SOFTWARE REQUIREMENTS SPECIFICATION

---

afetbilgi.com

GROUP 62

Denizcan Yılmaz 2444172
Ozan Cinci 2448223

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Purpose of the Software

"afetbilgi.com" is a website created by ODTÜ students and graduates with the goal of delivering accurate and verified information to aid disaster victims and those who want to assist in dealing with the Pazarcık Earthquake that occurred on February 6, 2023. It operates by utilizing a database that is entirely manually created by volunteers, and its aim is to provide prompt and useful information to people. Creators urge anyone who contacts them to provide only verified information to prevent the spread of misinformation and ensure that only accurate information is shared.

1.2 Scope

- System will provide its users a user-friendly interface which is easy to navigate through different categories of what they are searching for
- System will provide its users verified and up-to-date information which is contributed by volunteers
- System will let volunteers to contribute verified information for data set
- System will provide its users to search functionality to quickly find information
- System will contain contact forms for emergency requests or general inquiries
- System will be mobile responsive to provide access from mobile devices
- System will have accessibility features for people with disabilities, which they can use

1.3 Product Overview

1.3.1 Product Perspective

afetbilgi.com is not a part of a large system, but it utilizes some services of Amazon Web Services such as S3 bucket, CloudWatch, Athena, VPC. It does not have a database; rather it uses S3 which is an object storage system as a data warehouse. Volunteers shall contribute to it via sharing verified information about categories like general need, important resources, health services etc. Users shall view the website in four languages such as Turkish, English, Kurdish and Arabic.

The interface that users benefit from surfing the website is powered by ReactJS. All users shall download the related data with filtering as pdf into their devices. The system uses static websites to reduce the latency as much as possible in case of emergency and unstable internet connection access.

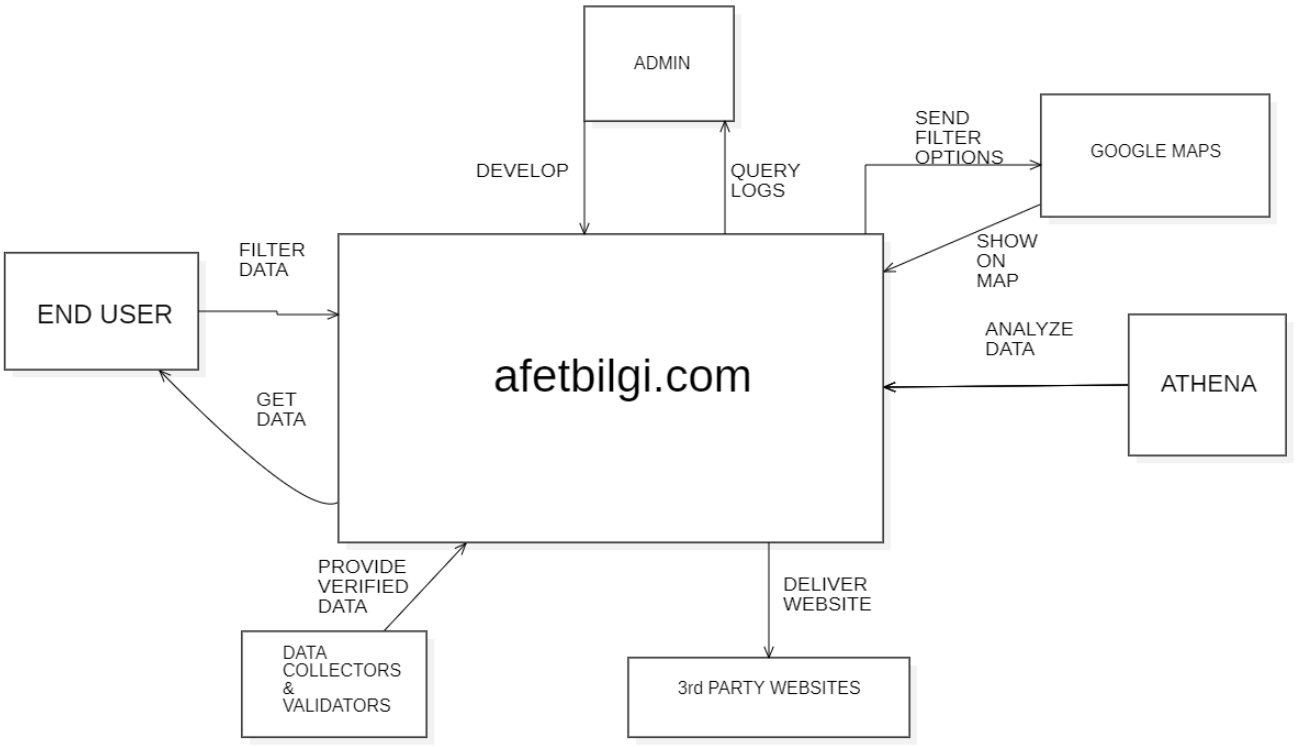

Figure 1: Context Diagram

1.3.1.1 System Interfaces

Google Maps Interface: This interface shows provided locations on Google Map application or on a browser which is a separate website from afetbilgi.com. Use cases are applications that require location based services. The location data that is provided by afetbilgi.com is shown on a 2D map to make users easily find specific locations such as help centers, pharmacies, hospitals, food-delivery centers, hotels, etc.

Data Collectors & Human Validators Interface: This interface provides verified information that can be placed into the system.

Athena Interface: This interface is a cloud-based query analysis platform that provides a wide range of services to maintain and analyze servers on remote machines. The system utilizes Athena to analyze and view logs

3rd Party Websites Interface: This interface provides 3rd party links to which users are redirected so that they can find official websites of authorities and various resources.

Admin Interface: This interface is a control panel for the admins of the project. Admins can improve and update the system and view and analyze the query error logs.

1.3.1.2 User Interfaces

The user interfaces of afetbilgi.com are designed only for the main users. The main user interface of the page is designed in a simple and efficient fashion, since the target users are the people living in earthquake affected areas and the main priority was to make the website available as early as possible . The data used in the project is kept as a json file, hence there is no user interface for system admins and IT staff, since there is no database implementation or admin-managed functionality in the page. More detailed information about the user interfaces will be provided in the External Interfaces section (3.1).

End-user Interface: The main user interaction between the webpage and the users is provided via the homepage of the website. There is no register or login interface, mainly because the target user group of the project is the people living in earthquake-affected regions and the top priority is effectiveness of the project. The homepage has 4 main sections, namely “General Needs”, “Important Resources”, “Health Services” and “To Help”. The users can easily find the related links and resources via these sections. The homepage also provides links to other two interfaces, which are listed below.

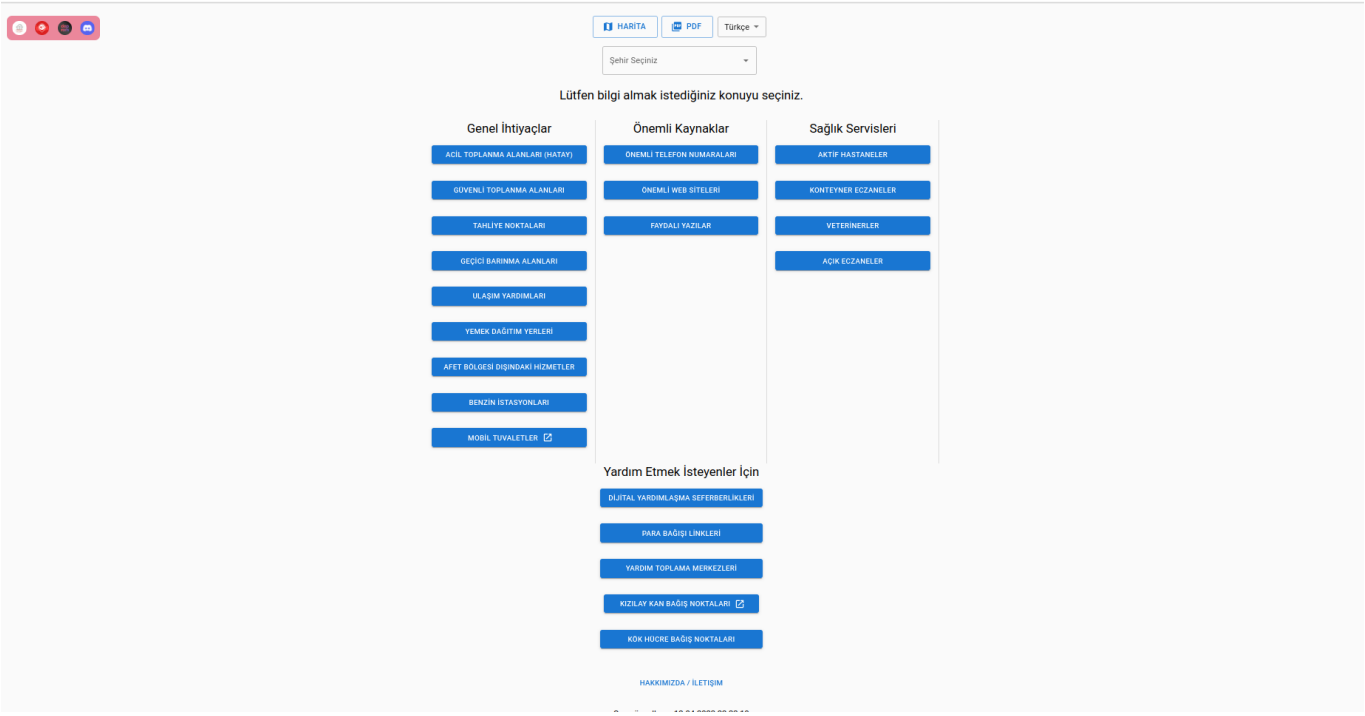

Figure 2: End-user Interface

End-user Map Interface: The map interface is accessible from the top of the homepage interface. It opens on the new tab and the main functionality of it to show locations for food, shelter, medical care etc. marked on the map via the Leaflet api. When clicked on a place, the corresponding location opens on

Google Maps. Users can easily find the places closest to their location for such needs using this map.

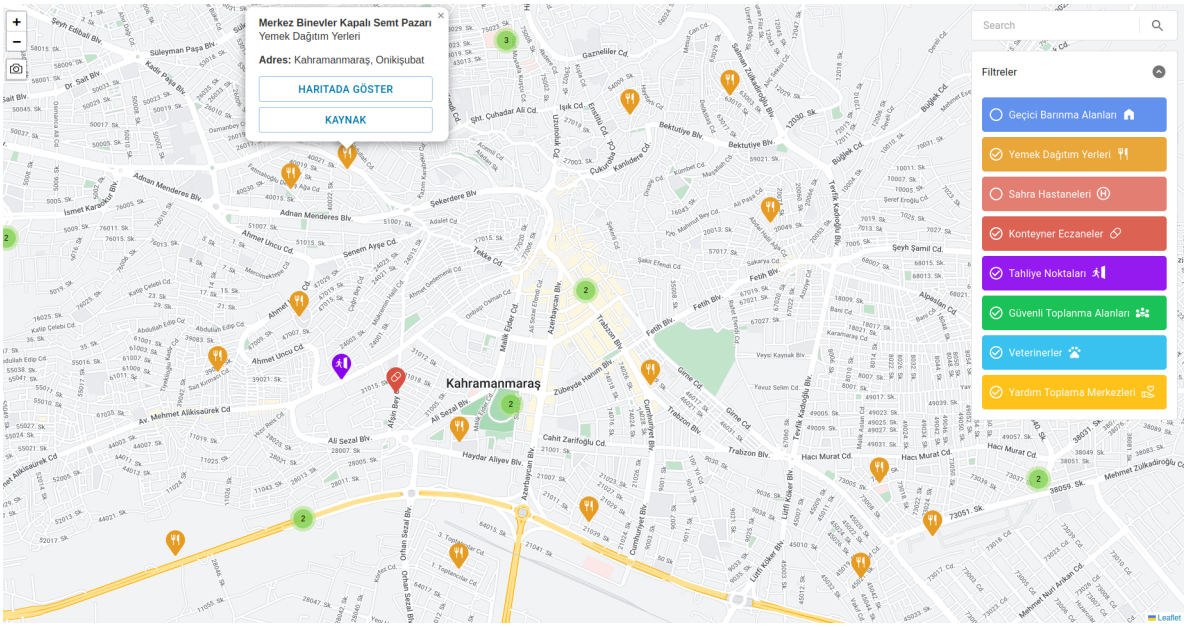

Figure 3: End-user Map User Interface

End-user PDF Interface: This interface is used for users to download the data about a chosen city as a pdf file for a more compact reading. It makes the data available offline and makes it more accessible. The modal greets the user with a field to choose a city and by clicking the download button, the data is downloaded as a pdf file to the target device. The interface has a quite compact and easy-to-use design.

Figure 4: End-user PDF Interface

1.3.1.3 Software Interfaces

Cloudflare Interface: it is a content delivery network and provides various services to improve website performance, security, and reliability. It operates a global network of servers that cache website content, reducing the load on origin servers and accelerating the delivery of content to end-users.

Amazon S3 Bucket Data Interface: It is basically used as a data warehouse. System shall not contain neither a rational nor non-relational database. For the sake of keeping the system simple for contributors, low-latency initializing time, and ease of development, Amazon s3 buckets are utilized as data sources. Data is stored as objects.

CloudWatch Interface: It is used for trailing requests from different resources, blocking IPs that may be considered as a source of Ddos attacks, monitoring remote servers' health situation etc. It is basically a monitoring and management service for AWS infrastructure.

Athena Interface: This interface is used to analyze and query the data that is stored in S3 buckets by using SQL. It is serverless and easy to manage which relieves the burden of developers for the sake of an easy development process.

S3 Backup Bucket Interface: It is used for backing the data up. In case of corruption in the primary bucket, accidently deleting or overwriting data is prevented. It provides versioning, back-up and high availability.

Data Parser and Markdown Generator Interface: It is used for parsing the raw data file according to projects' structure. After parsing it, markdown is placed into a pdf that is produced by the system.

1.3.1.4 Communication Interfaces

Afetbilgi.com uses HTTPS protocol for the communication between its users and the website. HTTP protocols ensure that data inflight is encrypted using certificates so it is secure and middle-man attacks are avoided. Also, since the website uses a static content delivery system, it provides an extra layer of security as well as faster access.

1.3.1.5 Memory Constraints

Memory constraints are not the main concern of afetbilgi.com. The system does not have a database. It serves its data from S3 Bucket as a JSON object. Small amount of memory in devices is enough to maintain the system.

1.3.1.6 Operations

The operations provided by afetbilgi.com can be divided in three:

User operations:

- Filter data by the selected city
- Download the data as pdf
- View the specific locations on Maps
- Contact the developers
- Provide verified information
- Get the data provided without downloading

System operations:

- Provide static data content as website

- Create pdf by filter choices
- Maintain backup in case of disaster situations

Staff operations:

- Maintain verified and up-to-date data
- View data bucket logs
- Analyze data on Athena
- Authorize developer members

1.3.2 Product Functions

The characteristics and abilities that the software system should offer to its consumers are referred to as product functions. With their thorough description tables, (3.2) provides a more in-depth explanation of how each product performs within the system. The product functions of the system is listed in the table below:

Functions	**Summary**
Contribute to the system	End users contribute voluntarily to the system via contacting admins
Contact admins	End users contact admin to communicate about contribution, data verification, etc.
Adding new data	Validators provide verified information, and after adminstory process of deployment, new data is updated into S3.
Accessing temporary accommodation places	Users can access the temporary accommodation places for a selected city.
Accessing evacuation points	Users can access the evacuation points for a selected city filters
Accessing safe gathering places	Users can access the safe gathering places for a selected city.
Selecting services on the map	Users select services on map by filtering according to what kind of service they are looking for.
Finding donation resources	Users are redirected to 3th party money donation organizations
Accessing important websites	Users can access the important websites by filtering where they need it.
Accessing crucial phone numbers	Users can access the crucial phone numbers by filtering where they need it.

Table 1: Product functions table

1.3.3 User Characteristics

There are three main users of afetbilgi.com. They are Admin, End Users, and Data Collectors & Validators.

End Users are the people who want to access websites to get help and information about where and how to find help. Since the website is served in Turkish, Kurdi, and English, basic understanding of any of these three languages and ability to interact with the website by clicking buttons to redirect is enough for End Users.

Admins are the one who regulates the website. They are in charge of maintaining the website, adding new features, and inspecting errors that may be caused. Basically solving technical issues are the main responsibility of the admins. Having high computer skills, knowing the internal mechanism of services used by afetbilgi.com, and producing the appropriate solutions to improve the system are essential for Admins.

Data Collectors & Validators are the ones who want to improve the website by somehow collecting the verified data and validating them. Since the website is served in 3 languages, basic understanding of these three languages, and basic knowledge of excel are essential for this role since data is collected in the form of excel.

1.3.4 Limitations

- **Regulatory policies:** The system does not hold any critical data related to users. However, the data transfer from/to the system should be regulated in order to assure the correctness and reliability of the system.
- **Hardware Limitations:** The data kept in AWS should be easy-to-access and scalable in case of any abrupt increase in the number of entries kept in the system. Also, internet bandwidth should be adequate to support user access, since the people who live in the earthquake-prone areas are the site's target audience.
- **Interfaces to other applications:** The system uses several interfaces for Google Maps and PDF API. The system should be compatible with these APIs and run smoothly on all devices - especially on mobile devices.
- **Parallel operation:** Parallelization is not a crucial limitation in afetbilgi.com, since most of the operations are read-only by end users.
- **Audit and control functions:** There are no audit functions, because the goal of the system is not to raise money. The control functions, on the other hand, should be accessible only by the system admin in order to ensure the reliability of the system. Hence, they need privileges for those operations.
- **Higher-order language requirements:** The frontend is written in React, it has a very clean and result-oriented interface. However, there is no backend in the system. Since the operations done on the system are mainly read-only, there are no hard-coded requirements for the system.
- **Signal handshake protocols:** Mostly, HTTPS is used for receiving - sending information to the system and the server.
- **Quality requirements:** Admins and the IT related people within the project are responsible for keeping the system fast and reliable. The security of the system is mostly provided by Amazon and AWS Cloudflare service.
- **Criticality of the application:** Failures in the system can cause people not to receive the help / aid they are seeking and might result in financial and emotional

damage. However, the system is not failure-prone since it has a very basic purpose and architecture.
- **Safety and security considerations:** The integrity of the data kept in Amazon services is very crucial, since otherwise it damages the reliability of the systems.
- **Physical/mental considerations:** Anyone who can use a device which has access to the internet is able to use afetbilgi.com.

1.4 Definitions

Term	Definition
UI	User interface
API	Application programming interface
Database	S3 object storage system
HTTP/S	Hypertext transfer protocol
VPC	Virtual Private Cloud
JSON	Javascript object notation
AWS	Amazon web services

Table 2: Definition table

2. References

This document is prepared by specifications according to the IEEE 29148-2011 standard:

1 - ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering.

3. Specific Requirements

3.1 External Interfaces

- **Google Maps:** The coordinates of the place selected on the map view is passed to the Google Maps interface and displayed on a separate page.
- **AWS S3:** This interface is a cloud based computing platform offered by Amazon.com that provides a wide range of services to maintain servers on remote machines. The

system utilizes S3 Bucket, CloudWatch, to provide itsservers a remote machine so that it will be accessible from everywhere in the world.

- **Cloudflare:** This interface is a cloud-based platform that provides a wide range of services to optimize and secure the web applications. It provides a content delivery network to worldwide users fast and secure. It prevents DDoS attacks and provides SSL/TLS encryption to HTTPS connections.
- **PDF Interface:** This interface is a software tool which enables developers to manipulate files using Javascript and create PDF (Portable Document Format). All the data in the primary data bucket can be easily manipulated,filtered and downloaded to users' devices.
- **Vercel Interface:** This interface provides a serverless deployment for web applications. It is used for static content delivery of React applications via continuous deployment and global content delivery network.
- **Data Parser API:** This interface enables the system to parse verified data into JSON (Javascript Object Notation) automatically, which enables verified data to be easily integrated into the system.

3.2 Functions

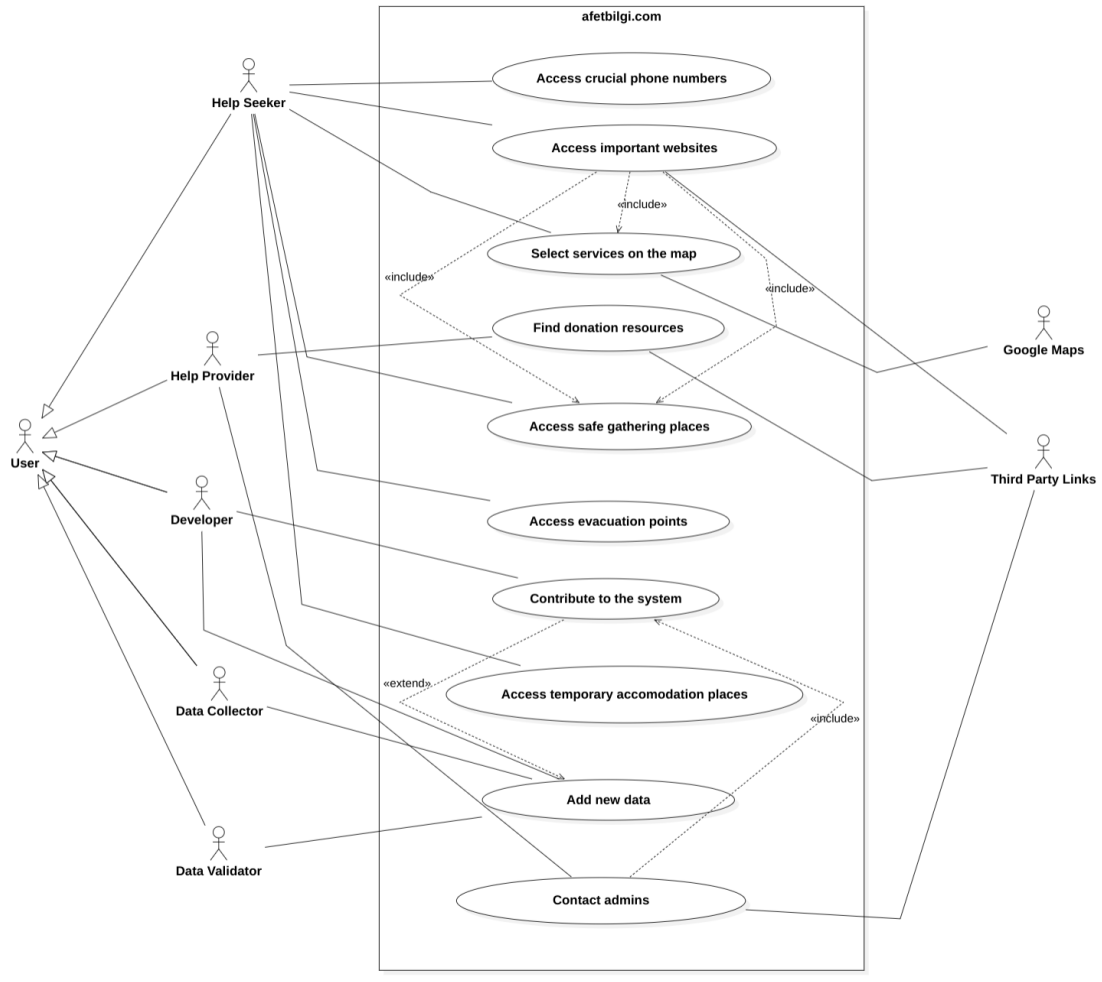

Figure 5: Use Case Diagram

Use case name	Contribute to the system
Actors	Developer
Description	End user contributes voluntarily to the system via contacting admins
Data	-
Preconditions	One of the developers must be available
Stimulus	End user wants to contribute
Basic Flow	Step1. End user enters website Step2. End user clicks contact Step3. End user clicks github icon Step4. End user is redirected to github repo where s/he can contribute directly to the project after approval of admin
Exception Flow	Contribution is not approved by the admin and the end user fails to contribute to the system.
Postconditions	End user contributes to the system after committing approval of the admin.

Table 3: Contribute to the system

Use case name	Contact admins
Actors	Help-provider, 3th party links
Descriptions	End users contact admin to communicate about contribution, data verification, etc.

Data	-
Preconditions	At least one admin must be available
Stimulus	End user want to contact admin
Basic Flow	Step1. End user enter the website Step2. End user click contact button in the main page Step3. End user sends email to the admin
Alternative Flow #1	Step3. End user clicks twitter icon Step4. End user is redirected to the twitter page of afetibilgi.com
Alternative Flow #2	Step3. End user clicks github icon Step4. End user accesses github repository to contact
Alternative Flow #3	Step3. End user clicks instagram icon Step4. End user redirected to instagram page to contact admin
Exception Flow	Error logs are sent to CloudWatch and let the admin know about causes of exceptions.
Postconditions	User contacts admin to communicate

Table 4: Contact admins

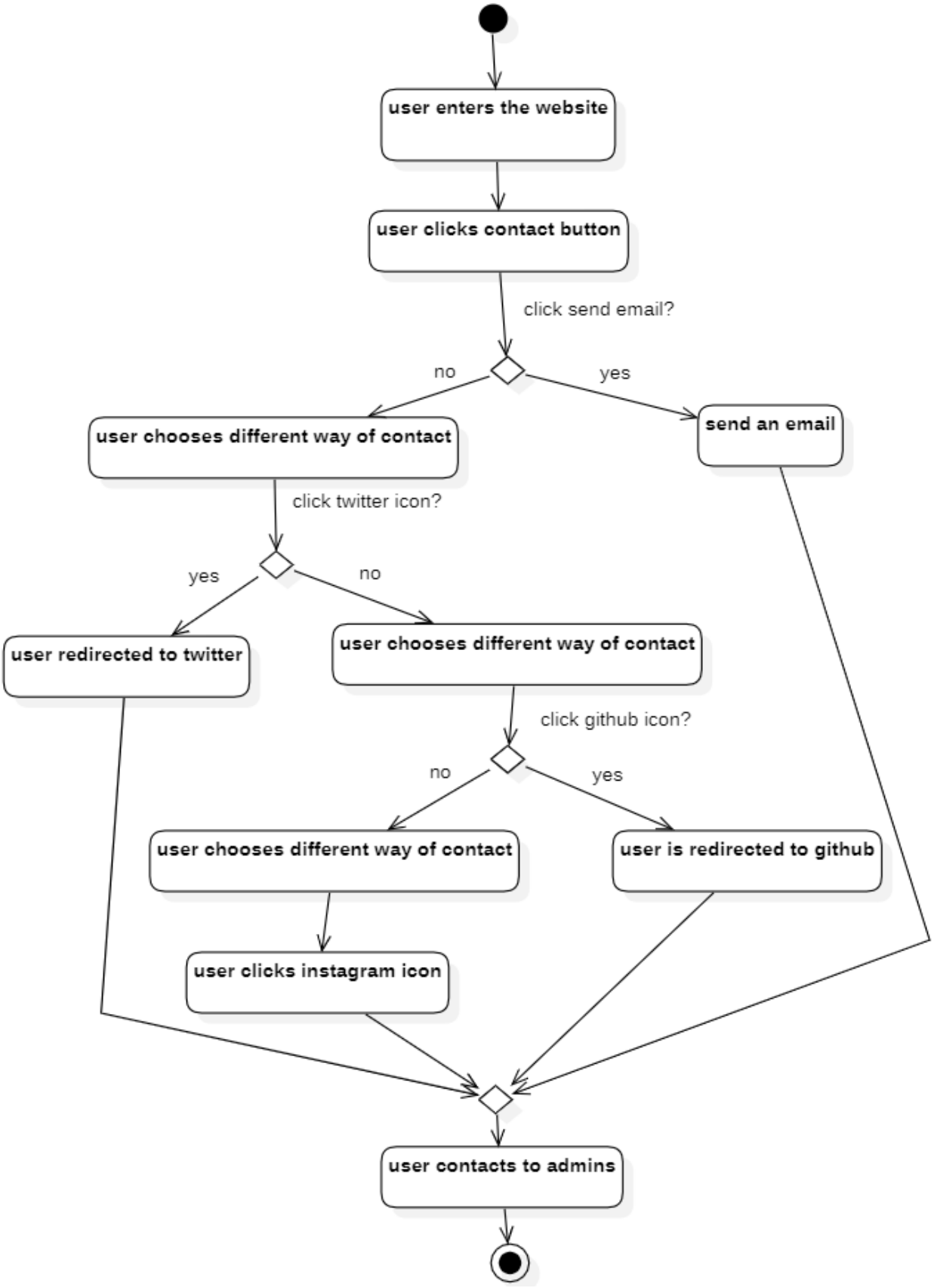

Figure 6: Activity diagram for contact admin

Use case name	Adding new data
Actors	Developer
Description	Human validators provide verified information, and after adminstory process of deployment, new data is updated into S3.
Data	The data provided by human validators, and selected city
Preconditions	Human validators shall contact admin and admin should be available to see their messages. AWS must be available.
Stimulus	Human validators wants to update or add new information
Basic Flow	Step1. Human validator fills excel file Step2. Human validator contacts admins Step3. Admin parse the data accordingly using data parser Step4. Admin generates data pdf using pdf api Step5. Admin uploads processed data into S3 Bucket Step6. Updated data is delivered to the end user.
Exception Flow	Error logs are sent to CloudWatch and let the admin know about causes of exceptions.
Postconditions	Data is updated with new information

Table 5: Adding new data

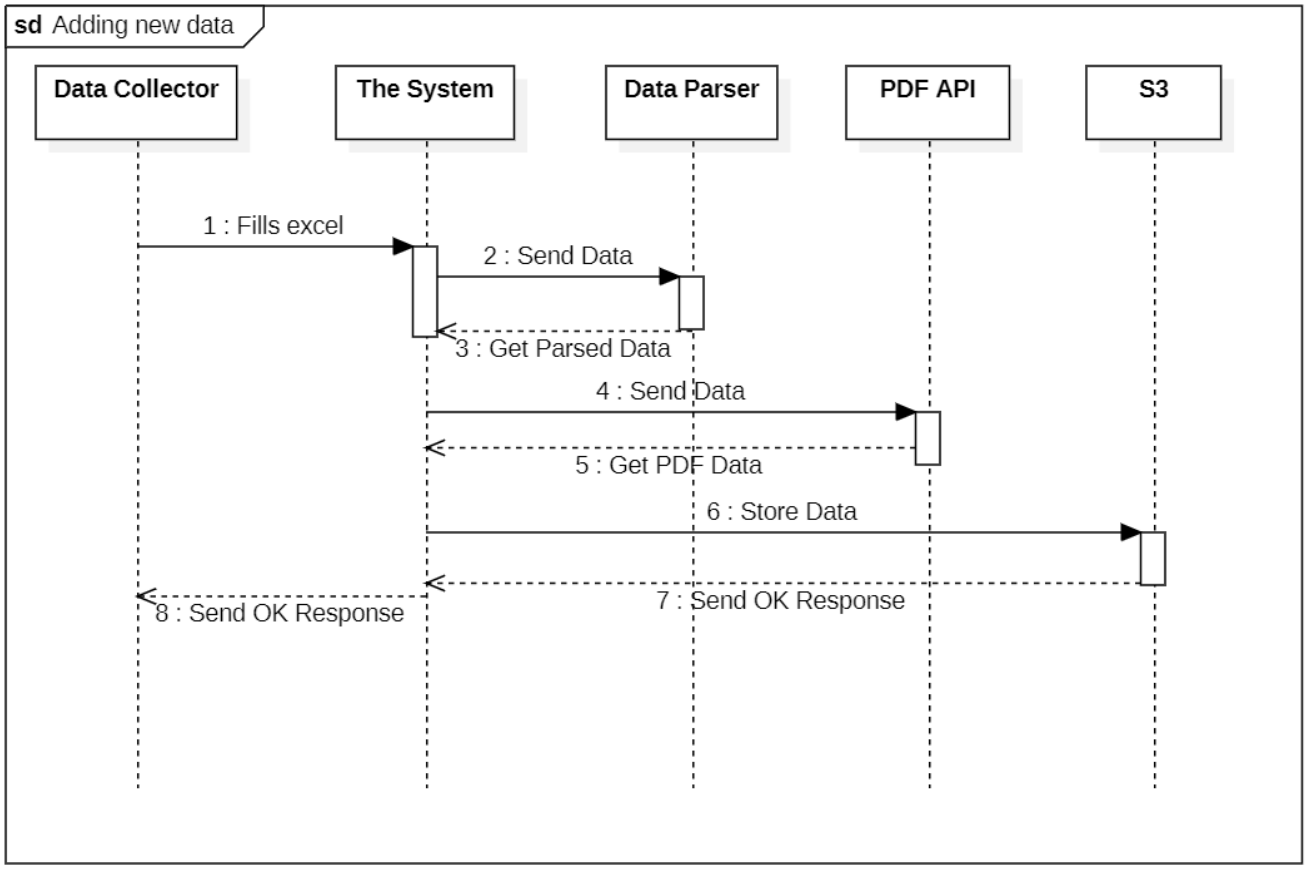

Figure 7: Sequence diagram for adding new data

Use case name	Accessing temporary accommodation places
Actors	End user
Descriptions	Users can access the temporary accommodation places for a selected city.
Data	Selected city
Preconditions	3th party source link, twitter link must be available
Stimulus	End user wants to access temporary accommodation place information

Basic Flow	Step1. Click “Accessing Temporary Accommodation Places” under the General Needs section. Step2. Select a city. Step3. Click the link for the map or the phone number.
Alternative Flow	Step4. Either Twitter link or the source link is clicked.
Exception Flow	There is no data for a selected city, hence nothing is listed.
Postconditions	User accesses the desired data.

Table 6: Accessing temporary accommodation places

Use case name	Accessing evacuation points
Actors	End-users
Description	Users can access the evacuation points for a selected city filters
Data	Selected city
Preconditions	Google Maps, phone call redirection
Stimulus	End user wants to access evacuation place information
Basic Flow	Step1. Click “Evacuation Points” under the General Needs section. Step2. Select a city. Step3. Click the link for the map or the phone number.

	Step4. Map is clicked.
Alternative Flow	Step4. Phone number is clicked.
Exception Flow	There is no data for a selected city, hence nothing is listed.
Postconditions	User accesses the desired data

Table 7: Accessing evacuation points

Use case name	Accessing safe gathering places
Actors	End-users
Description	Users can access the safe gathering places for a selected city.
Data	Selected city
Preconditions	Google Maps API, 3th party links
Stimulus	End user wants to access safe gathering place information
Basic Flows	Step1. Click safe gathering places under the General Needs section. Step2. Select a city. Step3. Click the link for the map or source. Step4. Map is clicked.
Alternative Flow	Step4. Source is clicked.
Exception Flow	There is no data for a selected city, hence nothing is listed
Postconditions	User accesses the desired data

Table 8: Accessing safe gathering places

Use case name	Selecting services on the map
Actors	End Users, Google Map
Description	Users select services on map by filtering according to what kind of service they are looking for
Data	Selected city
Preconditions	CloudFlare, Google Maps, 3th party links and Amazon s3 Bucket must be available to HTTPS requests.
Stimulus	End user wants to map and location information of selected services based on location
Basic Flow	Step1. Click the “map” button. Step2. User is redirected to the maps application Step3. Users select a location on a map. Step4. User clicks “Show on the map” button Step5. User is redirected to the google maps with related location
Alternative Flow #1	Step3. User filters by categories of listed locations Step4. User clicks “Show on the map” button Step5. User is redirected to the google maps with related location

Alternative Flow #2	Step4. User filters by categories of listed locations Step5. User clicks the “Resource” button. Step6. User is redirected to related 3th party source link
Exception Flow	Error logs are sent to CloudWatch and let the admin know about causes of exceptions.
PostConditions	User access a location information of a service

Table 9: Selecting services on the map

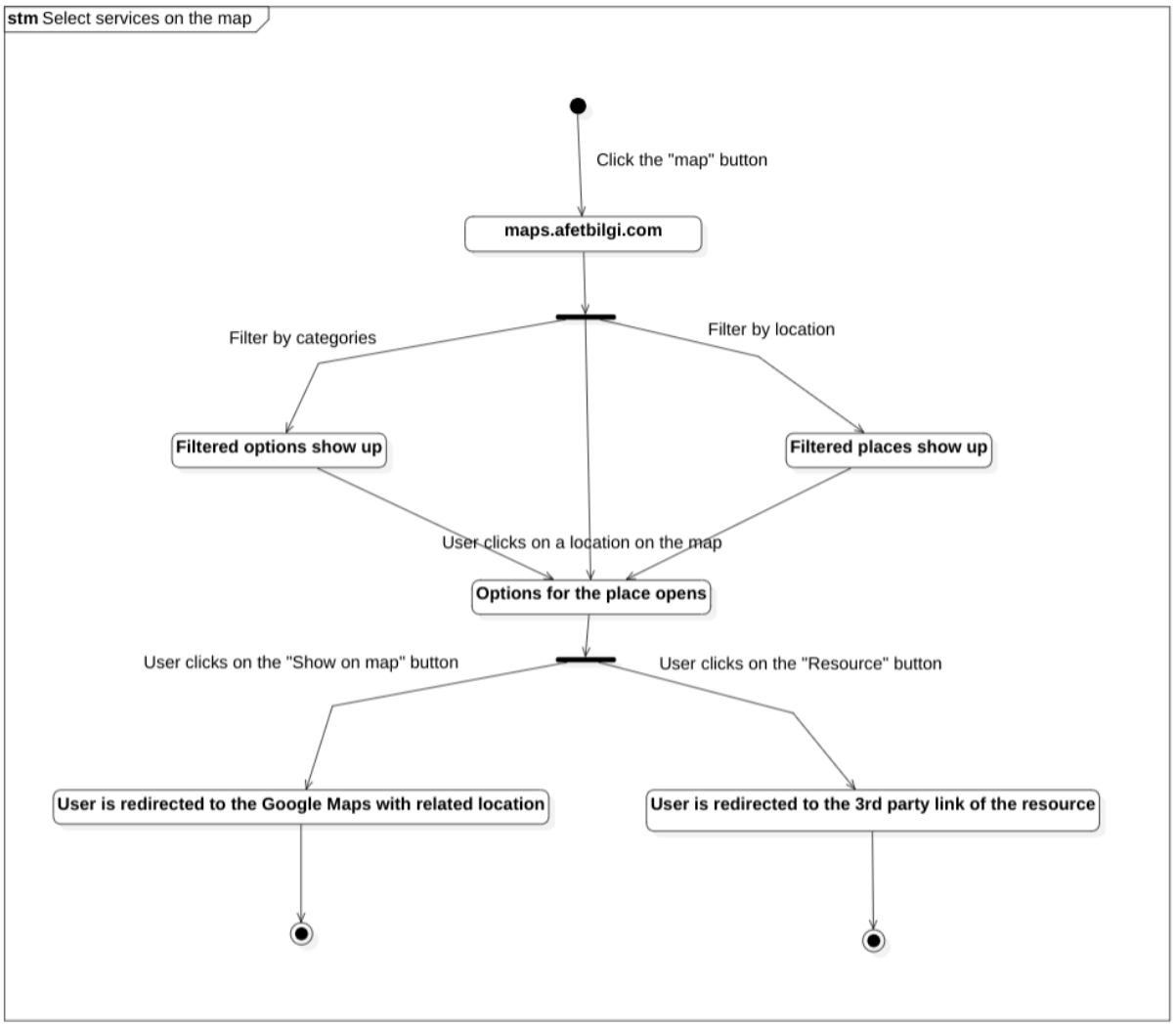

Figure 8: State diagram for select services on the map

Use case name	Finding donation resources
Actors	Help providers, 3th party links
Description	Users are redirected to 3th party money donation organizations
Data	-
Preconditions	3rd party links must be available
Stimulus	End user wants to access donation resource information
Basic Flow	Step1. User enters the website Step2. User clicks the “Monetary Donation Links” button Step3. User is redirected to a different page that all donation receivers listed. Step4. User selects and clicks one of the 3th party links Step5. User leaves the system to visit 3th party links
Exception Flow	Error logs are sent to CloudWatch and let the admin know about causes of exceptions.
Postconditions	Users are redirected to the 3rd part links to donation.

Table 10: Finding donation resources

User case name	Accessing important websites
Actors	Help seeker, 3rd party links

Description	Users can access the important websites by filtering where they need it
Data	selected city
Preconditions	CloudFlare and Amazon s3 Bucket must be available to HTTPS request
Stimulus	End user wants to access important websites
BasicFlow	Step1. User filters by city Step2. User clicks “Useful links” button on the homepage under “Important Resources” Step3. User accesses the listed links with which phone number detail is shared Step4. User clicks to the link selected and the system redirect user to the related website
Exception Flow	Error logs are sent to CloudWatch and let the admin know about causes of exceptions.
Postconditions	User accesses related 3th party website

Table 11: Accessing important websites

Use case name	Accessing crucial phone numbers
Actors	Help seeker
Descriptions	Users can access the crucial phone numbers by filtering where they need it
Data	Selected city

Preconditions	-
Stimulus	End user wants to access crucial phone number information
Basic Flow	Step1. User filters by city Step2. User clicks “Crucial Phone Number” button on the homepage under “Important Resources” Step3. User accesses the listed phone numbers with which phone number detail is shared
Exception Flow	Error logs are sent to CloudWatch and let the admin know about causes of exceptions.
Postconditions	User contacts the related phone number

Table 12: Accessing crucial phone numbers

3.3 Usability Requirements

- Users shall use all related functions of the system as long as they can have an active internet connection.
- Users shall use without signing in, no need for an account.
- Users shall download all the data in case they want to access the data offline.
- Users shall access all the related data at most 4 clicks to buttons.
- Admin shall reach any records of error logs due to the website or 3th party services and analyze it
- Data Collectors & Validators shall update the data that is used in the website after approval of admin

3.4 Performance Requirements

- The system shall be available to all users simultaneously, there is practically no limit to serve static website
- The system’s data shall have 99.999999999% durability and 99.99% availability over a year.

- The system provides static website with caching, so loading time shall only depend the traffic between CDN provider and end user
- Database of the system shall be synchronized with one backup database, and in case of failure of the main database, the backup database shall be served in under 1 minute. One of the databases shall be serving as the main database whereas the other one only is used in case of failure and backup.

3.5 Logical Database Requirements

There is no logical database used in the system. The data is kept as .json files on the frontend and stored in AWS. The data sheet generated by the data collectors & validators is converted into .json files and then stored in the AWS S3 bucket. However, the format of the .json files is closest to NoSQL databases at best.

3.6 Design Constraints

The system does not have any account system, so there is no private data of users such as identity information, password, or credit card information. Only data that the system owns is data to be shared publicly without any legal concern since the system is designed to be accessed to get crucial information.

Although all the data is shared publicly without any legal concerns, since the website is allowing HTTPS requests, data is encrypted in flight. Also, static files are encrypted while stored in S3 bucket. As a result, no information shall be transmitted in plain text.

3.7 Software System Attributes

a) **Reliability:**
   - Failure of the system is quite unlikely, since the website is mostly static. The system's data has 99.999999999% durability.

b) **Availability:**
   - The system should have 99.99% availability over a year.

c) **Security:**
   - Security is mostly provided by Amazon Cloudflare and other Amazon services.

d) **Maintainability:**
   - The data and the documentation of the system should be updated and checked regularly in order to ensure a reliable and up-to-date system.

e) **Portability:**
   - The website is responsive and compatible with all browsers. Therefore, the website is guaranteed to work on all devices with connection to the internet.

3.8 Supporting Information

afetbilgi.com is a system that was developed by volunteers to publicly share information for the ones affected by earthquake. The website only contains static content, so it shall be fast, secure, and light.

All the rights of data are publicly given so there are no legal constraints to be taken or accounted for. The system relies on 3rd party services such as AWS, CloudFlare, and Vercel. The website shall be accessed online and be downloaded to any device to access offline.

The validation of data is done by volunteers called “Data Collectors & Validators”. In case any information provided is wrong, then they shall be responsible for spreading corrupted information.

4. Suggestions to improve the existing system

We suggest that:

- afetbilgi.com should also be a mobile app.
- afetbilgi.com should have an account system.
- End users can register into the account system.
- There would be a concept such as Help Request.
- Users would create Help Requests.
- Admin would manage Help Requests.
- There would be an admin panel so that admin would easily contact authorities to answer help requests.

End users can download the mobile app of afetbilgi.com so that they can easily use the original version of the website instead of reading through the downloaded PDF. Thanks to mobile apps, there would be an account registration system that can be integrated with mobile apps. Users would easily register into this system by providing some information such as their phone number, name, location, blood type, relatives’ name, and emergency contact number. In case of emergency, registered users can easily access a roll call thanks to this system. After an emergency, the end user checks his/her status as alive so that no resource will be wasted. Also registered users would create Help requests for nutrients, paramedical help, or crane needs to lift rubble, etc. . When a help request is created, the admin can regulate this help request based on its location and would let authorities to react to an emergency so that only an internet connection would be enough to collect proper data about needed help.

4.1. System Perspective

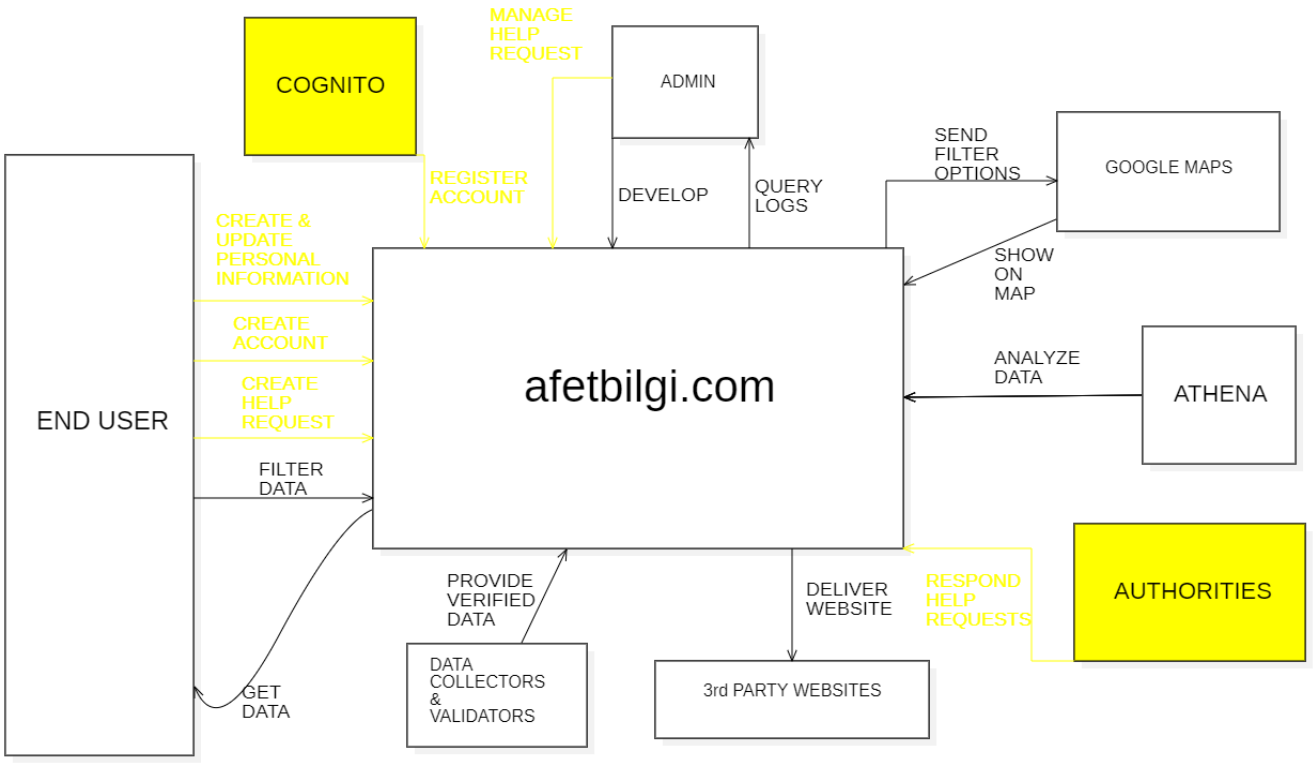

Figure 9: Context diagram

4.1.1. System Interfaces

Authority Interfaces: This interface enables users to create help requests. After creating various help requests, the admin gets a warning. According to the type of warning, admin contacts authorities to start operation. After the request is fulfilled, the admin marks the request as done. The types of requests are request nutrition, request tent, request paramedical, request security authorities (police, soldier, etc.), report crime, request for saving earthquake victims who are under a wreck, request crane, request clothes, etc. Authorities are guided according to the emergency of request and locations.

Cognito Interface: This interface enables fast and ordered communication as it provides a roll call system. Users who are enrolled in the system have to provide their emergency information. The status of the user is to be provided. In case the user is alive and does not need any help, the user shall check it is okay so no resource is wasted and all the sources are used efficiently. Status alive, dead, or missing are the fundamental information. Moreover; information of emergency contact, blood type, members of family, possible locations that users may be present can help authorities to help victims while using limited resources efficiently. According to the provided and updated information that was provided by earthquake victims, new help requests may be created or closed so that loved ones of the victims have proper and correct information.

4.1.2. User Interfaces

Account Interface: Account interface is where users can create their accounts and manage their actions on the website with the panel opened. An account can be of several types such as admin, help seeker and help provider. An account of admin type has access to all requests created by help seekers and can close them when the required condition is met. Help seeker and help provider accounts, on the other hand, do not have to have the privileges of an admin account. They can create requests on their demand. Different views must be available for different account types. Account interface is required because the system should hold data related to each specific user registered. The account interface is mainly about authentication.

Request Interface: Request interface is where users can open / close requests depending on their account types. A help seeker account can create new requests, which can be closed by admin accounts when the required criteria are met. This requires an authorization action, since a help seeker account should not perform a close request action.

4.1.3. Software Interfaces

Cognito: Cognito is used for the account management system. The moderation of the accounts should be run on the Cognito interface.

Google Play Store: Google Play Store is used for downloading the Android App of the project. In addition, updates to the app should be handled on Google Play Store as well.

App Store: App Store is used for downloading the iOS App of the project. In addition, updates to the app should be handled on the App Store as well.

4.1.4. Communication Interfaces

For user-website communication, Afetbilgi.com employs the HTTPS protocol. HTTP protocols make sure that data is encrypted while in transit using certificates to keep it safe and prevent middle-man attacks. Additionally, the website offers a higher level of protection and faster access because it uses a static content delivery system.

4.1.5. Memory Constraints

Prior to the improvements to the system, there was no need to implement a database. With the newly added features, we need to store data related to each account and hence it bears a need for a database. For authentication, Amazon Incognito service shall be used and for the account-related data and the request data, a database service should be implemented. The system should be fast, reliable and secure. Different requests should not interfere with each other.

4.1.6. Operations

For the sake of clarity, already existent operations are not mentioned. Only the new ones are listed. The operations suggested can be divided into three:
User operations:

- Create account
- Provide account information
- Create help request
- Download static mobile app

System operations:

- Provide static mobile app for offline use
- Register accounts

Admin operations:

- Answer requests

4.2. External Interfaces

4.3. Functions

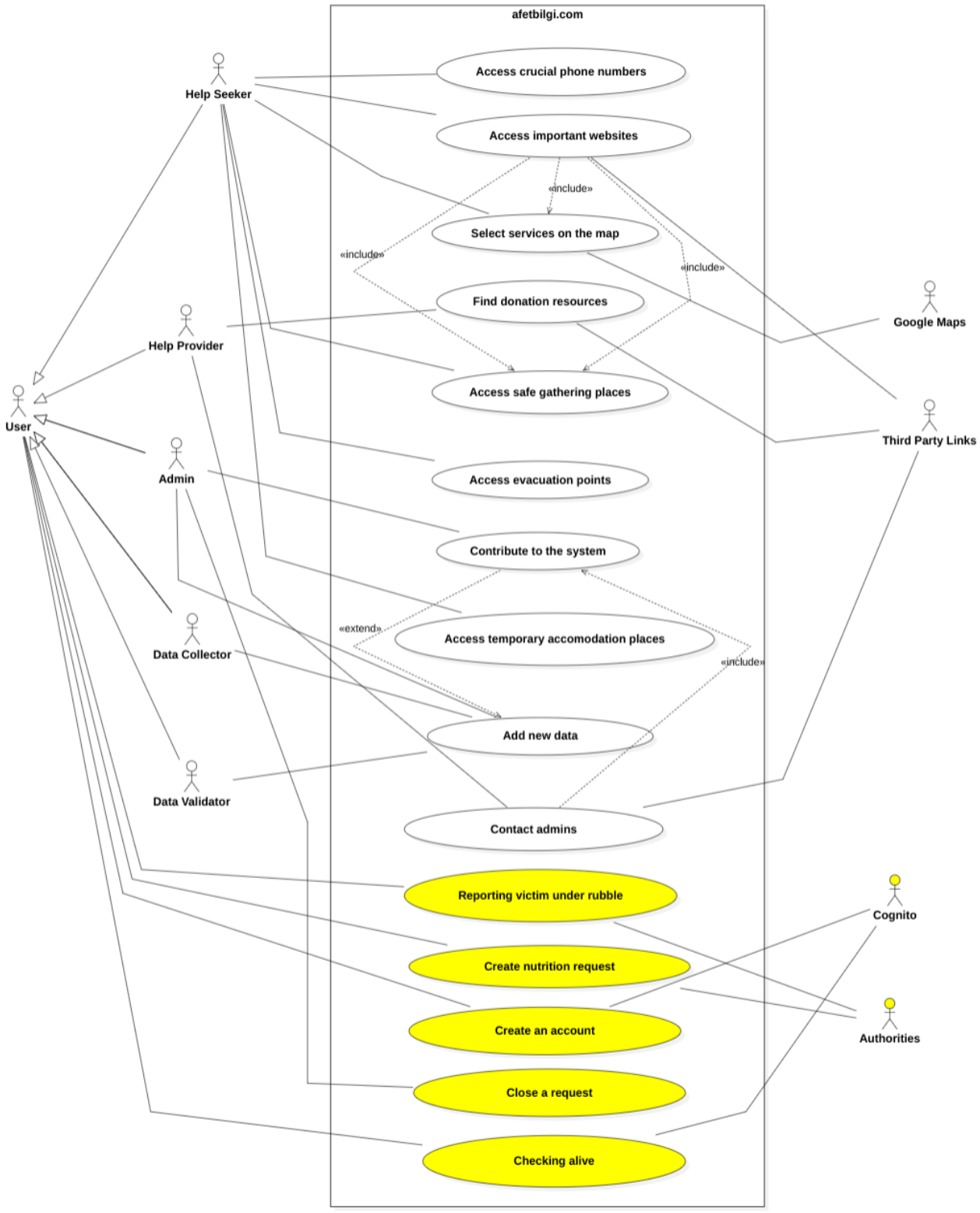

Figure 10: Use case diagram

Use case name	Report victim under a rubble
Actors	End user, admin, authorities
Description	End user reports his location as creating a help request about a victim under a rubble. Admin let authorities know and the saving operation is done.
Data	Victim’s location
Preconditions	End user must have an account
Stimulus	-
Basic flow	**Step1.** end user accesses the website or mobile app **Step2.** end user provides location information **Step3**. end user creates a help request **Step4.** admin gets a notification **Step5**. admin contacts authorities about station
Exception flow:	Error logs are sent to admin panel
Postconditions	end user reports a victim’s location who needs help

Table 13: Report victim under a rubble

Use case name	Create nutrition request
Actors	End users, admin, authorities
Description	Users can request a nutrition request, then admin contact authorities to fulfill this request
Data	Location
Preconditions	End user must have an account
Stimulus	-
Basic flow	**Step1.** user accesses the website **Step1.** user creates a nutrition request **Step2.** admin gets a notification when a request is created

	Step3. user waits admin response **Step4.** authorities contact back to admin about status **Step5.** admin marks the request as fulfilled **Step6.** end users gets notification about status
Alternative flow	**Step4.** admin closes the request
Exception flow	Error logs are sent to admin panel
Postconditions	End users get necessary nutrition.

Table 14: Create nutrition request

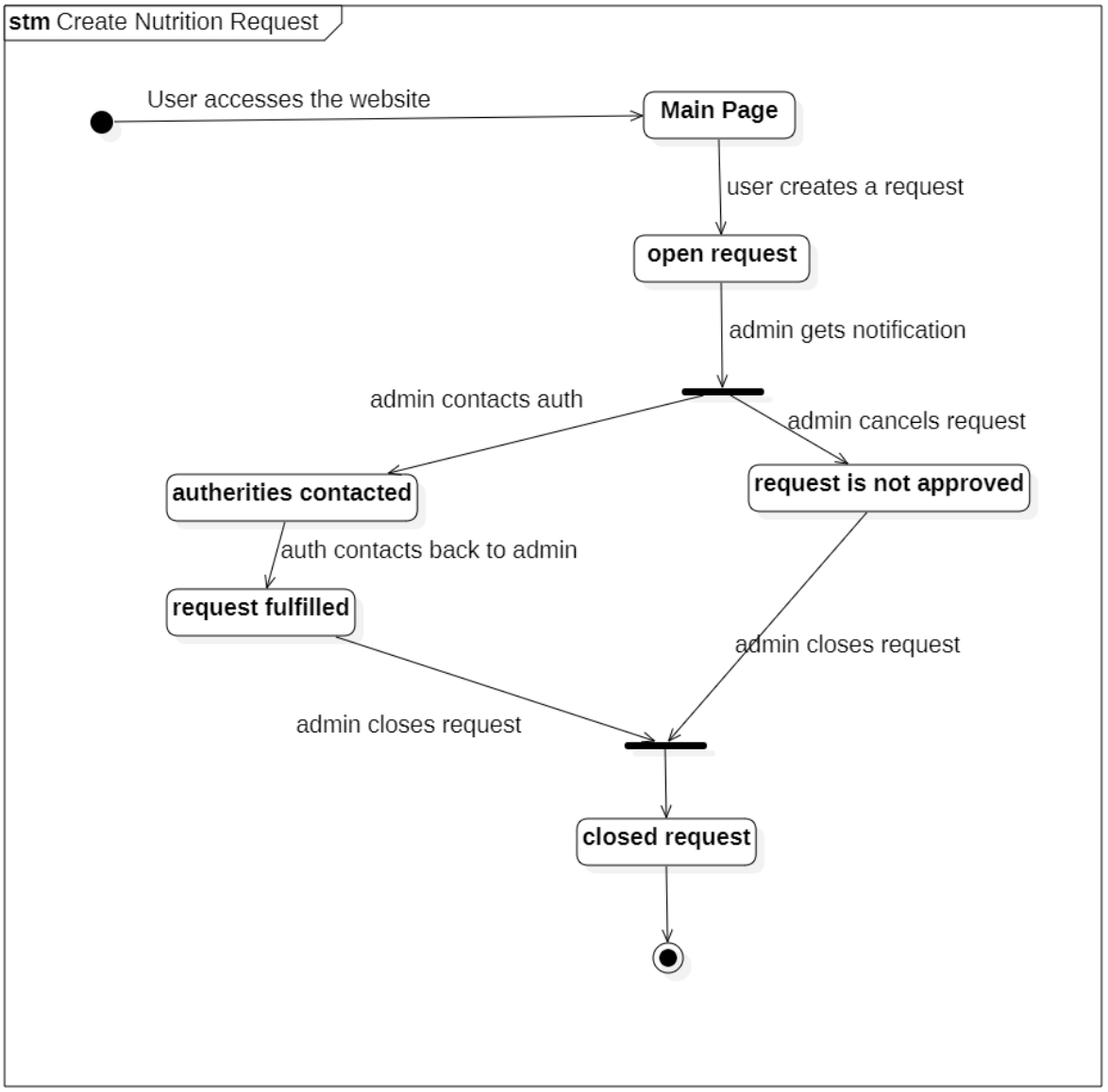

Figure 11: State diagram for create nutrition request

Use case name	Create an account
Actors	End users, cognito
Description	End users create an account using their devices in case of emergency cases
Data	End user’s emergency information
Preconditions	-
Stimulus	-
Basic flow	**Step1**. end user accesses mobile app or the website **Step2**. end user provides necessary information **Step3**. end user completes create account operation **Step4**. database records new user and the system stays vigilant in case of
Exception flow	Error logs are sent to admin panel
Postconditions	End user creates an account to use roll call system

Table 15: Create an account

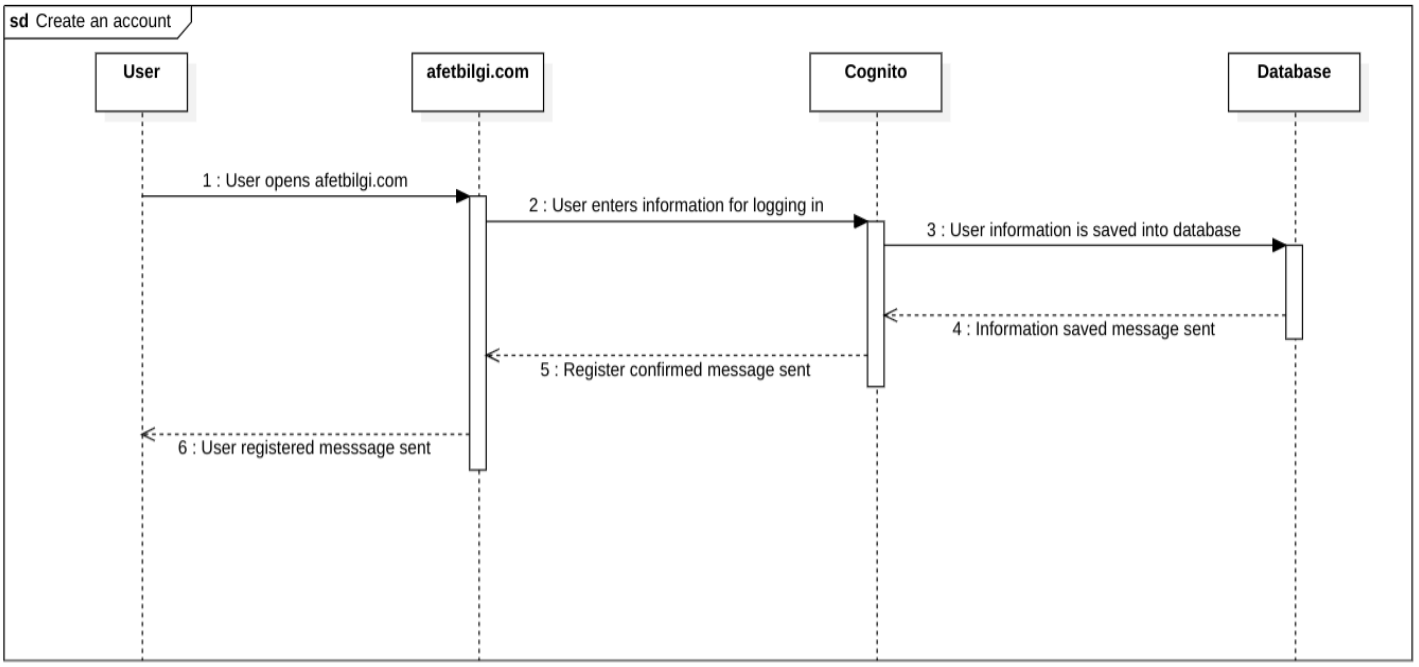

Figure 12: Sequence diagram for Create an account

Use case name	Close a request
Actors	Admin
Description	Admin closes a request. request may be fulfilled or scam
Data	An open help request id
Preconditions	There must be at least one open help request
Stimulus	-
Basic flow	**Step1**. admin decides whether it is a scam request or a real request **Step2**. admin fulfills the request by contacting authorities **Step3**. admin gets contacted by authorities about request **Step4**. admin closes the request
Alternative flow	**Step2**. account of the end user is deleted **Step3**. admin detects it is a scam and closes request directly
Exception flow	Error logs are sent to admin panel
Postconditions	Requests are regulated by admin.

Table 16: Closing a request

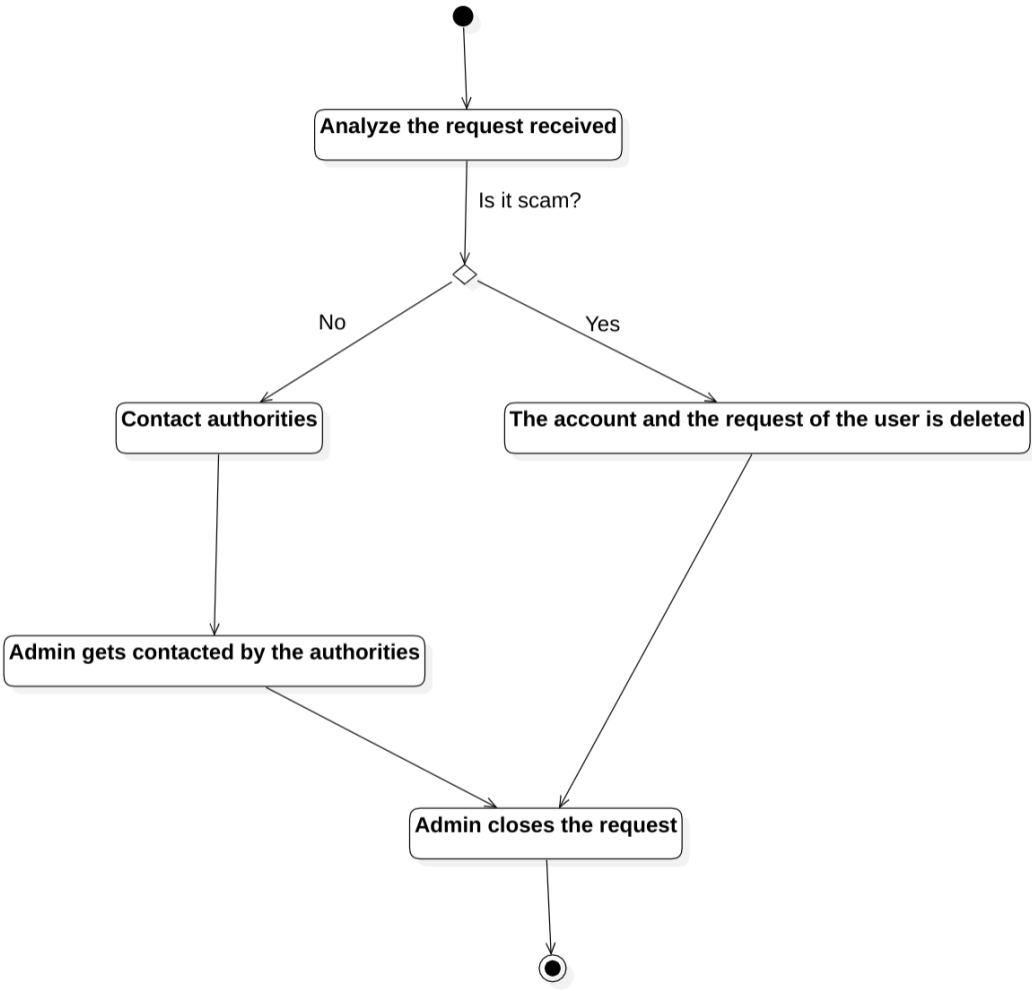

Figure 13: Activity diagram for close request

Use case name	Checking alive
Actors	End users, cognito
Description	In case of a disaster, end users check if s/he is alive so that his/her loved ones have information about his/her status.
Data	Whether end user is alive or not
Preconditions	End user must have an account
Stimulus	-

Basic flow	**Step1**. end user accesses the website or mobile app **Step2**. s/he check his/her status as alive **Step3**. saves the status in cognito **Step4**. admin gets a notification
Exception flow	Error logs are sent to admin panel
Postconditions	Disaster victim updates the data whether he is okay or not

Table 17: Checking alive

4.4. Usability Requirements

- Users shall use all related functions of the system as long as they can have an active internet connection.
- Users shall use the website whether by creating an account or not. Signing in is not a requirement but it is highly encouraged, since it has features that are highly useful in the case of a natural disaster.
- Users who have signed in can fill in their personal data, which will be shared with the authorities in case of a disaster.
- Users who have signed in will have an “I am safe” button, by clicking which they will be able to share their current situation in case of a disaster.
- Users who have signed in can create a “nutrition request” and “stuck under rubble request”. Their location and personal information will be shared with the authorities. Their location and situation will also be displayed on the map for any volunteers who want to help.
- Admins will have the authority to close any request that is deemed as spam or that is satisfied.

- Help provider accounts will be able share their location and related information in case of a disaster.
- All users shall download all the data in case they want to access the data offline.
- All users shall access all the related data at most 4 clicks to buttons.
- Admins shall reach any records of error logs due to the website or 3th party services and analyze it.
- Data Collectors & Validators shall update the data that is used in the website after approval of admin.

4.5. Performance Requirements

- The system shall be available to all users simultaneously, there is practically no limit to serve static website
- The newly added database for account and request information will be serverless on AWS, hence it will be available 100% of the time.

- The system provides static website with caching, so loading time shall only depend the traffic between CDN provider and end user
- Database of the system shall be synchronized with one backup database, and in case of failure of the main database, the backup database shall be served in under 1 minute. One of the databases shall be serving as the main database whereas the other one only is used in case of failure and backup.

4.6. Logical Database Requirements

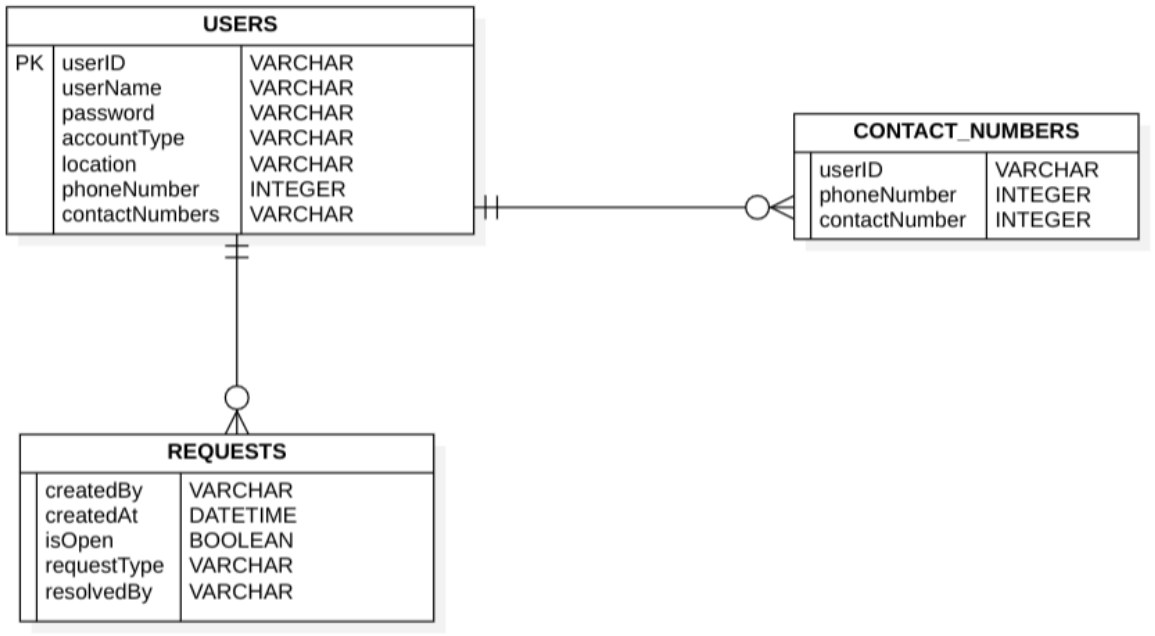

Figure 14: ER diagram for the logical database

With the new features added, such as creating an account and keeping user data, there arises a need for a logical database. The database keeps information related to each user and the type of the account created. When a user is created, userID is assigned automatically and username and password is taken as input from the user. The hash of the password is kept in the database, instead of the plain password for security reasons. The phone number is also used in Cognito. There exists a separate table for phone numbers and related contact numbers. At each request a new entry for the Requests table is inserted and the related information is kept.

There are 3 tables in the database: Users, Contact_Numbers and Requests. Users table keeps data related to each unique user and with userID as its primary key. Contact_Numbers is a table that keeps the phone number and the contact numbers to be called for the particular user. Requests table keeps information related to each request. The person who created the request, the date of the creation, a simple boolean as to if the request is still open, the type of the request and whether the request is still open.

4.7. Design Constraints

- System shall be highly available.

- For the account system, username and passwords shall not have to be recorded since the system uses Cognito.
- Private information of end users shall be stored as encrypted.
- Private information shall be only accessible to a single user according to the regulations.
- Phone number, location, blood type and various information shall be backed up as encrypted.
- Private information shall not be lost due to technical problems.

4.8. System Attributes

f) Reliability:
- Failure of the system is quite unlikely, since the website is mostly static. The system's data has 99.999999999% durability.
- The point in time recovery is possible since suggested attributes are serverless and backed up regularly and automatically.

g) Availability:
- The system should have 99.99% availability over a year.
- The suggested attributes are completely serverless so downtime shall be only seconds.
- The backup of suggested attribute databases are automatically taken by Cognito.

h) Security:
- Security is mostly provided by Amazon Cloudflare and other Amazon services. Also, the hashed version of the user passwords are kept in the database.
- End user shall need username, password, and his/her recorded device to access the account system. It provides an extra layer of security.

i) Maintainability:
- The data and the documentation of the system should be updated and checked regularly in order to ensure a reliable and up-to-date system. Admins should regularly.

j) Portability:
- The website is responsive and compatible with all browsers. Therefore, the website is guaranteed to work on all devices with connection to the internet.

4.9. Supporting Information

Private information shall be stored as encrypted according to the regulations and civil rights. System shall be highly available so suggested databases are completely serverless. Private information shall be backed up to prevent data loss. The remaining system shall be still static to be fast, secure, and light