Iteration 3: RAD and First Implementation

Jump to bottom

Jeff Caldwell edited this page 2 minutes ago · 32 revisions

Barrios Technology - Forecasting and Prediction Modeling

Team Noname

Jonathan Morgan

Carlos Cardenas

Josue Lozano

Jeremy Miles

Jeff Caldwell

Introduction

i. Purpose of the system

The ISS analysis tool is a web application that provides analysts at Barrios, an aerospace company, with logistics optimization, usage forecasting, and prediction accuracy analysis surrounding consumable delivery to the International Space Station (ISS).

ii. Scope of the system

The system encompasses secure data input and transport via HTTP, data storage in a relational database, prediction accuracy analysis, logistics optimization analysis, and predictive modeling. All system functionality will be presented to users through a simplified and intuitive web-based interface.

iii. Objectives and success criteria of the project

iv. Definitions, acronyms, and abbreviations

- **ISS**: International Space Station
- **IMS**: Inventory Management System - Barrios' ISS inventory management datastore

- **ORM**: Object Relational Mapper - A framework-specific set of classes and methods for interacting with a database

v. References

- Emmett: A "full-stack" web framework written in Python
- HTMX: A client-side JavaScript library that enhances HTML for easy communication with the server
- PostgreSQL: A relational SQL database
- Pandas: A Python library for exploring and manipulating tabular data
- Prophet: A forecasting algorithm

vi. Overview

The ISS analysis tool will give Barrios analysts the ability to quickly transform user-provided data into three types of analysis: prior prediction accuracy assessment, logistics optimization, and predictive modeling.

Current System

The current system is a simplified approach based on the requirements defined in iteration 2. With this iteration, the code's focus is basic file upload and user data persistence. Additional work is underway to prototype predictive modeling, optimization, and accuracy analysis functionality that will be integrated into the application during a future iteration.

Proposed System

The proposed system will use an HTTP server framework named Emmett to facilitate a basic Model-View-Controller application structure. With this structure, controllers will respond to HTTP requests and coordinate

i. Overview

ii. Functional Requirements

- Data upload, validation, and persistence functionality
- Accuracy analysis, logistics optimization, and predictive modeling algorithms
- Intuitive and aesthetically pleasing user interface

iii. Non-functional requirements

a. Usability

Interface:

- The web application should have an intuitive and appealing user interface that works on various screen sizes.

User management:

- Administrators should be able to create additional users with different levels of access.

Account management:

- Users should be able to log in, log out, and reset their passwords.

Ease of use:

- Users should be able to upload and verify data in a straightforward and intuitive way. Additionally, users should be able to view analyses in a high-level overview and have access to more detailed views of each analysis.

b. Reliability

Data:

- User data should be persisted to the appropriate database table and stored as a redundant flat file on the filesystem.

c. Performance

Client performance:

- Consideration for browser performance and interface response times should be paramount. Undue amounts of JavaScript should be avoided.
- The display of large amounts of tabular data should be approached iteratively — using pagination or "infinite scrolling" to load smaller subsets of the data at a time.
- Visualizations should be lightweight.

Server performance:

- Persistence of user data should be fast, especially in the face of multi-gigabyte source uploads.
- Analysis functions should take every measure to perform as quickly as possible.

d. Supportability

- The application will be easy to use on any modern web browser.
- Documentation of the underlying application code and application deployment guidance will be provided.

e. Implementation

- Must work well in all modern browsers.

f. Interface

- The interface should be responsive.
- The interface should look good and communicate intention.

g. Packaging

- The system will be deployed to cloud-based infrastructure using a containerized environment for the server.
- Could infrastructure must include a PostgreSQL database.

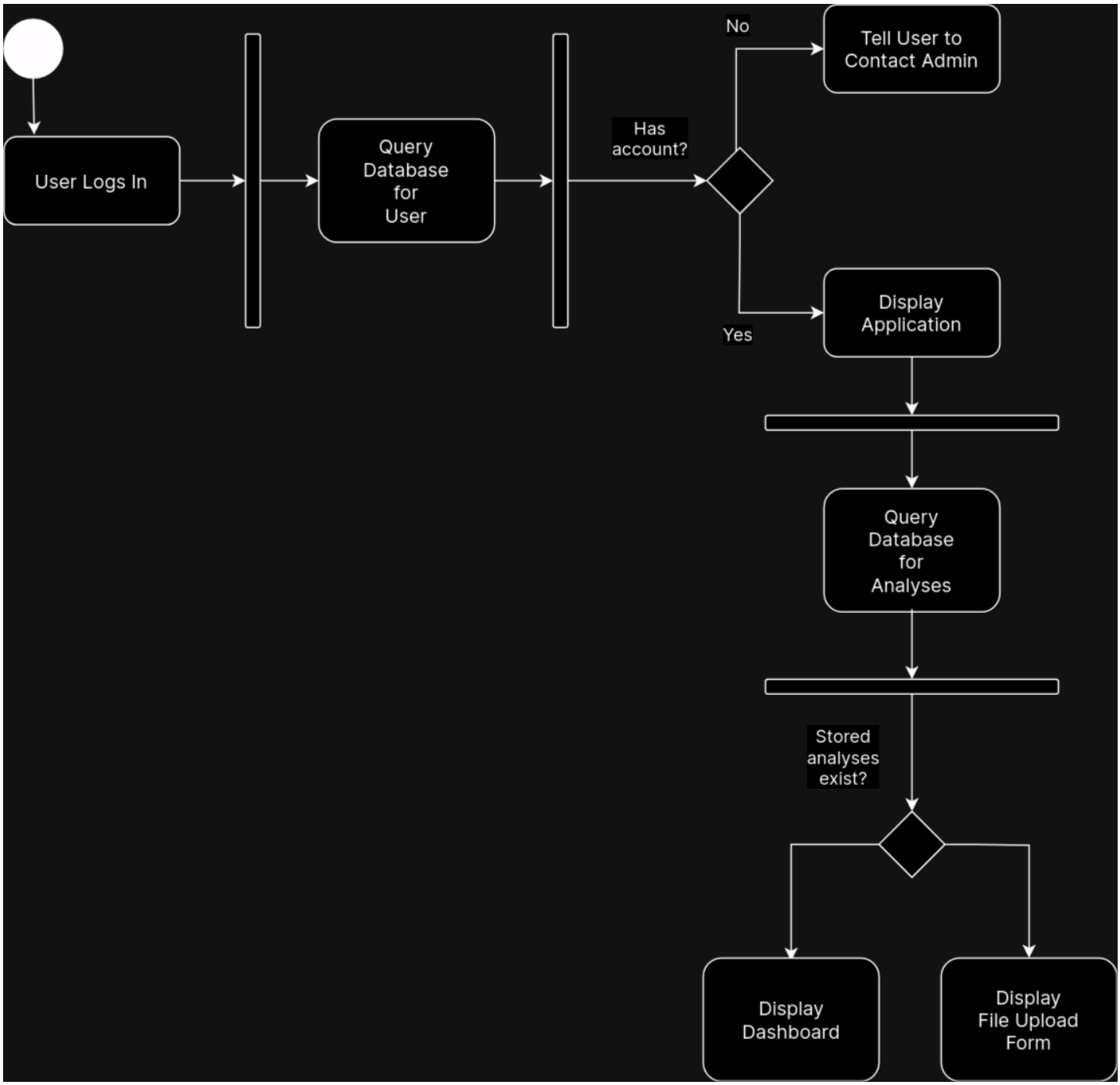
h. Legal

- The system should use open source libraries with permissive licensing.
- Client data should remain confidential
- Student should remain the intellectual property of the students

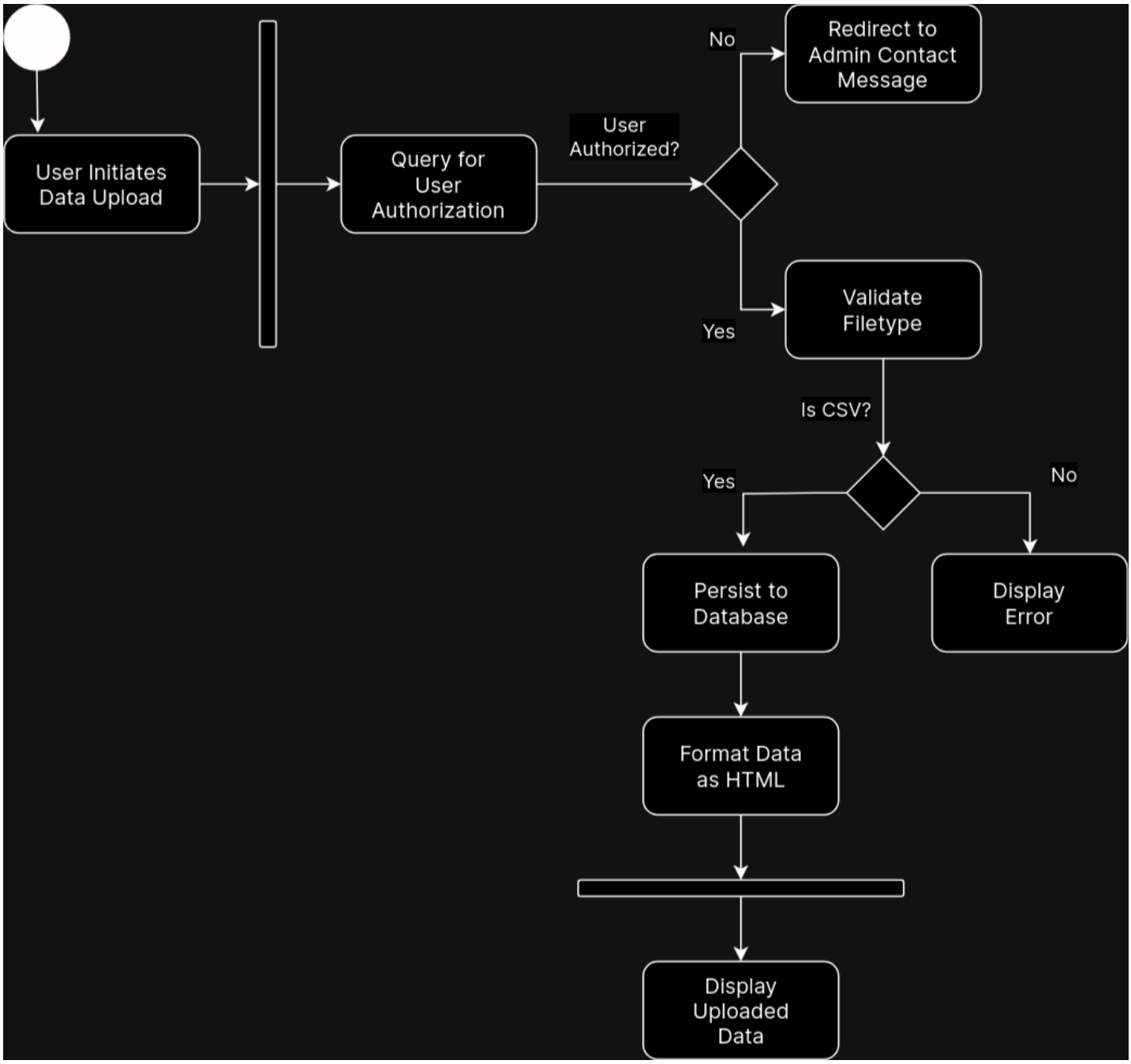
iv. System Models

a. Scenarios

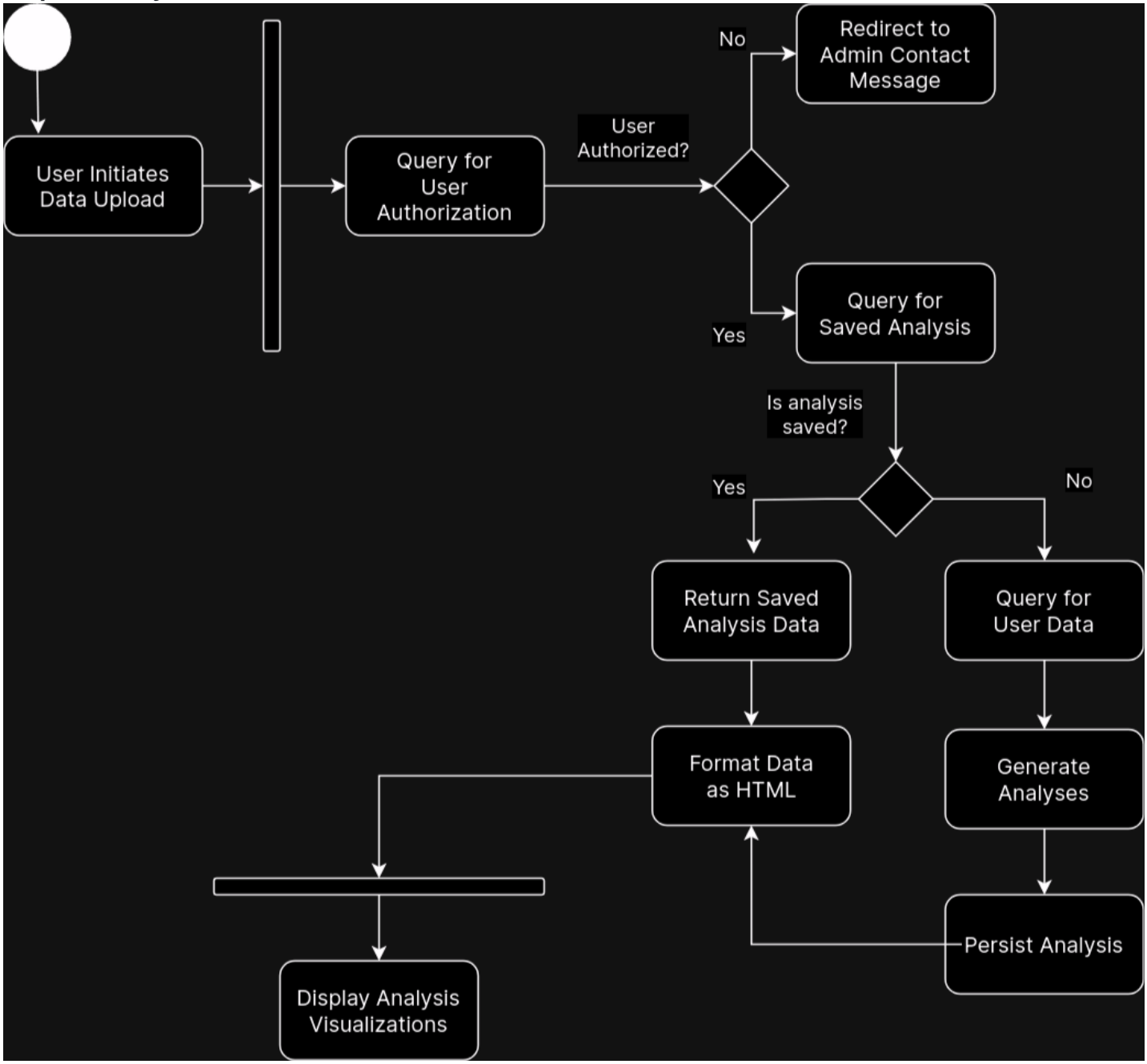
User Login Scenario



File Upload Scenario

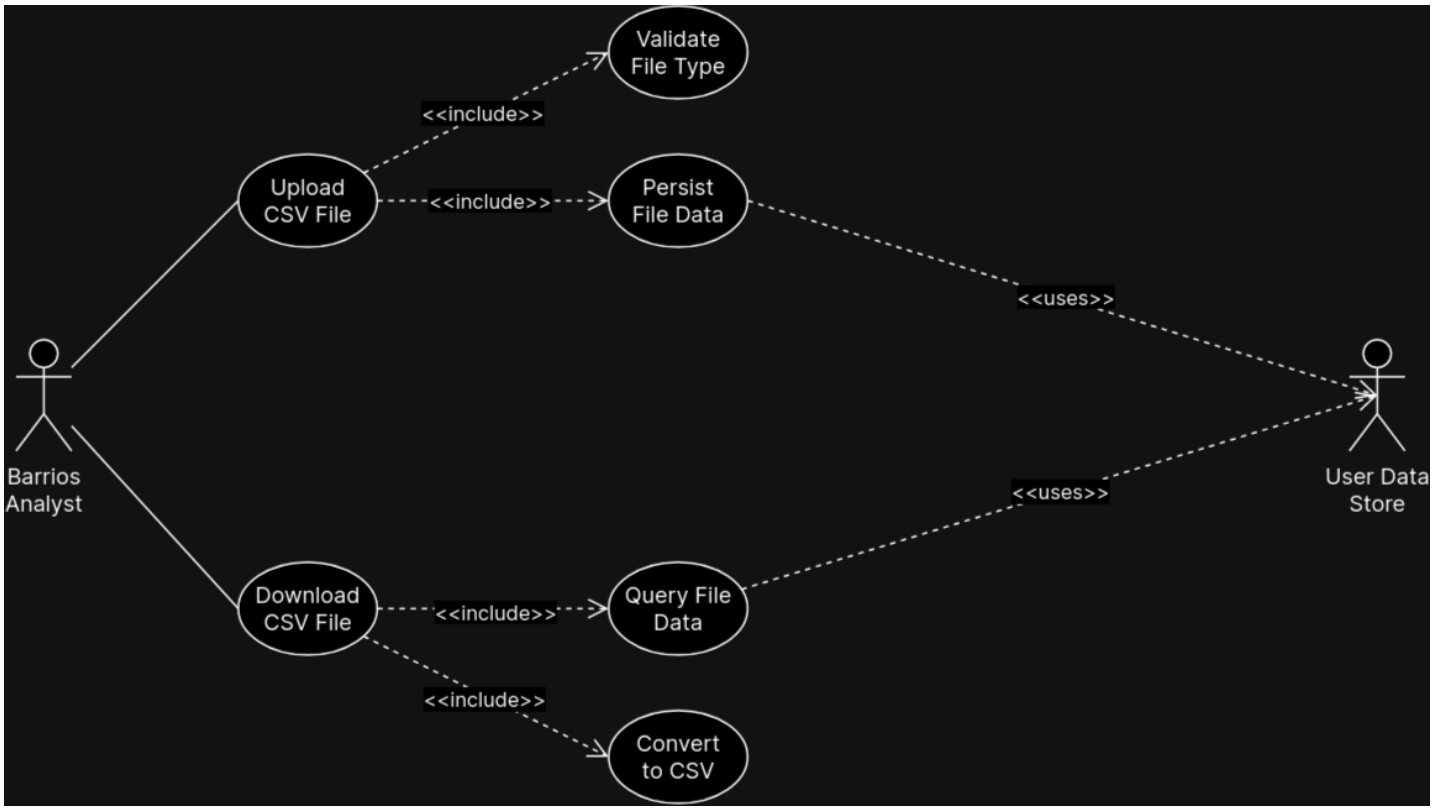

User Initiates Data Upload
Query for User Authorization
User Authorized?
No
Redirect to Admin Contact Message
Yes
Validate Filetype
Is CSV?
Yes
No
Persist to Database
Display Error
Format Data as HTML
Display Uploaded Data

Request Analysis Scenario

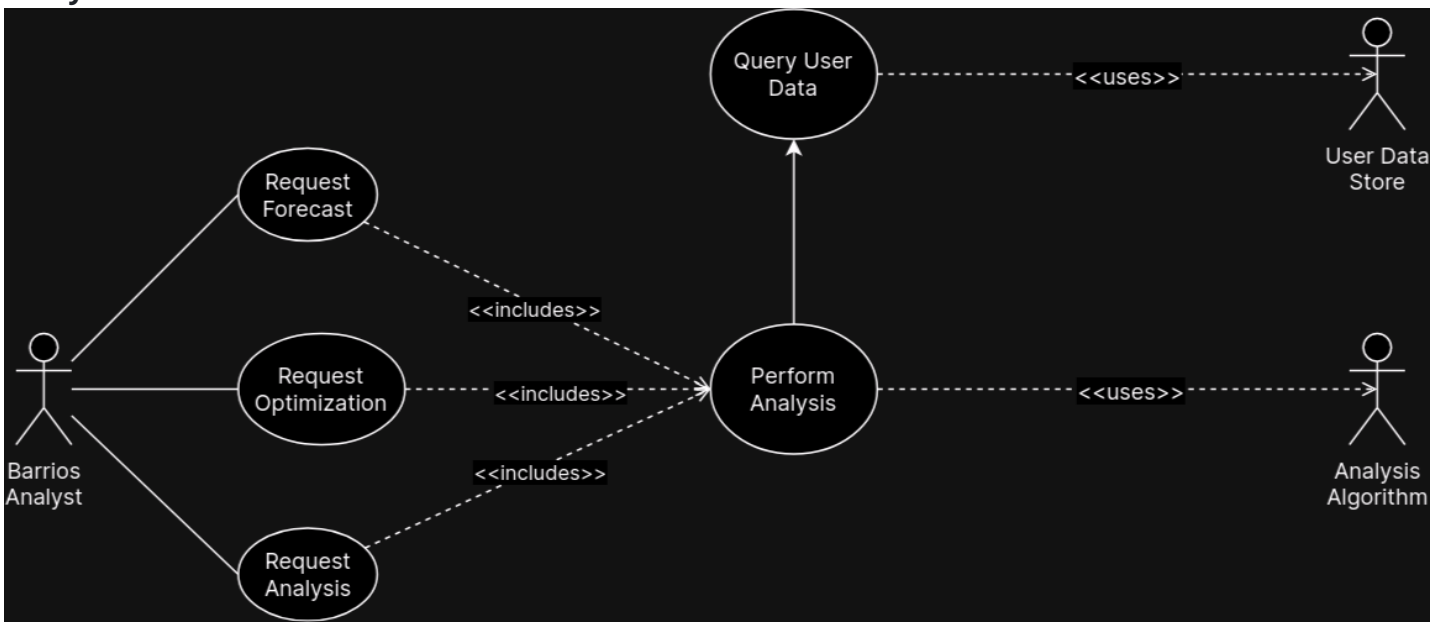


b. Use case model

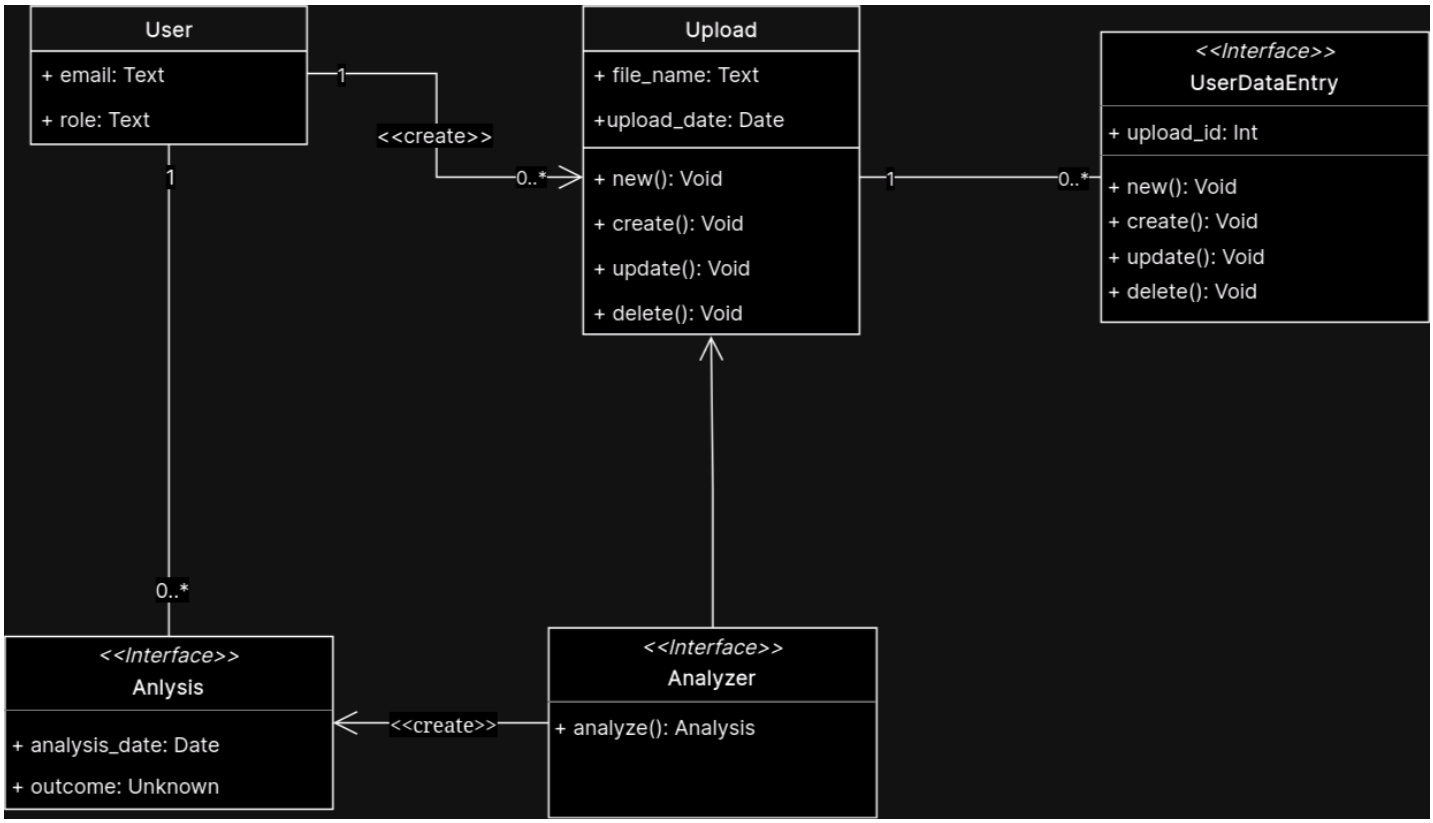
Upload Data Use Case



Analysis Use Case



c. Object model



d. User interface - screen mock-ups

BARRIOS

- DASHBOARD
- USAGE ANALYSIS
- RESUPPLY PREDICTIONS
- DETAILED FLIGHT PLAN

UPLOAD DATA

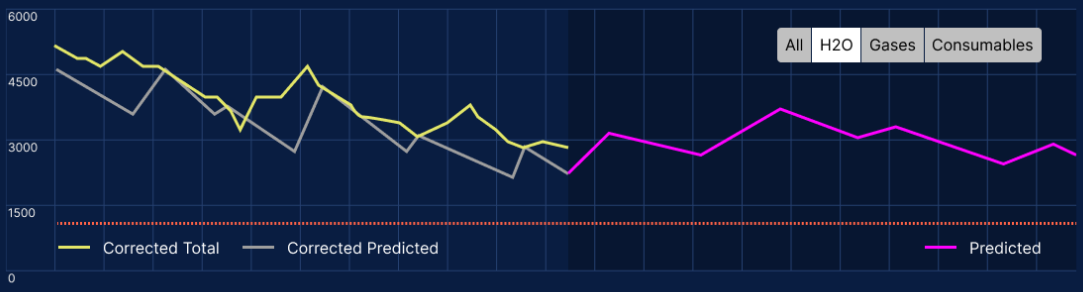
USAGE

BELOW PREDICTIONS

10%

RESUPPLY

NAME	MIN	MAX
O2	400 LBS	700 LBS
N2	100 LBS	200 LBS
H2O	500 L	700 L
FOOD	200 LBS	400 LBS
ACY INSERT	300	600
KTO	10	30



Flight Plan Overview

NAME	TYPE	DATE
69 SOYUZ	UNDOCK	09/27/2023
U (MICROORGANISMS)	EVA	10/12/2023
U (RFG RETRIEVAL)	EVA	10/19/2023
R-61	EVA	10/25/2023
NG-19	UNDOCK	10/30/2023
SPX-29	LAUNCH	11/01/2023
SPX-29	DOCK	11/02/2023
84 PROGRESS	UNDOCK	11/29/2023
86 PROGRESS	DOCK	12/01/2023

BARRIOS

- DASHBOARD
- USAGE ANALYSIS
- RESUPPLY PREDICTIONS
- DETAILED FLIGHT PLAN

UPLOAD DATA

USAGE

BELOW PREDICTIONS

10% ↓

USAGE ANALYSIS

NAME	MIN	MAX
O2	400 LBS	700 LBS
N2	100 LBS	200 LBS
H2O	500 L	700 L
FOOD	200 LBS	400 LBS
ACY INSERT	300	600
KTO	10	30
O2	400 LBS	700 LBS
N2	100 LBS	200 LBS
H2O	500 L	700 L
FOOD	200 LBS	400 LBS
ACY INSERT	300	600
KTO	10	30
O2	400 LBS	700 LBS
N2	100 LBS	200 LBS
H2O	500 L	700 L
FOOD	200 LBS	400 LBS
ACY INSERT	300	600
KTO	10	30

BARRIOS

DASHBOARD

USAGE ANALYSIS

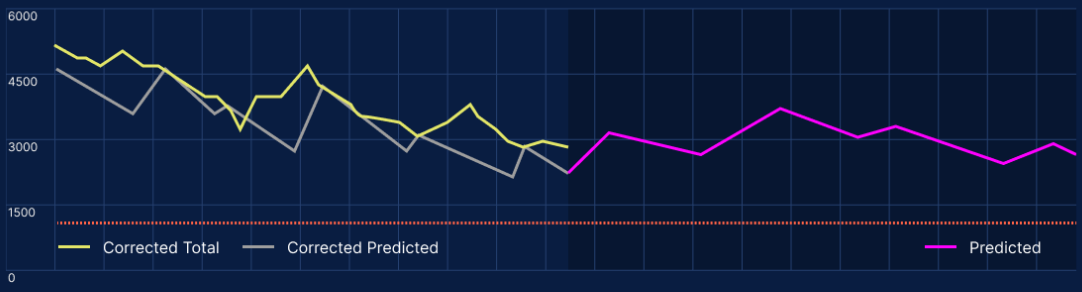
RESUPPLY PREDICTIONS

DETAILED FLIGHT PLAN

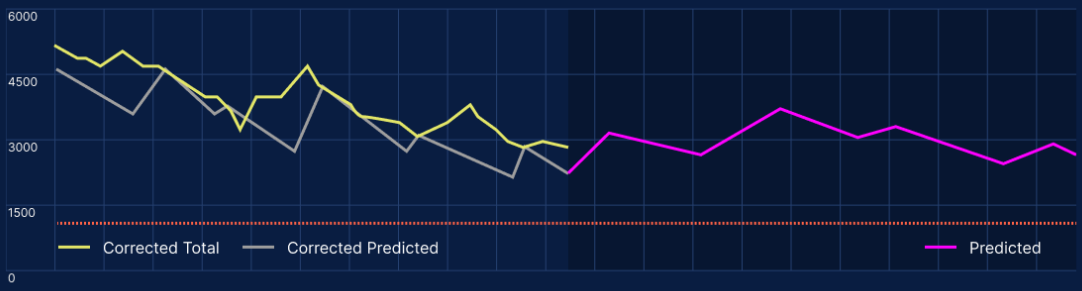
UPLOAD DATA

RESUPPLY PREDICTIONS

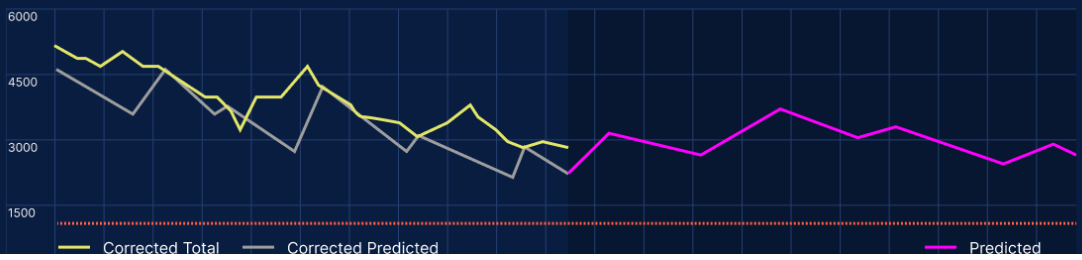
H2O



GASES



GASES



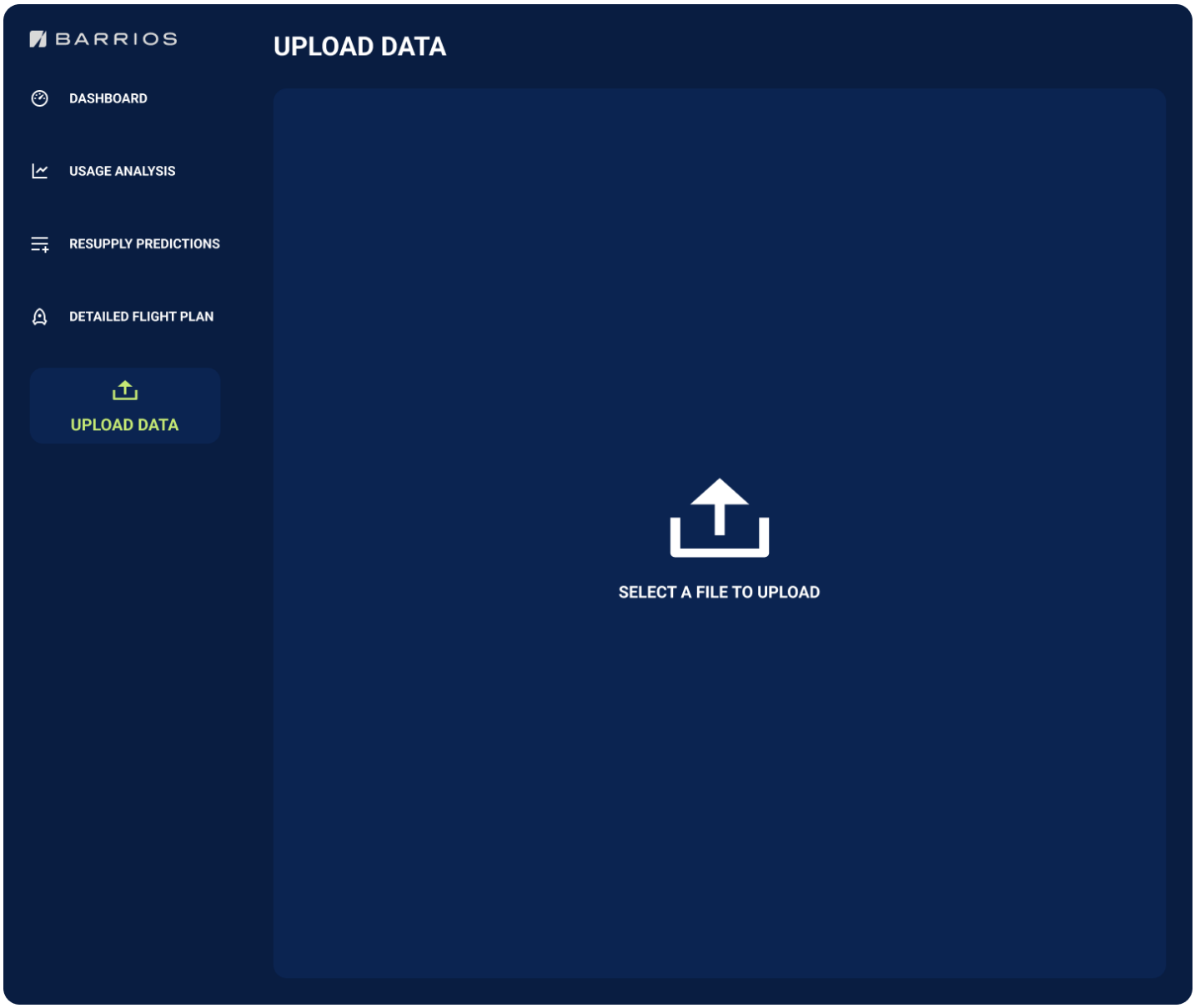
BARRIOS

- DASHBOARD
- USAGE ANALYSIS
- RESUPPLY PREDICTIONS
- DETAILED FLIGHT PLAN

UPLOAD DATA

Flight Plan

NAME	TYPE	DATE
69 SOYUZ	UNDOCK	09/27/2023
U (MICROORGANISMS)	EVA	10/12/2023
U (RFG RETRIEVAL)	EVA	10/19/2023
R-61	EVA	10/25/2023
NG-19	UNDOCK	10/30/2023
SPX-29	LAUNCH	11/01/2023
SPX-29	DOCK	11/02/2023
84 PROGRESS	UNDOCK	11/29/2023
86 PROGRESS	DOCK	12/01/2023
69 SOYUZ	UNDOCK	09/27/2023
U (MICROORGANISMS)	EVA	10/12/2023
U (RFG RETRIEVAL)	EVA	10/19/2023
R-61	EVA	10/25/2023
NG-19	UNDOCK	10/30/2023
SPX-29	LAUNCH	11/01/2023
SPX-29	DOCK	11/02/2023
84 PROGRESS	UNDOCK	11/29/2023
86 PROGRESS	DOCK	12/01/2023
69 SOYUZ	UNDOCK	09/27/2023
U (MICROORGANISMS)	EVA	10/12/2023
U (RFG RETRIEVAL)	EVA	10/19/2023
R-61	EVA	10/25/2023
NG-19	UNDOCK	10/30/2023



Glossary

Prophet: A timeseries forecasting algorithm developed by Facebook/Meta **Emmett**: A Python web framework that provides routing (controllers), templating (views), and models. The framework also includes an "object relational mapper" for working with relational databases. **Object Relational Mapper**: Library with methods or functions for interacting with a database without the need for writing queries directly. **Inventory Management System (IMS)**: The database/datastore used by Barrios for tracking logistics and consumables for the International Space Station.

First Iteration

Much of this iteration has been centered around exploring the client data, prototyping analysis functions, and implementing basic functionality.

1. Implemented the beginnings of a file upload mechanism, which currently takes multipart form data, saves a file to the filesystem, and reads the field names of the provided tabular data. In the future,

these field names will be used to determine which `UserData` table to save the user data to.

Data Controller - Upload Route

```python
@data.route('/upload', methods='post')
async def upload():
    response.meta.title = 'Upload | ISS Consumables'
    files = await request.files
    file = files.file
    id = str(uuid.uuid4())
    type = file.content_type.split('/',1)[0]
    ext = file.content_type.split('/',1)[1]
    # return an error if not csvg
    # otherwise, upload to a temp folder
    temp_file_location = f"storage/{id}.{ext}"
    await file.save(temp_file_location)
    # read the file from disk
    # Note: encoding='utf-8-sig' is important here
    #       it removes `\ufeff` from fields
    fields = []
    with open(temp_file_location, encoding='utf-8-sig', newline='') as csvfile:
        reader = csv.DictReader(csvfile, delimiter=',')
        data = []
        for row in reader:
            data.append(row)
        fields = data[0].keys()
        print(fields)
    # check the field names to determine what type of model
    # the data represents

    # instantiate and persist the appropriate models.

    # return an HTML table (possibly paginated)

    # delete the temporary file
    return {'name': file.filename, 'ext': ext, 'type': type}
```

Prophet Algorithm - [Prototype Notebook]

In [15]:
```
m = Prophet()
m.fit(df)
```

```
18:49:41 - cmdstanpy - INFO - Chain [1] start processing
18:49:41 - cmdstanpy - INFO - Chain [1] done processing
```

Out[15]: <prophet.forecaster.Prophet at 0x28c312c4af0>

In [16]:
```
# makes dataframe in the future 365 days
future = m.make_future_dataframe(periods=365)
future.tail()
```

Out[16]:

	ds
447	2024-09-01
448	2024-09-02
449	2024-09-03
450	2024-09-04
451	2024-09-05

In [17]:
```
#forcasting
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

Out[17]:

	ds	yhat	yhat_lower	yhat_upper
447	2024-09-01	-5065.721087	-5278.677850	-4836.000947
448	2024-09-02	-5072.011033	-5299.272679	-4834.838562
449	2024-09-03	-5078.303136	-5300.942589	-4853.118780
450	2024-09-04	-1463.548013	-1679.048152	-1224.735333
451	2024-09-05	-5090.881205	-5329.003611	-4844.824342

- iii. Objectives and success criteria of the project
- iv. Definitions, acronyms, and abbreviations
- v. References
- vi. Overview

Current System

Proposed System

- i. Overview
- ii. Functional Requirements
- iii. Non-functional requirements
  - a. Usability
  - b. Reliability
  - c. Performance
  - d. Supportability
  - e. Implementation
  - f. Interface
  - g. Packaging
  - h. Legal
- iv. System Models
  - a. Scenarios
  - b. Use case model
  - c. Object model
  - d. User interface - screen mock-ups

Glossary

First Iteration

Clone this wiki locally

```
https://github.com/4306-team-noname/barrios.wiki.git
```