MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

SOFTWARE ARCHITECTURE DESCRIPTION (SAD)

SPRING 2022-2023

---

afetbilgi.com

Group 24

Burak Metehan Tunçel
2468726

Saad Yousuf
2349819

Contents

List of Figures

List of Tables

1 Introduction

This document is the Software Specification Requirement (SRS) of a website designed to help earthquake victims to acquire the necessary information and give volunteers a chance to donate to help earthquake victims. The website is called afetbilgi.com[1], developed by Middle East Technical University (METU) students and graduates.

1.1 Purpose and Objectives of afetbilgi.com

afetbilgi.com, direct translation to English is 'disaster documentation', is an open-source efforted project led by students from METU in Ankara, Turkiye. It aims to provide a clean, verified, and correctly classified information interface for earthquake victims and helpers alike in the aftermath of the tragic earthquake on February 6th, 2023, in Pazarcik, Turkiye. It also offers quick information using confirmed website links, maps, and address tables, along with the relevant contact details of organizations and helpers involved.

1.2 Scope

afetbilgi.com was established to offer as much information as needed by users in three main categories:

- People who are affected by the earthquake (the victims).
- Individuals/Organisations who want to help and participate in other governmen-t/private efforted procedures in the affected areas.
- People from METU who verify and checked any presented links on the websites.

The website is primarily responsible for providing tables and datasheets with web-site links to third-party organizations/contacts details of web places/physical locations which offer/collect help. As indicated here, these links are external and lead out to other websites(outside from afetbilgi.com) whose efforts are verified by human resolves (METU students/helpers/site administrators) on the surface-level user experience.

Given how the world is connected with the internet and phones/televised communication, the project developers aim to create a website using these advantageous characteristics via a simple interface in multiple available languages to create fast and easy use of information with no additional and unnecessary obstacles. In areas lacking internet infrastructure that might have been disturbed by the earthquake activities, the website can be distributed via printed-out PDFs, which are shareable via ordinary computers and mobiles, and hand-forwarded physical versions in the forms of leaflets and so on.

Lastly, afetbilgi.com includes a map functionality if the victim/helper has an internet connection. Any user can locate helper geolocations via terrain/road routes while also being able to quickly view extra details such as written addresses, contact phone details, and previous reviews.

1.3 Stakeholders and Their Concerns

There are three main categories of people related to afetbilgi.com:

1. **Earthquake victims/ affectees:** These individuals whom the earthquake has directly impacted seek help, support, and information to recover from the disaster. They may be looking for information on how to find shelter, food, medical assistance, and other resources that can help them get back on their feet. The website may provide them with a platform to connect with relief organizations and volunteers and access information on navigating the recovery process.

2. **Volunteers:** These individuals want to offer their time, skills, and resources to support the relief and recovery efforts. They may include local volunteers, international volunteers, and disaster response teams. The website may attract volunteers by providing information on how to get involved, where to go, and what support is needed. Their primary use of the website could be to scout places to help from outside the main areas, such as centers transporting essential needs to stricken areas like farther cities such as Ankara and Istanbul. This is the target sector for the Donate or Help category, such as via blood donation, monetary donation, physical

volunteer help, etc. Other entities such as relief organizations, government agencies, more prominent sponsors, and potential media outlets can exist within this category.

3. **Web developers, Data Collectors, and Site administrators:** These are the website creators responsible for developing, designing, and maintaining the platform. They may include web developers, designers, and other professionals involved in creating and managing the website. These stakeholders may be vested in ensuring the website is accessible, user-friendly, effective, and, most importantly, providing simple, verified information to facilitate relief and recovery efforts without any hurdles.

2 References

This document is prepared with respect to IEEE 42010-2022 [2] standard.

References

[1] A. B. İşlem Merkezi, *Afetbilgi — afetler hakkında doğru ve güncel bilgiler*, `http://www.afetbilgi.com/`, February, 2023.

[2] IEEE, "Iso/iec/ieee international standard for software, systems and enterprise – architecture description," *ISO/IEC/IEEE 42010:2022(E)*, 2022. DOI: `10.1109/IEEESTD.2022.9938446`. [Online]. Available: `https://ieeexplore.ieee.org/document/9938446`.

3 Glossary

Term	**Definition**
Python	Computer Programming language to create applications, features, etc.
React	A JavaScript framework widely used to create websites
JavaScript	Scripting programming language used to create applets for the internet
CI/CD	Continuous Integration/Continuous Development
HTTPS	An internet protocol known as Hyper Text Transfer Protocol Secured
AWS	Amazon Web Services the provide web hosting servicing
CloudFlare	A secure DNS hosting service
Vercel	A web hosting service
UI/UX	User Interface/User Experience, meant to refer to the frontend part of a web app that the target audience of the website interact with
PDF	Known as Portable Document Format for easy sharing
IP	Internet Protocol
DNS	Domain Name Server
AWS IAM	Amazon Web Services Identity and Access Management is a service that enables you to manage access to AWS resources securely by creating and managing users, groups, and roles with assigned permissions.

Table 1: Glossary of Terms

4 Architectural Views

4.1 Context View

4.1.1 Stakeholders' Uses of This View

In this view, system context is provided, with the actors and other systems related to it in a general manner. Website maintainers and Data collections of afetbilgi.com will be the ones that primarily use this view to better understand how the different actors interact in the working of this website and the relationship that the system has with other external entities like the AWS components, Maps API component, and so on.

4.1.2 Context Diagram

afetbilgi.com[1] is not part of a more extensive system. It is a standalone and open-source efforted website to verify critical information in the fight against the 6 February 2023 Pazarcik Earthquake and deliver it to disaster victims and those who want to help in an understandable, concise manner in multiple languages.

This information is presented in either the form of legible tables with third-party governmental and private links or an interactable method via a map view interface. If deemed necessary, admin and maintainers can make changes to display newly created or edited data and upload it to the system upon any complaints or suggestions they may get on their contact details.

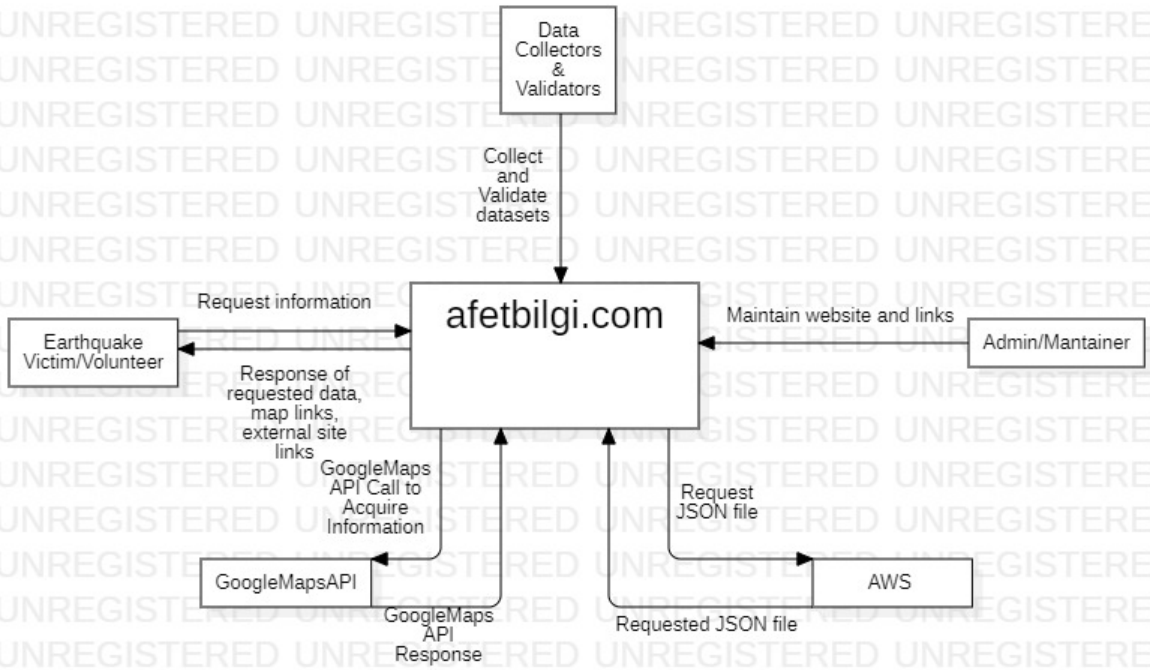

Figure 1: Context Diagram for afetbilgi.com

The afetbilgi.com consists of a combination of small physical and software parts. With the help of interfaces, these parts communicate among themselves and with the user.

4.1.3 External Interfaces

In this section, the external interfaces of the afetbilgi.com will be provided, as well as their operations and relationships.

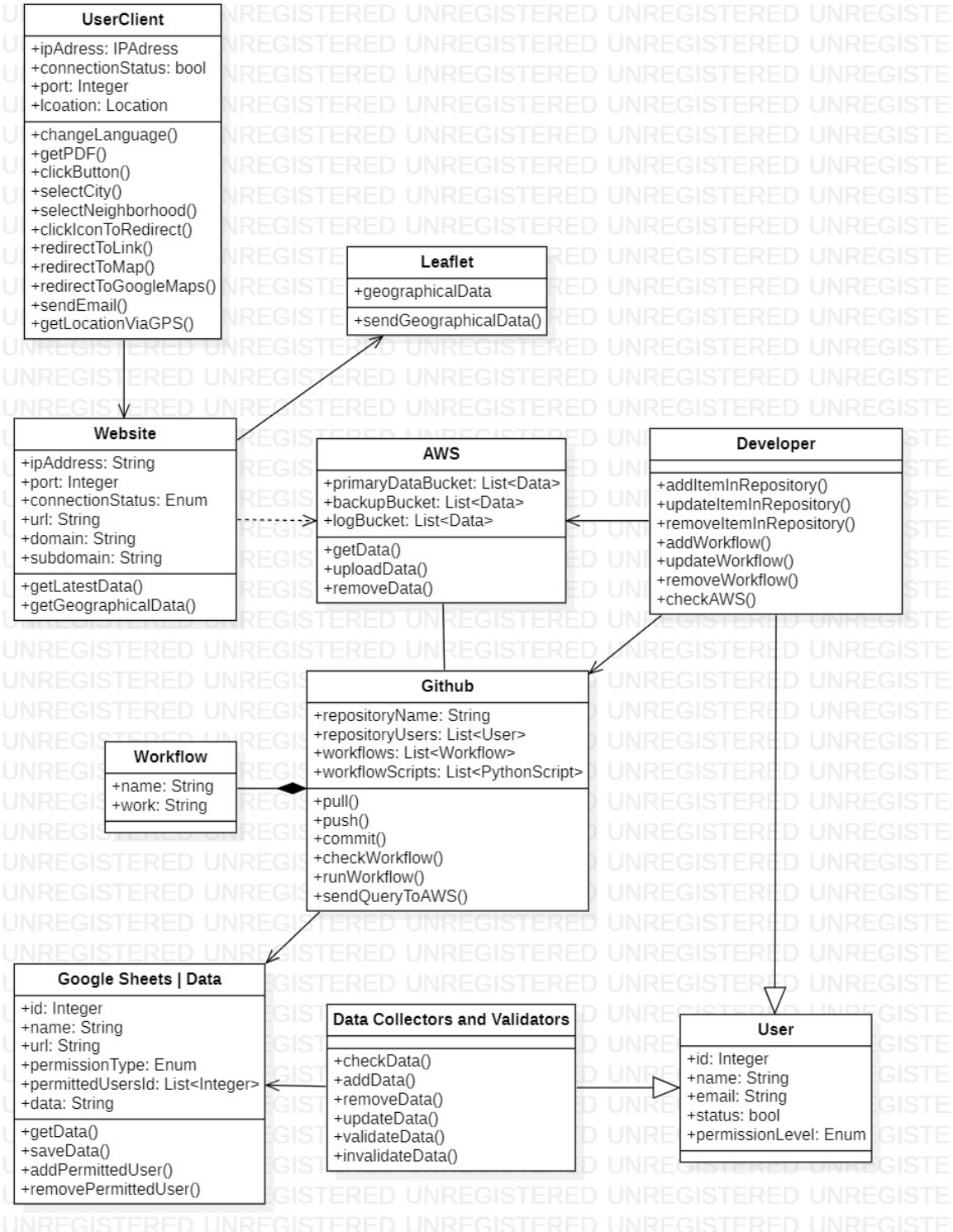

Figure 2: External Interfaces

As it can be observed from Figure 2, afetbilgi.com has multiple external interfaces. UserClient, Website, Leafler, AWS, Developer, Google Sheets for data, Data Collectors and Validators are external interfaces of the system. Github Repository for the afetbilgi.com may also ve considered an external interface since it is generally responsible for sustainability of the system. The operations given in the diagram can be summarized as follows:

- The Data Collectors and Validators collect data and validate it. After validation, data is added into a specificied data sheet in Google Sheets.

- Google Sheets mainly store the data. The data is divided into several files in Google Sheets. These sheets can be accessible for data collectors and validators.

- GitHub Repository is used storing the source code. Additionally, the GitHub workflows of the repository check, maintain and update the website in regular basis by executing the workflows in a determined period.

  Some workflows use the scripts in the repository to access sheets to get the recent data and create new updated `latest.json` and `schema.json` files. After completing the creation of these files, they are uploaded to Amazon Web Services (AWS). These workflows can be managed and updated by the developer.

- Leaflet is used for the map of the afetbilgi.com. There is additional subdomain, whose link is maps.afetbilgi.com for the complete map.

- UserClient initiate the connection with the website. It has some attributes that are provided to website and functionalities to control the website.

4.1.4 Interaction Scenarios

Two different interaction scenarios are provided:

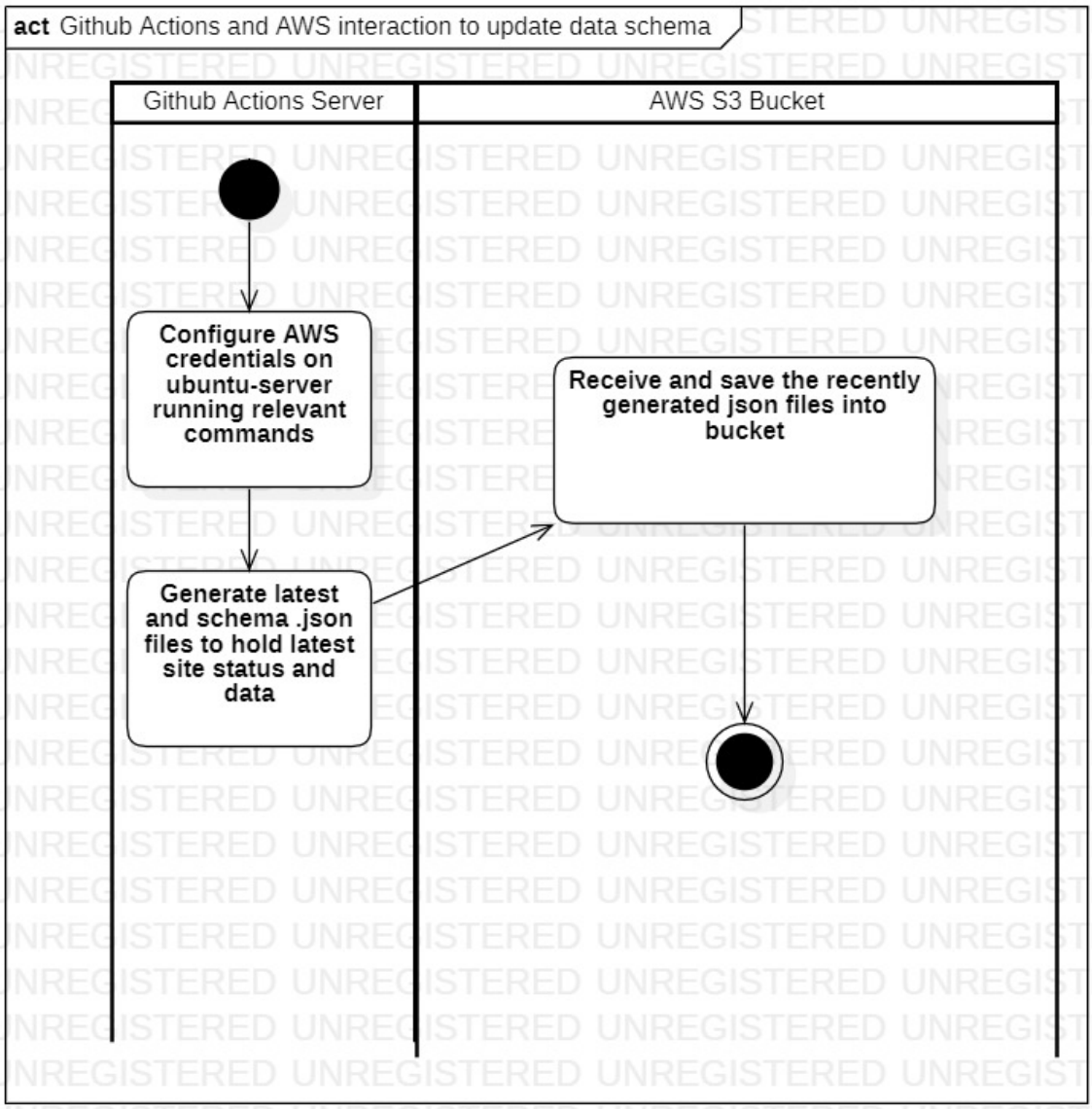

Figure 3: Activity Diagram — GitHub Actions and AWS Interaction to Update Data Schema

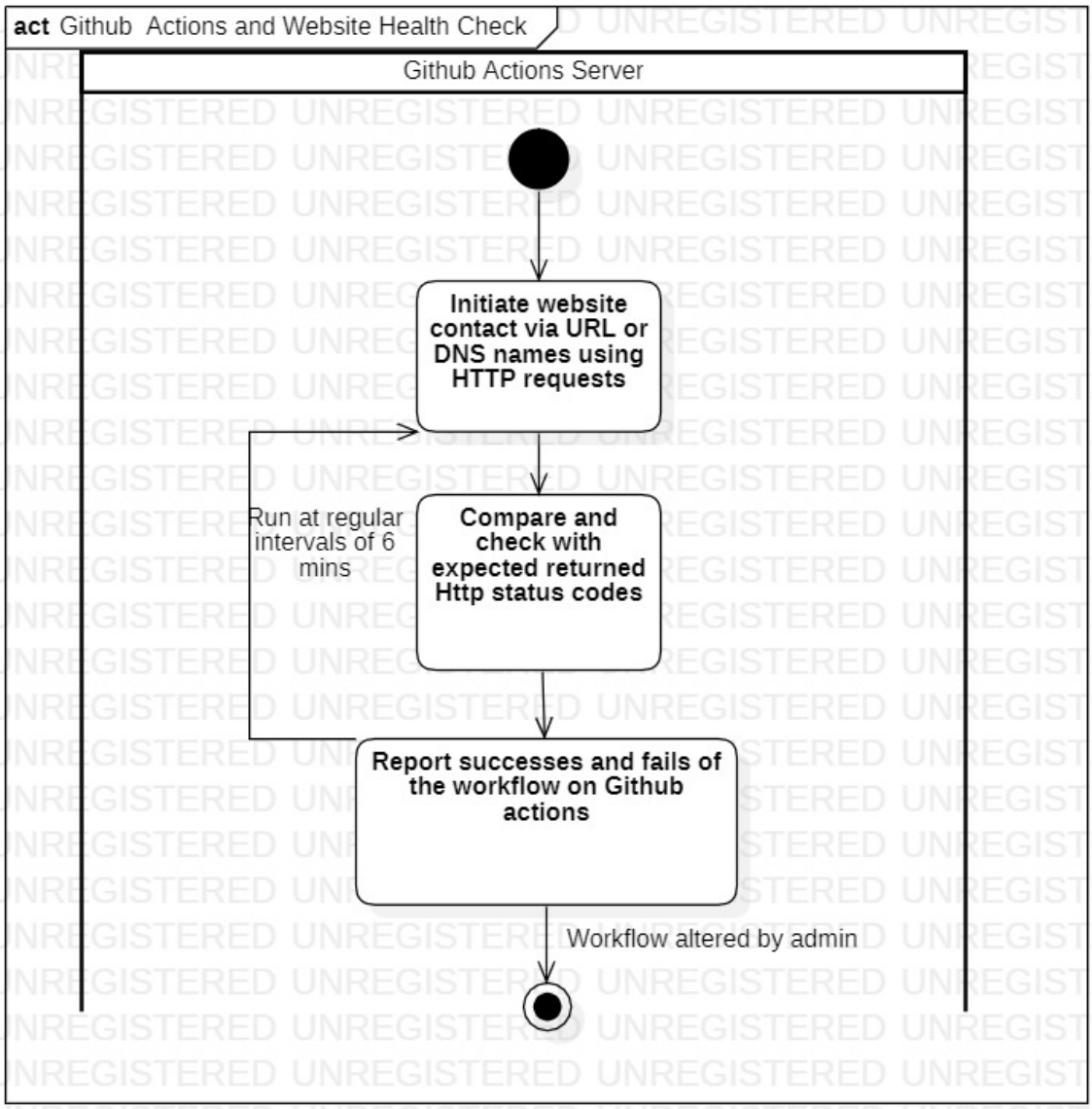

Figure 4: Activity Diagram — GitHub Actions and Website Health Check

4.2 Functional View

4.2.1 Stakeholders' Uses of This View

This view depicts the different components of the system and their interaction, including the internal interfaces. This is extremely important for the stakeholders as it gives information about the system's functionalities and the integral properties, it also gives way for other viewpoints and diagrams. As such victims and volunteers wont find this useful but website maintainers along with Data collectors would deeply appreciate this view to understand internal components and perhaps come up with ways to improve it.

4.2.2 Component Diagram

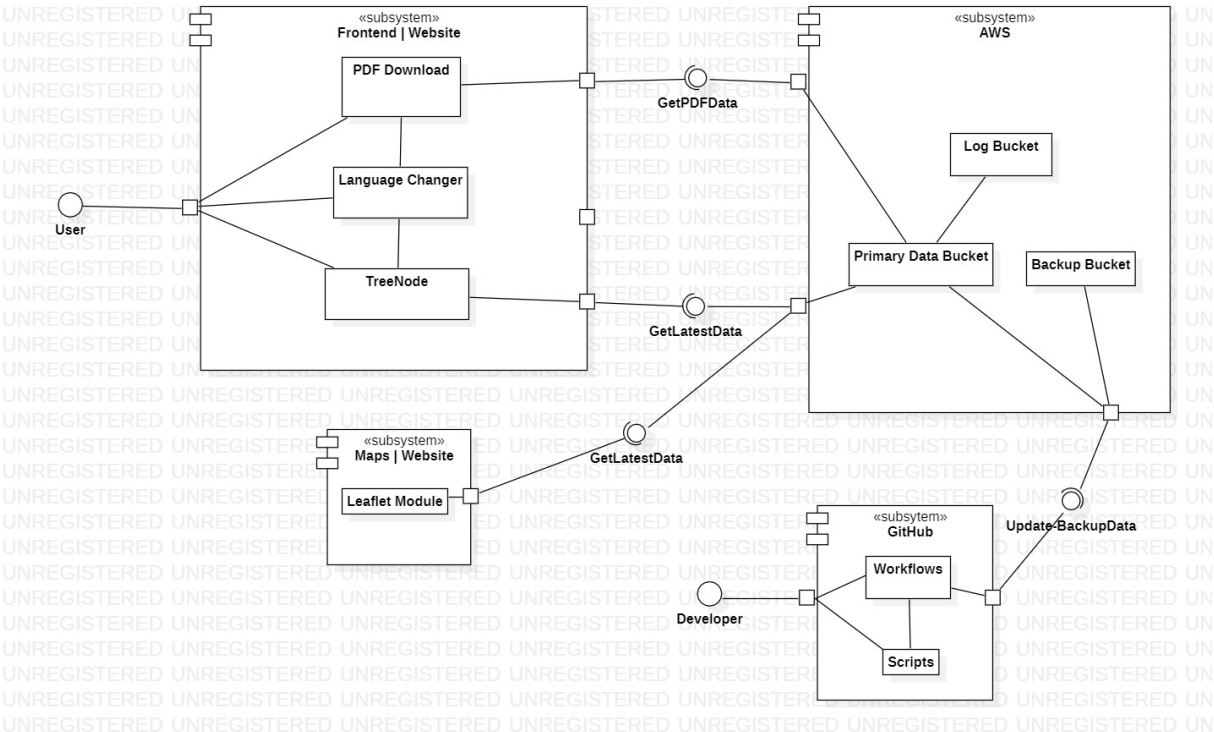

Figure 5: Component Diagram

Our system have four subsytems. Two of them subsystem is website subsytems.

- The website has two different subsystems which are the frontend, the main website, and the map, providing the map feature.
- The map subsystem use leaflet to show the data more effectively on the map.

- The frontend subsystem is divided into components to handle different request from the user.

  It includes PDF component to get the PDF version of the website to use and distribute the website in offline mode. The PDF files are generated by workflows and stored in AWS.

  This subsystem also has language changer to update the language setting of the website so that user can access the website in different language to use.

  TreeNode is the main display component. It parse the data after getting the latest available data, `latest.json`, from the AWS.

- AWS is mainly used to `latest.json` (data), `schema.json` (schema of the `latest.json`) and PDFs.

- Github subsystem includes scripts and workflows. Scripts are generally written in python and used by the workflows. Workflows are responsible for checking, maintaining and updating the website in regular basis.

4.2.3 Internal Interfaces

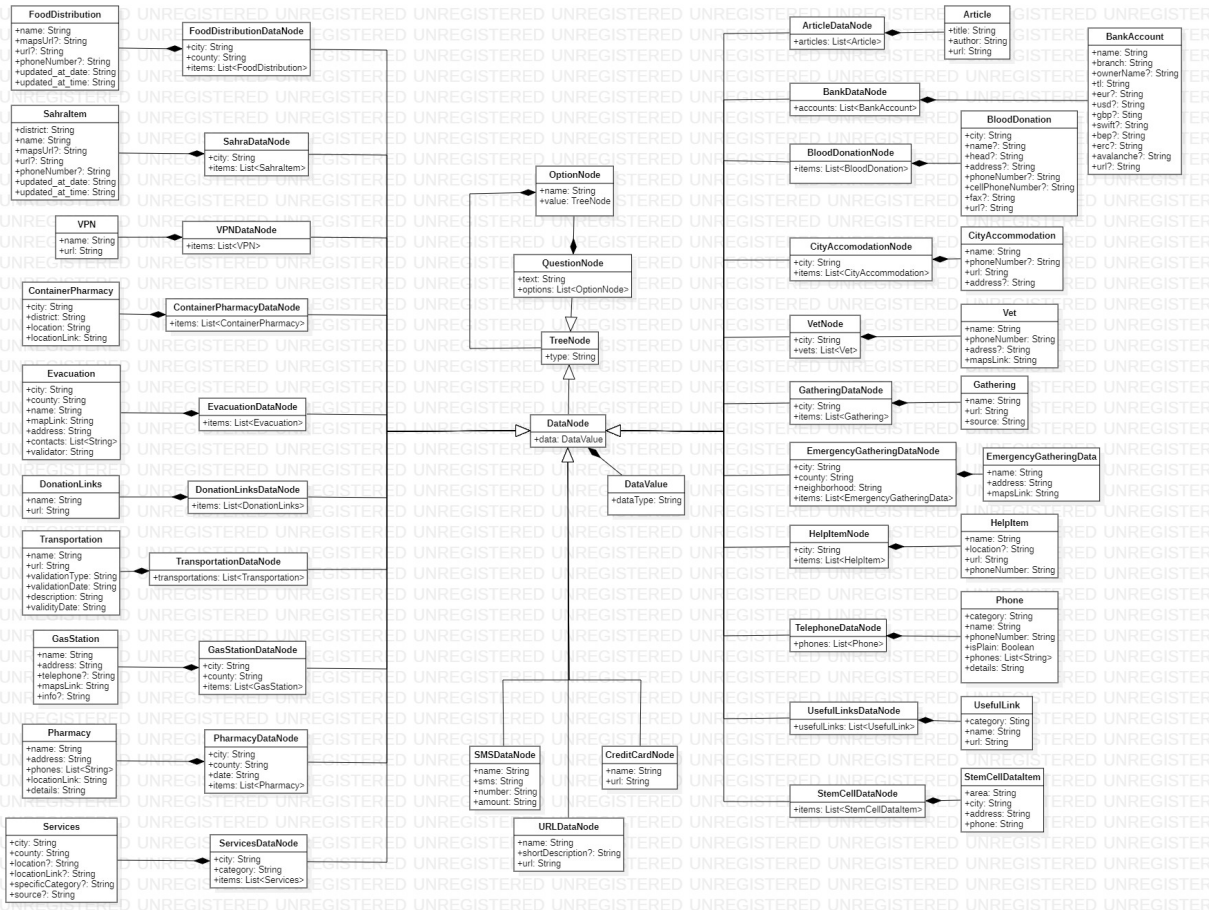

Figure 6: Internal Interfaces

There is no internal dynamism between interfaces. The internal interfaces are just the data interfaces to provide structured information for frontend code so that frontend code can parse the information correctly and fastly.

The main data is stored in the AWS in `latest.json` and retrieved from AWS so that the frontend code parses the data.

4.2.4 Interaction Patterns

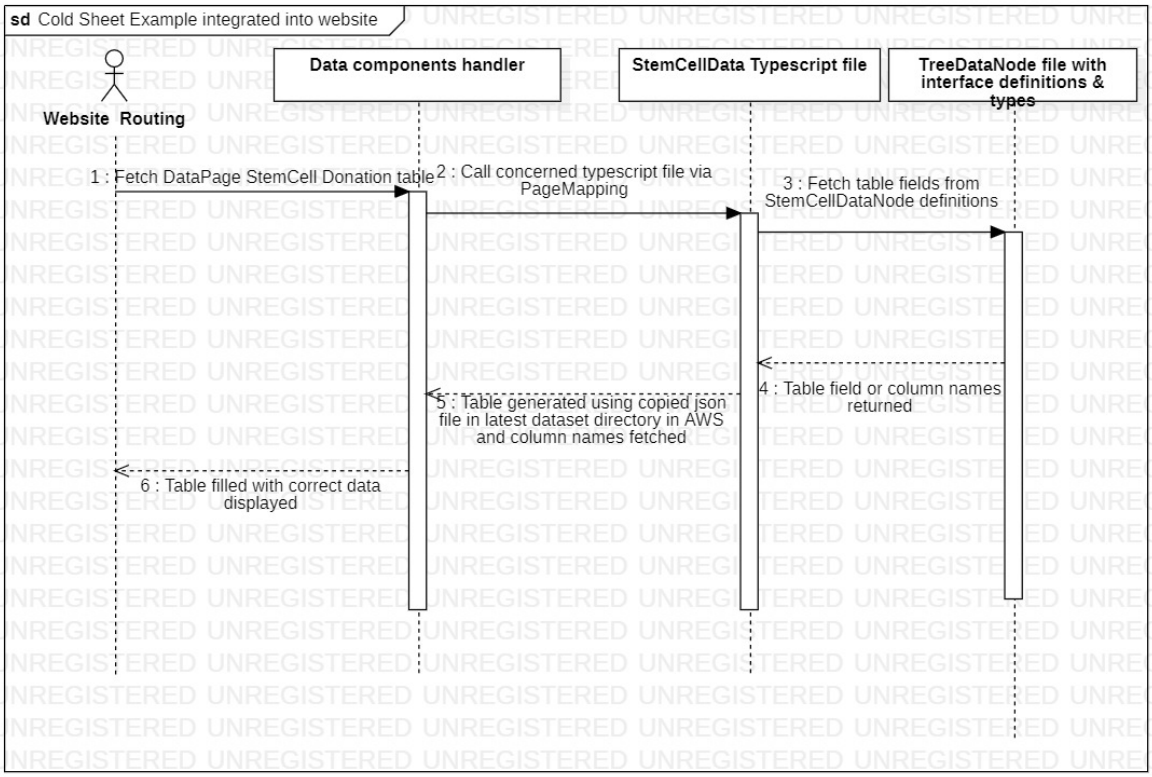

Figure 7: Sequence Diagram — Cold Sheet Example Integrated Into Website

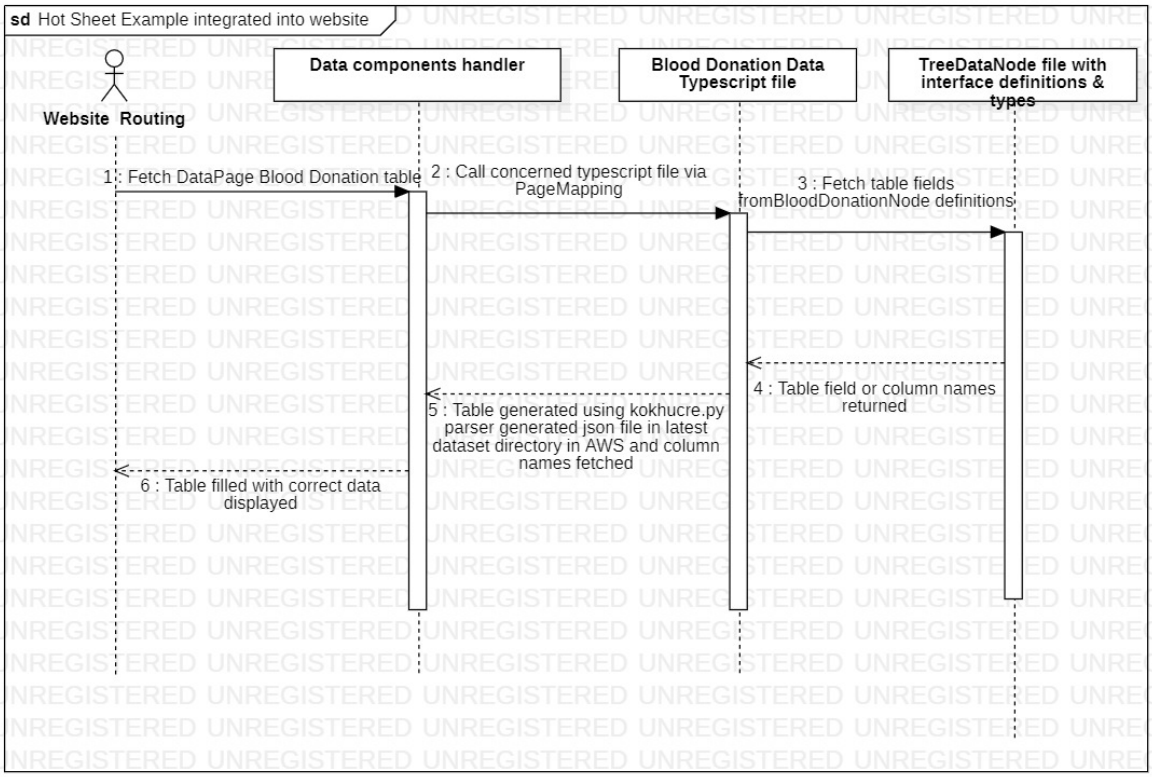

Figure 8: Sequence Diagram — Hot Sheet Example Integrated Into Website

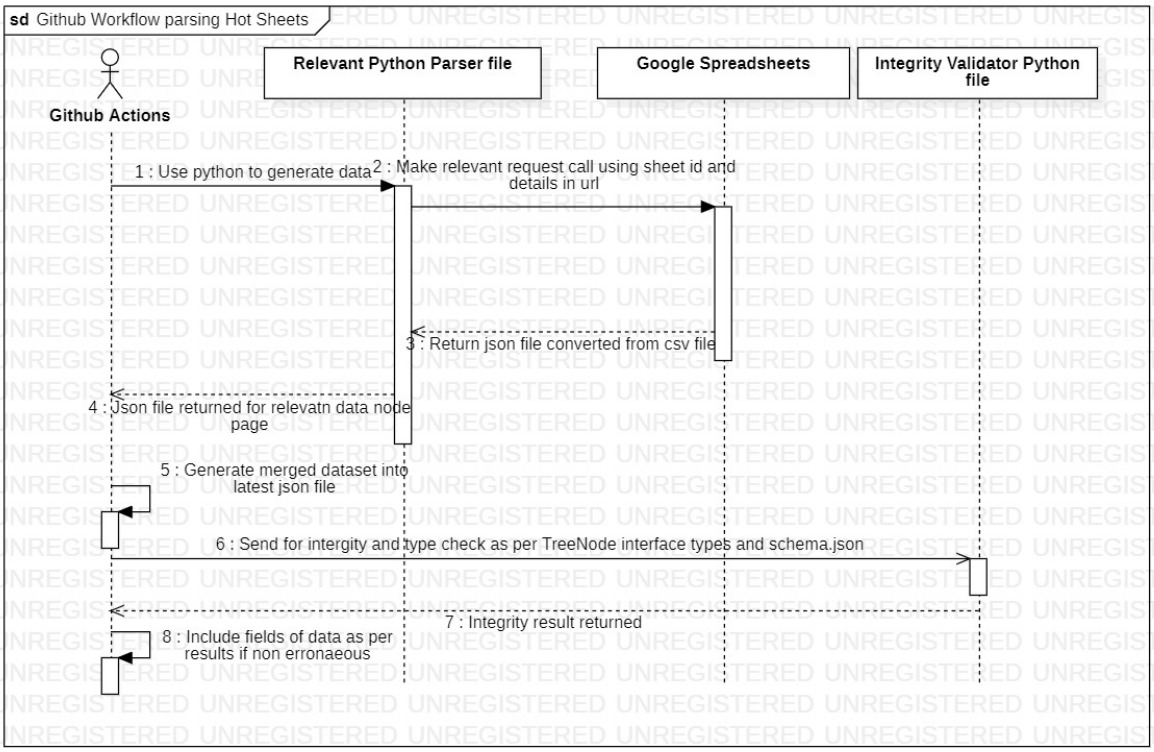

Figure 9: Sequence Diagram — GitHub Workflow Parsing Hot Sheets

4.3 Information View

4.3.1 Stakeholders' Uses of This View

This view concerns the variables and the data used and stored in this system during use as per Rozanski's explanations. While this open-source, static and nondynamic website doesn't employ a database infrastructure backend like mySQL, stakeholders such as website maintainers along with data maintainers/validators would again appreciate to know which data pages and nodes are being exchanged and whether any unreliable information is being or has been uploaded to the website assets and datasheets.

4.3.2 Database Class Diagram

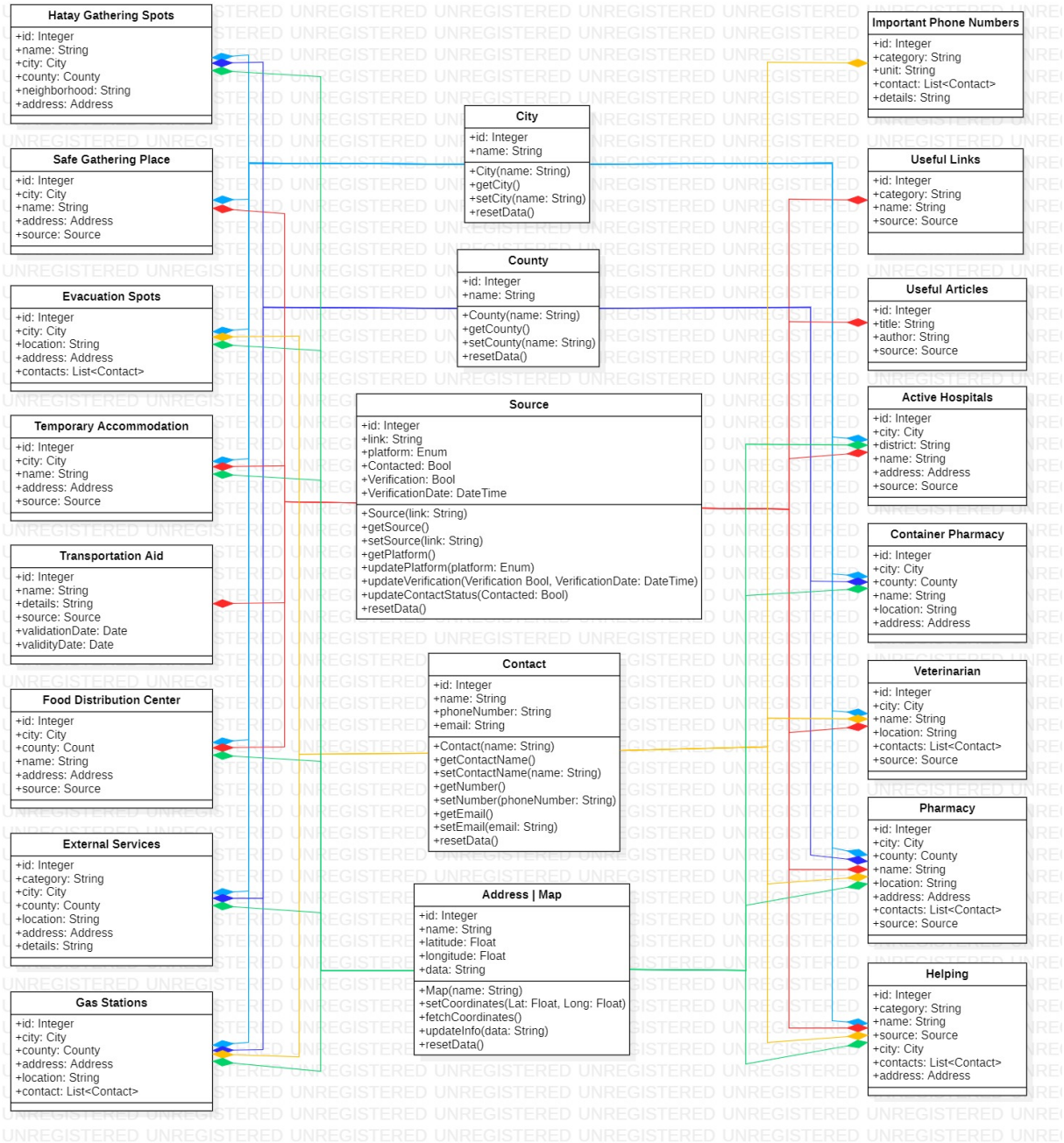

Figure 10: Database Class Diagram

4.3.3 Operations on Data

Operation	CRUD Operation
City :: City	Create: City Profile with name Read: Update: Delete:
City :: getCity()	Create: Read: City for selection Update: Delete:
City :: setCity()	Create: name for city name field Read: Update: Delete: previous city name
City :: resetData()	Create: Read: Update: restart with default empty/null values for fields Delete: existing field values
County :: County(name:String)	Create: County Profile with name Read: Update: Delete:
County :: getCounty()	Create: Read: County for selection Update: Delete:
County :: setCounty(name:String)	Create: county name Read:

	Update: Delete: previous county name field
County :: resetData()	Create: Read: Update: restart with default empty/null values for fields Delete: existing field values
Source :: Source(link:String)	Create: Source Profile with relevant link Read: Update: Delete:
Source :: getSource()	Create: Read: Source for selection Update: Delete:
Source :: setSource(link:String)	Create: source link updated Read: Update: Delete: previous source link
Source :: getPlatform()	Create: Read: Platform field(enumerated) for selection Update: Delete:
Source :: updatePlatform(platform:Enum)	Create: Read: Update: platform field(enumeration) Delete: previous platform field(enumeration)
Source :: updateVerification(Verification Bool,	Create: Read: Update: Verification Status and Time

VerificationDate:DateTime)	Delete: Previous Verification Status and Time
Source :: updateContactStatus(Contacted:Bool)	Create: Read: Update: Contacted status Delete: Previous Contacted status
Source :: resetData()	Create: Read: Update: restart with default empty/null values for fields Delete: existing field values
Contact :: Contact(name:String)	Create: Contact Profile with name Read: Update: Delete:
Contact :: getContactName()	Create: Read: Contact Name field for selection Update: Delete:
Contact :: setContactName(name:String)	Create: name for contact Read: Update: Delete: previous name for contact
Contact :: getNumber()	Create: Read: Contact Phone Number field for selection Update: Delete:
Contact :: setNumber(phoneNumber:String)	Create: phone number for contact Read: Update: Delete: previous phone number

Contact :: getEmail()	Create: Read: Contact Email field for selection Update: Delete:
Contact :: setEmail(email: String)	Create: email for contact Read: Update: previous email for contact Delete:
Contact :: resetData()	Create: Read: Update: restart with default empty/null values for fields Delete: existing field values
Map :: Map(name:String)	Create: Map Profile with name Read: Update: Delete:
Map :: setCoordinates(Lat:Float, Long:Float)	Create: longitude and latitude float values Read: Update: : longitude and latitude float values Delete:
Map :: fetchCoordinates()	Create: Read: Longitude and Latitude Coordinates for selection Update: Delete:
Map :: updateInfo(data:String)	Create: Read: Update: additional data is updated Delete: previous data is deleted

Map :: resetData()	Create: Read: Update: restart with default empty/null values for fields Delete: existing field values

Table 2: CRUD Operations on Data

4.4 Deployment View

4.4.1 Stakeholders' Uses of This View

Rozanski's document explained the deployment view as the environment in which the system in running(inclusive of hardware/external server elements). As such in this project's case, the website maintainers/admins would be the ones to appreciate a basic amount of knowledge from the deployment diagram below such as the basic relations such as how to run parser script files and set up hot/cold sheets which are regularly updated to latest.json via the AWS s3 bucket. On the other hand, future/long term site maintainers would be wanting to take the analysis of the deployment diagram even further to develop and focus on individual parts comprising the components such as improving coldsheets' static json files that can be secured perhaps a bit more before being merged to formed recent latest.json files.

4.4.2 Deployment Diagram

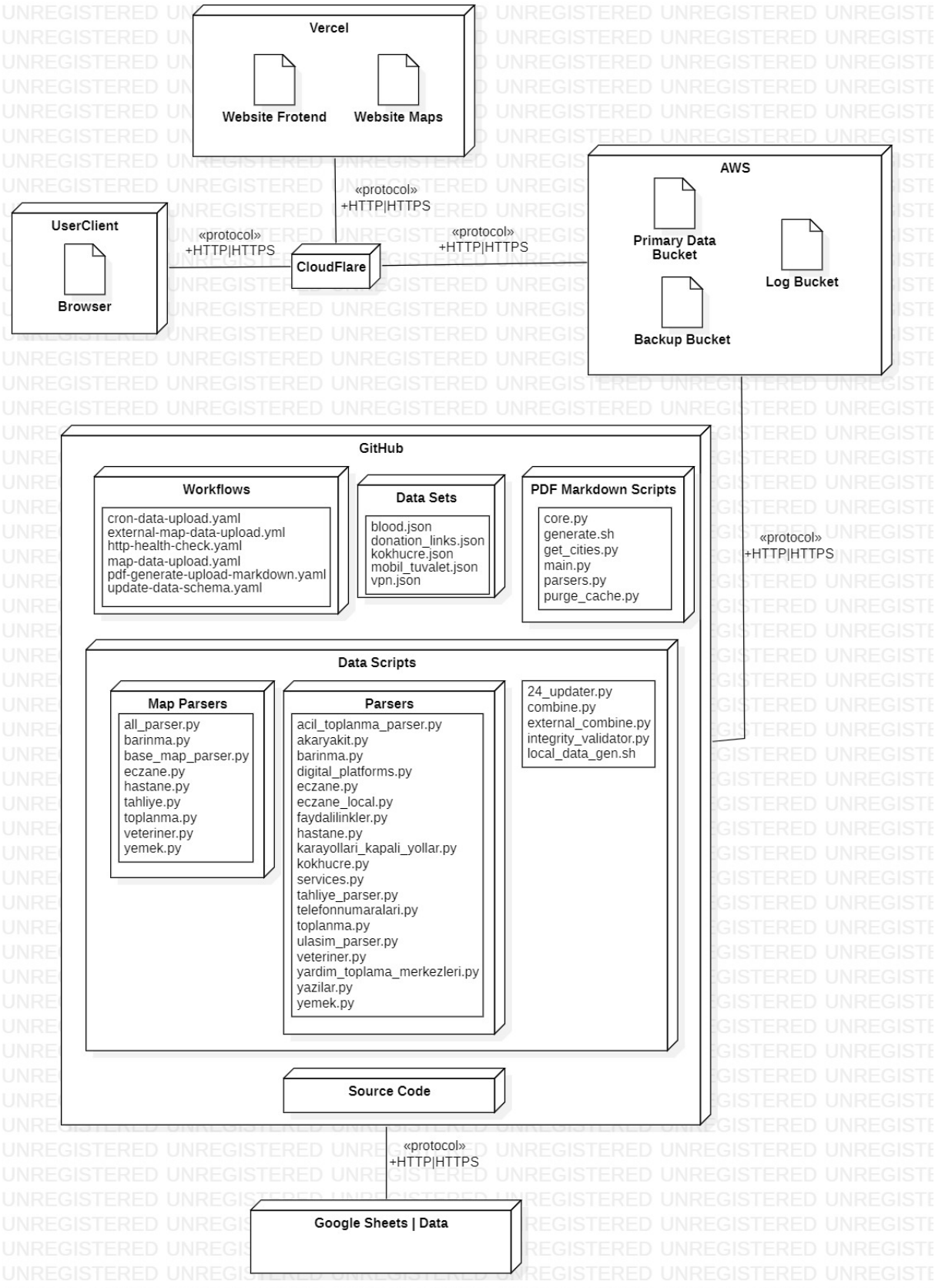

Figure 11: Deployment Diagram

In our deployment diagram viewpoint, we show the deployment environment of the afetbilgi.com.

- User should use a javascript supported browser to access the website.
- Requests are met by CloudFlare and CloudFlare maps the related request. CloudFlare also gives supports for general protection such as using HTTPS, TLS and DDoS protection as well as the general protection.
- Vercel contains the website with both frontend and maps.
- AWS includes the primary data which is `latest.json` and other data. It also stores the bakcups and logs.
- GitHub provides source code and workflows. Workflow of GitHub checks, maintains and updates the data and the website. These workflows uses scripts to get the latest and stored data data and combine them to upload to AWS. Also, PDFs are generated and upload to AWS.
- Google Sheets store the data updated by data collectors and validators.

4.5 Design Rationale

The whole afetbilgi.com website system is based on the premise and primary rationale that it's an open source, verified non-profit website, which aims to convey quick information to victims and volunteers alike without any user registration/related overhead requiring unnecessary time expenditure. It's based on a website with plain buttons and hierarchical routing leading to relevant datasheets as per need. All the views consisting of context, functional, information and deployment build on this.

In the context view, the basic setup is explained in the context diagram showing how maintainers have set up the website of afetbilgi.com to be used by victims and volunteers alike. The external interface diagram further improves on this showing how the data validators used external services like Google Datasheets to fetch recent data, as well as AWS to use assimilated data from latest.json. On the other hand site maintainers and

developers have also used Github Actions' CI/CD workflows to continuously update the website and check its data health regularly from the AWS S3 bucket where it's hosted and its DNS assignments are covered by Cloudflare.

In the functional view, we see the internal interfaces interact in a modern modular setup(as per object programming's interface and extension principles). On the computation component sides, it's explained how the website health is checked regularly using the workflow actions from Github on ubuntu environments with backups of data and the last health website saved to buckets in AWS, while on the internal interface we learn of modularized entities like TreeNode interfaces which strictly check how new data pages are added and how their typing match to the node description as per their specification. As the website is static and every such Data node(each having own webpage on the afetbilgi.com website) doesn't really interact with any other interface yet the sequence diagrams great demonstrate how new data(hot/cold) is added and conformed to the node specifications while the Workflows assimilate and eventually upload then to AWS as latest.json files(types checked and saved via schema.json too).

Information View reinforces the stand alone concept of this project with no backend databases and no area to upload any private information such user info/credit card detail,etc . The site data validators and maintainers are the main stakeholders to appreciate this section, knowing which variables transmit and exchange within which interfaces as shown on the Database Class diagram. There are no explicit SQL classes or tables yet there is some sort of validation/role allotment needed for a user to be allowed to update data on the website. All the tables/datasheets have common elements of City, Source, Location and Map components present. The CRUD operations' table helps in having a basic summary of the presented data-altering options from the Database Class Diagram without too much technicalities.

Lastly, the Deployment View again explains the project repository and the modular nature of this project, with the files clearly set out. Over here, we see the dependent nature of this project with other external entities to upload data to or interact with such as the AWS bucket components and exchanging latest.json files with them along with type conformations in schema.json. Parser python files are run via external Github actions' ubuntu server to run and assimilate hot data sheets after fetching from the relevant Google Datasheets while cold sheets(manually added data) are simply copied and merged.

5 Architectural Views for Suggestions to Improve the Existing System

5.1 Context View

5.1.1 Stakeholders' Uses of This View

In this view, system context is provided, with the actors and other systems related to it in a general manner. Website maintainers and Data collections of afetbilgi.com will be the ones that primarily use this view to better understand how the different actors interact in the working of this website and the relationship that the system has with other external entities like the AWS components, Maps API component, and so on. In addition, Authentication has been added and can use Google API as a reliable 3rd party user registration handling as well along with an embedded Mail subscription handling module to inform users of potential alerts given the ongoing emergency situation about the earthquake. Similar interfaces are further explained in our added components in the external diagram.

5.1.2 Context Diagram

afetbilgi.com is not part of a more extensive system. It is a standalone and open-source efforted website to verify critical information in the fight against the 6 February 2023 Pazarcik Earthquake and deliver it to disaster victims and those who want to help in an understandable, concise manner in multiple languages.

This information is presented in either the form of legible tables with third-party governmental and private links or an interactable method via a map view interface. If deemed necessary, admin and maintainers can make changes to display newly created or edited data and upload it to the system upon any complaints or suggestions they may get on their contact details.

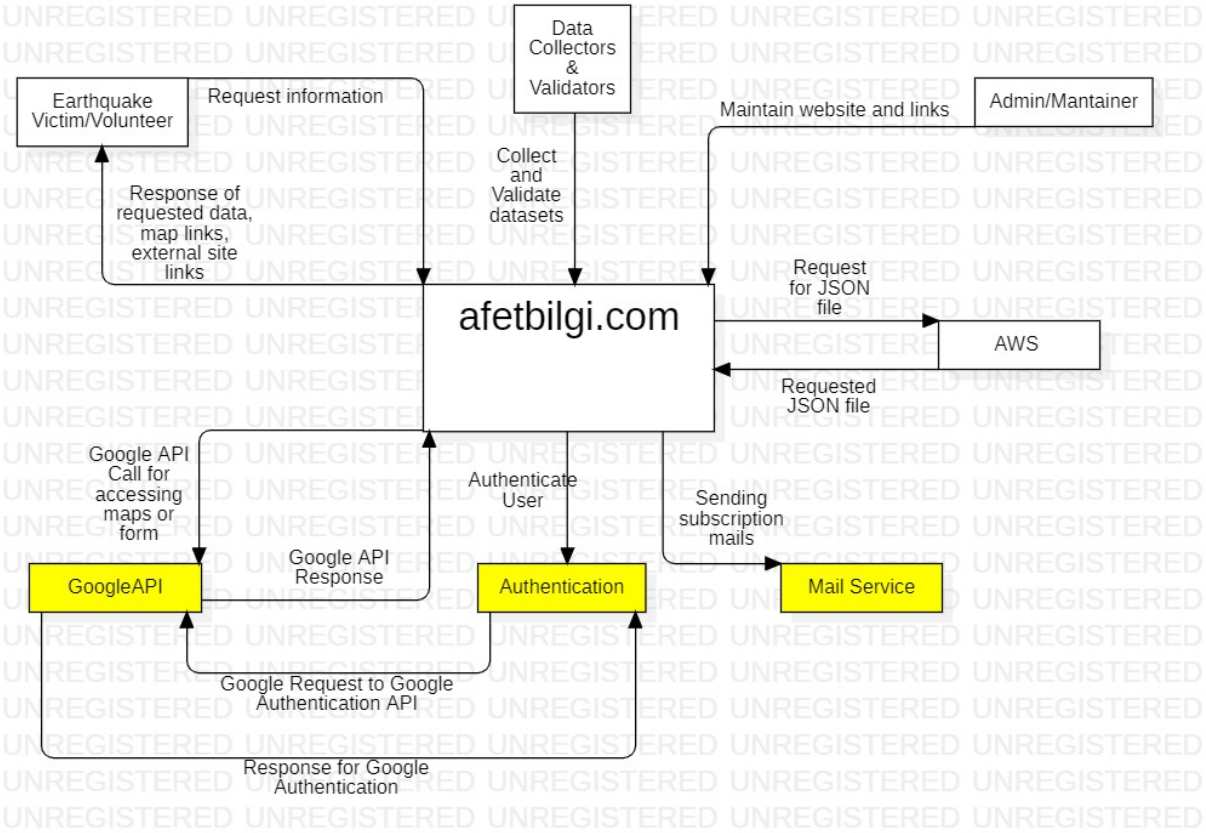

Figure 12: Suggested Context Diagram

The afetbilgi.com consists of a combination of small physical and software parts. With the help of interfaces, these parts communicate among themselves and with the user.

5.1.3 External Interfaces

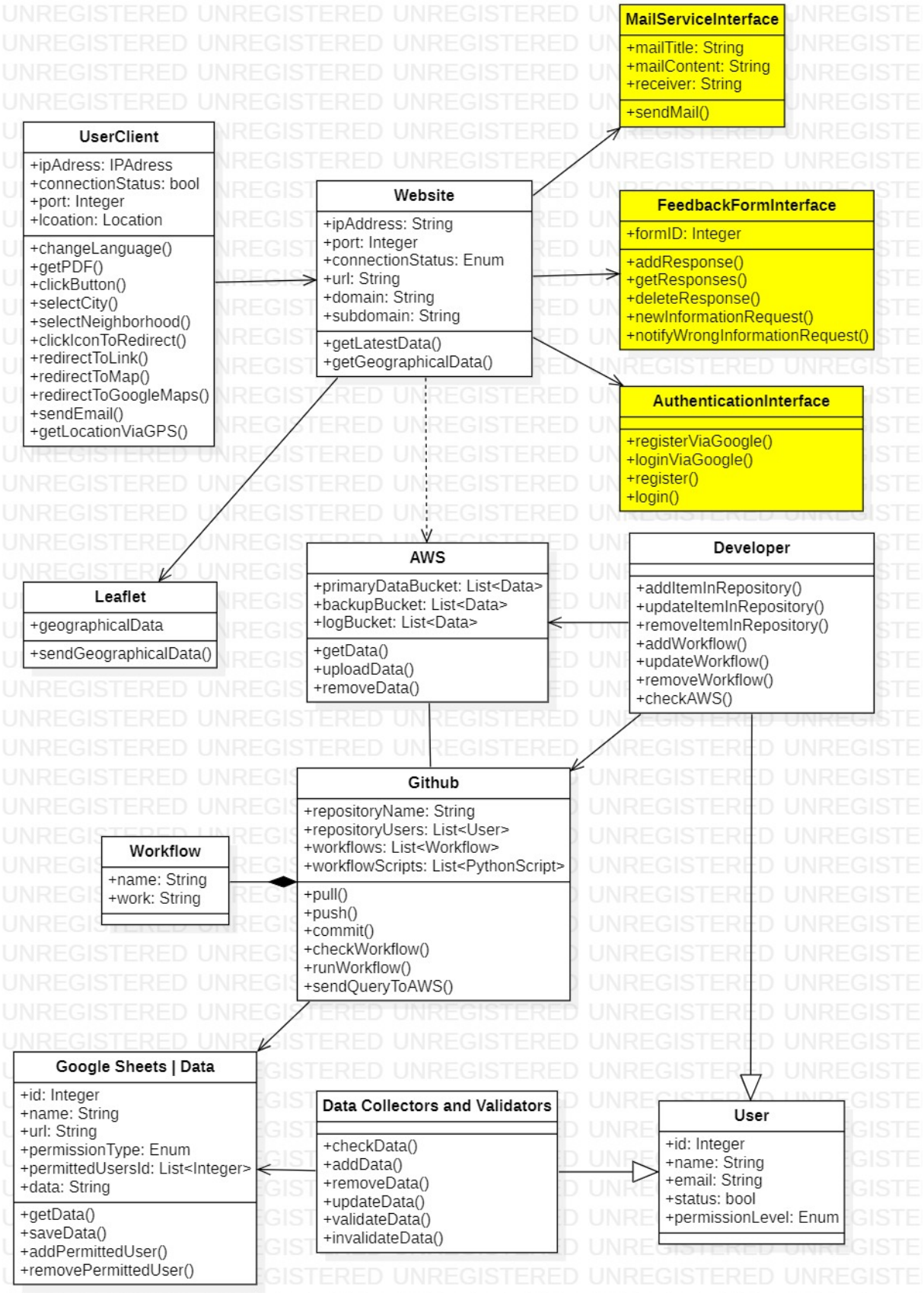

Figure 13: Suggested External Interfaces Diagram

As it can be observed from Figure 13, afetbilgi.com has multiple external interfaces. We have added MailServiceInterface, FeedbackFormInterface and AuthenticationInterface. The operations given in the diagram can be summarized as follows:

- MailServiceInterface provides the option of sending mail about the updated information to the subscribed users.
- FeedbackFormInterface provides the option of sending feedback related to website and the data. Users from the earthquake region can request for adding new information and notifying the wrong and expired infromation.
- AuthenticationInterface provides the option of logging in. The authentication system is used to provide opportunity to authorized institutions and organizations to update the information from the website directly.

5.1.4 Interaction Scenarios

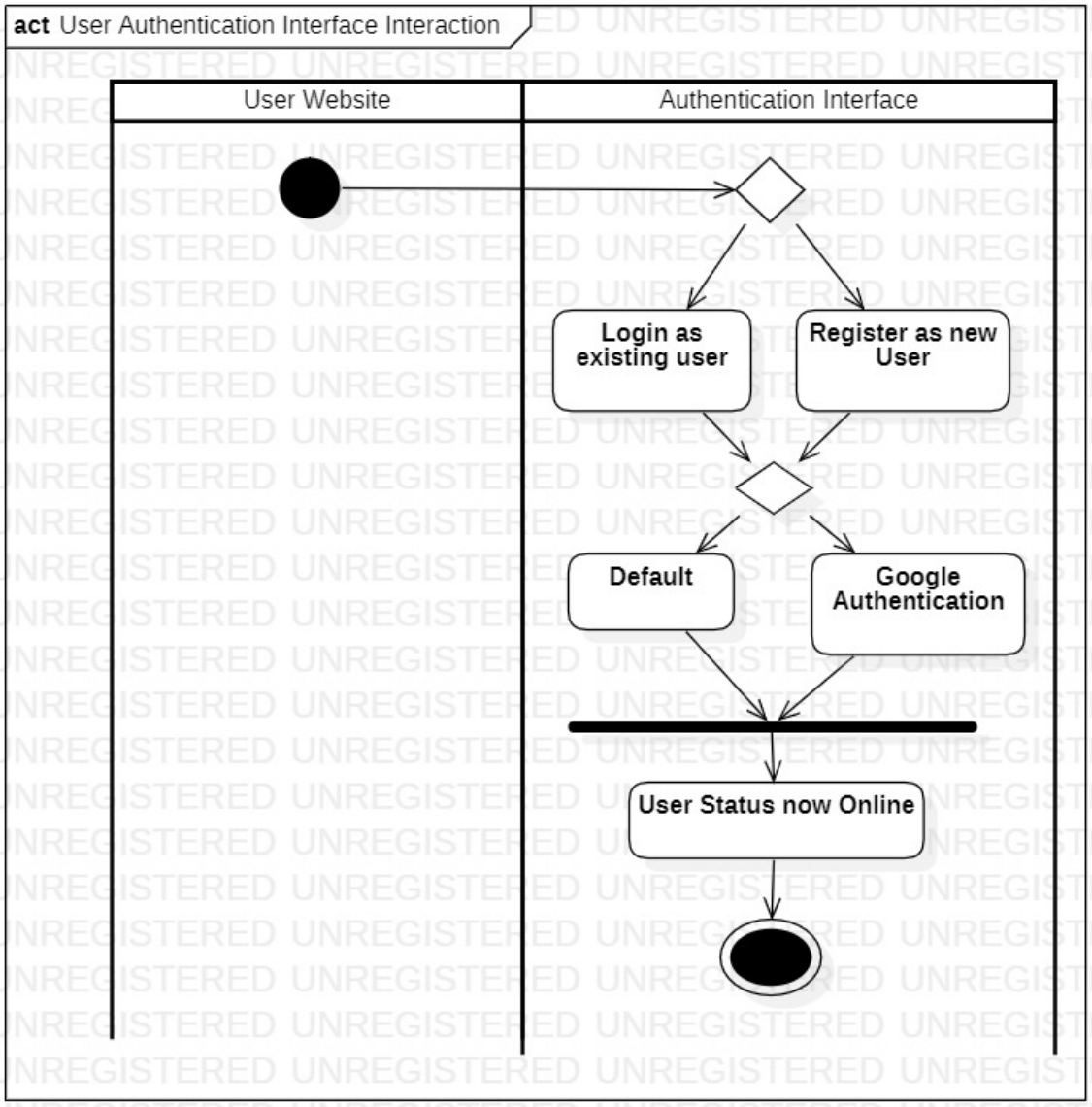

Figure 14: Activity Diagram — User Authentication Interface Interaction

5.2 Functional View

5.2.1 Stakeholders' Uses of This View

This view depicts the different components of the system and their interaction, including the internal interfaces. This is extremely important for the stakeholders as it gives information about the system's functionalities and the integral properties, it also gives

way for other viewpoints and diagrams. As such victims and volunteers wont find this useful but website maintainers along with Data collectors would deeply appreciate this view to understand internal components and perhaps come up with ways to improve it. Additional components such as Google authentication would be handled by external API request to handle user registration as well another internal component to handle mail subscription to registered users. Similar new internal interfaces handle these arrears and interact with these additional external components.

5.2.2 Component Diagram

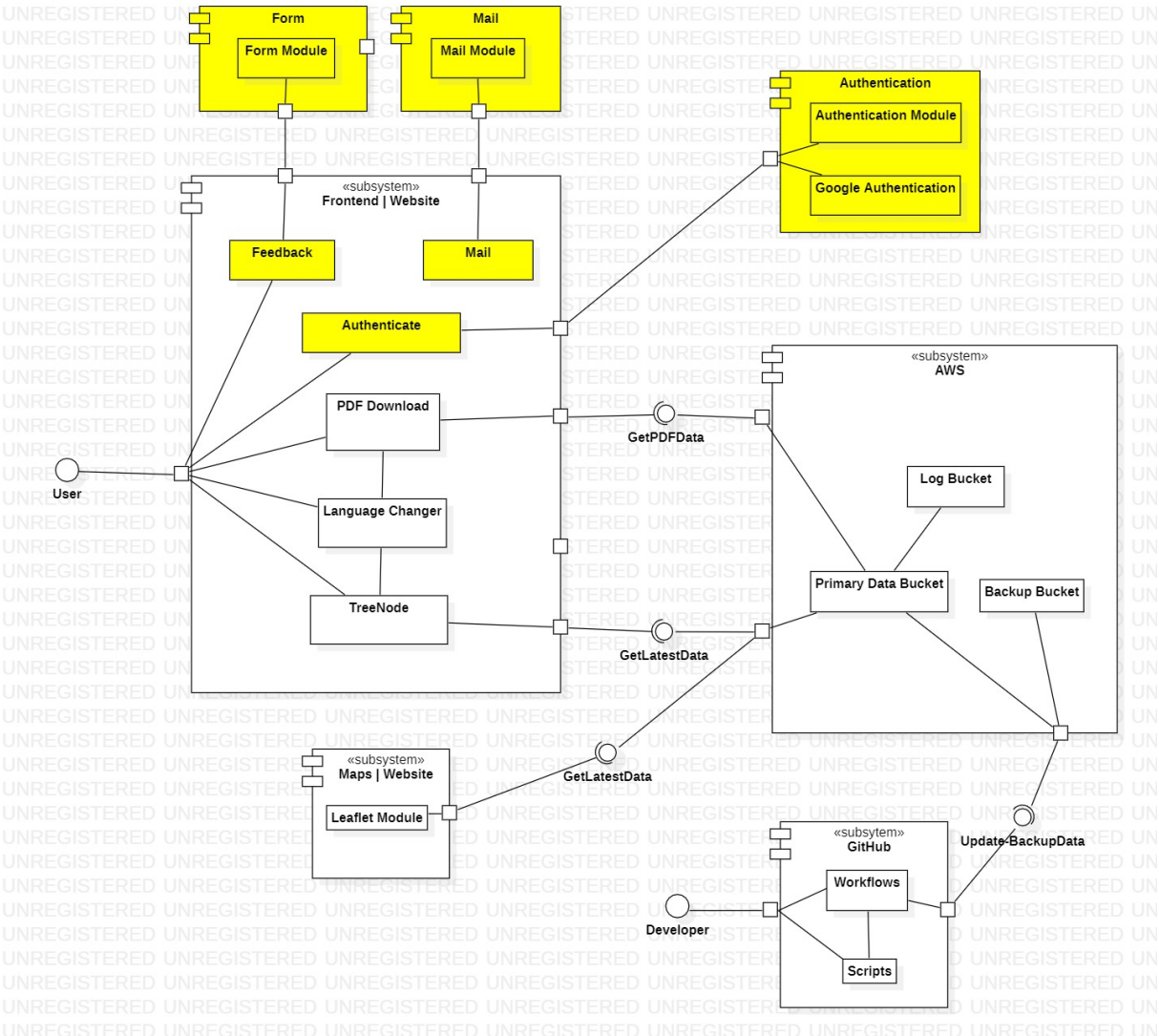

Figure 15: Suggested Component Diagram

We have updated component diagram by adding the components for form, mail and authentication components:

- Form provides user to send feedback about website and data so that the website and data can be updated according to the feedback by the normal users and victims.
- Mail component sends mail about the updated information to the subscribed users.
- Authentication component provide authorized users to register and login to the website by using Google Authentication system or built-in Authentication Module.

5.2.3 Internal Interfaces

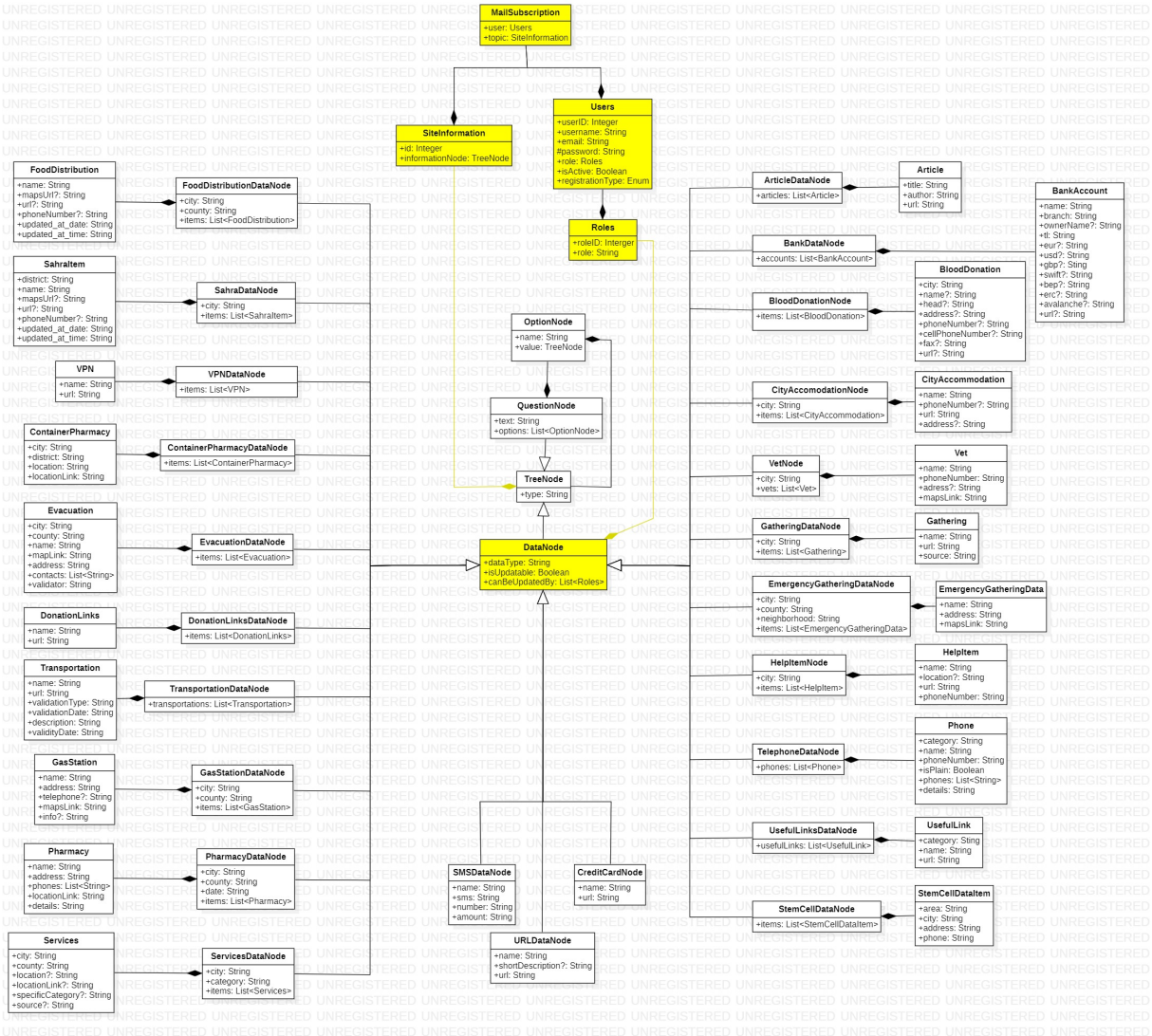

Figure 16: Suggested Internal Interfaces

Since, there is no internal dynamism between interfaces initially, we embraced the same approches. Therefore, the internal interfaces are just the data interfaces to provide structured information for frontend code so that frontend code can parse the information correctly and fastly.

The main data is stored in the database and retrieved from database, so the frontend code parses the data.

5.2.4 Interaction Patterns

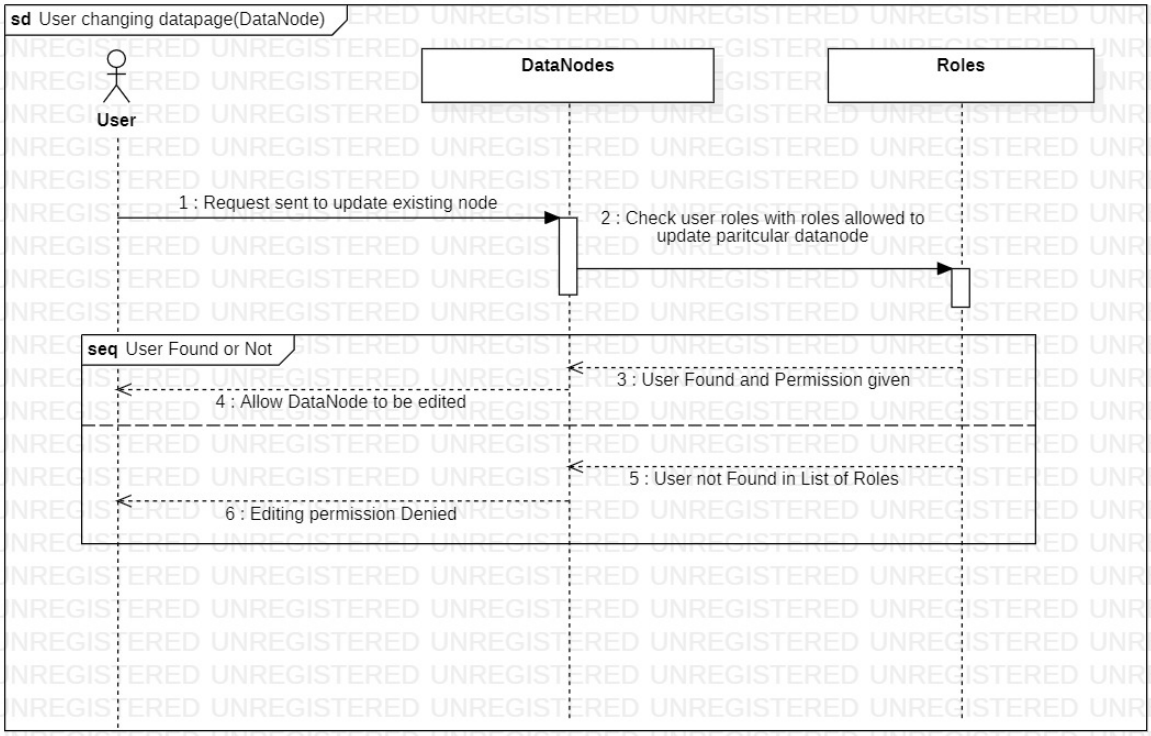

Figure 17: Sequence Diagram — User Changing Datapage

5.3 Information View

5.3.1 Stakeholders' Uses of This View

This view concerns the variables and the data used and stored in this system during use as per Rozanski's explanations. While this open-source, static and nondynamic website doesn't employ a database infrastructure backend like mySQL, stakeholders such as website maintainers along with data maintainers/validators would again appreciate to know which data pages and nodes are being exchanged and whether any unreliable information is being or has been uploaded to the website assets and datasheets. In additional an essential user database class has been added with additional mail subscription classes to formalize and secure the website even further. As per allotted roles the user would have the autheorization to make changes onto the website.

5.3.2 Database Class Diagram

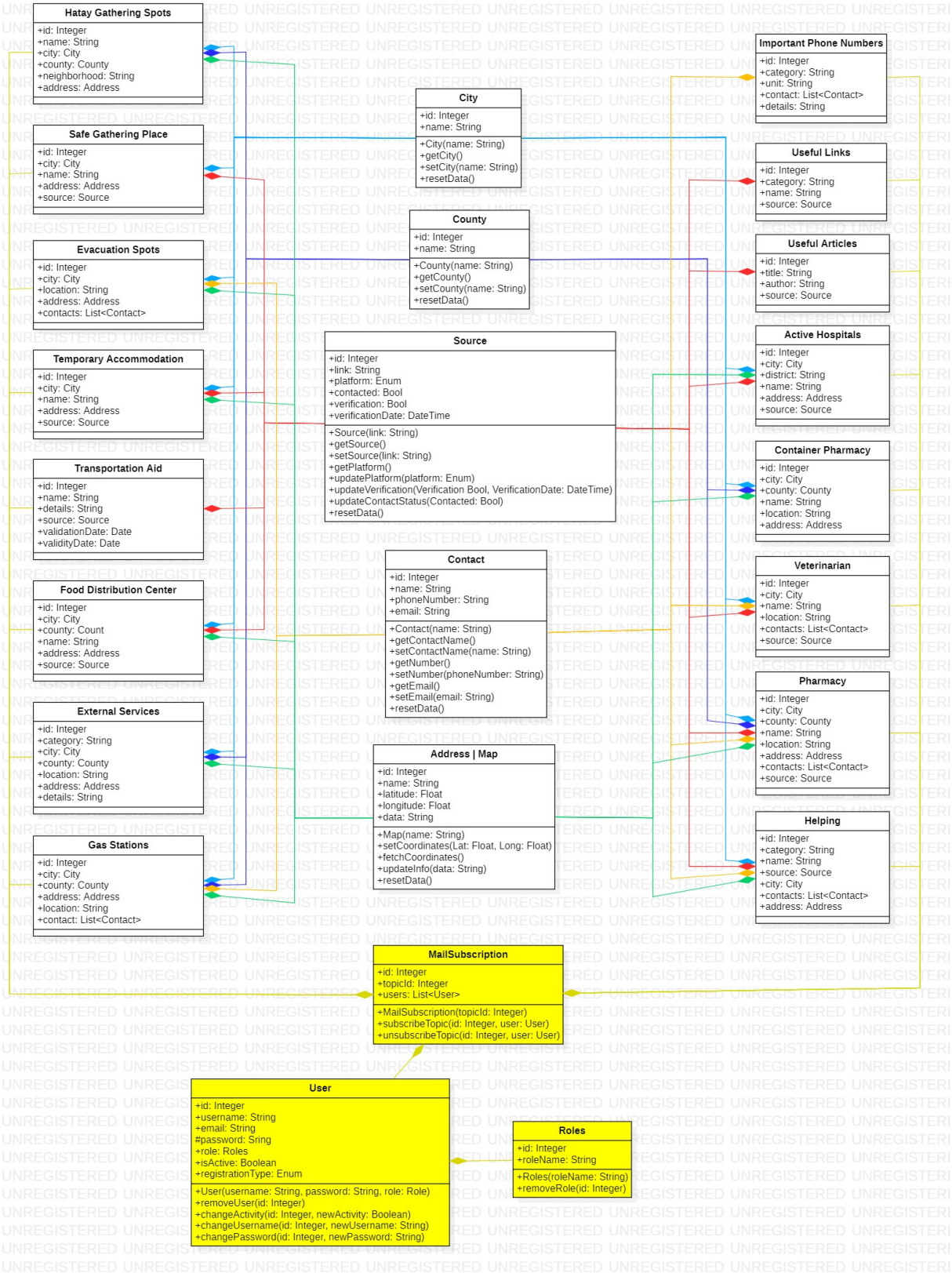

Figure 18: Suggested Database Class Diagram

5.3.3 Operations on Data

New Operations	CRUD Operations
MailSubscription :: MailSubscription()	Create: Create new MailSubscription item with given topicId Read: Update: Delete:
MailSubscription :: subscribeTopic()	Create: Read: Update: Add user to the users list of the subcription item Delete:
MailSubscription :: unsubscribeTopic()	Create: Read: Update: Remove user from the users list of the subcription item Delete:
User :: User()	Create: Create new user with given arguments Read: Update: Delete
User :: removeUser()	Create: Read: Update: Delete: Delete user with given id
User :: changeActivity()	Create: Read: Update: Update the activity status of the user Delete:
User :: changeUsername()	Create:

	Read: Update: Update the username of the user Delete:
User :: changePassword()	Create: Read: Update: Update the password of the user Delete:
Roles :: Roles()	Create: Create new role with the given name Read: Update: Delete:
Roles :: removeRole()	Create: Read: Update: Delete: Remove the role with given integer

Table 3: CRUD Operations on Data

5.4 Deployment View

5.4.1 Stakeholders' Uses of This View

Rozanski's document explained the deployment view as the environment in which the system in running(inclusive of hardware/external server elements). As such in this project's case, the website maintainers/admins would be the ones to appreciate a basic amount of knowledge from the deployment diagram below such as the basic relations such as how to run parser script files and set up hot/cold sheets which are regularly updated to latest.json via the AWS s3 bucket.

On the other hand, future/long term site maintainers would be wanting to take the analysis of the deployment diagram even further to develop and focus on individual parts comprising the components such as improving coldsheets' static json files that can be secured perhaps a bit more before being merged to formed recent latest.json files. In addition, our static website will now have some dynamism such as a new interface(with a relevant js file in the react project) to generate forms for users to enter feedback after they have logged in/registered via the proper authentication webpage(also new).

5.4.2 Deployment Diagram

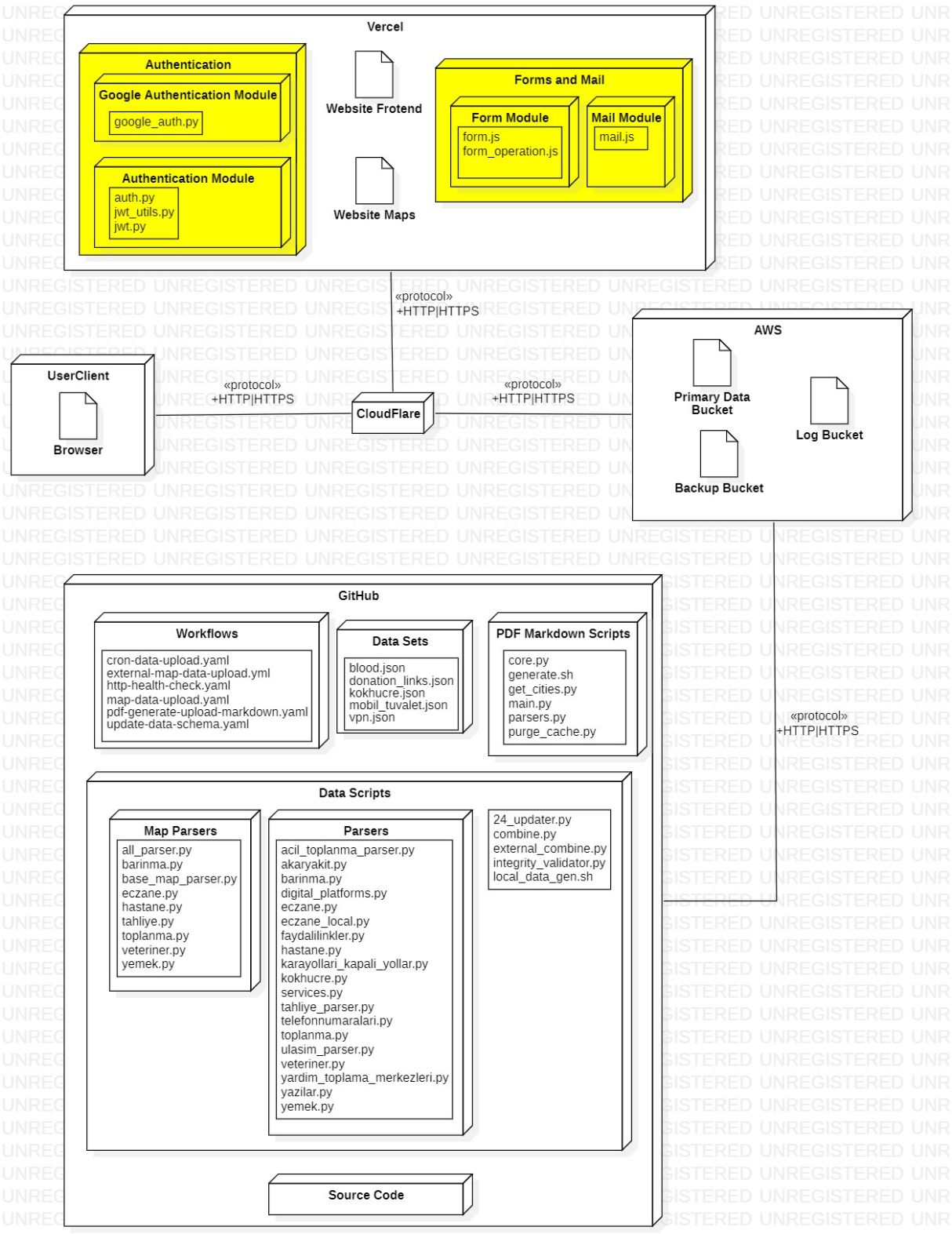

Figure 19: Suggested Deployment Diagram

- Authentication provides authorized users to register and login. It uses two systems which are Google Authentication and built-in Authentication System.

  Google Authentication access the Google API for authentication. Built-in authentication system use stored database information with JSON Web Tokens (JWT) for token based authentication system.

- Form provides user to send feedback about website and data. Forms to provide feedbacks are defined according to needs. Some operations such as uploading image as well as the basic form operations such as submitting.

- Mail modeule sends mails if there is an update in the website.

5.5 Design Rationale

The entire afetbilgi.com website system is based on the fundamental principle and primary objective of being an open-source, verified non-profit website. Its purpose is to provide quick information to victims and volunteers without requiring user registration or unnecessary time expenditure. The website is designed with simple buttons and hierarchical routing to relevant datasheets based on user needs. All the different views, including the context, functional, information, and deployment views, are built upon this foundation. In summary, we have proposed enhancements to increase the website's security and dynamism.

In the context view, the diagram explains the basic setup of the afetbilgi.com website for use by victims and volunteers. The external interface diagram demonstrates how data validators utilize external services like Google Datasheets to retrieve recent data, and AWS to incorporate data from latest.json. Site maintainers and developers also employ Github Actions' CI/CD workflows to continuously update the website and regularly check its data integrity from the AWS S3 bucket where it is hosted. DNS assignments are managed by Cloudflare. Additionally, we have added authentication using Google API as a reliable third-party user registration mechanism. A module for handling mail subscriptions has also been embedded to inform users about potential alerts related to ongoing earthquake

emergencies. Similar interfaces are further elaborated in the added components of the external diagram.

In the functional view, we observe the interaction of internal interfaces within a modern modular setup based on object programming principles of interfaces and extensions. The computation components involve regular website health checks using Github workflow actions on Ubuntu environments, with data backups and the last recorded website health saved to AWS buckets. On the internal interface side, modularized entities like TreeNode interfaces ensure proper addition of new data pages and validate their typing according to node descriptions specified. Although the website is static and each data node (having its own webpage on afetbilgi.com) does not directly interact with other interfaces, the sequence diagrams effectively demonstrate the process of adding and conforming new data (hot/cold) to node specifications, with workflows assimilating and eventually uploading them to AWS as latest.json files (type-checked and saved via schema.json). Additional components such as Google authentication are handled through external API requests for user registration, and there are internal components for managing mail subscriptions to registered users. Similar new internal interfaces handle these aspects and interact with the additional external components.

The Information View reinforces the project's concept of being stand-alone, with no backend databases or areas for uploading private information such as user details or credit card information. The main stakeholders who appreciate this section are the site data validators and maintainers. The Database Class diagram illustrates the transmission and exchange of variables within interfaces. Although there are no explicit SQL classes or tables, some form of validation and role assignment is required for users to be allowed to update data on the website. All tables and datasheets have common elements like City, Source, Location, and Map components. The CRUD operations table provides a basic summary of the available data-altering options based on the Database Class Diagram, without delving into excessive technical details. Furthermore, an essential user database class has been added, along with additional mail subscription classes, to further formalize and secure the website. Based on the assigned roles, users have authorization to make

changes to the website.

Lastly, the Deployment View explains the project repository and its modular nature, with clearly outlined files. This view highlights the project's dependencies on other external entities for data upload and interaction, such as AWS bucket components and exchanging latest.json files with them, along with type conformations defined in schema.json. Parser Python files are executed through external Github Actions' Ubuntu server to fetch and assimilate hot data sheets from relevant Google Datasheets, while cold sheets (manually added data) are simply copied and merged. Finally, the static website will now have some level of dynamism such as a new interface(with a relevant js file in the react project) to generate forms for users to enter feedback after they have logged in/registered via the proper authentication webpage(also new).