AUTUMN 2020 TERM PROJECT

Software Design Report

INSTRUCTOR: DOC. DR. MERT ÖZKAYA

TAHİR GÖRKEM ŞAKALAK

İLAYDA SUVARİOĞLU

SİMGE PINAR ERDOĞAN

Contents

1.Cover Page 1

2.Background 4

2.1 Existing Systems 4

2.2 Lack Of The Existing Systems 4

2.2.1 Platform addressing a single operating system 4

2.2.2 Internet Requirement 4

2.2.3 Failure to deliver the alert simultaneously 5

2.2.4 Few users appeal 5

3.Problem Statement 5

3.1 Problem Definition 5

3.2 Proposed System 5

3.3 Charts 6

4. Requirements Specification 8

4.1 Inability to meet the instant assistance needs of earthquake victims both offline and online 8

4.2 Failure to provide information on both offline and online earthquakes 8

4.3 Not giving awareness training before the earthquake 9

4.4 Ability of people to make their voices heard for specific earthquakes,public reach of resolved issues 9

4.5 Failure to contact the right people to resolve critical issues in the forum 9

4.6 Inability to prevent information pollution due to the lack of access to the campaigns conducted by people on a single platform 10

4.7 Failure to determine the needs of the society such as transportation, shelter, heating, food in the earthquake (task creation) and inability to supply 10

4.8 Inability to contact the right people during the supply of needs 10

4.9 Functional Requirements 11

4.9.1 Uml Use-Case Diagram 11

4.10 Non-Functional Requirements 12

4.10.1 Volere Template 12

5. Architectural Specification 14

5.1 XCD Software Architecture Modeling 15

6. Traceability Between Requirements and Architectures 19
6.1 Functional Requirements 19
6.1.1 SOS System 19
6.1.2 Offline Earthquake System 19
6.1.3 Awareness System 19
6.1.4 Forum System 19
6.1.5 Communication System 20
6.1.6 Campaign System 20
6.1.7 Task System 20
7. Model-To-Code Transformation Algorithm Specifications 21
8. Transformed Code 22
9. Mock-Ups 29
10.Lessons Learned 34
11.Conclusion.....34
12.References 35

2.BACKGORUND

2.1 EXISTING SYSTEMS

Until today, many applications related to natural disasters have been developed.Many applications have produced products for specific needs.

These can be listed as follows:

- Disaster Management
  - It provides life-saving information for pre- and post-disaster processes.

- Zello PTT Walkie Talkie
  - It offers low frequency radio application for communication during and after the disaster, so that we can reach the information that our family and loved ones are safe.

- Earthquake Alert
  - It sends a warning about the location and extent of the disaster during or after the disaster.

- PawBoost
  - It ensures that user-based content is created by specifying the location for the animals lost after a disaster to be delivered to their owners.

2.2 LACK OF THE EXISTING SYSTEMS

2.2.1 Platform addressing a single operating system

- Disaster Management

  Only appeals to Android users. This limits the target audience.

2.2.2 Internet Requirement

- Zello PTT Walkie Talkie

  In order to use the platform, at least 2G network is required. In case of disaster, the internet need may not always be met.

2.2.3. Failure to deliver the alert simultaneously

- Earthquake Alert

  The alert, including the size and location of the disaster, cannot always be reported simultaneously.There may be a delay of up to 5-8 minutes.

2.2.4. Few users appeal

- PawBoost

  Since it was developed for searching pets only, it appeals to fewer users.This situation reduces the awareness of the platform.

3.PROBLEM SPECIFICATION

3. 1 PROBLEM DEFINITION

Earthquakes are events that deeply affect social life.An earthquake zone as a road map does not have to help the community of citizens before the earthquake and after the earthquake in Turkey.

We can list it under the following subheadings:

- Failure to determine and supply critical needs of the society such as transportation, shelter, heating, food in case of an earthquake
- Not knowing the life-saving actions of earthquake victims due to the lack of awareness about what to do during the earthquake in the period before the earthquake
- Failure to prevent information pollution arising from different applications due to the lack of access to the campaigns carried out by people on a single platform.

3.2 PROPOSED SYSTEM

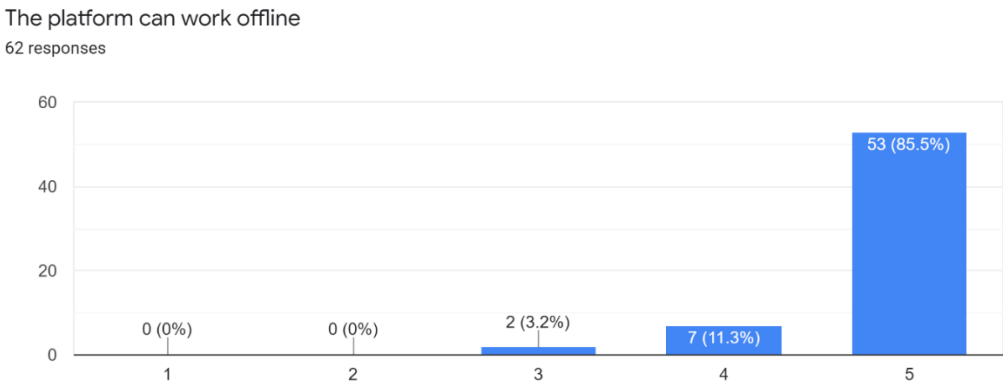
- It will be offered to users of two operating systems, Android and IOS.
- Depending on the functions of the application, it will operate both offline and online.It will operate offline to ensure the communication of the victims during and after the disaster.The systems that provide this are sos and eartquake alarm systems.
- Since it is an application that includes many disaster-related functions, it gathers the masses under the same platform.
- In this way, it appeals to a wide audience.Ability to meet the instant assistance needs of earthquake victims both offline and online
- Provide information on both offline and online earthquakes
- Giving awareness training before the earthquake

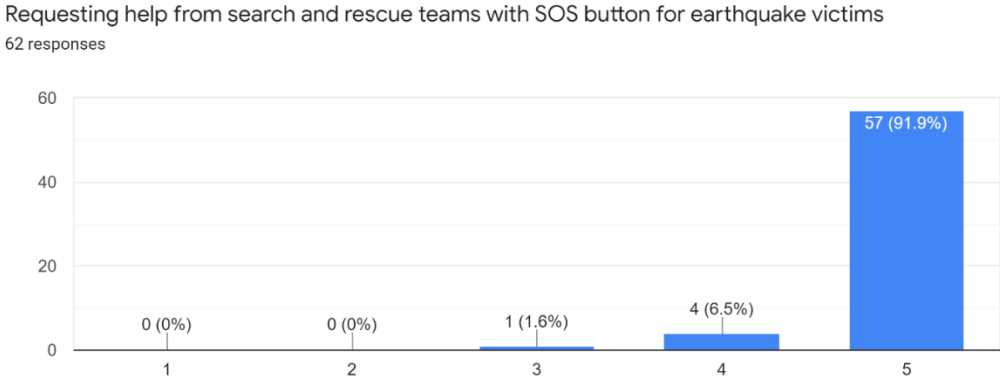
- Ability of people to make their voices heard for specific earthquakes, and public reach of solved issues
- Contact the right people to resolve critical issues in the forum
- Ability to prevent information pollution due to the lack of access to the campaigns conducted by people on a single platform
- Determine the needs of the society such as transportation, shelter, heating, food in the earthquake (task creation) and ability to supply
- Ability to contact the right people during the supply of needs

3.3 CHARTS

In order to get user opinions, we asked the public about the requirements determined by ourselves with certain questions.The following charts were created with the reponse from 62 people.

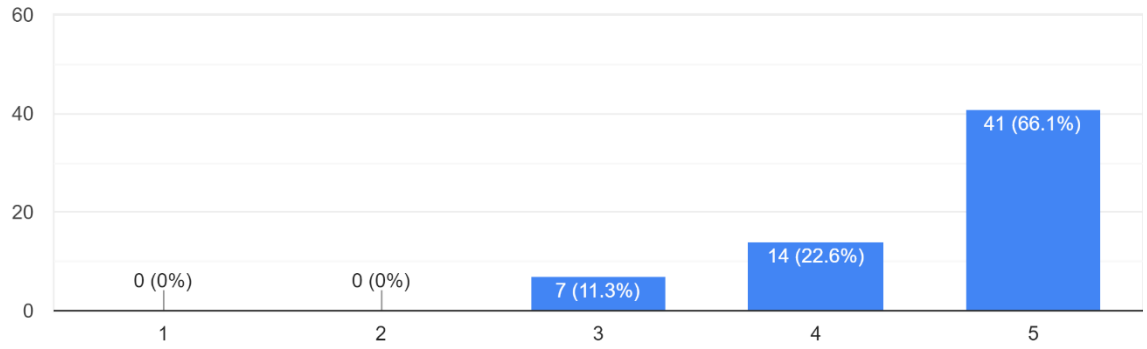
The x-axis of the charts is scored from 1 (lowest) to 5 (highest).The Y-axis also indicates the number of people.





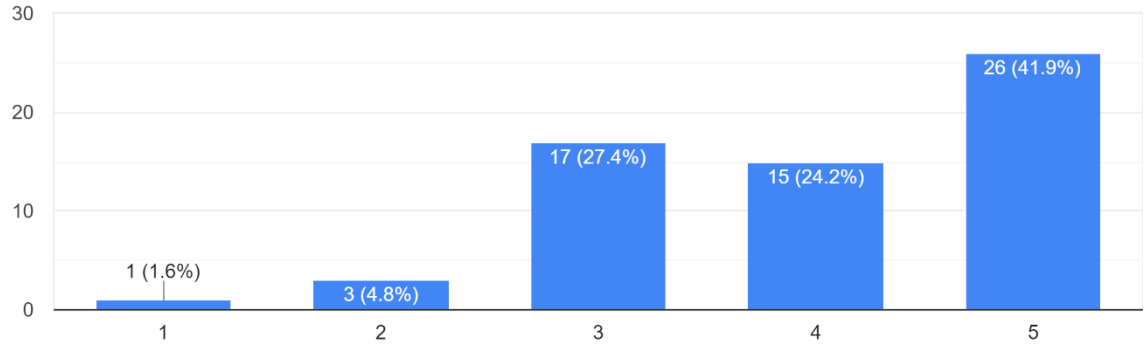
Supply of needs (transportation, heating, shelter, food, other) after an earthquake through assignments

62 responses



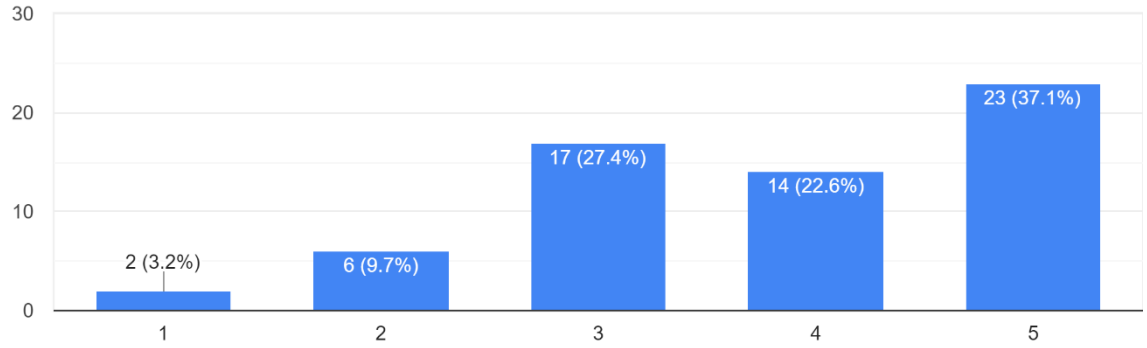
Providing awareness training before the earthquake

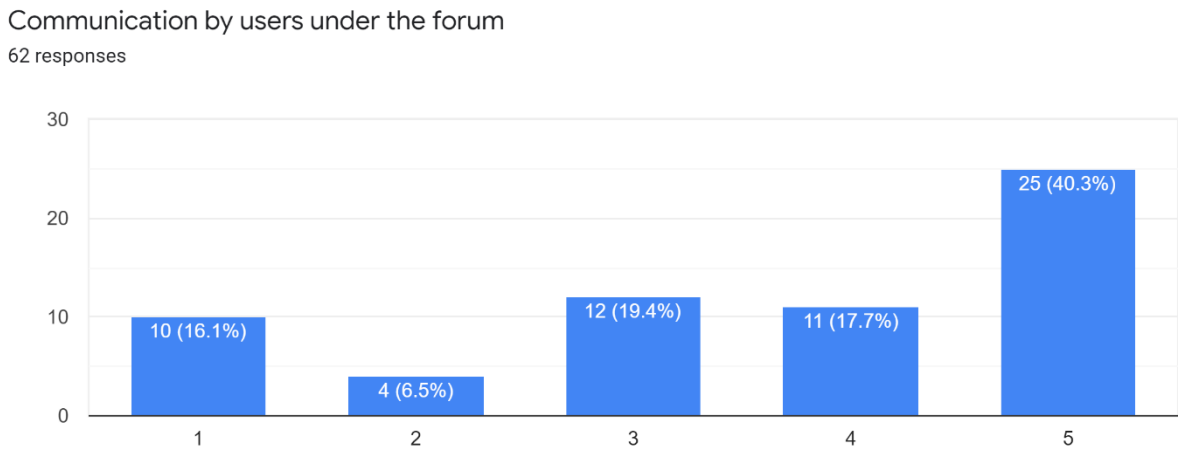
62 responses



Following the campaigns carried out by institutions after an earthquake on a single platform

62 responses





4. REQUIREMENTS SPECIFICATION

4.1 Inability to meet the instant assistance needs of earthquake victims both offline and online

The basic requirements for the solution of this problem are Bluetooth, Wifi, Multipeer Connectivity features.Support peer-to-peer connectivity and the discovery of nearby devices. Support peer-to-peer connectivity and the discovery of nearby devices.The basic working principle is to transmit information by creating an end-to-end network.These applications provide them to be operated using wifi and bluetooth.

Unlike other applications, the solution offered to this problem is the SOS system.In this system, with the wifi, bluetooth, multipeer connectivity infrastructure, which we considered as a requirement above, the users are presented as a notification to the systems of search-rescue teams such as AFAD and AKUT along with the personal information and navigation of the user requesting sos, offline and online.

4.2 Failure to provide information on both offline and online earthquakes

The basic requirement for the solution of this problem is Bluetooth, Wifi, Multipeer Connectivity features.Support peer-to-peer connectivity and the discovery of nearby devices.The basic working principle is to transmit information by creating an end-to-end network.These applications allow them to be operated using wifi and bluetooth.

Unlike other applications, the solution to this problem is the Offline Earthquake Alert system.In this system, with the wifi, bluetooth, multipeer connectivity infrastructure, which we considered as a requirement above, users can access the location, magnitude, date and time information of earthquakes with data from the Kandilli observatory, offline and online.

4.3 Not giving awareness training before the earthquake

The basic requirement for the solution of this problem is the platforms of professional institutions that share the process from the preparation of the earthquake bag to what to do after the earthquake with the public.

Unfortunately, in our country, pre-earthquake awareness raising cannot be done sufficiently and experience is gained mostly as a result of earthquakes.In the new system we have developed, it is to provide this awareness without having more bad experience.Unlike other applications, the solution to this problem is the awareness section.In this system, the platform of professional institutions, which we have accepted as a requirement above, provides users with critical information transfer.

4.4 Ability of people to make their voices heard for specific earthquakes, and public disclosure of resolved issues

For specific earthquakes, earthquake survivors and non-earthquake victims can find their relatives and pets they lost under the forum platform, make people in need of help make their voices heard, and inform the public about the issues and people that have been resolved

The basic requirement for the solution of this problem is the need to minimize information pollution and prevent the distribution of information on different platforms and to combine it on a single platform.Unlike other applications, the solution to this problem is the Forum system.In this system, the common platform, which we considered as a requirement above, provides users with the chance to find large audiences in the same environment.

4.5 Failure to contact the right people to resolve critical issues in the forum

The basic requirement for the solution of this problem is to minimize information pollution and prevent the distribution of information on different platforms, and for this, the need to combine it on a single platform.

Unlike other applications, the solution to this problem is the communication part of the Forum's system.In this system, the forum system, which we accept as a requirement above, enables users to connect directly with specific people related to the subject under their entries and to resolve the problems of the entries faster.

4.6 Inability to prevent information pollution due to the lack of access to the campaigns conducted by people on a single platform

The basic requirement for the solution of this problem is the need for many campaigns to be announced to more people by minimizing information pollution and financial losses and to combine them on a single platform.

Unlike other applications, the solution offered to this problem is the Campaigns system.In this system, it provides direct access to the campaigns hosted by institutions and organizations with the common platform that we considered as a requirement above.In this way, the number of people participating in the campaigns is supported, as the users can access the information of other campaigns besides the campaign they want to be included.

4.7 Failure to determine the needs of the society such as transportation, shelter, heating, food in the earthquake (task creation) and inability to supply

The basic requirement for the solution of this problem is the need to gather aid for earthquake victims in a single environment and to combine these aids under the name of assignment in order to categorize and facilitate their provision.

Unlike other applications, the solution to this problem is the assignment system.In this system, for the assignments that we consider as a requirement above, it is the completion of the task between the assisted and the assistants, thanks to the officers assigned by the institution.

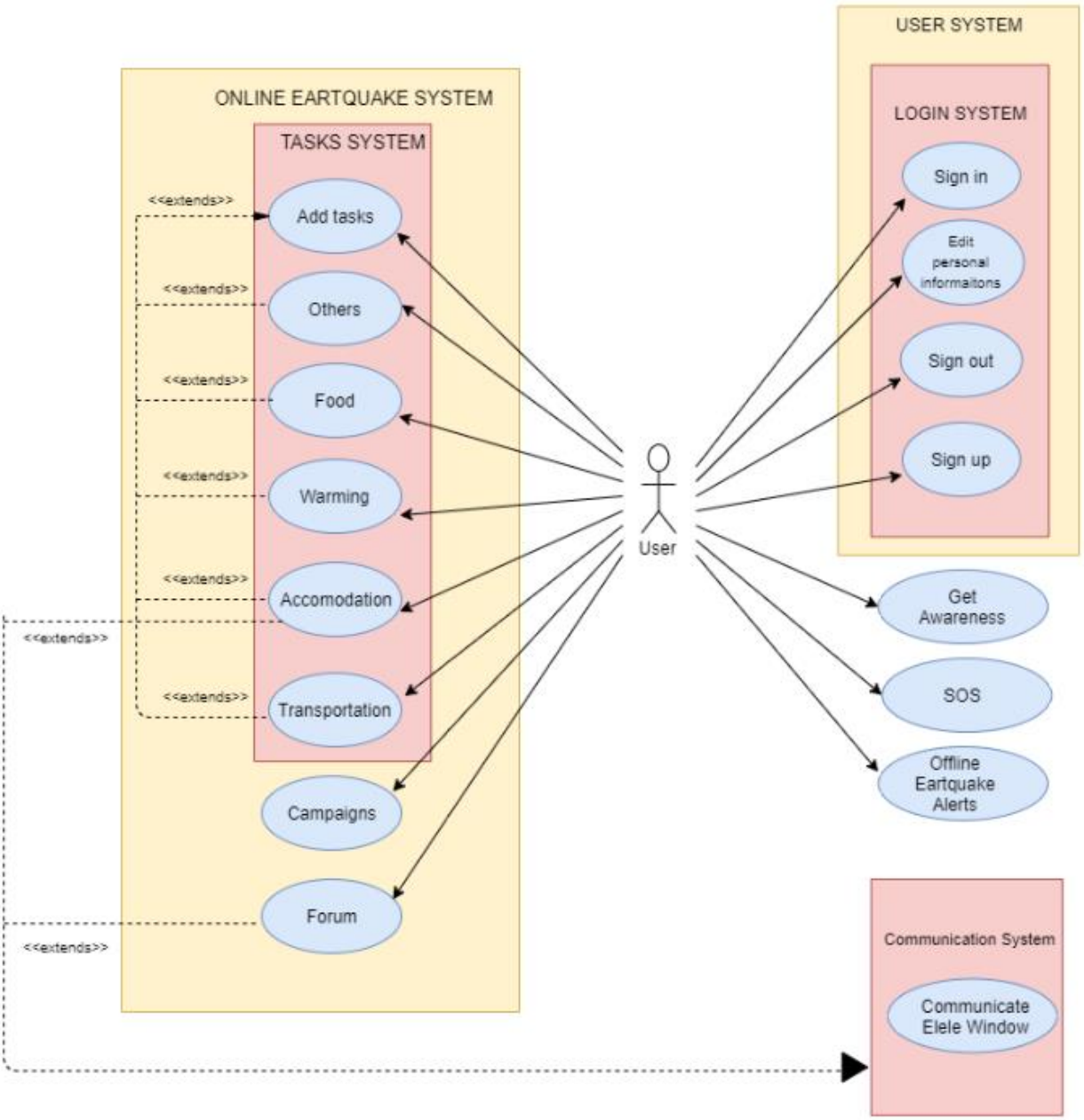
4.8 Inability to contact the right people during the supply of needs

The basic requirement for the solution of this problem is the need to gather aid for earthquake victims in a single environment and to combine these aids under the name of assignment in order to categorize and facilitate their provision.

Unlike other applications, the solution to this problem is the communication part of the assignment system.In this system, with the assignment system we considered as a requirement above, it enables those who request help to establish a reliable connection and find a solution directly under the task opened by them.

4.9 FUNCTIONAL REQUIREMENTS

4.9.1 Uml Use-Case Diagram



4.10 NON-FUNCTIONAL REQUIREMENTS

4.10.1 Volere Template

Usability

Description 1: Look and feel standards:

- Consistent layout and flow throughout the system
- Positioning the buttons in order of importance

Striking colors of the buttons

Fit Criteria 1: In case of an earthquake, earthquake victims need to reach help easily and quickly.

Rationale 1: In order to meet this criterion, the SOS button has been placed on the main page of the application and access to help has been accelerated.

Description 2: educational help about earthquake

Fit Criteria 2: Raising people's awareness about earthquakes is our focus

Rationale 2: Providing easy access to the awareness section, whether people are members of the application or not

Reliability

Description 1: Structuring the aid process of the institutions with the public

Fit Criteria 1: The institutional credibility of the aid process after the earthquake

Rationale 1: We have provided the aid network of the heating, food and other categories in the assignment department with the staff that the institution is responsible for.

Performance

Description 1: Response times:

- Application loading not more than 3 seconds at most
- Screen open and refresh time not more than 3 seconds at most.

Fit Criteria 1: System performance will be examined every day to make sure there are no delays at any given time for any process made.

Rationale 1: This is important because in order to achieving the highest level of output, the application loading time and screen refresh rate should not take longer than it should.

Maintainability

Description 1: On system downtime, repairs should not exceed 3 hours maintenance when restoring the system.

Fit Criteria 1: System will be tested every minute to avoid any failures in the process of restoration.

Rationale 1: This is essential as there should be a smooth recovery operation on the system to ensure that it will be up and again as soon as possible.

Reliability

Description 1: Mean time between Failures expected no more than twice a month.
Fit Criteria 1:
Rationale 1: This is essential, as it will allow users to use the system smoothly without any system breakdowns.

Capacity

Description 1: The network will accommodate over 1000 simultaneous users.
Fit criteria 1: Every day the system's performance should be checked to ensure that it can satisfy the amount of users logged in at one time without causing the system to slow down.
Rationale 1: This is useful in the system if every member of staff is logged into the system all at once, the system should still perform to a very high and efficient level.

Description 2: The system should be able to handle over 1GB amount of data stored a day.
Fit criteria 2: Every day the data stored in the system is examined to check that it can handle over 1GB of data stored.
Rationale 2: This is useful as it keeps tracks of how much data flows within the system.

Security

Description 1: ELELE should allow for a restricted level of access:
Fit Criteria 1: System will be tested every day to ensure that each member of staff has the correct login details.
Rationale 1: This is essential as each member of staff have their own access right.
Description 2: Password Encryption to log-in details:

- No restriction to password length or characters.
- Can contain numbers and symbols.

Fit Criteria 2: For each account created passwords can be of any length and any character.
Rationale 2: This gives users the flexibility as he or she have the ability to chose their own personal passwords.

Privacy

Description 1: Customer Loyalty:

- Customer details are to be kept private
- Details of customer are to be utilized only when necessary

Fit Criteria 1: This principle will be maintained and revised every day to insure customers details are kept safe and not leaked without authorization.
Rationale 1: This requirement is essential to gain customer loyalty and trust at a professional standard. Their details have to be kept private and only used when necessary as part of company policy.

5. ARCHITECTURAL SPECIFICATION

We have modeled the ELELE application, which is an earthquake platform with classical architecture, for N users.The user reaches the Sos button for help on the homepage without having to login.There is a consumer port in the showInfo component, which creates a click event via the emitter port in the Sos component and receives the information to ask for help.The user reaches the earthquake alarm button for earthquake notifications on the homepage without having to login.It creates the click event through the port emitter in the OfflineEarth component.This is another consumer port in the showInfo component listening to this event.

The user reaches the awareness button to raise earthquake awareness on the homepage without having to login.It creates the click event via the emitter port in the GetAware component.This is another consumer port in the showInfo component listening to this event.The user must access the module in the usersystem in order to register.For this, a method call is created with the required port in the SignUp component.There is provided port in SignIn component for returning this call.In order for the logged in user to log out, the required port in the SignIn component requests two way communication.The answer to this request is provided by the provided port in the SignOut component.Creates an event to update the logged-in user personal information with the emitter port of the EditInfo component.This asynchronous communication is listening with the consumer port of showInfo component.

Logged in users can access tasks (transportation, accomodation, food, warming, others) for the specific earthquake.For this, it is the consumer port in showInfo component that establishes one way communication with the emitter port in the Task component. The user can use the add button to add a new task.This two way communication is set up between the required port of the Add component and the provided port of the task component.The user clicks on the Hand in Hand button to communicate with the person responsible for the task they want to help.Thus, there is a synchronous communication between the required port of the Task component and the provided port of the Communication component.

The logged in user can access the campaigns created to help with specific earthquakes.These campaigns create an event on the emitter port of the Campaign component. This event is listening for the consumer port of the showInfo component.Logged in users can access the forum platform, which facilitates the communication of other users with each other.For this, it creates an event with the emitter port of the Forum component.This is the consumer port of the showInfo component listening for the event.At the same time, users can communicate with each other directly in the entries in the forums.This is provided between the reuquired port of the Forum component and the provided port of the Communication component.

5.1 XCD SOFTWARE ARCHITECTURE MODEL

```
enum User := {None,Single,Pair};
enum TaskName := {Tr,Ac,Wa,Fo,Ot};
enum Campaign := {Off,On};
enum Task := {Off,On};
enum Info := {Off,On};
enum Forum := {Off,On};

typedef byte ID;


//COMPONENT ShowInfo

component ShowInfo(){

  bool eleleClick := false;
  User chosenUser :=  Single;

  consumer port TaskElele{
    @interaction{
      waits: ! eleleClick;
    }
    @functional{
      ensures:  eleleClick := true;
                chosenUser[task]:= user ;
    }
      validTaskInfo(Task task,User user);
}


  consumer port CampaignElele{
    @interaction{
      waits: ! eleleClick;
    }
    @functional{
      ensures:  eleleClick := true;
                chosenUser := campaign[@];
    }
    validCampaign(Campaign campaign);
}

  consumer port ForumElele{
    @interaction{
      waits: ! eleleClick;
    }
    @functional{
      ensures:  eleleClick := true;
                chosenUser := forum[@];
    }
    validEntry(Forum forum);
}

  consumer port InfoElele{
    @interaction{
      waits: ! eleleClick;
    }
    @functional{
      ensures:  eleleClick := true;
                chosenUser := info[@];
    }
    validInfo(Info info);
}

  consumer port AwareElele{
    @interaction{
      waits: ! eleleClick;
    }
    @functional{
      ensures:  eleleClick := true;
                chosenUser := user[@];
    }
   validAware(User user);
}

  consumer port SosElele{
    @interaction{
      waits: ! eleleClick;
    }
    @functional{
      ensures:  eleleClick := true;

    }
   validSos();
}

  consumer port OfflineElele{
    @interaction{
      waits: ! eleleClick;
    }
    @functional{
      ensures:  eleleClick := true;

    }
   validOffline();
}


}
```

```
//COMPONENT TASK

component Task(){

  bool requestClick := false;
  bool isTaskAvailable := false;
  User chosenUser :=  Single,Pair;
  Task noTask[N] := None;
  bool chosenTask := On;

  emitter port ShowElele{
      Task TaskType[3] := Wa,Fo,Ot;
      @interaction{
      waits: !requestClick;
      }
      @functional{
      promises: task;
                user;
      ensures:  requestClick := true;
                chosenTask[task] := user;
      }
      validTaskInfo(Task task,User user);
  }


  provided port showTask{
    Task TaskType[5] := Tr,Ac,Wa,Fo,Ot;
    @interaction{
      accepts: requestClick==true;
    }
    @functional{
    requires: requestClick != None;
    ensures: isTaskAvailable[@]:=true;
         \result := taskNum[@];
    otherwise
    requires: requestClick == None;
    throws: NoTaskException;
    }
    Task validTask() throws NoTaskException;
  }

  required port requestCommiTask{
    Task TaskType[2] := Tr,Ac;
    bool requestClick := false;
    @interaction{
      waits: requestClick==true;
    }
    @functional{
      promises: \nothing;
         requires: \result == TaskName[@];
         ensures: requestClick:=false;
         otherwise:
         requires: !(\result == TaskName[@]);
         ensures: \nothing;
         otherwise:
         requires: !(\exception == NoTaskEleleException);
         ensures: \nothing;
    }
    Task validTaskElele() throws NoTaskEleleException;
}
```

```
//COMPONENT ADD

  component Add( ){

      required port clickAdd{
        bool requestAdd := false;
        @interaction{
          waits: requestAdd==true;
      }
      @functional{
        promises: \nothing;
           requires: \result == TaskName[@];
           ensures: requestAdd:=false;
           otherwise:
           requires: !(\result == TaskName[@]);
           ensures: \nothing;
           otherwise:
           requires: !(\exception == NoTaskException);
           ensures: \nothing;
      }
      Task validTask() throws NoTaskException;
  }
}
```

```
//COMPONENT CAMPAIGN

  component Campaign( ){

    bool campaignClick := false;
    bool chosenCampaign := On;
    User chosenUser :=  Single;

    emitter port showCampaign{
      @interaction{
      waits: !campaignClick;
      }
      @functional{
      promises: campaign;
      ensures:  campaignClick := true;
                chosenCampaign := campaign[@];
      }
      validCampaign(Campaign campaign);
  }

}
```

```
//COMPONENT EDITINFO

component EditInfo( ){
    bool infoClick := false;

    emitter port showInfo{
      @interaction{
      waits: !infoClick;
      }
      @functional{
      promises: info;
      ensures:  infoClick := true;
                chosenInfo := info[@];
      }
      validInfo(Info info);
    }
}
```

```
//COMPONENT GETAWARE

  component GetAware( ){
    User chosenUser :=  Single;
    bool awareClick := false;

    emitter port Aware{
      @interaction{
      waits: !awareClick;
      }
      @functional{
      promises: user;
      ensures:  awareClick := true;
                chosenUser := user[@];
      }
      validAware(User user);
    }
}
```

```
//COMPONENT FORUM

  component Forum(){
    bool forumClick := false;
    Forum chosenForum := On;
    User chosenUser :=  Single;

    emitter port showEntry{
      @interaction{
      waits: !forumClick;
      }
      @functional{
      promises: forum;
      ensures:  forumClick := true;
                chosenForum:= forum[@];
      }
      validEntry(Forum forum);
  }

  required port requestCommiForum{
    bool requestClick := false;
    Forum chosenForum := On;
    @interaction{
      waits: requestClick==true;
    }
    @functional{
      promises: \nothing;
         requires: !(\result == chosenForum[@]);
         ensures: \nothing;
         otherwise:
         requires: (\result == chosenForum[@]);
         ensures: requestClick:=false;
         otherwise:
         requires: !(\exception == NoForumEleleException);
         ensures: \nothing;
    }
    Forum validForumElele() throws NoForumEleleException;
    }
}
```

```
//COMPONENT COMMUNICATION+

component Comminucation(){
  bool requestClick := false;
  bool isCommiAvailable := false;

  provided port showCommiTask{
    Task TaskType[2] := Tr,Ac;
      @interaction{
        accepts: requestClick==true;
      }
      @functional{
      requires: requestClick != None;
      ensures: isCommiAvailable[@]:true;:;
             \result := commiNum[@];
        otherwise
        requires: requestClick == None;
        throws: NoTaskEleleException;
      }
    Task validTaskElele() throws NoTaskEleleException;
  }

  provided port showCommiForum{
    @interaction{
      accepts: requestClick==true;
    }
    @functional{
    requires: requestClick != None;
    ensures: isCommiAvailable[@]:= true;
         \result := commiNum[@];
    otherwise
    requires: requestClick == None;
    throws: NoForumEleleException;
    }
  Forum validForumElele() throws NoForumEleleException;
  }
}
```

```
//COMPONENT SOS

component Sos(){
  bool sosClick := false;

  emitter port GetSos{
    @interaction{
    waits: !sosClick;
    }
    @functional{
    promises: \nothing;
    ensures:  sosClick := true;
    }
    validSos();
  }
}
```

```
//COMPONENT OFFLINE EARTHQUAKE

component OfflineEarth(){

  bool offlineClick := false;

  emitter port Offline{
    @interaction{
    waits: !offlineClick;
    }
    @functional{
    promises: \nothing;
    ensures:  offlineClick := true;

    }
    validOffline();
  }
}
```

```
//COMPONENT SIGNIN

component SignIn(){

  bool requestClick := false;
  User chosenUser := Single;
  bool isSignAvailable := false;
  bool currentUser := true;
  bool currentPassword := true;

  provided port receiveSignIn{
    @interaction{
      accepts: requestClick==true;
    }
    @functional{
    requires: currentUser;
              currentPassword;
    ensures: isSignAvailable[@]:= false;
         \result := userId[@];
    otherwise:
    throws: AlreadyException;
    }
  User validUser() throws AlreadyException;
  }

  required port requestSignout{
    @interaction{
      waits: requestClick==true;
    }
    @functional{
      promises: \nothing;
         requires: !(\result == chosenUser[@]);
         ensures: \nothing;
         otherwise:
         requires: (\result == chosenUser[@]);
         ensures:  requestClick := true;
                   chosenUser := userId[@];
         otherwise:
         requires: !(\exception == NoSignInException);
         ensures: \nothing;
    }
    User validSignUser() throws NoSignInException;
    }

}
```

```
//COMPONENT SIGNOUT

component SignOut(){

  bool requestClick := false;
  User chosenUser[2] := Single,None;
  bool isSignInAvailable := true;
  bool currentUser := true;

  provided port receiveSignout{
    @interaction{
      accepts: requestClick==true;
    }
    @functional{
    requires: isSignInAvailable;
    ensures: chosenUser[@] := None ;
    otherwise:
    throws: NoSignInException;
    }
    User validSignUser() throws NoSignInException;
  }
}
```

```
//COMPONENT SIGNUP

component SignUp(){

  required port requestSignUp{
    bool requestClick := false;
    User chosenUser := Single;
    @interaction{
      waits: requestClick==true;
    }
    @functional{
      promises: \nothing;
         requires: !(\result == chosenUser[@]);
         ensures: \nothing;
         otherwise:
         requires: (\result == chosenUser[@]);
         ensures:  requestClick := true;
                   chosenUser := userId[@];
         otherwise:
         requires: !(\exception == AlreadyException);
         ensures: \nothing;
    }
    User validUser() throws AlreadyException;
  }

}
```

6.TRACEABILITY BETWEEN REQUIREMENTS AND ARCHITECTURES

6.1 Functional Requirements:

6.1.1. SOS System

There is an emitter port in the sos component.The value of the boolean type sosClick in the sos component is defined as false by default. As the communication protocol, sosClick expects the boolean to be true. When osClick returns true, it calls the validSos () method.

6.1.2 Offline Earthquake System

There is an emitter port in the Offline Earthquake Alert component. The value of offlineClick of boolean type in the OfflineEarthquake component is defined as false by default.As the communication protocol, offlineClick expects the boolean to be true.When offlineClick returns true, it calls the validOffline () method.

6.1.3 Awareness System

There is an emitter port in the GetAware component. The value of awareClick of boolean type in GetAware component is defined as false by default. The default index of the User array for GetAware component is defined as Single. As a communication protocol, awareClick expects the boolean to be true and assigns the user parameter to promises. When awareClick returns true, it calls the validAware (User user) method.

6.1.4 Forum System

There is an emitter port in the forum component.The value of awareClick of boolean type in the forum component is defined as false by default.For the Forum component, the default index of the User array is defined as Single and the default index of the Forum array is On.The value of forumClick of boolean type in the forum component is defined as false by default.As the communication protocol, forumClick expects the boolean to be true and assigns the forum parameter to promises. When forumClick returns true it calls the validEntry (Forum forum) method.

There is also a required port in the forum component. For the Forum component, the default index of the Forum array is defined as On.The value of requestClick of boolean type in the forum component is defined as false by default.As the communication protocol, requestClick expects the boolean to be true.If the chosenForum does not exist, nothing is done, but if it does, requestClick is returned false and it waits until it becomes true again.This shows the communication protocol.

6.1.5 Communication System

There are 2 provided ports in the communication component.In the first, the value of the boolean type requestClick in the Forum component is false by default and isCommiAvailable in the boolean type is false by default. As the communication protocol, requestClick expects the boolean to be true.

The first provided port is defined for the transportation and accomodation indexes.As the communication protocol, requestClick expects the boolean to be true and determines the functional contruct as follows:The value of isCommiAvailable for the current task is true unless the requeastClick is None.It returns commiNum for the current task or sends it to NoTaskEleleException as long as requestClick is None.

As the communication protocol for the second provided port, requestClick expects the boolean to be true and sets the functional configuration as follows:The value of isCommiAvailable for the current forum is true unless requeastClick is None.It returns commiNum for the current forum or sends it to NoForumEleleException as long as requestClick is None.

6.1.6 Campaign System

There is an emitter port in the Campaign component.In the Campaign component, the value of campaignClick of boolean type is false by default, the default value of chosenCampaign of type boolean is On, and the default index of the User array is defined as Single.As the communication protocol, campaignClick expects the boolean to be true and assigns the campaign parameter to promises.When campaignClick returns true, it calls the validCampaign (Campaign campaign) method.

6.1.7 Task System

There is an emitter port in the Task component.In the Task component, the value of requestClick of type boolean is false by default, the default value of isTaskAvailable of type boolean is On, the default value of chosenTask of type boolean is On, the default index of the User array is Single-Pair, and the default index of the Task array is None.

Emitter port is defined for Warming, Food, Others indexes. As the communication protocol, requestClick expects the boolean to be true.It assigns task and user parameters to promises.requestClick returns true.The chosenTask value is synchronized to the task where the user is.It then calls the validTaskInfo (Task task, User user) method.

Task component's only provided port is defined for all indexes of the task array.As the communication protocol, requestClick expects the boolean to be true and the functional contract determines as follows:The value of isTaskAvailable for the current task is true unless requeastClick is None.It returns taskNum for the current task or sends it to NoTaskException as long as requestClick is None.Task component also has a required port.

For the Task component, it is defined for the transportation and accommodation indexes of the Task array.The value of requestClick of type boolean in the Task component is

defined false by default.As the communication protocol, requestClick expects the boolean to be true.If the task does not exist, nothing is done, but if it does, requestClick is returned false and it waits until it becomes true again.This shows the communication protocol.If the exception is not happening, still nothing should be done.

7. MODEL-TO-CODE TRANSFORMATION ALGORITHM SPECIFICATION

We have used interfaces to provide multiple inheritances .These interfaces are ShowInfo,Comminucation,Add.These interfaces implemented by some classes which used functions.Based on this,other components which will be included later ,can also take properties from related interfaces.

Abstract classes are generally used to gather objects with common properties under one roof. Abstract classes that we appended are Task,CheckUserInfo. An abstract class can be extended using keyword "extends".An interface can be implemented using keyword "implements".

The pseudo code of the XCD software model is given below:

8. TRANSFORMED CODE

```
User = [None,Single,Pair]
TaskName = [Tr,Ac,Wa,Fo,Ot]
Campaign = [Off,On]
Task = [Off,On]
Info = [Off,On]
Forum = [Off,On]

byte ID

abstract class Task {
   function ShowElele()
  function showTask()
  function requestCommiTask()
}

abstract class CheckUserInfo {
   function requestSignUp()
   function receiveSignIn()
   function requestSignout()
   function receiveSignout()
}

interface ShowInfo(){
  function show()
}

interface Comminucation(){
  function communicate()
}

interface Add{
  function clickAdd()
}
```

```
class Food EXTENDS Task IMPLEMENTS Add {
  function ShowElele(){
    WHILE requestClick is not False:
      requestClick is true
      call validTaskInfo(Task,User)
    END WHILE
  }
  function validTaskInfo(){
    isTaskAvailable is True
  }
  function show(){
    WHILE requestAdd is True:
    IF currentTask is Food
      requestAdd is False
    END IF
    ELSEIF currentTask is not Add
      do nothing
    END ELSEIF
    ELSE
      throw NoTaskException
    END ELSE
  }
  function showTask(){
    WHILE requestClick is not False:
      IF requestClick is not None then
        show currentTask
      END IF
      ELSE
        throw NoTaskException
      END ELSE
    END WHILE
  }
}
```

```
class Warming EXTENDS Task IMPLEMENTS Add{
  function ShowElele(){
    WHILE requestClick is not False:
      requestClick is true
      call validTaskInfo(Task,User)
    END WHILE
  }
  function validTaskInfo(){
    isTaskAvailable is True
  }
  function show(){
    WHILE requestAdd is True:
    IF currentTask is Food
      requestAdd is False
    END IF
    ELSEIF currentTask is not Add
      do nothing
    END ELSEIF
    ELSE
      throw NoTaskException
    END ELSE
  }
  function showTask(){
    WHILE requestClick is not False:
      IF requestClick is not None then
        show currentTask
      END IF
      ELSE
        throw NoTaskException
      END ELSE
    END WHILE
  }
}
```

```
class Others EXTENDS Task IMPLEMENTS Add {
  function ShowElele(){
    WHILE requestClick is not False:
      requestClick is true
      call validTaskInfo(Task,User)
    END WHILE
  }
  function validTaskInfo(){
    isTaskAvailable is True
  }
  function show(){
    WHILE requestAdd is True:
    IF currentTask is Food
      requestAdd is False
    END IF
    ELSEIF currentTask is not Add
      do nothing
    END ELSEIF
    ELSE
      throw NoTaskException
    END ELSE
  }
  function showTask(){
    WHILE requestClick is not False:
      IF requestClick is not None then
        show currentTask
      END IF
      ELSE
        throw NoTaskException
      END ELSE
    END WHILE
  }
}
```

```
class Transportation EXTENDS Task IMPLEMENTS ShowInfo,Add,Comminucation {
  function showTask(){
    WHILE requestClick is not False:
      IF requestClick is not None then
        show currentTask
      END IF
      ELSE
        throw NoTaskException
      END ELSE
    END WHILE
  }
  function validTaskInfo(){
    WHILE eleleClick is not False:
      show validTaskInfo
    END WHILE
    }
    function show(){
      WHILE requestAdd is True:
      IF currentTask is Food
        requestAdd is False
      END IF
      ELSEIF currentTask is not Add
        do nothing
      END ELSEIF
      ELSE
        throw NoTaskException
      END ELSE
    }
    function communicate{
      WHILE requestClick is True:
        IF requestClick is not None then
          start Commi
        END IF
        ELSE
          throw NoTaskEleleException
        END ELSE
      END WHILE
    }
    function requestCommiTask(){
      WHILE requestClick is not False:
      IF currentTask is Transportation
        requestClick is False
      END IF
      ELSEIF currentTask is not Transportation
        do nothing
      END ELSEIF
      ELSE
        throw NoTaskEleleException
      END ELSE
  }
}
```

```
class Accomodation EXTENDS Task IMPLEMENTS ShowInfo,Add,Comminucation{
  function showTask(){
    WHILE requestClick is not False:
      IF requestClick is not None then
        show currentTask
      END IF
      ELSE
        throw NoTaskException
      END ELSE
    END WHILE
  }
  function validTaskInfo(){
  WHILE eleleClick is not False:
    show validTaskInfo
  END WHILE
  }
  function show(){
    WHILE requestAdd is True:
    IF currentTask is Food
      requestAdd is False
    END IF
    ELSEIF currentTask is not Add
      do nothing
    END ELSEIF
    ELSE
      throw NoTaskException
    END ELSE
  }
  function communicate{
    WHILE requestClick is True:
      IF requestClick is not None then
        start Commi
      END IF
      ELSE
        throw NoTaskEleleException
      END ELSE
    END WHILE
  }
  function requestCommiTask(){
    WHILE requestClick is not False:
    IF currentTask is Accomodation
      requestClick is False
    END IF
    ELSEIF currentTask is not Accomodation
      do nothing
    END ELSEIF
    ELSE
      throw NoTaskEleleException
    END ELSE
  }
}
```

```
class Campaign IMPLEMENTS ShowInfo{
  function showCampaign(){
    WHILE campaignClick is not False:
      campaignClick is true
      call validCampaign(Campaign)
    END WHILE
  }
  function validCampaign(){
    chosenCampaign is On
  }
  function show(){
    WHILE campaignClick is True
       show currentCampain
    END WHILE
  }
}
```

```
class EditInfo IMPLEMENTS Show{

  function show()
  {
    WHILE infoClick is True
       show currentInfo
      IF currentInfo is change then
        click SaveButton
      END IF
      ELSE
        do nothing
      END ELSE
    END WHILE

  }
}
```

```
class SignUp EXTENDS CheckUserInfo{
  function requestSignUp(){
    WHILE requestClick is True:
    IF currentUser is not chosenUser
      do nothing
    END IF
    ELSEIF currentUser is chosenUser
      chosenUser is equal to userId
    END ELSEIF
    ELSE
      throw AlreadyException
    END ELSE
  }
}
```

```
class Sos IMPLEMENTS ShowInfo{
  function GetSos(){
    WHILE sosClick is True
       show currentSos
       call validSos()
    END WHILE
  }
  function validSos(){
    currentSos is True
  }
  function show(){
    WHILE sosClick is True
       show currentSos
    END WHILE
  }
}
```

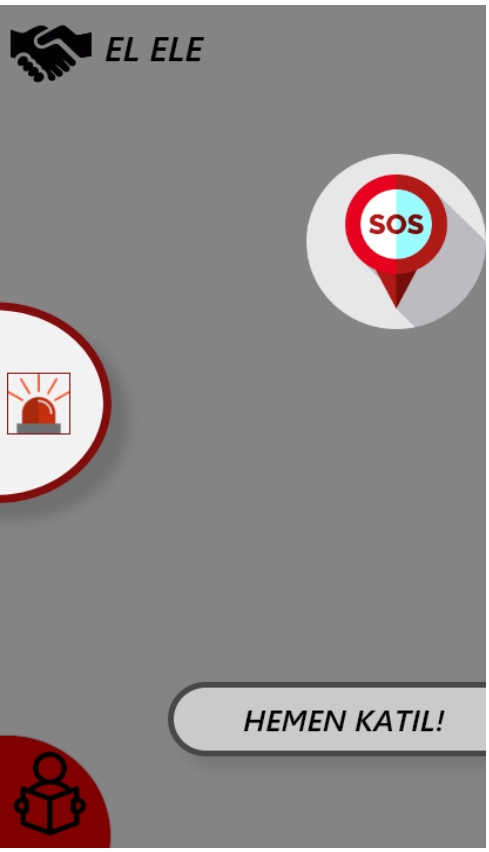
```
class Forum IMPLEMENTS ShowInfo,Comminucation{
  function showEntry(){
    WHILE forumClick is not False:
      forumClick is true
      call validEntry(Forum)
    END WHILE
  }
  function validEntry(){
    chosenForum is On
  }
  function show(){
    WHILE forumClick is True
       show currentForum
    END WHILE
  }
  function communicate{
    WHILE requestClick is True:
      IF requestClick is not None then
        start Commi
      END IF
      ELSE
        throw NoForumEleleException
      END ELSE
    END WHILE
  }
  function requestCommIForum(){
    WHILE requestClick is not False:
    IF currentForum is On
      requestClick is False
    END IF
    ELSEIF currentForum is not On
      do nothing
    END ELSEIF
    ELSE
      throw NoForumEleleException
    END ELSE
}
}
```

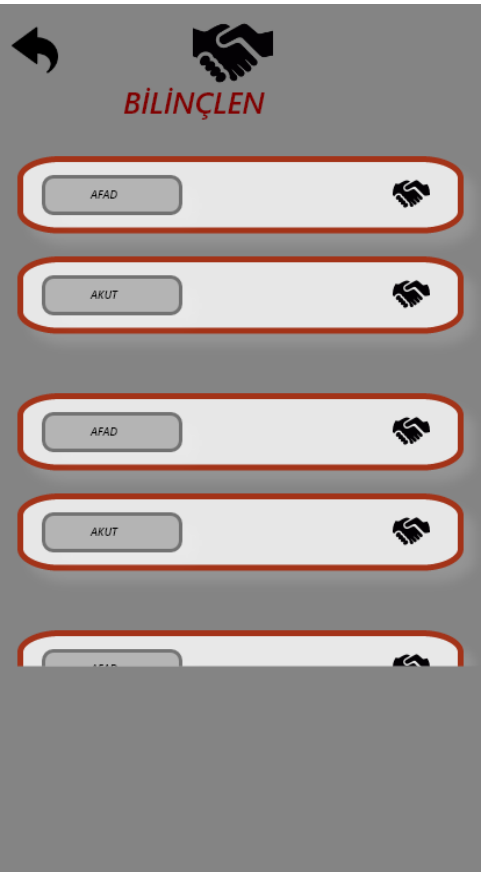
```
class OfflineEarth IMPLEMENTS ShowInfo{
function Offline(){
  WHILE offlineClick is True
     show currentAlert
     call   validOffline()
  END WHILE
}
function validOffline(){
  currentAlert is True
}
function show(){
  WHILE offlineClick is True
     show currentAlert
  END WHILE
}
}
```

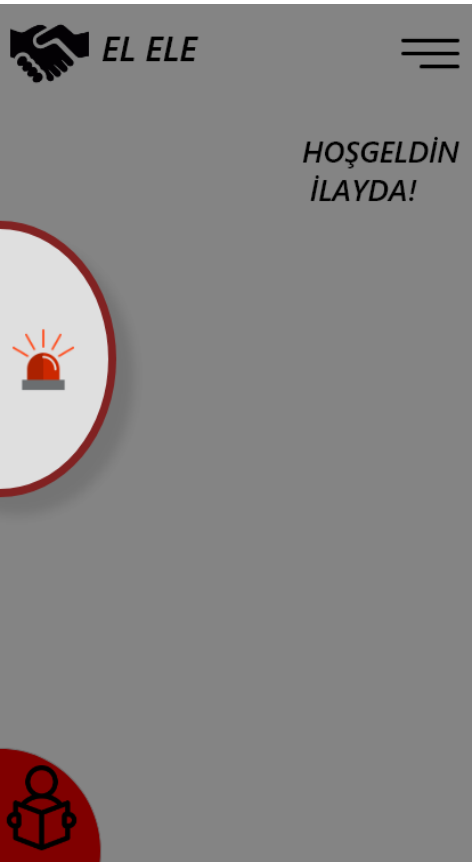
```
class GetAware IMPLEMENTS Show{
  function show(){
    WHILE awareClick is True
       show currentAware
    END WHILE
  }
}
```

```
class SignIn EXTENDS CheckUserInfo{
  function receiveSignIn(){
    WHILE requestClick is True:
    IF currentUser is chosenUser AND currentPassword is chosenPassword
      chosenUser is equal to userId
    END IF
    ELSEIF currentUser is not chosenUser OR currentPassword is not chosenPassword
      chosenUser is not equal to userId
    END ELSEIF
    ELSE
      throw AlreadyException
    END ELSE
  }
  function requestSignout(){
    WHILE requestClick is True:
    IF currentUser is chosenUser
      chosenUser is equal to userId
    END IF
    ELSEIF currentUser is chosenUser
      do nothing
    END ELSEIF
    ELSE
      throw NoSignInException
    END ELSE
}
}
```

9. SCREEN MOCK -UPS

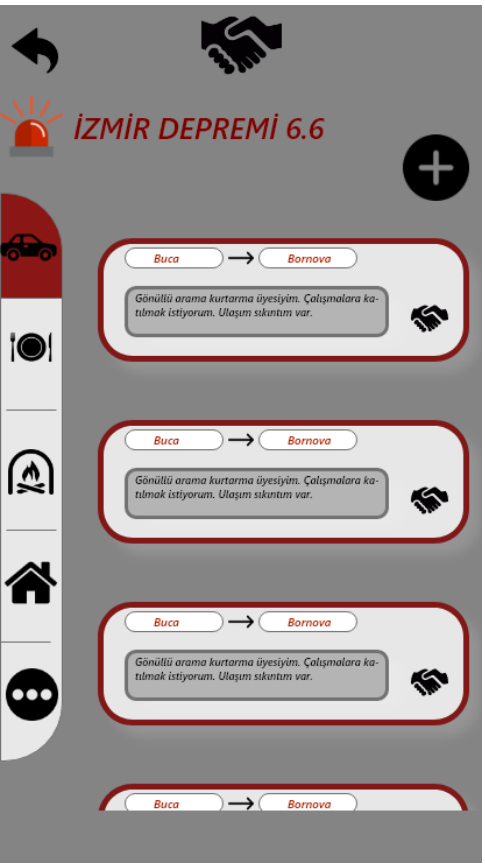


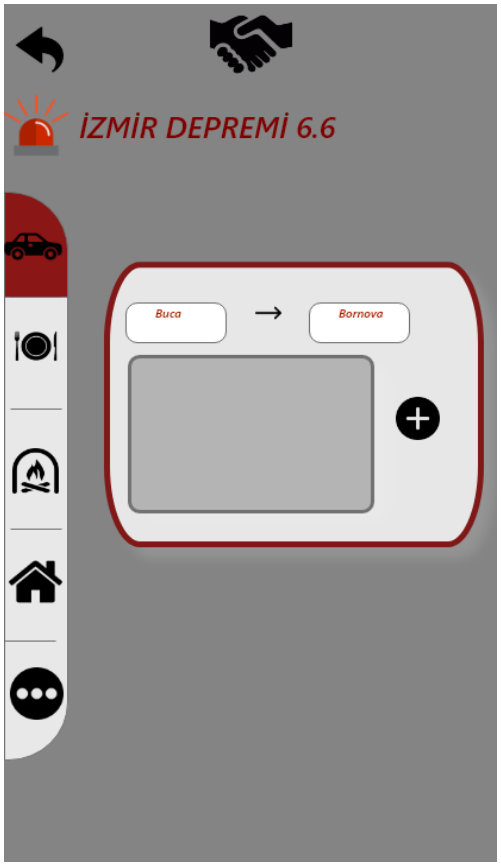


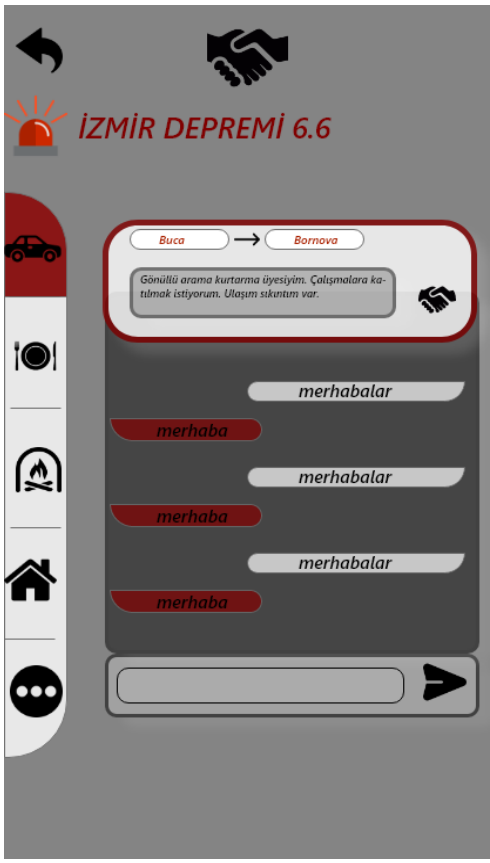

EL ELE
HOŞGELDİN
İLAYDA!

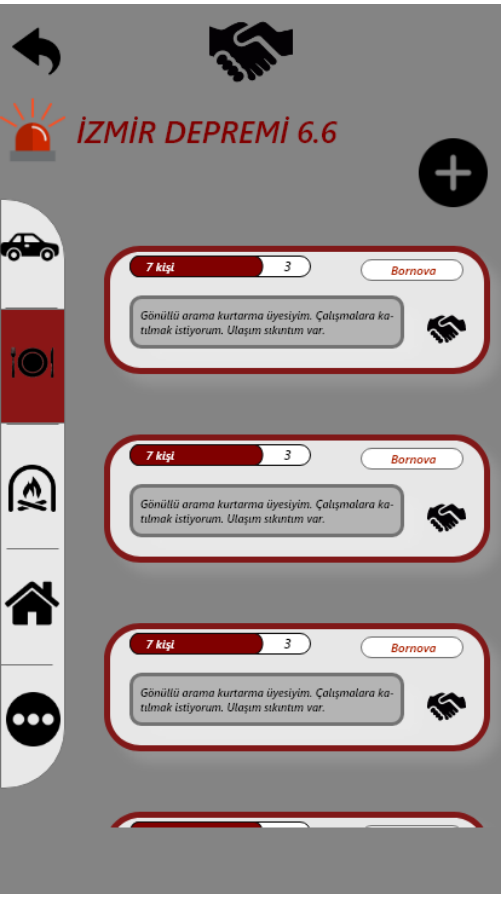

EL ELE
AFET ALARMLARI
İZMİR
deprem
30.10.20 14.51
6.6
ANTALYA
deprem
05.12.20 15.00
5.4
İZMİR
deprem
30.10.20 14.51
6.6
ANTALYA
deprem

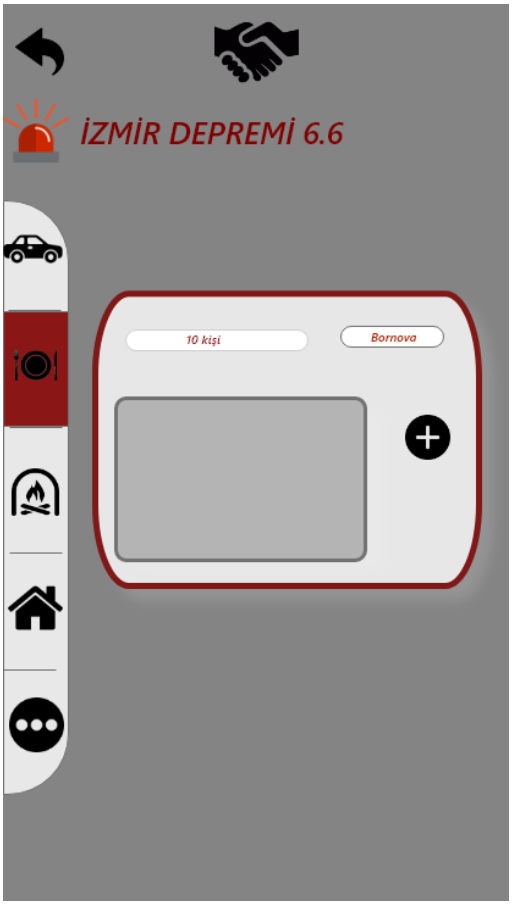

EL ELE
İZMİR DEPREMİ 6.6
GÖREVLENDİRME
KAMPANYALAR
FORUM


İZMİR DEPREMİ 6.6
Buca
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.

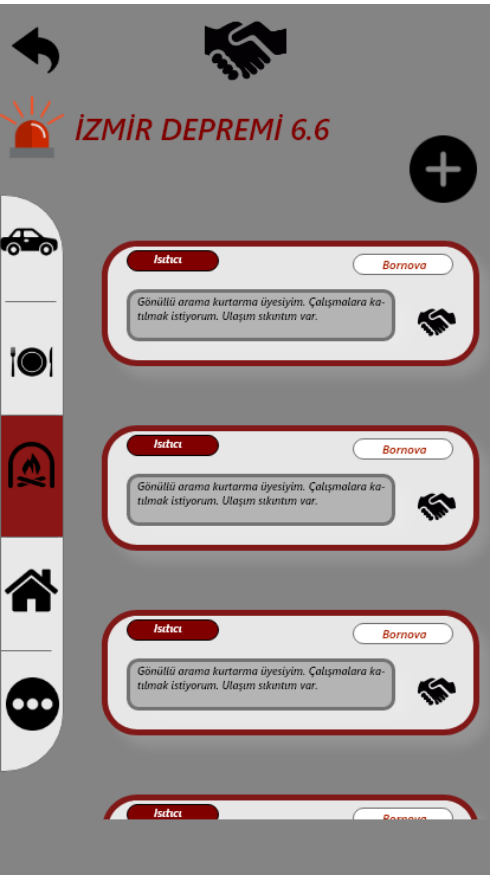

İZMİR DEPREMİ 6.6
Buca
Bornova

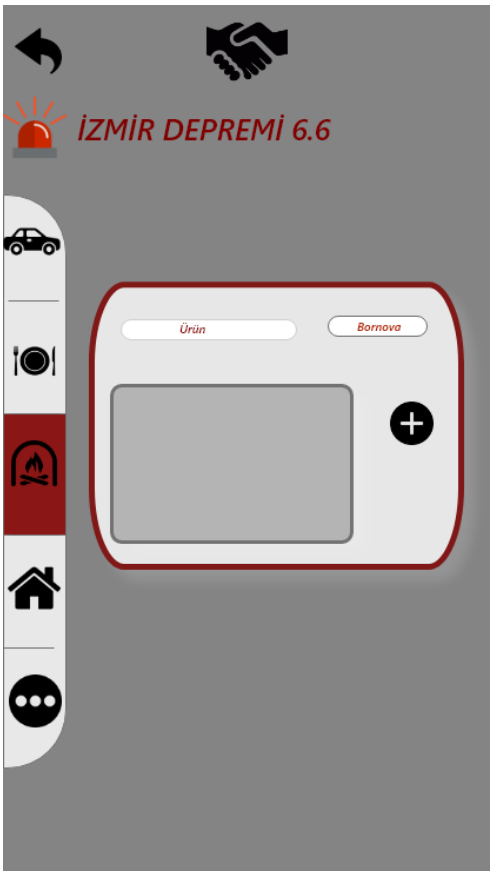

İZMİR DEPREMİ 6.6
Buca
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
merhabalar
merhaba
merhabalar
merhaba
merhabalar
merhaba


İZMİR DEPREMİ 6.6
7 kişi
3
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
7 kişi
3
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
7 kişi
3
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.

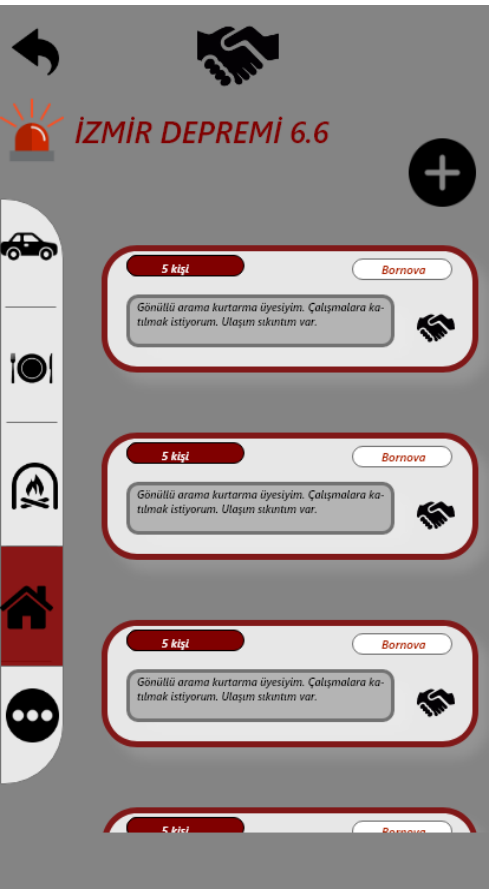

İZMİR DEPREMİ 6.6
10 kişi
Bornova

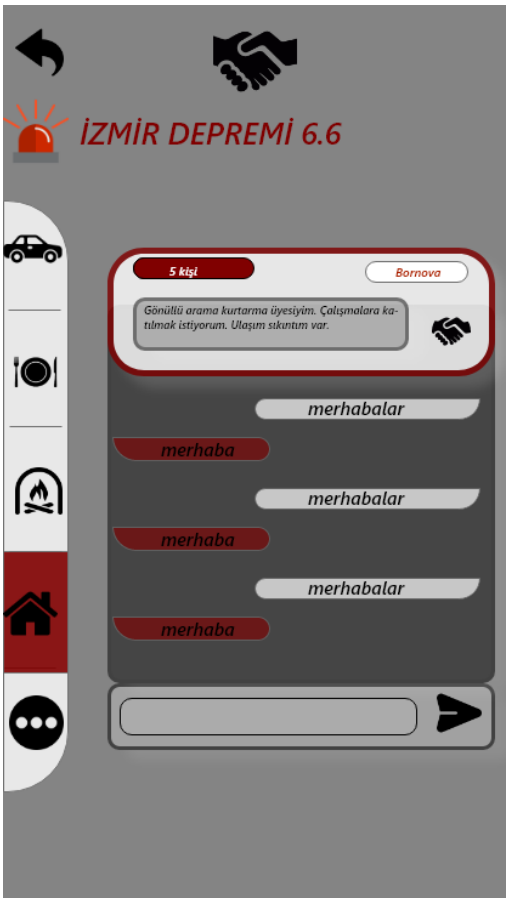

İZMİR DEPREMİ 6.6
Yardımınız için teşekkür ederiz!
Görevliler iletişim bilgilerinizden size ulaşacaktır.

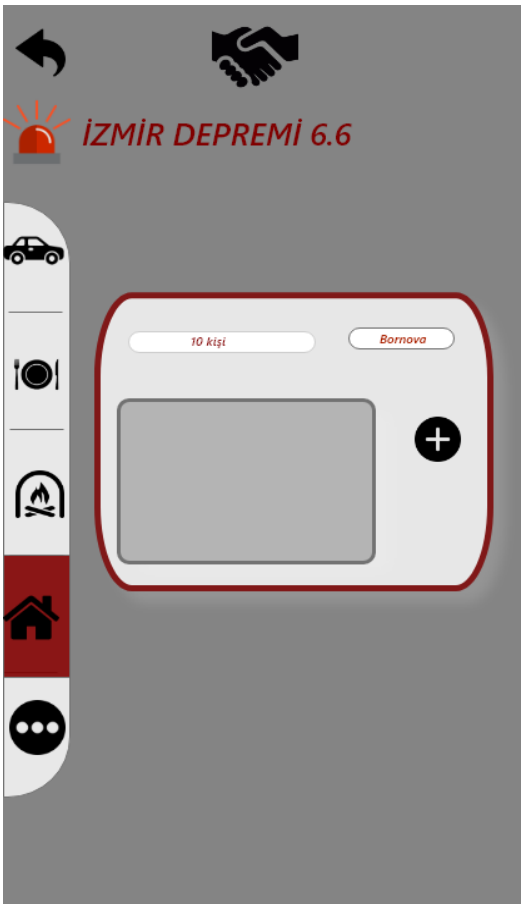

İZMİR DEPREMİ 6.6
İsitıcı
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
İsitıcı
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
İsitıcı
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.

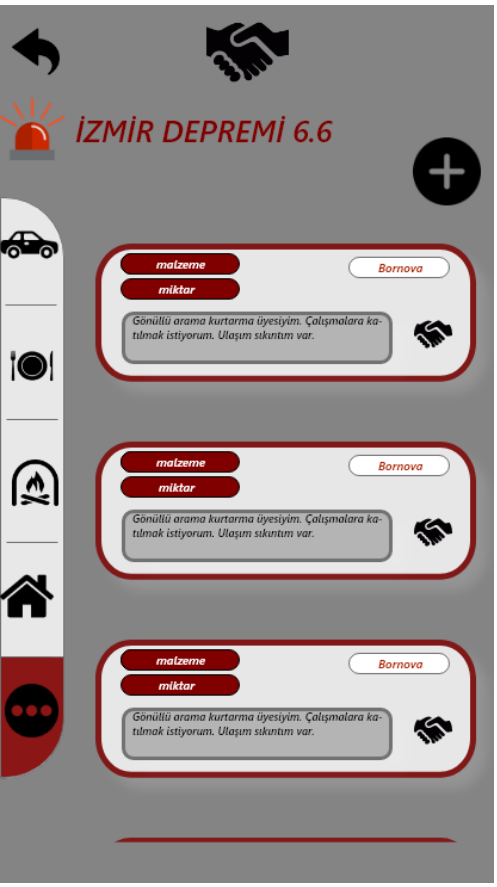

İZMİR DEPREMİ 6.6
Ürün
Bornova

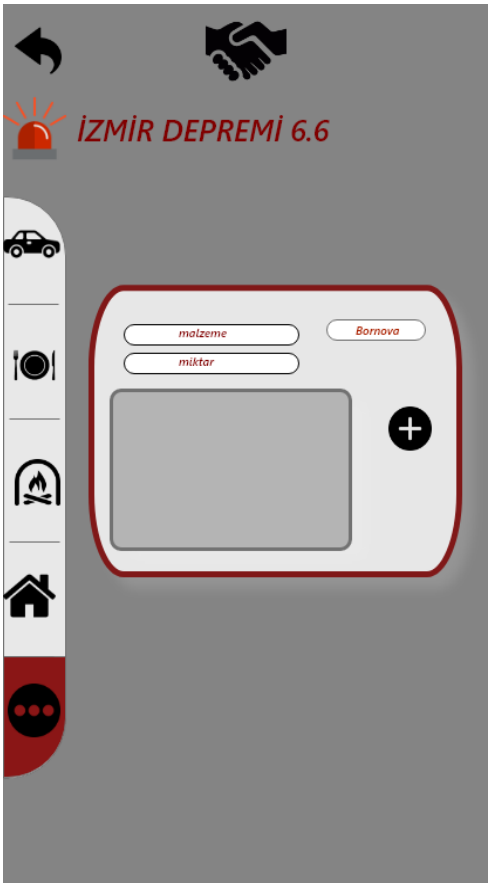

İZMİR DEPREMİ 6.6
Yardımınız için teşekkür ederiz!
Görevliler iletişim bilgilerinizden size ulaşacaktır.


İZMİR DEPREMİ 6.6
5 kişi
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
5 kişi
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
5 kişi
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.


İZMİR DEPREMİ 6.6
5 kişi
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
merhabalar
merhaba
merhabalar
merhaba
merhabalar
merhaba


İZMİR DEPREMİ 6.6
10 kişi
Bornova


İZMİR DEPREMİ 6.6
malzeme
miktar
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
malzeme
miktar
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.
malzeme
miktar
Bornova
Gönüllü arama kurtarma üyesiyim. Çalışmalara katılmak istiyorum. Ulaşım sıkıntım var.


İZMİR DEPREMİ 6.6
malzeme
miktar
Bornova

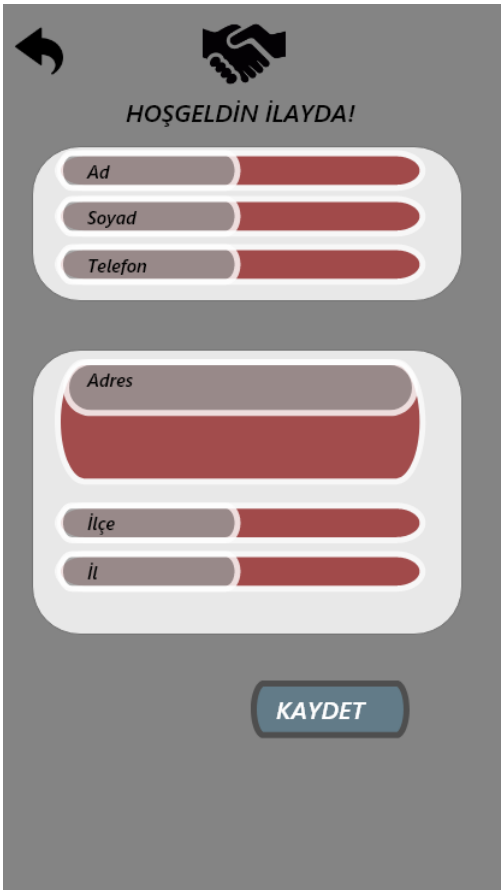

İZMİR DEPREMİ 6.6
Yardımınız için teşekkür ederiz!
Görevliler iletişim bilgilerinizden size ulaşacaktır.


İZMİR DEPREMİ 6.6
1 KİRA 1 YUVA
Depremzedelere 1 kira da sen bağışla!
KIZILAY
Siz de Kızılay'a bağış yaparak depremzede-lere yardım edebilirsiniz!
1 KİRA 1 YUVA
Depremzedelere 1 kira da sen bağışla!


EL ELE
İZMİR DEPREMİ 6.6
Kedimi kaybettim
Ekrem Orhon mahallesinde kaybolmuştur.Boynunda gökkuşağı tasması vardır. Bulanlar +90 555 5555 numaralı telefondan ulaşsın lütfen.
Kedimi kaybettim
Ekrem Orhon mahallesinde kaybolmuştur.Boynunda gökkuşağı tasması vardır. Bulanlar +90 555 5555 numaralı telefondan ulaşsın lütfen.
Kedimi kaybettim
Ekrem Orhon mahallesinde kaybolmuştur.Boynunda gökkuşağı tasması vardır. Bulanlar +90 555 5555 numaralı telefondan ulaşsın lütfen.


İZMİR DEPREMİ 6.6
Kedimi kaybettim
Ekrem Orhon mahallesinde kaybolmuştur.Boynunda gökkuşağı tasması vardır. Bulanlar +90 555 5555 numaralı telefondan ulaşsın lütfen.
kullanıcı 1
entry 1...
kullanıcı 1
entry 1...
kullanıcı 1
entry 1...
kullanıcı 1
entry 1...


Kişisel Bilgilerim
Çıkış yap


HOŞGELDİN İLAYDA!
Ad
Soyad
Telefon
Adres
İlçe
İl
KAYDET

10.LESSONS LEARNED

Understanding the Big Picture:

After determining our problem domain, we determined the requirements for the system we will develop based on the deficiencies we identified in existing systems.Before specifying the requirements, we divided the problems into sub-problems and specified requirements for each sub-problem.Following that, we created an uml class.

Organizing the Development Effort:

A single architectural description language was used to design the system.In addition, an interface was created using adobe xd in order to easily examine the interdepartmental connection in the system.

Facilitating the Possibilities for Reuse:

Components are usable. We configured the component for communication for usability and the component for information display according to this function.

Evolving the System:

It must have an architecture that must be built in such a way that changes in one part of the system do not have a negative impact on other parts of the system.In the system we designed based on this, the sauce and awareness departments do not have any negative effects on the earthquake section.

Guiding the Use Cases:

We have created several systems specific to functions in the use case diagram.The requirements of these systems designed differently from each other do not conflict with each other.Some components have been specified as extended relationship with systems.

11.CONCLUSION

This report presents the modeling of the aid platform based on the activities before and after the earthquake identified as the problem.There are 5 systems in the use-case created to solve the problem.Modeling was created with XCD architectural definition language with these systems and their components.

Algorithms have been designed with modeling and these algorithms have interface and abstract classes for reusability.We have supported Facilitating the Possibilities for Reuse, one of the categories of the architecture-centric perspective, with two components.Thus, the usability of the platform is ensured even if different components are added.

12. REFERENCES

- XCD - An Architecture Description Language, https://sites.google.com/site/ozkayamert1/home/xcd
- Pseudo Code, https://www.geeksforgeeks.org
- Drawio –UML Diagrams. Application
- Adobe Xd – User Interface Design. Create Cloud Application
- Software Analysis. Wikipedia. Web. April.2020.
- BOOK. R. N. Taylor, N. Medvidovic, E. M. Dashofy. 2009. Software Architecture: Foundations, Theory, and Practice. 1 st edition. Wiley