Software Requirements Specification for

# Afet Bilgi

Emre Geçit, Baran Yancı

April 24, 2023

# Contents

# List of Figures



# List of Tables

# 1 Introduction

This document is the Software Requirements Specification for the **Afet Bilgi** project, developed by a group of METU students to verify and deliver important information to fight against the "February 6, 2023, Pazarcık Earthquake".

## 1.1 Purpose of the System

The purpose of the system is to provide information to those who are affected by the earthquake.

## 1.2 Scope

- The system is used by two groups of people: the people who are affected by the earthquake and the people who want to help the people who are affected by the earthquake.
- People affected by the earthquake can reach important phone numbers, or the locations of important places or services.
- People who want to help the people affected by the earthquake can reach donation centers; such as blood, stem cells, money and so on. People can also create help points and share the locations of the help points, with the help of the system.
- People can also help fight against the earthquake by providing information about the earthquake. Information reaches by email, gets verified by the system managers, and then gets published on the website.

**Figure 1:** Context Diagram

## 1.3 System Overview

### 1.3.1 System Perspective

The `afetbilgi.com` product is not an element of a larger system. The project is split into two main parts. The first part is the front end of the website. The second part is the cloud services that are used to store and process the data. The front end is a web application that is developed using TypeScript and the ReactJS framework. The front end uses packages like MUI and is hosted on the static website `afetbilgi.com`. For the cloud services, the project uses Amazon Web Services (AWS) and the serverless framework. Alongside AWS, GitHub Actions is used for continuous integration and continuous deployment (CI/CD). The cloud services process the data and store it in a database. The data comes from individuals who enter and/or validate the data. The data is collected in Google Sheets and then processed by the cloud services. The cloud services are hosted on AWS. GitHub actions are also responsible for generating PDF files including information about affected areas, from the data in the database. The PDF files are then stored in the cloud services and can be accessed by the front end.

### 1.3.2 System Functions

Functionalities of `afetbilgi.com` are summarized below. More detailed information about the functionalities can be found in Section 3.2.

| Function | Summary |
|---|---|
| **See PDF About a City** | User can see a PDF about an affected city, that includes important information about the city and the earthquake. |
| **Kızılay Donation Centers** | User can see the locations of Kızılay donation centers. |
| **Generate PDF Documents** | Automatic PDF generation. The PDF files include information about affected areas. |
| **Show Maps** | Shows a map of useful locations in the affected area. |
| **Change Language** | User can change the language of the website. |
| **Reach to depremyardim.com** | User can click a link to reach depremyardim.com. |
| **Reach to afetharita.com** | User can click a link to reach to afetharita.com. |
| **Reach to deprem.io** | User can click a link to reach deprem.io. |
| **Join the Discord Server** | User can join the Discord server using the website. |
| **City Selection** | User can select a city to see the information about the city. |

**Table 1:** System Functions

### 1.3.3 Stakeholder Characteristics

Stakeholders of the system can be divided into four main groups:

- Victims of the disaster: People who are affected by the earthquake. They can reach information using the system.
- People who want to help: People who want to help the people who are affected by the earthquake. They can reach information using the system. They can also share the information of their help.
- Information providers: People who voluntarily provide information about the earthquake. They can reach out to the maintainers of the system to provide information.
- Maintainers / Developers: People who maintain and develop the system. They develop the software and maintain it by verifying new information and adding it to the system.

### 1.3.4 Limitations

The limitations of the system are listed below:

- The system should be available on mobile devices, as mobile devices are the most used devices in disaster times.

- The system should not be demanding on the network as the network may be unavailable during disaster times.
- The system should respond quickly as the people who are affected by the earthquake may need the information quickly.

### 1.4 Definitions

- **Earthquake**: Refers to the earthquake that happened on 6th of February 2023, Kahramanmaraş, Turkey.
- **Disaster**: For the time being, an earthquake is the only disaster that is considered in this project, although the system can be used for other disasters as well in the future.
- **Victim**: A person who is affected by the disaster.

## 2 References

This document is prepared with respect to **IEEE 29148-2018 standard**: *Systems and software engineering — Life cycle processes — Requirements engineering*

## 3 Specific Requirements

### 3.1 External Interfaces

There is no external interface for the system. The system is a web application that is hosted on a static website.

### 3.2 Functions

**Figure 2:** Use Case Diagram

| | |
|---|---|
| **Use case name** | See PDF About a City |
| **Actors** | User, Static Website, PDF Viewer |
| **Description** | If a user wants to see the PDF document containing information about a city, the city selection dialog is shown, then the desired is picked on the dialog. |
| **Data** | PDF file about the city |
| **Preconditions** | The PDF for the city file must be refreshed previously. |
| **Stimulus** | User clicks on the "PDF" button and picks a city |
| **Basic flow** | Step 1: User clicks the "PDF" button<br>Step 2: The pop-up dialog is shown<br>Step 3: User picks the desired city on the city selection dialog |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | User is redirected to the PDF viewer |

**Table 2:** See PDF About City Function

| | |
|---|---|
| **Use case name** | Kızılay Blood Donation Places |
| **Actors** | User, Static Website, Kızılay Website |
| **Description** | The user wants to see the blood donation locations. |
| **Data** | Blood Donation Locations |
| **Preconditions** | - |
| **Stimulus** | User clicks on the "Kızılay Blood Donation Places" button |
| **Basic flow** | Step 1: User clicks the "Kızılay Blood Donation Places" button |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | User is redirected to the Kızılay website |

**Table 3:** Kızılay Blood Donation Places Function

| | |
|---|---|
| **Use case name** | Generating PDF Documents |
| **Actors** | GitHub Actions, automation code, Data Providers & Validators |
| **Description** | The PDF files are generated by the automated code running on GitHub Actions. Those PDF files include information about evacuation points, food distribution centers, pharmacies, gas stations and more, based on the districts of the given city. All the information comes from Google Sheets, which holds the data coming from voluntary individuals. |
| **Data** | Open pharmacies, evacuation points, food distribution centers, gas stations, accommodation places, veterinarians. |
| **Preconditions** | The information should be on Google Sheets before generation. |
| **Stimulus** | GitHub actions run this task periodically. |
| **Basic flow** | Step 1: The data from sheets are fetched<br>Step 2: The PDF file is generated using the Python script<br>Step 3: The file is stored in AWS Cloud. |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | The updated PDF file is stored on the cloud, and is ready to be seen on the front end. |

**Table 4:** Generating PDF Documents Function

**Figure 3:** Activity Diagram for Generating PDF Documents

| | |
|---|---|
| **Use case name** | Showing Maps |
| **Actors** | Leaflet, Data Providers & Validators |
| **Description** | The map is generated by the map provider Leaflet. The map includes information about donation centers, temporary accommodation places, food distribution places, pharmacies, gas stations and more, and where they are on the map. All the information comes from voluntary individuals. |
| **Data** | Open pharmacies, evacuation points, food distribution centers, gas stations, accommodation places, veterinarians. |
| **Preconditions** | The information should be available on the sources before generation. |
| **Stimulus** | User clicks the "map" button on the page. |
| **Basic flow** | Step 1: The user clicks the "map" button<br>Step 2: The user is redirected to maps.afetbilgi.com<br>Step 3: The map is shown. |
| **Alternative flow** | Step 1: The user opens maps.afetbilgi.com. |
| **Exception flow** | - |
| **Post conditions** | - |

**Table 5:** Showing Maps Function

| | |
|---|---|
| **Use case name** | Changing Language |
| **Actors** | User, Static Website |
| **Description** | There are millions of people living in the affected areas belonging to different ethnicities and background, and although the main language of the site is Turkish, support for different languages is a must.<br><br>The project comes in four different languages which the users can choose from Turkish, English, Kurdish and Arabic. |
| **Data** | Visual messages |
| **Preconditions** | The translations are done previously. |
| **Stimulus** | User clicks the language button on the site |
| **Basic flow** | Step 1: The user clicks the language button<br>Step 2: A dropdown menu for language selection is shown<br>Step 3: The desired language is selected |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | The site is now in the desired language. |

**Table 6:** Changing Language Function

| | |
|---|---|
| **Use case name** | Reaching to depremyardim.com |
| **Actors** | User, Static Website |
| **Description** | The user wants to help the people affected by the earthquake. |
| **Data** | URL of the website. |
| **Preconditions** | - |
| **Stimulus** | User clicks the related button. |
| **Basic flow** | Step 1: The user hovers over the button.<br>Step 2: A description of the website is shown.<br>Step 3: The user clicks the button.<br>Step 4: The user is redirected to depremyardim.com. |
| **Alternative flow** | The user opens depremyardim.com . |
| **Exception flow** | - |
| **Post conditions** | The user is redirected to depremyardim.com. |

**Table 7:** Reaching to depremyardim.com Function

| | |
|---|---|
| **Use case name** | Reaching to afetharita.com |
| **Actors** | User, Static Website |
| **Description** | Afetharita.com is a website that provides map based information about the earthquake.<br>The map includes information about the earthquake, the aftershocks, the shelters, the hospitals, the schools and more. |
| **Data** | URL of the website. |
| **Preconditions** | - |
| **Stimulus** | User clicks the related button. |
| **Basic flow** | Step 1: The user hovers over the button.<br>Step 2: A description of the website is shown.<br>Step 3: The user clicks the button.<br>Step 4: The user is redirected to afetharita.com. |
| **Alternative flow** | The user opens afetharita.com . |
| **Exception flow** | - |
| **Post conditions** | The user is redirected to afetharita.com. |

**Table 8:** Reaching to afetharita.com Function

**Figure 4:** State Diagram of Reaching to afetharita.com

| | |
|---|---|
| **Use case name** | Reaching to deprem.io |
| **Actors** | User, Static Website |
| **Description** | Deprem.io is a website that users can use to help earthquake victims. |
| **Data** | URL of the website. |
| **Preconditions** | - |
| **Stimulus** | User clicks the related button. |
| **Basic flow** | Step 1: The user hovers over the button.<br>Step 2: A description of the website is shown.<br>Step 3: The user clicks the button.<br>Step 4: The user is redirected to deprem.io. |
| **Alternative flow** | The user opens deprem.io . |
| **Exception flow** | - |
| **Post conditions** | The user is redirected to deprem.io. |

**Table 9:** Reaching to deprem.io Function

| | |
|---|---|
| **Use case name** | Join the Discord server |
| **Actors** | User, Static Website |
| **Description** | The Discord server is where the developers of the project develop their projects and communicate.<br>Users can join the Discord server. |
| **Data** | Link for the Discord server. |
| **Preconditions** | - |
| **Stimulus** | User clicks the button. |
| **Basic flow** | Step 1: The user hovers over the button.<br>Step 2: A description of the Discord server is shown.<br>Step 3: The user clicks the button.<br>Step 4: The user is redirected to join the Discord server. |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | The user can join the Discord server after being redirected. |

**Table 10:** Join the Discord server Function

| | |
|---|---|
| **Use case name** | City Selection |
| **Actors** | User, Static Website |
| **Description** | Users may need information about only one city.<br>To eliminate unnecessary information<br>about other cities and focus on the desired city<br>could save time for users. |
| **Data** | Information about the given city |
| **Preconditions** | Information should be available beforehand. |
| **Stimulus** | User interacts with the dropdown menu. |
| **Basic flow** | Step 1: The user clicks the "Select a city" button.<br>Step 2: A dropdown menu is shown.<br>Step 3: The user selects the desired city.<br>Step 4: The information is filtered for the selected city. |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | All the information shown on the home page is<br>about the selected city. |

**Table 11:** City Selection Function

**Figure 5:** Sequence Diagram for City Selection Function

## 3.3 Usability Requirements

- A user shall use the system when a network connection is available.
- A user shall be able to find the needed information in at most 3 steps.
- All users shall be able to navigate through the system without basic computer knowledge.
- A user shall be able to use the system on any device and any browser.
- A user shall always be provided with the latest information.
- afetbilgi.com shall be available 24/7.
- afetbilgi.com shall be available in the 3 major languages spoken in Turkey plus English.

## 3.4 Performance Requirements

- All operations shall be completed in less than 3 seconds.
- Web GUI shall use less than 100 MB memory to ensure that application can be run from the majority of modern devices.
- Users shall be redirected to a required page such as another website or a form in 3 seconds after clicking a button.
- The user interface shall be interactive for users while another action is being performed.
- The GitHub actions workflow shall be able to continue functioning indefinitely (without any external factors) to ensure that automated plugins can run without problems.

## 3.5 Logical Database Requirements

There is no database.

## 3.6 Design Constraints

System concerns to serve users in a free and reliable way therefore open source hardware designs and open-source software development methods are chosen. Collaboration and contributions are encouraged to improve the system.

## 3.7 System Attributes

### 3.7.1 Reliability

- The system shall be able to provide reliable information to users.
- All of the hardware and software code of the system must be open-source.
- All the data sources must be verified and reliable.

### 3.7.2 Avaliability

- The site should be available for all platforms.
- The site should be available 24/7.
- In case of a system restart, the whole system shall be available in less than 5 minutes.
- The site should not consume big amounts of memory.

### 3.7.3 Security

- The system shall not store any user information.
- The system components shall be tested regularly to avoid zero-day attacks.

### 3.7.4 Maintainability

- Documentation shall be provided for all the components of the system.
- The system shall be able to be maintained by a single person.
- The tasks shall be divided into small and manageable parts.

- The automated tests shall be able to run on all the components of the system.

#### 3.7.5 Portability

- The system shall be able to be run on any device that has a web browser.
- Programming language which is chosen for the development of the system shall not depend on OS.
- Libraries that are used in the development shall be available for various programming languages.

# 4 Suggestions for Future Work

This system can be improved and held ready for future earthquakes and any possible disasters. New data sources can be added to the system. These data sources should better be verified and reliable. With reliable data sources and good programming interfaces, the need for manual data entry and human verification may be eliminated and the system can be made more reliable and quick.

## 4.1 System Perspective

## 4.2 External Interfaces

There are no external interfaces.

## 4.3 Functions

**Figure 6:** Use Case Diagram for Future Work

| | |
|---|---|
| **Use case name** | Get Data from Verified Information Source |
| **Actors** | User, Static Website, Data sources |
| **Description** | System fetches data from the verified information sources automatically. |
| **Data** | Information about gathering places, food distribution locations, shelters, gas stations etc. |
| **Preconditions** | - |
| **Stimulus** | - |
| **Basic flow** | Step 1: System fetches data from the verified information sources. |
| **Alternative flow** | The user directly visits the sources |
| **Exception flow** | - |
| **Post conditions** | - |

**Table 12:** Get Data from Verified Information Source Function

| | |
|---|---|
| **Use case name** | Submit Information |
| **Actors** | User, Static Website |
| **Description** | System allows users to submit information about the disaster situation. |
| **Data** | Information about gathering places, food distribution locations, shelters, gas stations etc. |
| **Preconditions** | User has significant information about the disaster. |
| **Stimulus** | - |
| **Basic flow** | Step 1: The user clicks the submit information button.<br>Step 2: The user enters the information.<br>Step 3: The user clicks the submit button.<br>Step 4: The system verifies the information. |
| **Alternative flow** | The user submits info to one of afetbilgi.com's sources |
| **Exception flow** | - |
| **Post conditions** | The information is submitted. |

**Table 13:** Submit Information Function

**Figure 7:** State Diagram for the Submit Information Function

| | |
|---|---|
| **Use case name** | Filter Information for Disaster |
| **Actors** | User, Static Website |
| **Description** | System filters the information according to a specific disaster. |
| **Data** | Information about gathering places, food distribution locations, shelters, gas stations etc. |
| **Preconditions** | User wants to learn about a given disaster. |
| **Stimulus** | - |
| **Basic flow** | Step 1: The user clicks the filter information button.<br>Step 2: The user picks the disaster.<br>Step 3: The user clicks the filter button.<br>Step 4: The system filters the information. |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | The information is filtered. |

**Table 14:** Filter Information Function

**Figure 8:** Sequence Diagram for the Filter Information Function

| **Use case name** | Information About What to Do |
|---|---|
| **Actors** | User, Static Website |
| **Description** | System provides information about what to do in a disaster situation for future disasters. |
| **Data** | Vital information about what should a person do in a case of a disaster such as an earthquake, flood etc. |
| **Preconditions** | User wants to learn what to do about a given disaster. |
| **Stimulus** | - |
| **Basic flow** | Step 1: The user clicks the learn button.<br>Step 2: The user is shown the info. |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | The information is shown. |

**Table 15:** Information About What to Do Function

| **Use case name** | Heat Map |
|---|---|
| **Actors** | User, Static Website |
| **Description** | System provides information a heat map showing the density of people who need help. |
| **Data** | Locations of people who need help. |
| **Preconditions** | A heat map is generated automatically. |
| **Stimulus** | - |
| **Basic flow** | Step 1: The user clicks the heat map button.<br>Step 2: The user is shown the heat map. |
| **Alternative flow** | - |
| **Exception flow** | - |
| **Post conditions** | The heat map is shown. |

**Table 16:** Heat Map Function

**Figure 9:** Activity Diagram for the Heat Map Function

## 4.4 Usability Requirements

- **User-friendly interface**: The system should have a better and more user-friendly interface. The interface should be easy to use and understand.
- **Easy to use**: The system should be easy to use. The user should be able to use the system without any prior knowledge.
- **Offline mode**: In disaster situations, the network can be down. The system should be able to work in offline mode. A PWA (Progressive Web App) can be used to make the system work offline. Native mobile apps can also be developed.

## 4.5 Performance Requirements

- **Low network usage**: The system should only transfer data when necessary. In disaster situations, the network may be restricted, the system should be able to work with limited network usage.
- **Fast response time**: In disaster situations, time is a crucial factor. The system should be able to respond quickly to the user's requests.

## 4.6 Logical Database Requirements

- **Database Schema**: The database schema should be designed in a way that it is easy to maintain and extend.
- **Database Security**: The database should be secure. The data should be encrypted and the database should be protected from unauthorized access.
- **Database Backup**: The database should be backed up regularly. This will make the system more reliable.

- **Database Migration**: The database should be able to be migrated to a different database system easily. This will make the system more reliable.
- **Database Contents**: The database can be used to store the data that is fetched from the data sources. The database can also be used to store the data that is generated by the system. Moreover, the database can be used to store the user's data.

## 4.7 Design Constraints

- **Different Repositories**: The source code of the system should be in different repositories. This will make the system more modular and easier to maintain.
- **Continuous Integration**: The system should be able to be tested automatically. This will make the system more reliable.
- **Used Packages**: The system should use packages that are well-maintained and have a good community.

## 4.8 System Attributes

### 4.8.1 Reliability

- The sources of the data should be verified and reliable. The system should be able to fetch data from these sources automatically.
- All of the data sources must be shown to the user.
- The process of filtering the data should be transparent to the user.

### 4.8.2 Avaliability

- The system should be able to work in offline mode, provided that the user has the app version of the system.

### 4.8.3 Security

- The data should be encrypted before being sent to the server.

### 4.8.4 Maintainability

- Automation tools like dependabot can be used to keep the system up to date.

### 4.8.5 Portability

- The system can be provided as a mobile app. This will make the system more available to use in disaster situations.