SpySat

ENG210-SC - PROJECT PROPOSAL

Cambry Partridge

# Table of Contents

[**Overview 3**](#_pwxl39881v8)

[Description 3](#_cv31fvv8p4bk)

[Concept Visual 4](#_lwprwbujljlw)

[Diagram 4](#_8iugb7lnd8)

[**Features 5**](#_e9a137oms0us)

[Core Requirements 5](#_yyok3su5wv46)

[Stretch Goals 6](#_qxtvwiyyb137)

[**Technical Process 8**](#_p4zf9ebj7mn)

[Tech Stack 8](#_ukoeg0wpdfch)

[Proposed Schedule 11](#_uql7rj2enpty)

[**Conclusion 15**](#_5zaxwn5sd33u)

[**Research References 16**](#_ue51jom06qz9)

# 

# Overview

## Description

For my Capstone I want to make a web application that can be used as a resource for visualizing and interacting with satellites. I am very passionate about space and I love learning more about what's beyond Earth’s atmosphere. Throughout my life I have frequently used tools to get a better understanding of the cosmos, to varying degrees of success. Many of the websites I have used and have found available have the common downfall of having too much information or being overly complicated making it hard for curious minds to begin their exploration.

SpySat will be a website to fix this issue. Taking the same information that these other sites use, but providing it in a way that anyone could understand and appreciate. It will feature an interactive 3D model that displays satellites within Earth’s orbit that will allow users to explore and interact with individual satellites. Users will be able to sort through and filter the display of satellites, get the most recent image captures from satellites, and create accounts to receive updates and facts about satellites they are interested in.

My main goals with SpySat is to create a microservice architecture to better solidify my knowledge in technologies I already know, learn new technologies for fun and to expand my understanding of a wider range of topics, create a reliable application with fail-safes built in, a highly optimized database, and work with specialized satellite data as it is my goal to work within the space industry developing software for satellites.

## Concept Visual

## 

## Diagram

## 

# Features

## Core Requirements

(ordered by importance)

* Optimized database querying– My biggest goal is being able to have a highly optimized database that allows for speed of information. I want to quickly sort through information and return that data just as quickly. To achieve this I’ll utilize multiple indexes, use both Redis and MongoDB together for better performance, and I will also do more research to find more ways to increase performance. To make sure my databases are performing well I will set up a dashboard to monitor the statistics.
* Interactive 3D model of satellites– From the database populated with satellite information from tracking APIs I will query the satellites and display all satellites in their approximated orbits around a 3D globe of Earth. Each satellite will be able to be interacted with and pull up information about the specific satellite (i.e: Name, Country of Origin, Function, Launch Date and Year, Size, etc.). This model will also allow users to drag and zoom in on the model to change perspective.
* Satellite imagery– Users will have an option to display images from satellites that specialize in imagery of Earth. Using an API and passing in the current latitude and longitude coordinates of a satellite, I can return and display an approximation of what a specific satellite is currently seeing. There is no way to get satellite specific images, so by this way I am creating the closest thing to getting images directly from satellites.
* Filtering satellites– There will be a dropdown that will allow users select and sort through satellites by specific parameters. This filtering will update the display to fit only satellites that fulfill these parameters. Filters will include Region of Orbit (LEO, MEO, GEO, SSO, and GTO), Country of Origin, which Company Owns the satellites, Functionality (Weather, Communications, Exploration, Navigation, Scientific Research, Earth Observation).
* Searching satellites– The search function will be very similar to the filter function. The difference will be that users can type into the search bar and the display will update to match the search parameters to satellite names or official identification numbers.
* User Accounts functionality– Users can register and login to the site with an email and password. Once accounts are created users can view their profile and do basic crud operations on their account.

## Stretch Goals

(ordered by importance)

* Favorite/Bookmark satellites– Users are able to mark satellites as favorited/bookmarked which will allow them to have an additional filter that displays just their favorited/bookmarked satellites. These satellites and their information will be displayed on the user's account page.
* Email Service– Service that will send emails to users when the following events happen: any updates to the user account (creation, updated email or password, deletion), recover password, space news or facts, any updates to their bookmarked satellites, or a favorited/bookmarked satellite is overhead.
* Educational facts– A small pop up window with educational facts along the side of the website, triggering every couple of minutes. This feature is to keep users engaged and keep the website from feeling stagnant.
* Deploy to cloud– If time allows, I will move this application that is already in microservice architecture to the cloud using AWS.
* AI chatbot– Only if time allows, I will train an AI and link it to a small chatbot on the site to make interacting with the satellites more engaging. Allowing users to feel like they are actually communicating with the satellite, however it's just AI giving them the information I was already displaying but in a conversational format.

# Technical Process

## Tech Stack

**\*:** Technology to learn

?: Technology I might use if time allows

* **Node.js**
  + While doing research into what languages NASA uses, I found that they use Node.js to keep astronauts safe during spacewalks and keep the data accessible as it is fairly flexible. As I am familiar with Node.js it will be used to create my website front-end and back-end connections.
* **\*Nuxt**
  + Nuxt will be used as my front-end framework to make my site visually appealing and interactive. I have touched it in the past and I know it is very flexible and lightweight, but I will need to relearn it to use it well.
* **MongoDB**
  + MongoDB will be used as my database to save satellite data as it is highly scalable, flexible, and has high performance. MongoDB is also document based which translates very nicely with Javascript and the satellite APIs I will be using.
* **MySQL**
  + MySQL works well with SuperTokens, so it will be used as my User database to hold their information and session details.
* **\*Redis**
  + I want to learn how to use Redis well. I know it can be used to lighten loads from querying other databases so I want to learn how to have Redis be a middle man between my services and MongoDB in a way that will increase performance.
* **\*CesiumJS/Worldwind/D3.js/Three.js**

[Only one of these technologies]

* + One of these 4 technologies will be used for my 3D modeling. They all have the capabilities for it however I need to test and find which will be the best one for my application. I am leaning towards D3.js as it is meant for modeling of data however I know that CesiumJS and Three.js have been used by others to do what I am doing.
* **\*satellite.js**
  + This npm package takes satellite TLEs (two-line element set) and calculates the orbits, so I will use this to get the orbits for my display.
* **Imaging api**
  + An outside API that takes Earth Latitude and Longitude coordinates and returns a recent satellite image of that position
* **Tracking api**
  + An outside API that returns satellite data including TLEs
* **\*KrakenD**
  + This will serve as my API gateway. While researching it seems relatively simple to set up, has high performance compared to competitors, and it is AWS compatible, so if time allows I will be able to move this to the cloud as well. I haven’t heard of this and so I will need to account for the learning curve.
* **?\*AWS**
  + If time allows I will move my entire microservice to AWS in the cloud.
* **SuperTokens**
  + Handles the authentication and authorization of my users. As well as sending emails for updates and notifications.
* **\*Prometheus**
  + Will be used to scrape data off my databases and services to pass to Grafana for monitoring. I’ve used Prometheus, but I want to have a better understanding of it and the data I am gathering.
* **\*Grafana**
  + Will take the data scraped from Prometheus and have a dashboard display of my microservice. I want to have monitoring, so I can see how my data and database is handling calls. This will also make it easier to see how quickly my database is functioning. Same as Prometheus, I’ve used it before, but I want to learn how to use it well and have valuable information to show to others.
* **GitHub**
  + GitHub is the go-to for source management and I will use it to keep my code up to date, minimize corruption, and save my code so that it can’t be lost.
* **Docker**
  + Will be used to containerize my application to aid in a quick deployment.
* **Postman**
  + Tool for testing of API endpoints.
* **draw.io/figma**
  + Visualization tools to create diagrams for planning.
* **?\*ChatGPT**
  + If time allows, I will train a chatbot based off of the data I have saved in my database and any information it can learn from the internet about satellites. It will be hooked up to the front-end and allow users to have a conversation about satellites.

## Proposed Schedule

Sprint Length: 5-6 days and 4-5 hours a day

**Sprint 0: Preparations before Capstone Quarter**

* Test and decide which 3D modeling software to use
* Diagrams and outlines finalized.
* Practice and use any technologies I am unfamiliar with so that I don’t need to learn it during capstone.
* Communications with experts on my topic (already started) to get any API keys or valuable information dealing with tracking or communications

**Sprint 1-2: Database and Tracking service**

* MongoDB and Redis setup
* API Documentation
* Tracking Service
* Monitoring Setup
  + Using Prometheus and Grafana, set up a dashboard to see how the database is performing to monitor how my service and database are performing with all the data.

**Sprint 3: Image Service, User Service, and API gateway configuration**

* Image Service
  + Pass approximate Earth latitude/longitude coordinates to an outside API and receive the most recent image taken of that area
* User Service
  + Set up SuperTokens with a MySQL database for authentication and security of users
  + Endpoints for Users to do CRUD interactions with their profile
* Set up Services with KrakenD API gateway

**Sprint 4: Basic Front-End and connections to backend**

* Basic front-end layout
  + Minimum looks, but the necessary functionality for what is needed for my minimum viable product
* Connection to my back end

**Sprint 5-6: 3D Display and Display Interactions**

* Connection to Tracking Service
  + Use Tracking service to get TLEs and calculate satellite orbits using satellite.js
* 3D Model
  + Create and render globe
  + Implement the dragging and zooming of display to change the orientation of the globe
  + Display satellites in their approximate orbit from data gotten from the tracking service
* Satellite Interactions
  + Allow satellites to be interacted with by clicking on that satellite and pulling up relevant information
* Filter Satellites
  + Using the tracking service, query by set filters and alter the 3D display to match the returned satellites
* Search Satellites
  + Using the tracking service, query satellite names and official IDs by searched parameters and alter the 3D display to match the returned satellites

**Sprint 8: Front-End and Bug-Fixing**

* Finalizing front-end with Nuxt
  + Overhaul the front-end to make it look good and
* Once front-end is more put together bug fixing and cleanup

**Sprint 9: Finalization**

* More testing and clean up
* Creation of presentation and practice presenting
* Final adjustments

**Sprint 10: Presenting**

* Present SpySat

# Conclusion

SpySat will be a microservice architecture focused on bringing satellite data to users in an interesting way. It will manage this by having a highly optimized database that will give data to my frontend to make an interactive 3D display of interactive satellites in their approximate orbit. SpySat is a project I want to see come to life, as it would have been something I would have used frequently while growing up. As a result, I am committed to creating a well-polished and unique capstone while never losing motivation during the 10 weeks of capstone. It will also expand on my knowledge of technologies I am already familiar with as well as learning new ones. If accepted I guarantee that SpySat will continue to inspire myself, other students, and future consumers.

# Research References

Example Model- <https://satellitetracker3d.com/>

D3.js- <https://d3js.org/>

CesiumJS- <https://cesium.com/platform/cesiumjs/>

three.js- <https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene>

WorldWind- <https://files.worldwind.arc.nasa.gov/artifactory/apps/web/examples/BasicExample.html>

Tracking software- <https://celestrak.org/software/satellite/sat-trak.php>

Sources of satellite data- <https://skywatch.com/free-sources-of-satellite-data/>

TLE- <https://celestrak.org/NORAD/elements/>

satellite.js- <https://www.npmjs.com/package/satellite.js>

KrakenD- <https://www.krakend.io/?gad_source=1&gclid=CjwKCAjwtNi0BhA1EiwAWZaANHjBIur7ZHu6EtbWcDP16hRZpZdtsbJifwtl2sxF44KNTNMeGp48ChoCHMIQAvD_BwE>