# DS3002 – Data Project 1
**25 points**

The goal of this data project is to demonstrate (1) an understanding of and (2) competence implementing and using basic data science systems such as pipelines, containers, APIs, data transformations, and relational databases.

Select a project:

1. API-based chatbot – container or API
2. ETL data processor – container
3. Discord integration – container or API

## 1. API-based Chatbot

a. Deliverable: Write an interactive, CLI-based chatbot and publish it either as a service (Lightsail or Chalice) or a container image that can be run as a web-based API. You must also submit a URL to a GitHub repository for your solution.

b. Benchmarks:
   i. Your chatbot should recognize at least 5 unique commands (i.e. 5 endpoints).
   ii. At least 4 of the 5 commands must generate dynamic responses, i.e. not a "static" response that is the same every time. Any endpoint/method may require parameters if you like.
   iii. The bot should perform proper error handling if a non-command is sent.

c. Grading:
   i. ☐ Successful deployment to Lightsail or a Docker image (with Dockerfile) – 10 points
   ii. ☐ Functionality that meets all benchmarks – 10 points
   iii. ☐ Creativity / Innovation / Quality – 2 points
   iv. ☐ Documentation – Describe how to use the chatbot and the elements that make it operational – 3 points.

2. **ETL data processor**

   a. Deliverable: Author a segment of an ETL pipeline that will ingest or process raw data. This should be published as a container image and Dockerfile. You must also submit a URL to a GitHub repository for your solution.

   b. Benchmarks:
      i. Your data processor should be able to ingest a pre-defined data source and perform at least three of these operations:
         1. Fetch / download / retrieve a remote data file by URL, S3 key, or ingest a local file mounted to the container. Suggestions for remote data sources are listed at the end of this document.
         2. Convert the general format and data structure of the data source (from TSV to CSV, from CSV to JSON, from JSON into a SQL database table, etc.)
         3. Modify the number of columns from the source to the destination, reducing or adding columns.
         4. The converted (new) file should be written to disk, or pushed to S3, or written to a SQL database.
         5. Generate a brief summary of the data file ingestion including:
            a. Number of records
            b. Number of columns

      ii. The processor should produce informative errors should it be unable to complete an operation.

   c. Grading:
      i. ☐ Successful build of the solution as a container image and Dockerfile – 10 points
      ii. ☐ Functionality that meets all benchmarks – 10 points
      iii. ☐ Creativity / Innovation / Quality – 2 points
      iv. ☐ Documentation – Describes how to use the data processor and the elements that make it operational – 3 points

## 3. Discord Integration

a. Deliverable: Write a Dockerized Python3 application that can be run interactively, taking a single parameter. Using a remote, publicly-accessible API of your choice you will retrieve remote data based on the parameter and post it to a channel in our Discord server. You must also submit a URL to a GitHub repository for your solution. (This project choice can also be published as an API if you desire.)

b. Benchmarks:
   i. Your container should take a single parameter when run. If no parameter is given, the application should provide an informative "help" screen to explain how to use it.
   ii. Your container should take the parameter, perform a request against the remote API using that parameter, and post the results to the #bot channel of our course Discord server. (Instructions on that will be provided.)
   iii. Your container should not contain sensitive information such as the Discord API key. You should use a secure method for passing this sensitive data into your container when it is run.
   iv. Your container should return an informative error if given a "bad" parameter.

c. Grading:
   i. ☐ Successful build of the solution as a container image and Dockerfile – 10 points
   ii. ☐ Functionality that meets all benchmarks – 10 points
   iii. ☐ Creativity / Innovation / Quality – 2 points
   iv. ☐ Documentation – Describes how to use the Discord integration and the elements that make it operational – 3 points.

Publicly-available datasets:

- https://www.kaggle.com/datasets
- https://data.world/
- https://www.data.gov/
- https://opendata.charlottesville.org/

Publicly-available APIs:

- https://docs.github.com/en/rest
- https://developer.twitter.com/en/docs/twitter-api
- HUGE LIST: https://github.com/public-apis/public-apis