company is **willing to pay your team for up to 80 hours per person for this project**[[5]](#footnote-5), and would rather see ***something that works – at least partially! – by the deadline.***

* Your project plan and your weekly progress updates should reflect an appropriate amount of time for the project, e.g. 6-9 hours per team member per week. If you wait until the last minute, Sarah will be nervous, pull the plug on the project, and might break off future contracts with your company.
* If you begin to deviate from your planned timeline, you should submit a **revised** plan as part of a weekly update. That way Sarah can keep management abreast of progress and aware of any changes in the functionality that will be delivered.
* You should plan your project in such a way that you can deliver incremental value to Sarah even if you cannot complete all of her requirements.
  + Recall the aircraft requirements document from the Requirements Engineering unit – one of the final chapters designated useful subcomponents that the vendor could deliver.
  + One of the slides from class has an excerpt.
* I will be surprised if any team were able to complete the full set of functionality by the deadline. **In your design document, you should have enumerated and organized the requirements, estimated the cost of each feature, and identified the subset of the functionality that you think your team can reasonably deliver. The total desired cost for your team should be roughly 80 hours per person. Estimate and negotiate accordingly.**

## Guidance

You can complete this project with a “Cloud-First” approach – all code is always deployed on the Cloud. You might prefer, however, to have one teammate study the GCP options while others start the implementation locally.

When offering a web service such as the Trustworthy Module Registry, *some* component of your system must handle incoming requests. After this point, however, these requests are handled through a sequence of function calls and interactions with internal components. If you prefer, you can focus on the entities *behind* the interface first, and add the “web interface” part later.

Initially, scaffolding and mocking can be used in place of components. This helps you focus on inter-component interfaces first, and facilitates testing each component in isolation. At deployment time you can replace a mock with the real component.

1. It is possible that their “winning” implementation did not win on *technical* grounds, but rather on pricing, trust in their company due to a long-term relationship, and so on. You should review their implementation carefully before you proceed, and you may flag weaknesses and opportunities for improvement (e.g. if their code crashes, it will affect the service you build around it). [↑](#footnote-ref-1)
2. These requirements sound a bit tricky, and it might be cheaper for your company to subcontract some of the work (e.g. buy an existing system and configure it for your customer). In this vein, you looked for any companies that currently offer registry-as-a-server. As it turns out, npm itself formerly sold this product (“[npm enterprise](https://docs.npmjs.com/sunsetting-npm-enterprise)”) before npm was purchased by GitHub in 2020. Now this product is sold under the name “GitHub Packages” and includes other module registries as well. You recall, of course, that [GitHub is itself owned by Microsoft](https://news.microsoft.com/announcement/microsoft-acquires-github/). Since Microsoft is a competitor of ACME Corporation, ACME Corporation does not want to leave its private software in their clutches. It would be possible to construct the registry that your customer desires by interfacing with GitHub Enterprise/Packages, but your contract forbids this solution. [↑](#footnote-ref-2)
3. It should be compliant with WCAG 2.1 at level AA. See <https://www.w3.org/TR/WCAG21/> for details. As a tool, I suggest the Chrome plugin for <https://github.com/microsoft/accessibility-insights-web>. [↑](#footnote-ref-3)
4. A “one fat VM” configuration would not be scalable and thus could not sustain a high load. However, it would be relatively straightforward to configure, so it might be useful for an initial prototype. [↑](#footnote-ref-4)
5. “80 hours per person for this project” – Since the project will run for 9 weeks, that will average out to 8-9 hours per teammate per week. The postmortem week should be lighter and the earlier weeks a little heavier. [↑](#footnote-ref-5)