MapONE

Technological Feasibility Analysis

November 2, 2021

Sponsors:
Planetary Geologic Mapping Program, USGS Astrogeology Science Center
Dr. Sarah Black, Research Physical Scientist
Marc Hunter, IT Specialist

Faculty Mentor:
Melissa D. Rose

Team Members:
Samantha Milligan
Michael Nelson
Ricardo McCrary
Jacob Stuck

Overview: The purpose of the Technological Feasibility Analysis document is to outline the project's proposed methods for developing an end product. The following pages discuss the technological requirements and selected tools.

Table of Contents

1. Introduction

From the 1957 Sputnik 1 mission to the 2020 Mars Rover mission, scientists continue to enhance and create new ways to explore space. For many researchers, the discovery of other planets and galaxies challenges current scientific research and pushes technological boundaries. New space missions provide an opportunity to tackle unknown mapping areas in the planetary science community. Similar to geologic mapping on Earth, missions require identifying key features such as craters, canyons, and fault lines on other planets. Scientists can map exploration sites and show what missions will encounter with knowledge of the surrounding area.

The client, the United States Geological Survey (USGS) Planetary Geologic Mapping (PGM) Program, is responsible for many of today's existing planetary maps. USGS supports the National Aeronautics and Space Administration (NASA) in developing these resources for space missions. Many other research teams, such as the Lunar and Planetary Institute, also support the planetary science community by developing maps [1]. These non-USGS organizations often publish their mapping efforts in various journal articles and conference papers. For the client, these publications are valuable map products when gathering resources for a specific region of interest. Currently, the client and the planetary science community lack a centralized system for maps published in journal articles and conference proceedings.

In response to the client's needs, the project team will create a centralized system for all non-USGS map products. The envisioned product, MapONE, is a user-friendly interface on the client's website. The interface will display map images, sources, and other related information for a selected region of interest. The interface will connect to one database containing all non-USGS maps. As part of the product, a web scraper will use machine learning techniques to collect and identify new maps in articles and conference papers. Overall, the product will allow researchers to access critical non-USGS map information.

2. Technological Needs

To collect and display non-USGS map information for the client, the project team will need to develop and integrate the following components:

1. GUI Implementation

A web-based user interface will be the foundation of MapONE. The client will need a simple web application to view and filter map information by author, source, region, etc. Most importantly, the interface needs to display map images using interactive, visual components. This way, researchers can easily visualize the geologic area of interest. A Graphical User Interface (GUI) will provide this interactivity.

2. Web Framework

A web application framework will be the skeleton of MapONE. The framework will be responsible for connecting the interface to the database. When the client uses the product's interface, they will be searching and gathering map information stored in the database. To filter and display the correct map information, the framework can be used to facilitate communication between these two systems. When the client searches for a select criteria (author, date, region, etc.), MapONE's framework must locate the correct information in the database and output it to the interface.

3. Central Database

MapONE must store all map data into one centralized database. Any number of maps may be pulled at any time by the client. The information may also consist of unstructured data such as images, coordinates, source links, publication dates, etc. depending on availability. Thus, the project team needs a database framework that supports changing data sizes with minimal risk on performance. The data will be organized by its most notable characteristics including publication dates, mapping areas, and data sources. This will provide convenience for both the client and the developers when integrating data from the database to the web interface.

4. Machine Learning Enhanced Web Scraper

The project's success heavily relies on MapONE's ability to accurately identify and collect non-USGS maps from various online sources. To accomplish this, the project team needs to develop a data extraction tool known as a web scraper. The team will need to use machine learning techniques, including image recognition, to detect maps. The accuracy of the image detection is essential to the client. Without proper configuration, the tool may collect unrelated information, bypass necessary map products, or add duplicate data. To avoid this potential issue, the project team will need to develop a machine learning model familiar with the key components of a planetary map. Examples could include coordinates, legends, and scales. Finding machine learning libraries with mapping information and related image detection tools will provide this functionality.

5. Remote Storage and Servers

As required by the client, the final product must be hosted on the USGS's website. To create a product, there must be remote servers to handle data storage. With cloud computing, the client and team will have instant access to the project's data at any given time. This flexibility means there is no need for physical storage space, as the cloud will automatically increase storage capacity and update software as needed. Data stored within the cloud will be automatically backed up and synced to all devices for consistency and ease of access.

Other limitations should be considered as well during the development of MapONE. Based on the client's requirements, MapONE must be a Python-based open-source tool. Python is a popular scripting language, a programming language that automates execution. This limits the project's technologies to compatible software. The challenge will be to encompass the project's needs within a Python, open-source environment. The client also requires the product to be packaged in a Docker container. Docker is an open-source containerization platform that packages data and any dependencies into a standard unit of software [2]. Containers offer a simple, portable structure to store data and facilitate easy application deployment. Because of these limitations, all proposed technologies discussed in this document need to be open-source, Python-based or compatible, and can be contained in a Docker.

3. Technological Analysis

For each of these components, the project team outlines potential technologies that can satisfy MapONE's technological needs. Based on the following analysis, the technologies best suitable for product development will be integrated into the final product.

3.1 GUI Implementation

Technological Need

As the foundation of MapONE, the GUI needs to navigate researcher requests for map information. The interface needs to be visually appealing and user-friendly. Overall, MapONE needs to display a variety of demographics to ensure each planetary region in the system offers source information and interactive map images.

Criteria

For each proposed GUI framework outlined in this section, the project team will use the following criteria to rank the best solution.

Criteria	Description
Implementation	The ability to integrate into the overall project solution. May include the project team's familiarity with the technology.
Documentation	The technology is well-documented and user-friendly.
Cross Compatibility	The technology can effectively complete runtime conversions.

Proposed Technologies

Flutter

Flutter is a GUI framework designed by Google and built upon the Dart programming language [3]. Dart is a Java Virtual Machine (JVM) based language which indicates its ability to be compiled into Java bytecode [4]. This can be used to the project team's advantage as an easy way

to convert code from one language and then import into Dart. Because Java and Kotlin are built upon JVMs, these languages provide an easy way to interact with Dart. For MapONE's purposes, Python can be converted into Dart using a simple library [5].

Flutter utilizes material design for its icons, interactivity, and appearance. Flutter is based on the concept of widgets. Widgets are objects that heavily rely upon Object Orientated Programming (OOP) concepts [3]. Figure 1 shows the encapsulation of an AppBar widget which the project team found can easily create a search bar. Similar to this example, Flutter offers a variety of widgets to create icons, place images, and add text to an interface. MapONE will need these features to create a simple but interactive interface for the client.

```
class _HomeScreenState extends State<HomeScreen> {
 Icon customIcon = const Icon(Icons.search);
 Widget customSearchBar = const Text('My Personal Journal');
 @override
 Widget build(BuildContext context) {
 return Scaffold(
  appBar: AppBar(
  title: customSearchBar,
  automaticallyImplyLeading: false,
  actions: [
   IconButton(
   onPressed: () {},
   icon: customIcon,
   )
  ],
  centerTitle: true,
  ),
```

Figure 1. A simple search bar implementation in Futter [6].

Although Flutter offers an interactive environment, the development environment requires a high level of proficiency and comfort in a Unix-based terminal. The technology requires the user to

edit the main configuration files `./bashrc` and `./zshrc` depending on the terminal environment [3]. This may cause implementation issues for the project team as Flutter's complexity may require more expertise. Fortunately, the project team is familiar with the software at the configuration level.

React.js

React.js is a Javascript web-based framework owned by Facebook [7]. The framework can create simple, concise, and visual UIs. React.js also pairs objects associated with markup and logic. For product purposes, the project team can easily edit markup-related objects inside the framework instead of in a separate file [7]. This reduces development errors since the project team would only be editing within the framework. Similar to Flutter, rather than decoupling markup and logic, React.js creates concrete classes using widgets with a degree of inheritance [7]. This would provide modularity to MapONE's functions. React.js also uses a syntax extension called JSX which defines these objects as an expression instead of a markup; this syntax is similar to HTML which is easier to understand for the project team [7].

Unlike Flutter, React.js does not have runtime compatibility with other languages. This can be problematic because the project will need to integrate other languages and solutions. For MapONE purposes, using Python may require multiple calls to the backend for any required frontend service.

Analysis

The above technologies are rated based on the following criteria and scored using a rating system between 1 (inadequate) and 5 (exceptional). The adequacy rating (total score over possible score) determines how well-suited a technology is for the final product expectations.

Technology	Implementation	Documentation	Cross Compatibility	Adequacy Rating (%)
Flutter	4	5	5	93%

React.js	2	5	2	60%

Decided Solution & Feasibility

Based on the above analysis, Flutter is the choice of technology. This decision was based on the implementation criteria as both frameworks have excellent documentation. The project team's familiarity allowed for ease of implementation. Because Flutter supports runtime conversions between multiple languages (including a plug-in for Python), implementing a wide range of technologies will be most suitable. Furthermore, the limited cross-compatibility React.js offers does not support the envisioned solution.

3.2 Web Framework

Technological Need

Since MapONE is a web-based tool, a web framework is necessary for the project's interface and database connection. The framework facilitates the communication between the two and structures their input and output to each other. In this way, map data stored in the database can be easily accessed by the interface.

Criteria

For each proposed web framework outlined in this section, the project team will use the following criteria to rank the best solution.

Criteria	Description
Implementation	The ability to integrate into the overall project solution. May include the project team's familiarity with the technology.
Documentation	The technology is well-documented and user-friendly.
Production-Ready	The technology is intended for deployment purposes.

Proposed Technologies

Django

Django is a Python-based web framework. The full-stack framework provides both a frontend and backend to easily create full web applications. The project team found Django's easy setup useful for MapONE as a production-ready application. Figure 1 shows how quickly the Django server can be launched on localhost. The framework also establishes security, administrative permissions, and configuration settings. Figure 2 shows how Django's settings can be configured to run on different hosts and has built-in security measures. Because MapONE will be hosted on the client's website, Django can create a secure, portable application. Because of its variety of features and the project team's familiarity, Django is easy to implement and deploy. Lastly, Django is one of the most popular web technologies and provides thorough documentation.

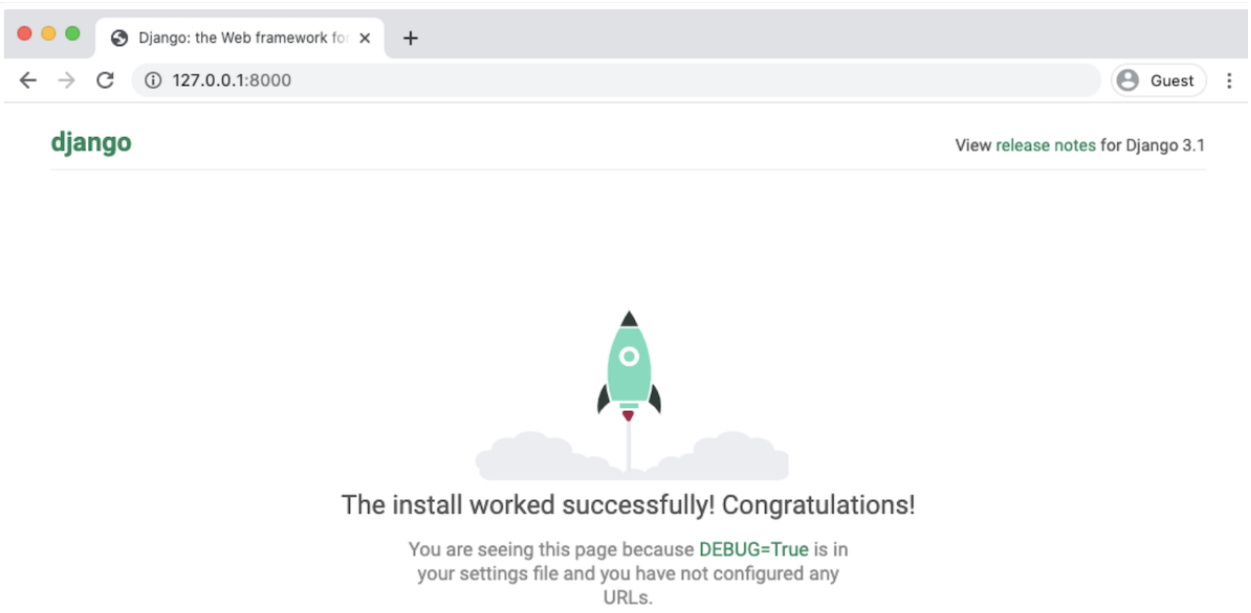

Figure 1. Django server operating on localhost.

```
21  # Quick-start development settings - unsuitable for production
22  # See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/
23
24  # SECURITY WARNING: keep the secret key used in production secret!
25
26  with open('secret_key.txt') as f:
27      SECRET_KEY = f.read().strip()
28
29  # SECURITY WARNING: don't run with debug turned on in production!
30  DEBUG = False
31
32  ALLOWED_HOSTS = ['radiant-anchorage-77650.herokuapp.com']
33
```

Figure 2. Django's configuration settings for a simple web application.

Flask

Flask is a lightweight, Python-based web framework and is one of the most popular web application technologies [8]. Whereas Django is structured, Flask does not use a specific layout. The project team noted developers have full control over design decisions in Flask. Whereas Django's application components must adhere to its built-in template, Figure 3 shows how the Flask can be configured in numerous ways. This may be useful for MapONE as client needs and requirements change. However, unlike Django, Flask is not a "production-ready" framework [8]. Although the project team is less familiar with Flask, the technology is known for its ease of implementation. Lastly, because of its popularity and easy setup, Flask is well-documented and provides user guides.

```
from flask import session

# Set the secret key to some random bytes. Keep this really secret!
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

@app.route('/')
def index():
    if 'username' in session:
        return f'Logged in as {session["username"]}'
    return 'You are not logged in'
```

Figure 3. Flask's development settings and the interface can be configured in the same file [9].

Analysis

The above technologies are rated based on the following criteria and scored using a rating system between 1 (inadequate) and 5 (exceptional). The adequacy rating (total score over possible score) determines how well-suited a technology is for the final product expectations.

Technology	Implementation	Documentation	Production-Ready	Adequacy Rating (%)
Django	5	5	5	100%
Flask	4	5	2	73%

Decided Solution & Feasibility

Based on the above analysis results, Django is the choice of technology for the web framework. Although both technologies are well-documented and easy to implement, the team's familiarity with Django over Flask will allow for easier development. Most importantly, because Flask is not intended for heavy-weight production-ready applications, Django would be more suitable for a deployed finished project. Overall, Django provides a simple framework to access and structure MapONE's planetary data for the interface view.

3.3 Central Database

Technological Need

A database is an organized collection of structured information that is stored on a server. In this sense, a database can be seen as an electronic archive where specific information is stored for use. The team will use the database as a central repository for all maps and journals found through the use of a web scraper. Through web scraping, any map or journal will be collected and stored in MapONE's database for ease of access and display.

Criteria

For each proposed database structure outlined in this section, the project team will use the following criteria to rank the best solution.

Criteria	Description
Scalability	The technology can account for and support changing data sizes, functionality, and architecture.
Performance	The technology can timely execute functions and operations.
Data Management	The technology can efficiently manage large data amounts.
Cost	The technology is cost-effective to the client.

Proposed Technologies

PostgreSQL

PostgreSQL is a database structure recommended by the client. With over thirty years of development, this reliable, stable database provides a variety of features [10]. Plug-ins, including a Geographical Information System (GIS), can be used for mapping information [11]. Figure 1 shows how data can be structured using GIS. MapONE would store map coordinates and region information in a similar way. Figure 2 also gives an example of how PostgreSQL can be organized.

As an open-source database, PostgreSQL requires no cost and offers detailed documentation. The database also supports web applications that are necessary for the product's web-based

design. Despite these benefits, PostgreSQL's performance is comparatively lower than other common database structures. For example, the database often demonstrates slower data uploads. Since MapONE will be using an automated web scraper for uploads, these operations may experience delays.

PostgreSQL poses potential issues for scalability as well. The server running the structure needs to be upgraded when the database expands. Thus, the database installation must be the same for all operating systems. Otherwise, the structure may not support all possible functions. This means the same database version is required to minimize formatting errors. The project team can reduce this risk by configuring the web framework's settings; installing a specific database version can be easily accomplished each time the application is run.

```
 num |         street          |  city   | state |  zip
-----+-------------------------+---------+-------+-------
 1   | Devonshire Place PH301  | Boston  | MA    | 02109
```

Figure 1. Data output from the GIS plug-in [12].

Field	Type	Length	Description
pd_lsn	PageXLogRecPtr	8 bytes	LSN: next byte after last byte of WAL record for last change to this page
pd_checksum	uint16	2 bytes	Page checksum
pd_flags	uint16	2 bytes	Flag bits
pd_lower	LocationIndex	2 bytes	Offset to start of free space
pd_upper	LocationIndex	2 bytes	Offset to end of free space
pd_special	LocationIndex	2 bytes	Offset to start of special space
pd_pagesize_version	uint16	2 bytes	Page size and layout version number information
pd_prune_xid	TransactionId	4 bytes	Oldest unpruned XMAX on page, or zero if none

Figure 2. An example of how PostgreSQL handles data organization [13].

MySQL

MySQL is the most common database for novice developers. Thorough documentation is available on many online resources. The versatile database allows for multiple program language

connections; these connections include languages such as Python and PHP [14]. Development tools can also be easily integrated using MySQL. The database is capable of effectively managing the Central Processing Unit (CPU) resources which would improve the execution time of processes and requests [15]. In turn, this would improve the web scraper portion of the project. With faster CPU times and memory management, the web scraper has the potential to accomplish more tasks on a single run cycle. MySQL also offers cross-platform support; the database can support all operating systems with minimal compatibility issues.

Although MySQL offers system flexibility, the database is not suitable for large data management. Figure 3 shows how MySQL handles spatial data storage. This does not provide the full functionalities required of MapONE; a mass collection of planetary maps is necessary for the project solution. Security may also be a priority for MapONE, and this database is known for infrequent security updates.

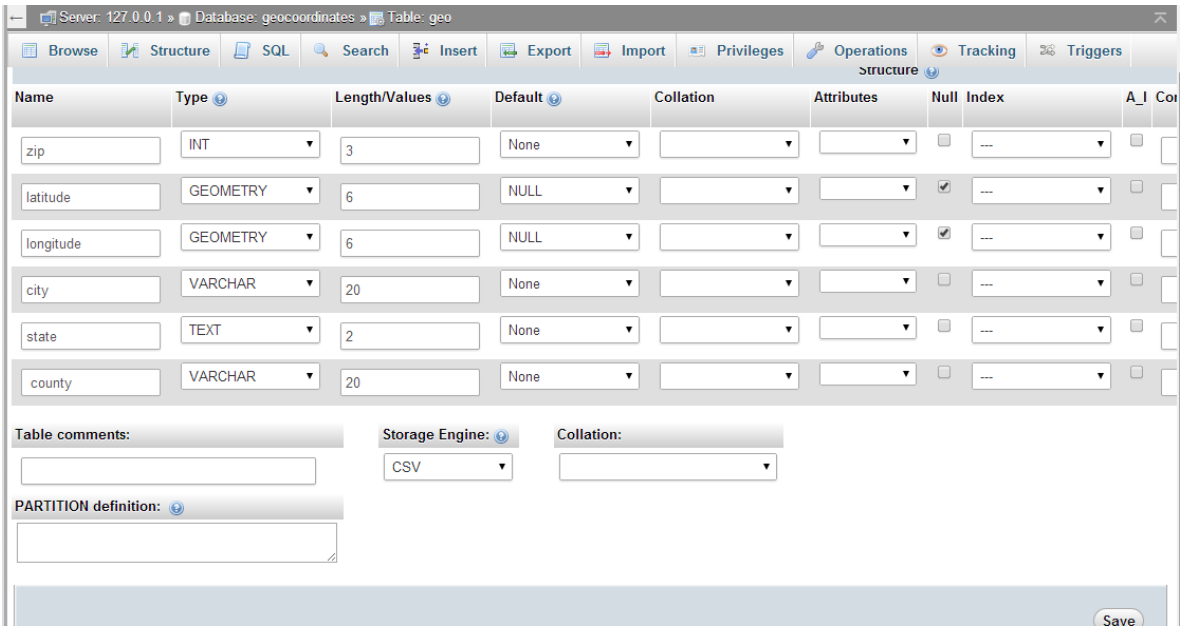

Figure 3. User view of MySQL's database [16].

MongoDB

Unlike PostgreSQL and MySQL, MongoDB is not a SQL-based database. MongoDB is used to structure unformatted data instead of relational tables [17]. Figure 4 shows an example of how

the database handles data organization. MongoDB also offers faster query execution as the data is stored in a computer's Random-Access Memory (RAM) instead of the hard drive while operating. This would be beneficial to the client's request that the project is hosted on a constant running server. The database also provides documentation on storage and indexing, and the query syntax is easier to understand compared to SQL.

MongoDB offers high-speed performance and querying. Thus, incorrect indexing can create data discrepancies and slow performance. The largest data size this database can store is 16 megabytes [18]. This is a critical issue for the project team since MapONE requires a large storage capacity. Lastly, since MongoDB is closed-source, it requires an additional cost to the clients.

name	quantity	size	status	tags	rating
journal	25	14x21,cm	A	brown, lined	9
notebook	50	8.5x11,in	A	college-ruled,perforated	8
paper	100	8.5x11,in	D	watercolor	10
planner	75	22.85x30,cm	D	2019	10
postcard	45	10x,cm	D	double-sided,white	2

Figure 4. An example of how MongoDB may store data [19].

Analysis

The above technologies are rated based on the following criteria and scored using a rating system between 1 (inadequate) and 5 (exceptional). The adequacy rating (total score over possible score) determines how well-suited a technology is for the final product expectations.

Technology	Scalability	Performance	Data Management	Cost	Adequacy Rating (%)
PostgreSQL	4	3	5	5	85%
MySQL	4	4	2	5	75%
MongoDB	4	5	5	2	80%

Decided Solution & Feasibility

Based on the above analysis results, PostgreSQL is the choice of technology. The fact USGS uses this database to structure its maps is one of the major deciding factors. The project team would also have plug-in access for handling the specific mapping data. MySQL's limited mass data management is not suitable for MapONE's application. MongoDB is also unsuitable as a closed-source database; the client would be required to pay an additional cost which is not necessary for the project's scope at this time.

3.4 Machine Learning Enhanced Web Scraper

Technological Need

Artificial Intelligence (AI) is the general term for any technique used by machines to imitate human intelligence. Machine Learning (ML) is a branch of AI that uses deep learning algorithms to improve a machine's capability with certain tasks via repetition. The project team must locate and identify planetary maps throughout the internet. ML can assist and increase the speed of the web scraping process. A solid machine learning library with image recognition capabilities is necessary for extracting viable data from the internet. As a design choice, the project team will also use existing Python libraries to fluidly integrate machine learning into MapONE.

Criteria

For each proposed machine learning-assisted library outlined in this section, the project team will use the following criteria to rank the best solution.

Criteria	Description
Scalability	The technology can account for and support changing data sizes, functionality, and architecture.
Implementation	The ability to integrate into the overall project solution. May include the project team's familiarity with the technology.
Documentation	The technology is well-documented and user-friendly.
Performance	The technology can timely execute functions and operations.

Proposed Technologies

Pandas

Pandas is an open-source Python library used for data manipulation and analysis. Unlike the other technologies in this category, Pandas is built over Numpy, a package used for data science [20]. Pandas can evaluate data from a variety of sources, including SQL, using ML techniques. Most notably, Pandas can coincide with other Python libraries for more versatile functionality. Overall, Pandas is a powerful tool used for machine learning and has become the "backbone" of many data science projects [21].

Beyond data organization, Pandas requires assistance from other ML libraries for image detection; thus, it cannot be used as a standalone software for MapONE's development. However, Pandas' data sorting is a convenience the project team will likely consider after the data collection. Figure 1 shows how convenient reading in data and creating subsets for analysis are.

```
[1]  import pandas as pd

     dataset = pd.read_csv('cancer.csv')
```

```
[ ]  x = dataset.drop(columns=["diagnosis(1=m, 0=b)"])
```

```
[ ]  y = dataset["diagnosis(1=m, 0=b)"]
```

Figure 1. Creating and organizing a dataset into a table using Pandas.

TensorFlow

With a focus on AI training and deep neural networks, TensorFlow is an open-source library accessible via Python and Javascript. Analysts typically use TensorFlow for large numerical computations with its flexible architecture [22]. This allows users to quickly and easily deploy ML tools harnessing advanced techniques, such as image recognition and text generation. TensorFlow is also well-documented with numerous tools and demos available to the community. The project team can take advantage of these tools since the web scraper will require accurate planetary map detection.

A reliable architecture will also benefit the implementation of image recognition given the complexity of select planetary images. TensorFlow alone cannot classify an image as a planetary map. Instead, Keras provides this functionality.

Keras

Created as an extension of the TensorFlow platform, Keras is a powerful interface used to solve ML problems of large iterations. Keras is used by NASA for its performance and scalability [23]. Many developers also use Keras as their primary ML software for its readable syntax. Considering the large dataset the web scraper will predictably collect, Keras may be a more practical solution for handling the unknown size of the database.

Keras creates datasets using validation splitting, a tool commonly used in ML to "fine-tune" model performance [24]. This means it can create training and testing sets to ensure the final model is accurate. Keras can also implement data augmentation on the training set; this includes performing random transformations on existing data to accommodate the learning model [25]. Figures 2 and 3 are examples of these tools.

Lastly, since Keras is an extension of TensorFlow, documentation for the library mostly appears in TensorFlow forums and can be easily accessed.

```
[ ]  train_ds = tf.keras.utils.image_dataset_from_directory(
       data_dir,
       validation_split=0.2,
       subset="training",
       seed=123,
       image_size=(img_height, img_width),
       batch_size=batch_size)
```

```
[ ]  val_ds = tf.keras.utils.image_dataset_from_directory(
       data_dir,
       validation_split=0.2,
       subset="validation",
       seed=123,
       image_size=(img_height, img_width),
       batch_size=batch_size)
```

Figure 2. A dataset is split into training and validation sets.

```
[ ]  data_augmentation = keras.Sequential(
       [
         layers.RandomFlip("horizontal",
                           input_shape=(img_height,
                                        img_width,
                                        3)),
         layers.RandomRotation(0.1),
         layers.RandomZoom(0.1),
       ]
     )
```

Figure 3. Augmentation of an image to be flipped horizontally from its original position.

Analysis

The above technologies are rated based on the following criteria and scored using a rating system between 1 (inadequate) and 5 (exceptional). The adequacy rating (total score over possible score) determines how well-suited a technology is for the final product expectations.

Technology	Scalability	Integration	Documentation	Performance	Adequacy Rating (%)
Pandas	3	5	5	4	85%
TensorFlow	4	5	5	4	90%
Keras	5	5	4	5	95%

All three solutions prove to be effective technologies for the client. Pandas provides exceptional readability, documentation, and can be easily integrated with either of the remaining technologies. TensorFlow is the most documented library, providing extensive demos that directly correlate to MapONE's needs. Keras is simply a more advanced version of TensorFlow designed to efficiently read large datasets and maintain simplicity.

Decided Solution & Feasibility

Based on the above analysis results, the most practical solution for the project team will use Keras as an extension to TensorFlow. Given the project's database will expand over time, Keras will manage data uploads. Keras is imported through TensorFlow as a Python library. Thus, the project team can utilize TensorFlow's strong documentation for Keras. For performance, Keras is designed with less extensive implementation to promote "fast experimentation" for quicker results [23].

3.5 Remote Storage and Servers

Technological Need

Cloud computing services provide a remote way to access and secure data. Effective data storage would prevent data loss and facilitate continuous product deployment. With this service, USGS can access MapONE's map data on a remote, secure service.

Criteria

For each proposed cloud computing service outlined in this section, the project team will use the following criteria to rank the best solution.

Criteria	Description
Scalability	The technology can account for and support changing data sizes, functionality, and architecture.
Implementation	The ability to integrate into the overall project solution. May include the project team's familiarity with the technology.
Documentation	The technology is well-documented and user-friendly.
Cost	The technology is cost-effective to the client.

Proposed Technologies

Amazon Web Services (AWS) S3

AWS S3, a technology recommended by the client, is the leading industry's data storage service at 33% of the total market share [26]. Amazon offers several storage classes with uses ranging from frequent to infrequently accessed data. Specifically, the S3 Intelligent-Tiering storage class is used for both types and accounts for changing data patterns [27]. This class would be the most beneficial to monitor the unpredictable volume of data; MapONE's web scraper will be automatically and continuously adding new map data. Thus, a storage class must account for these frequent changes.

Amazon also supports a wide range of clients and datasets. Companies, especially startups, use AWS regardless of size or overhead which is ideal for a project of this scale. S3 is also easy to implement given the project team's familiarity with the software and the thorough documentation. Amazon provides an S3 Representational State Transfer (REST) API as a reference for its cloud computing services. Creating an account, setting up servers, and managing costs are all documented on the Amazon website [27].

Microsoft Azure

Microsoft Azure, a technology recommended by the client, is a fierce competitor of AWS and second in the industry. For USGS's purposes, Azure's storage class Blobs "allows unstructured data to be stored" and supports "random access" [28]. As previously mentioned, the planetary map datasets may frequently change; this service provides the ability to account for unpredictable access and updates.

Similar to S3, because of Azure's popularity, the service is well-documented and includes training and developer guides. The ease of implementation would be similar, although the project team is less familiar with its services than AWS. Azure is most known for its large clientele with ninety-five percent of Fortune 500 companies as clients [28]. Many of these companies partner with Microsoft for its wide range of products. For this reason, Azure is a prime candidate for USGS to partner with and may be more cost-effective; Microsoft offers bundle purchases for a variety of their products. According to CAST AI, a cloud expert platform, an upfront

commitment plan (compute optimized) costs about fifty percent less for Azure than AWS. However, Azure and AWS offer similar pricing for on-demand rates (general purpose and memory) [29].

Analysis

The above technologies are rated based on the following criteria and scored using a rating system between 1 (inadequate) and 5 (exceptional). The adequacy rating (total score over possible score) determines how well-suited a technology is for the final product expectations.

Technology	Scalability	Implementation	Documentation	Cost	Adequacy Rating (%)
AWS S3	5	5	5	4	95%
Microsoft Azure	4	4	5	5	90%

Notably, both services provide similar capabilities. However, because of its notoriety for large applications and easy setup, S3 provides a better beginner-friendly environment. However, Azure provides a more cost-effective approach for large organizations like USGS. Overall, as the two most popular cloud computing services, both AWS S3 and Microsoft Azure are well-documented interfaces.

Decided Solution & Feasibility

Based on the above analysis results, AWS S3 is the choice of technology. Because Amazon offers scalable solutions, ease of implementation, and thorough documentation, the project team will use this service for cloud computing. To integrate into the final solution, AWS S3 will serve as the basis for storing all planetary map data including images, source links, and publication dates. USGS servers will be hosted remotely on AWS.

4. Technology Integration

Based on the previous analysis, the following components form MapONE's web application tool.

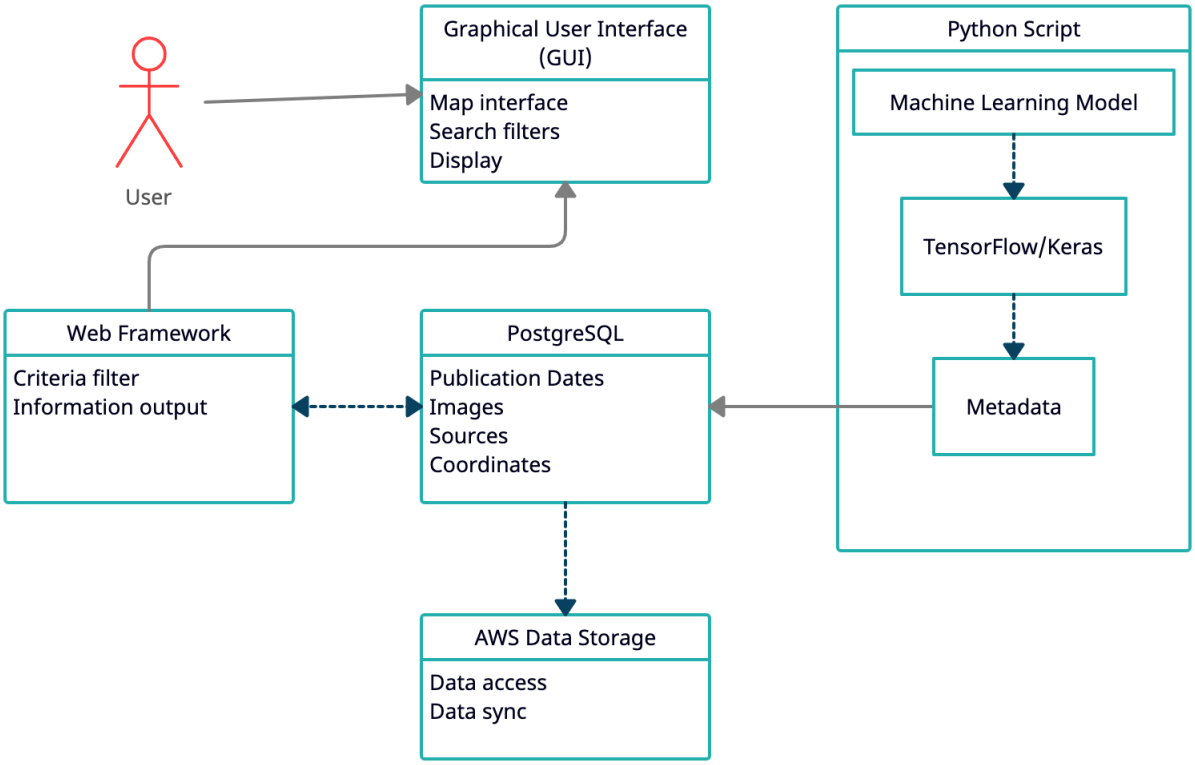

Figure 1. System diagram for MapONE.

In this figure, the system process is broken down into five main components. At the center is the database, where all metadata collected from the machine learning model will be stored. AWS's cloud computing service will host and store all information passed into the database. Django's web framework will filter the dataset to satisfy USGS criteria and display it to the GUI. Lastly, the user will be able to interact with the database using a Flutter interface.

5. Conclusion

As previously mentioned, the client currently lacks a centralized system for all planetary map publications. Many non-USGS maps are often published in journal articles and conference papers. This limits USGS's ability to gather all resources for a specific mapping area of interest. MapONE, a planetary map interface, will display metadata on these valuable map products. This product will support the planetary science community as a central point for map information.

As shown in the system diagram, the envisioned product encompasses many aspects as a web application. A web scraper will be used to scan journals and other academic sources for notable map information (region, images, etc.). This machine-learning assisted tool will then identify and collect relevant data. Most importantly, MapONE's interface will display this to researchers.

Throughout this document, the project team analyzed potential technologies for MapONE's GUI, web framework, database structure, web scraper, and data storage. These components are necessary to build an automated web-based tool capable of identifying, collecting, and displaying map data. From the project team's research and analysis, the following technologies will be used for this tool:

1. **Flutter**: A user interface tool capable of displaying visual map components.
2. **Django**: A web framework that can connect MapONE's interface and database.
3. **PostgreSQL**: A database structure to support and store map data.
4. **Keras and TensorFlow**: Machine learning libraries used to identify map information.
5. **AWS**: A cloud computing and data storage service to host the product's application.

Although these technologies and their integration form MapONE's foundation, additional steps are necessary to create a fully functional product. Next, the project team will expand on the individual functional and non-functional requirements for each component discussed in this document.

References

[1] "Lunar and Planetary Institute." Lunar and Planetary Institute. https://www.lpi.usra.edu/ (accessed October 28, 2021).

[2] "What is a Container?" Docker. https://www.docker.com/resources/what-container (accessed October 28, 2021).

[3] "Beautiful native apps in record time." Flutter. https://flutter.dev/ (accessed October 15, 2021).

[4] "Binding to native code using Dart:FFI." Flutter. https://flutter.dev/docs/development/platform-integration/c-interop (accessed October 15, 2021).

[5] H. Javaid. "How to run Python scripts on Flutter." Medium. https://medium.com/@ihassanjavaid/how-to-run-python-scripts-on-flutter-d6a4aedb6227 (accessed October 15, 2021).

[6] "How to create a search bar in Flutter." LogRocket Blog. https://blog.logrocket.com/how-to-create-search-bar-flutter/ (accessed October 28, 2021).

[7] "React – a JavaScript library for building user interfaces." React.js. https://reactjs.org/ (accessed October 15, 2021).

[8] "Flask vs Django in 2021: Which Framework to Choose?" Hackr.io. https://hackr.io/blog/flask-vs-django (accessed October 18, 2021).

[9] "Quickstart." Flask. https://flask.palletsprojects.com/en/2.0.x/quickstart/#deploying-to-a-web-server (accessed October 28, 2021).

[10] R. Peterson."What is PostgreSQL? Introduction, Advantages & Disadvantages" Guru99. https://www.guru99.com/introduction-postgresql.html (accessed October 15, 2021).

[11] "PostgreSQL." PostgreSQL https://www.postgresql.org/ (accessed October 15, 2021).

[12] "PostGIS 3.1.5MDev manual." PostGIS. https://postgis.net/stuff/postgis-3.1.pdf (accessed October 31, 2021).

[13] "70.6. Database Page Layout." PostgreSQL. https://www.postgresql.org/docs/current/storage-page-layout.html (accessed October 31, 2021).

[14] "MySQL." MySQL. https://www.mysql.com/ (accessed October 15, 2021).

[15] "Pros and Cons of MySQL 2021." Trustradius. https://www.trustradius.com/products/mysql/reviews?qs=pros-and-cons (accessed October 15, 2021).

[16] "Table structure for Geo Spatial Data." Stack Overflow. https://stackoverflow.com/questions/25232316/table-structure-for-geo-spatial-data/25237180 accessed (October 31, 2021).

[17] "Understanding the Pros and Cons of MongoDB." KnowledgeNile. https://www.knowledgenile.com/blogs/pros-and-cons-of-mongodb/ (accessed October 15, 2021).

[18] Xtivia. "The Pros and Cons of MongoDB." Virtual-DBA. https://www.virtual-dba.com/blog/pros-and-cons-of-mongodb/ (accessed October 15, 2021).

[19] "Structure your data for MongoDB." MongoDB. https://docs.mongodb.com/guides/server/introduction/ (accessed October 31, 2021).

[20] "What is Pandas In Python? Everything You Need to Know." ActiveState. https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/ (accessed October 18, 2021).

[21] "Python Pandas Tutorial: A Complete Introduction for Beginners." Learndatasci. https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/\ (accessed October 18, 2021).

[22] "An end-to-end open source machine learning platform." TensorFlow. https://www.tensorflow.org/ (accessed October 18, 2021).

[23] "About Keras." Keras.io. https://keras.io/about/ (accessed October 18, 2021).

[24] "The Train, Validation, and Test Sets: How to Split Your Machine Learning Data." V7Labs. https://www.v7labs.com/blog/train-validation-test-set (accessed October 31, 2021).

[25] "Image Classification." TensorFlow. https://www.tensorflow.org/tutorials/images/classification (accessed October 31, 2021).

[26] V. Sharma, V. Nigam, and A. Sharma, "Cognitive analysis of deploying web applications on microsoft windows azure and amazon web services in global scenario," *Materials Today: Proceedings*, vol. 10, no. 126, Nov. 2020, doi:10.1016.

[27] "Amazon S3." Amazon. https://aws.amazon.com/s3/?nc=sn&loc=1 (accessed October 18, 2021).

[28] "Microsoft Azure: Cloud Computing Services." Microsoft. https://azure.microsoft.com/en-us/ (accessed October 18, 2021).

[29] P. Santis. "Ultimate cloud pricing comparison: AWS vs. Azure vs. Google Cloud in 2021." CAST AI. https://cast.ai/blog/ultimate-cloud-pricing-comparison-aws-vs-azure-vs-google-cloud-in-2021/ (accessed October 18, 2021).