



DOKUZ EYLÜL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING



CME 2210 OBJECT ORIENTED ANALYSIS AND DESIGN

Tınaztepe Yerleşkesi, Buca-Kaynaklar, Dokuz Eylül Üniversitesi, İZMİR, TÜRKİYE

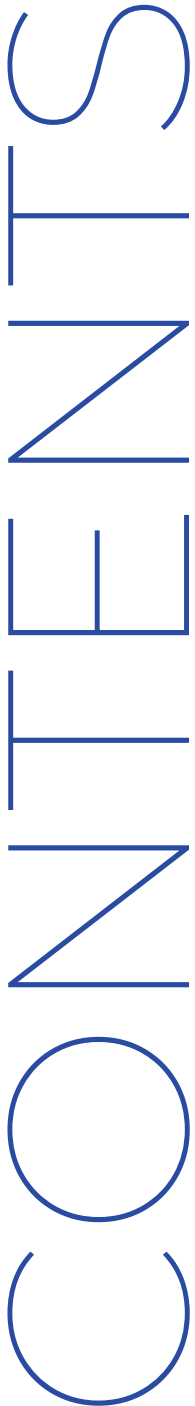
30.05.2023

Disaster Coordination System (DCS)

by
Yusuf Gassaloğlu
Berkay Dinç
Alperen Aydın

2023

Table of Contents



02.

INTRODUCITON

03.

REQUIREMENTS

07.

UML DIAGRAMS

- 07 - CLASS DIAGRAM
- 08 - USE CASE DIAGRAM
- 10 - SEQUENCE DIAGRAMS
 - 10 - *User Sequence Diagram*
 - 11 - *Personnel Sequence Diagram*
 - 12 - *Coordination Unit Sequence Diagram*
- 13 - ACTIVITY DIAGRAMS
 - 13 - *User Activity Diagram*
 - 14 - *Personnel Activity Diagram*
 - 15 - *Coordination Unit Activity Diagram*
 - 16 - *General View*
- 16 - STATE DIAGRAMS

20.

IMPLEMENTATION

There are many natural disasters in Turkey. Earthquakes, floods, forest fires, avalanches, etc. take too many lives each year. Moreover, many people face difficulties because of the coordination problem after the disaster. As an example, after disasters, victims report the need for a search and rescue team. They report post-disaster needs such as tents, water, food, etc. If there is no coordination, these requests are fulfilled randomly. Most of the helps go too late. Because of these, deaths increase. Disaster Coordination System (DCS) is the best system for the best coordination.

The scope of the project area is İzmir / Turkey. The scope of the project is managing post-disaster urgent calls and needs.

Users (disaster victims) report their status to DCS. Status example:

Name: Joe Doe,

GSM: 05551234567

Location information

tent – ambulance – rescue team, 2 children 2 adults.

REQUIREMENTS

The disaster coordination system will be divided into two parts. The first one is the Coordination Engine and the other one is the Client Endpoint. The coordination engine will mostly implement the decision-making part of the system, and the client endpoint will be the interface for reaching and interacting with the users of the system.

Before diving into either of the parts, the usage and the flow of the system must be explained. The usual flow of the system starts with a client interaction. A “client” is one of the following three:

01

User

Requests a list of needs and several different status information with location

02

Personnel

Awaits new tasks that will be assigned to them by their Coordination Units.

03

Coordination Unit

Employs personnel, awaits new tasks that will be assigned to them by the system itself, when the tasks are received, decides which personnel/s will resolve the task, assigns the task to the personnel/s.

STUDENT REQUIREMENTS

If a user performs a request, the system will decide which coordination unit/s to take action. For example, if a user needs an ambulance and security, the system will assign tasks to the nearest coordination units, in this case a hospital, and a police station nearby. Remember that this example does not fully define or explain the behavior of the Coordination Engine. When the tasks are assigned to the coordination units, they will then assign tasks to their personnel. For example, the hospital will respond to the ambulance call by assigning some of their personnel to the task. And the police station will do the same for their case. Later, when the user requests are fulfilled by the personnel, the personnel will mark the task as “resolved” and the users will be notified that their requests have been resolved.

The clients will communicate and interact with the system through the Client Endpoint. The endpoint will forward user requests to the Coordination Engine, push new notifications or tasks to the clients, and store the data of several different parts of the system, such as information of all the coordination units, users’ personal information

For demonstration purposes and proof of concept, the system will employ the observer pattern on clients for the interaction scheme. The clients will subscribe for new notifications that will be pushed from the Client Endpoint, and when their state is changed, for example by receiving a notification, the clients will respond with invoking handlers.

In order to apply the observer pattern, Users, Personnel and Coordination Units will be derived from an abstract Client class including methods for event handling, some of them are:

```
void on(Event event, Function handler)
```

```
void notify  
(Event event, Notification notification)
```

And the user endpoint will provide methods similar to the following ones for communication and interaction (this list does not reflect the whole method table):

```
void createRequest  
(User user, Request request)
```

```
void assignTask  
(Personnel personnel, Task task)
```

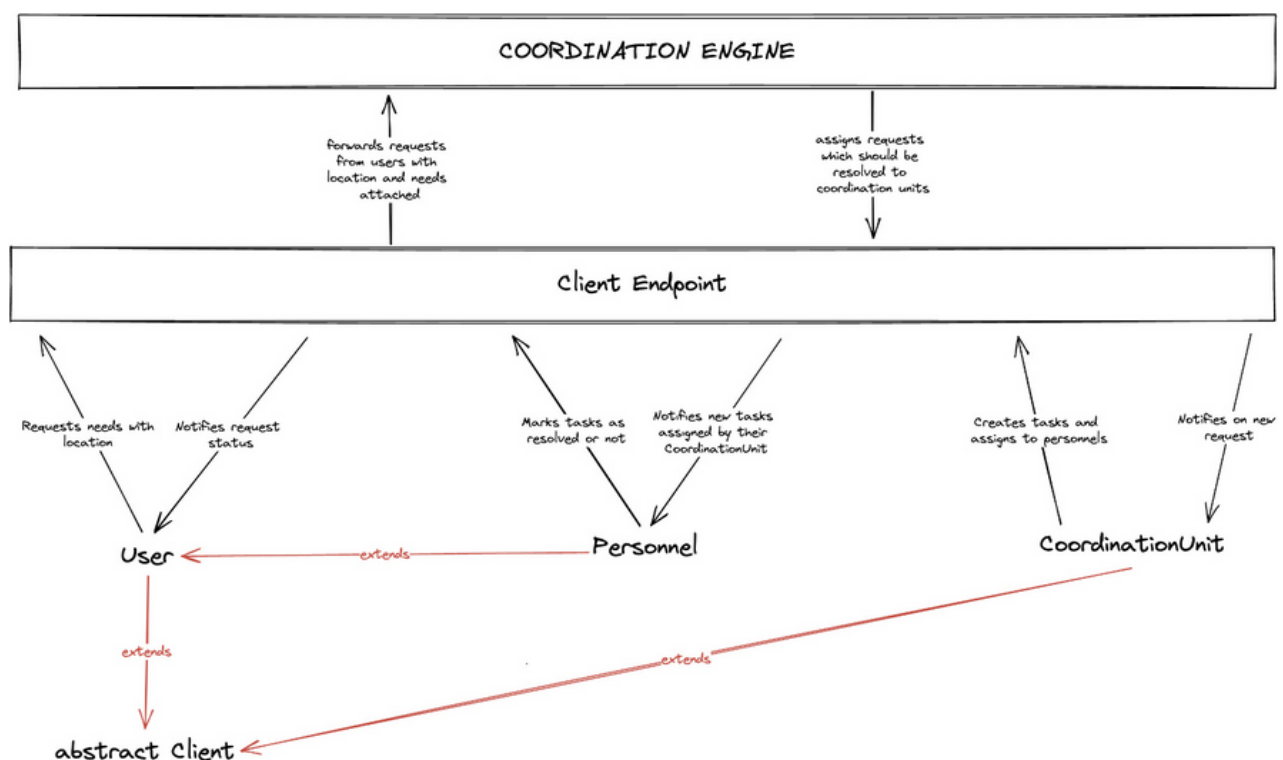
```
void assignTask  
(Personnel personnel, Task task)
```

Finally, the system will decide which task will be assigned to which coordination unit using the Coordination Engine. The behavior or the algorithm of the Coordination Engine is not fully defined, and can't be explained yet. But the main motivation of this section of the project is to create a decision-maker that will take action in such a way that all the incoming requests will be fulfilled in the minimum amount of time.

Most probably, there will only be one method interface for the Coordination Engine to interact with other components, which is:

```
CoordinationUnitTask[] createTasksForRequest(Request request, User user)
```

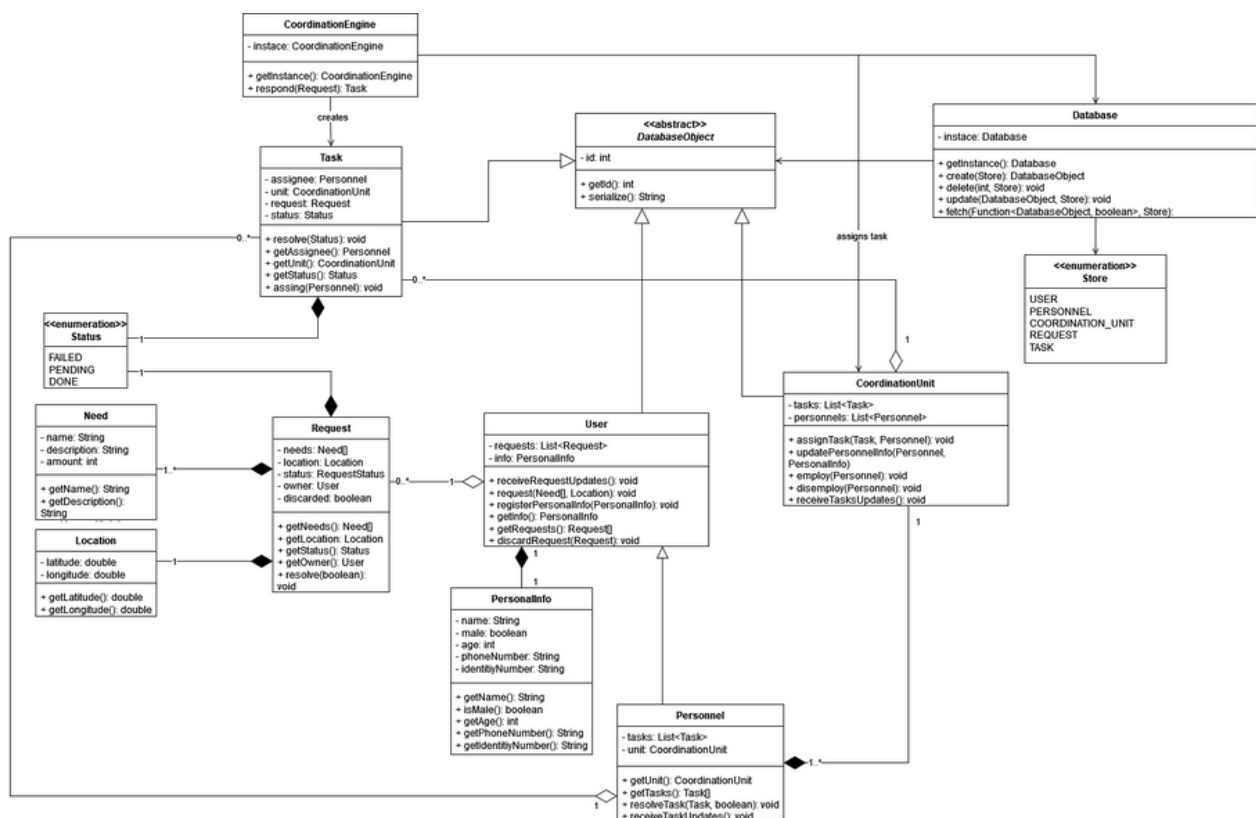
The method will decide the actions will be taken in order to fulfill the user



UML DIAGRAMS

Class Diagram

The class diagram contains several different components derived from **DatabaseObject**, which represents an object that can be sent to the **Database** to be stored in a Store. The first derived class is the **User** class, which represents a standard user with the given **PersonInfo**. User class aggregates zero or more **Requests** which are made of a **Location**, one or more **Needs**, and a **Status**. The **Personnel** class extends the User class and represents personnel that work in a **CoordinationUnit**. Coordination units employ personnels, and units can't exist without them, so there is no coordination unit with no personnels. Coordination units also assign personnels to **Tasks**, which are created by the **CoordinationEngine**, in the response of a Request created by a user.



Use case diagram

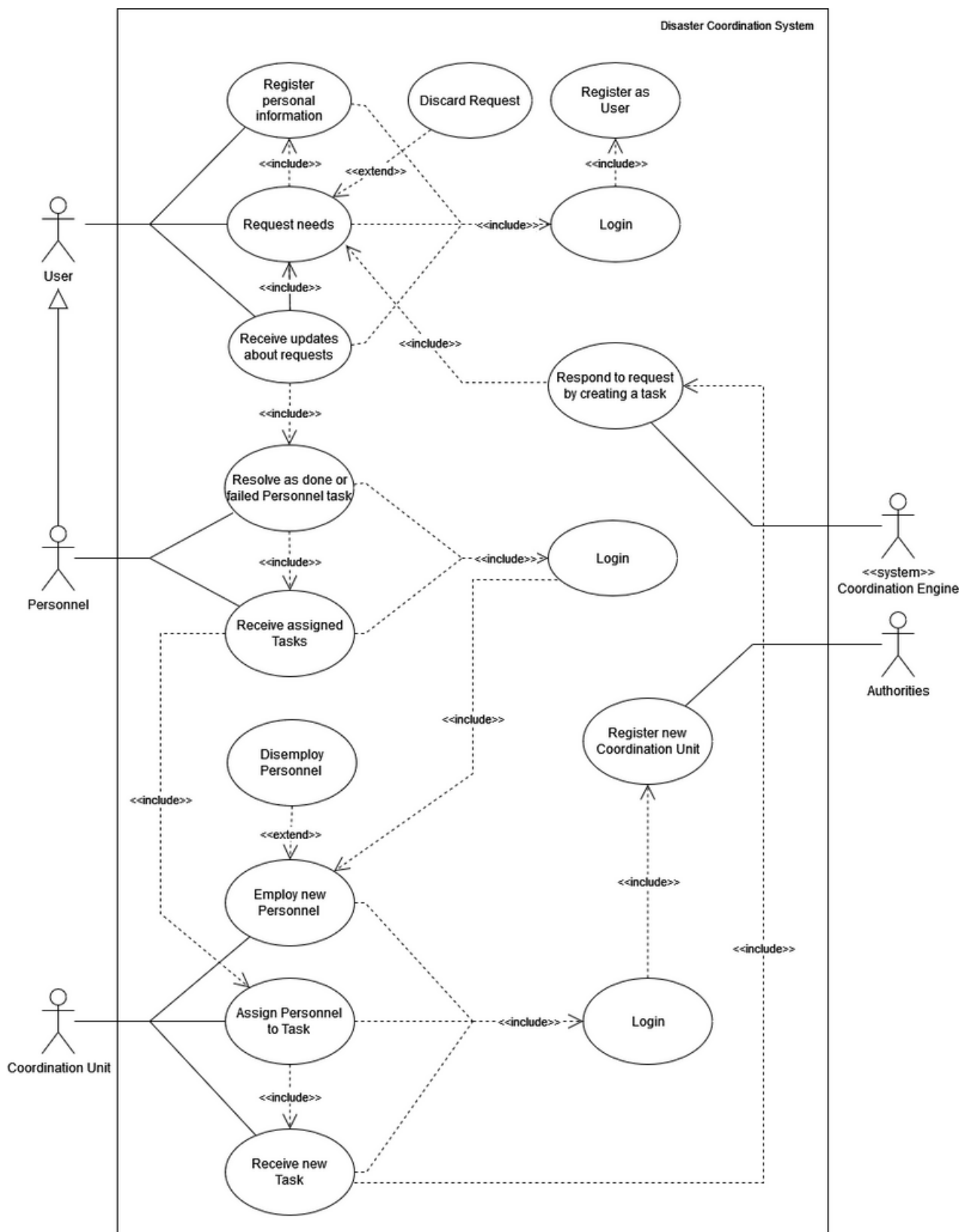
The use case diagram consists of three different “users” of the system. Before each operation they can perform, they must be logged in.

The first one is the **User**, the User is capable of creating a **Request**, but before that, the **User** must provide a **PersonalInfo**, so no anonymous requests can be done. After making a request, the user can monitor the updates on the requests, if they are done or failed or still pending. Also, the user can discard a request if they wish to.

The second one is the **Personnel**, which can monitor their Task's, and receive updates about them, as well as resolve the tasks. Personnels are employed by **CoordinationUnits** and they can't log in before they are registered to the system by their coordination units.

The third one is the **CoordinationUnit**, which can employ, disemploy a new personnel, assign tasks to a specific personnel, and receive the tasks they are assigned to by the **CoordinationEngine**. The coordination units are registered by the authorities of the government.

The **CoordinationEngine** creates tasks as a response to the **Requests** of the users, and creates tasks to be resolved by the appropriate coordination units based on its decision algorithm.

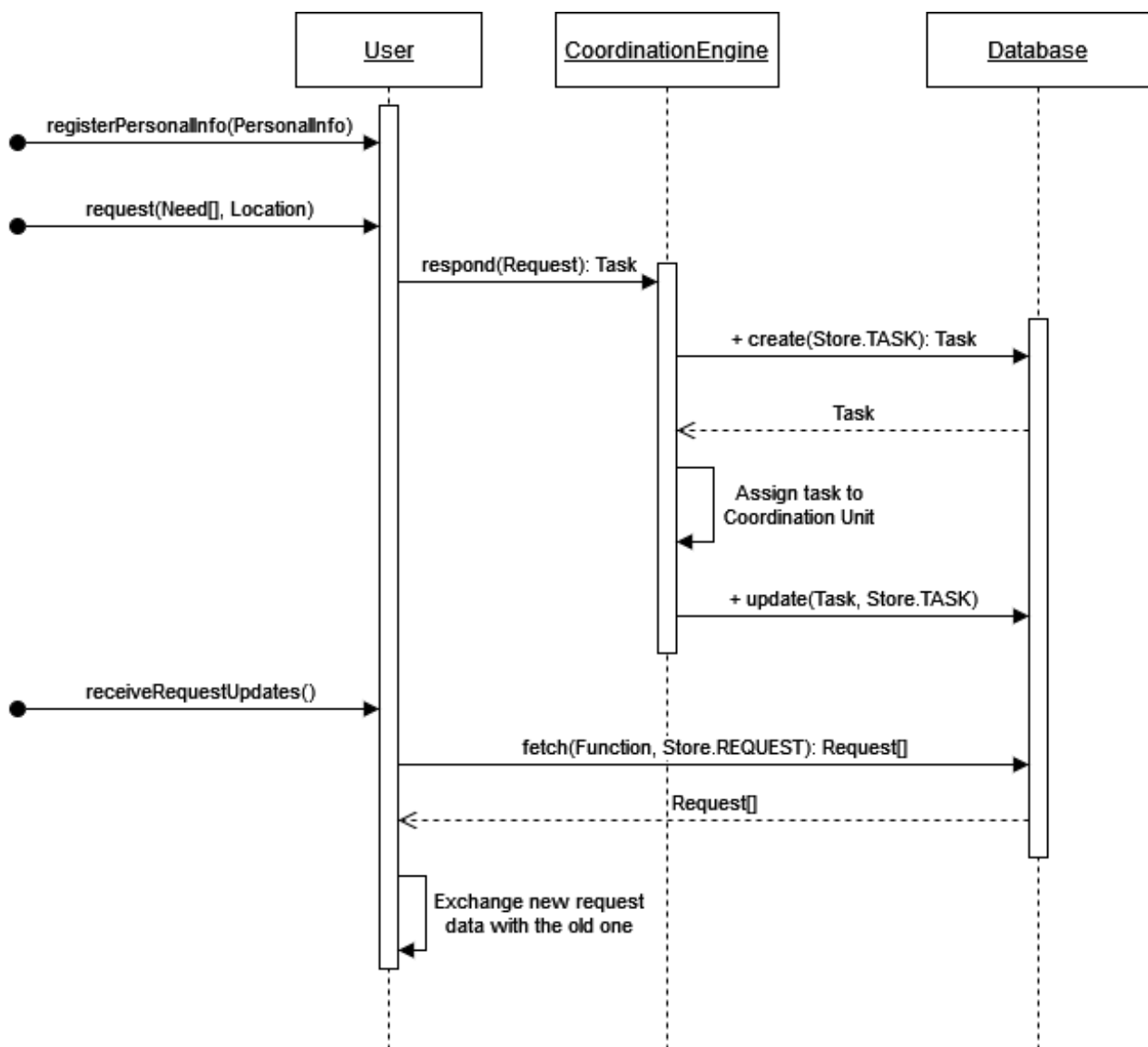


Use Case Diagram

SEQUENCE DIAGRAMS

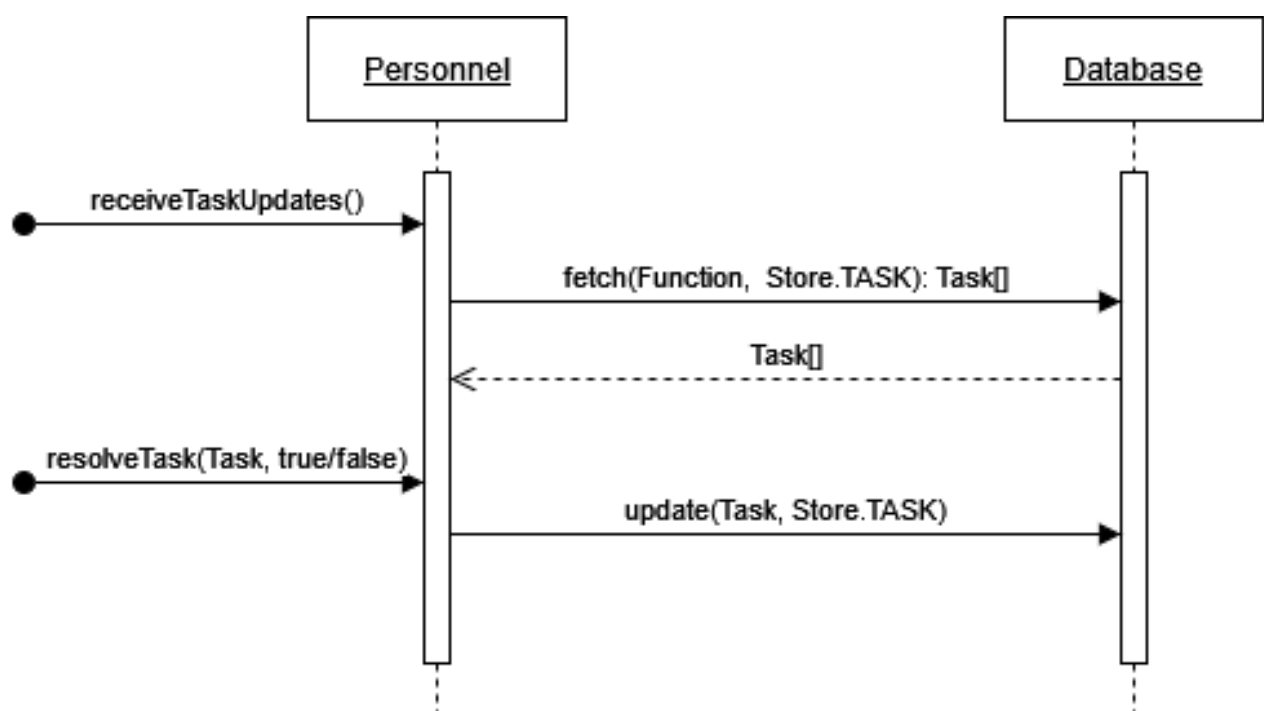
User Sequence Diagram

The user sequence diagram shows all of the actions a user can perform, and it shows the relationship between the coordination engine and the database of the system.



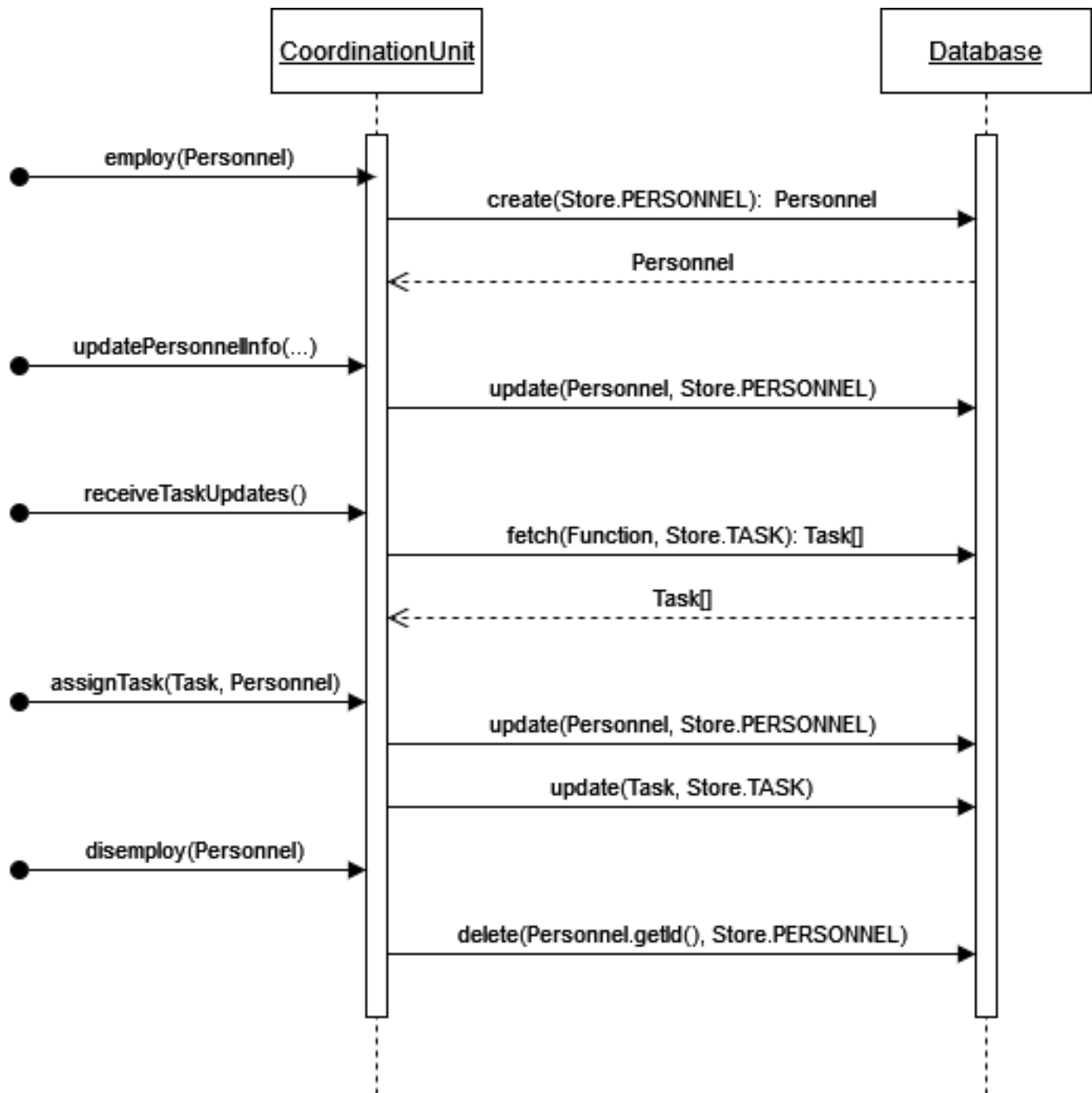
Personnel Sequence Diagram

The personnel sequence diagram is as simple as possible, since a personnel can do little, and no other actions are possible for this class.



Coordination Unit Sequence Diagram

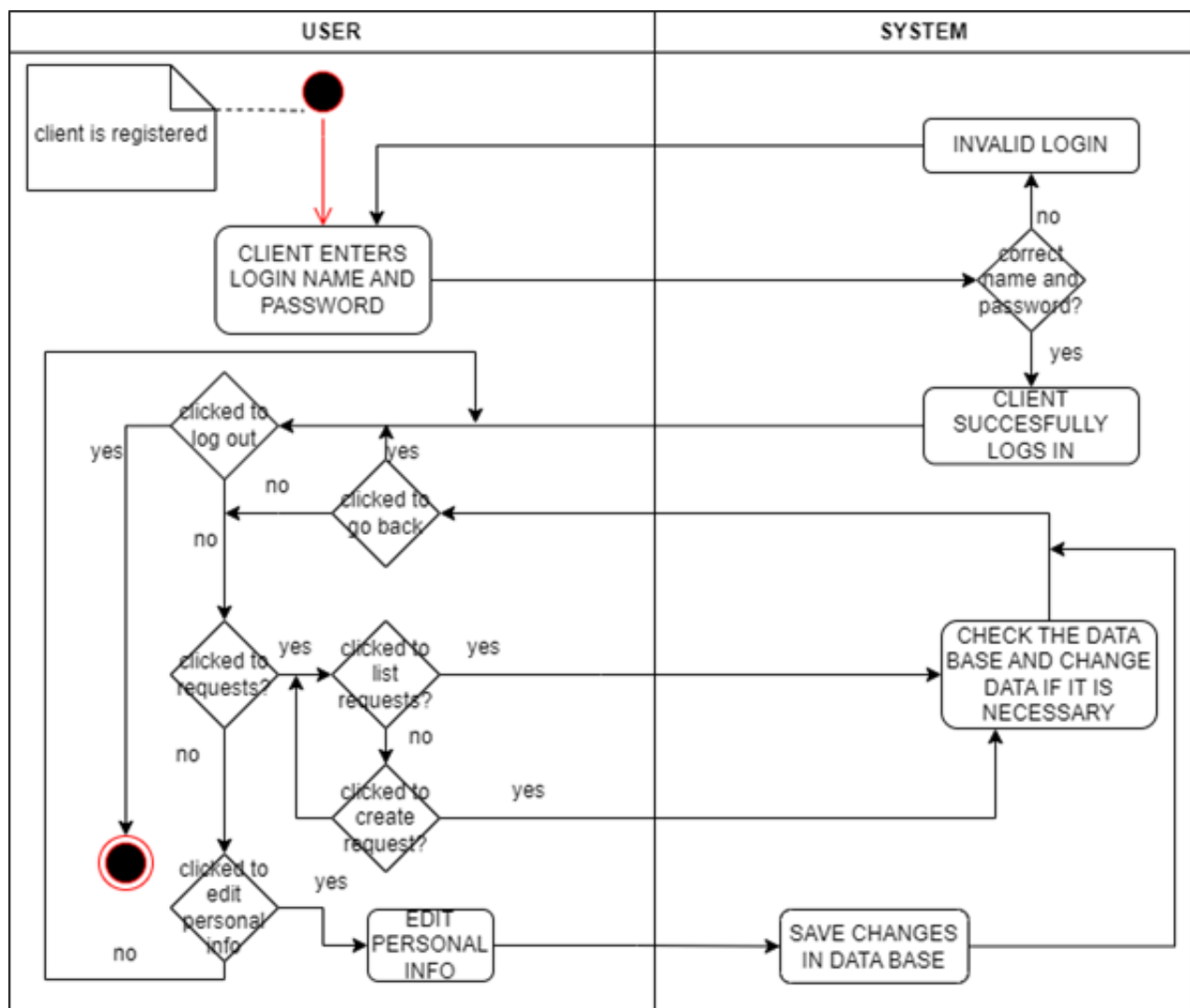
The personnel sequence diagram is as simple as possible, since a personnel can do little, and no other actions are possible for this class.



ACTIVITY DIAGRAMS

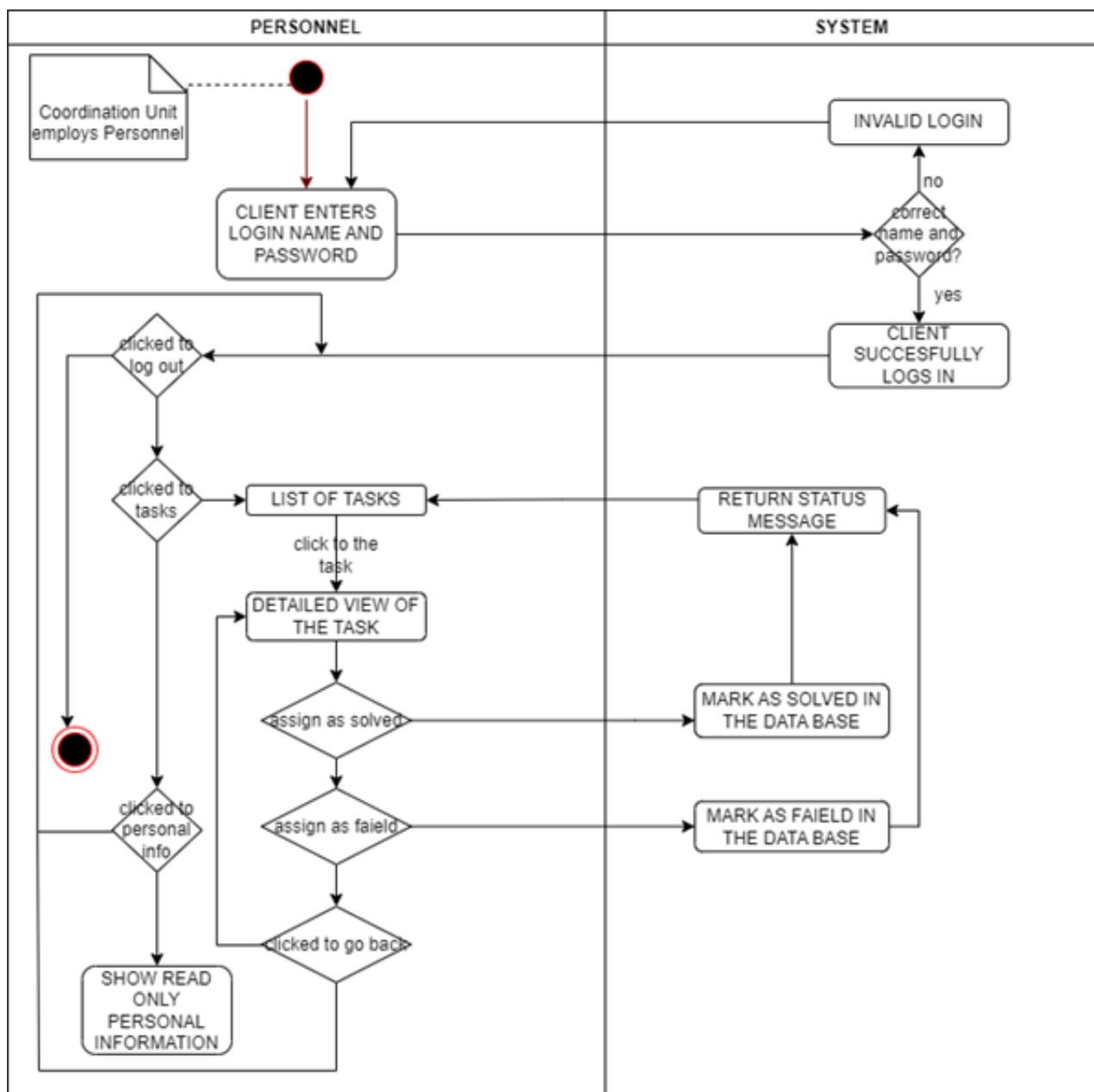
User Activity Diagram

Users log into the system and the system checks login information from the users database. They can create requests or list past requests. They can edit personal information. The system saves new information in the database. When the request is forwarded, the request is added to the database on the system side. The client must be registered before.



Personnel Activity Diagram

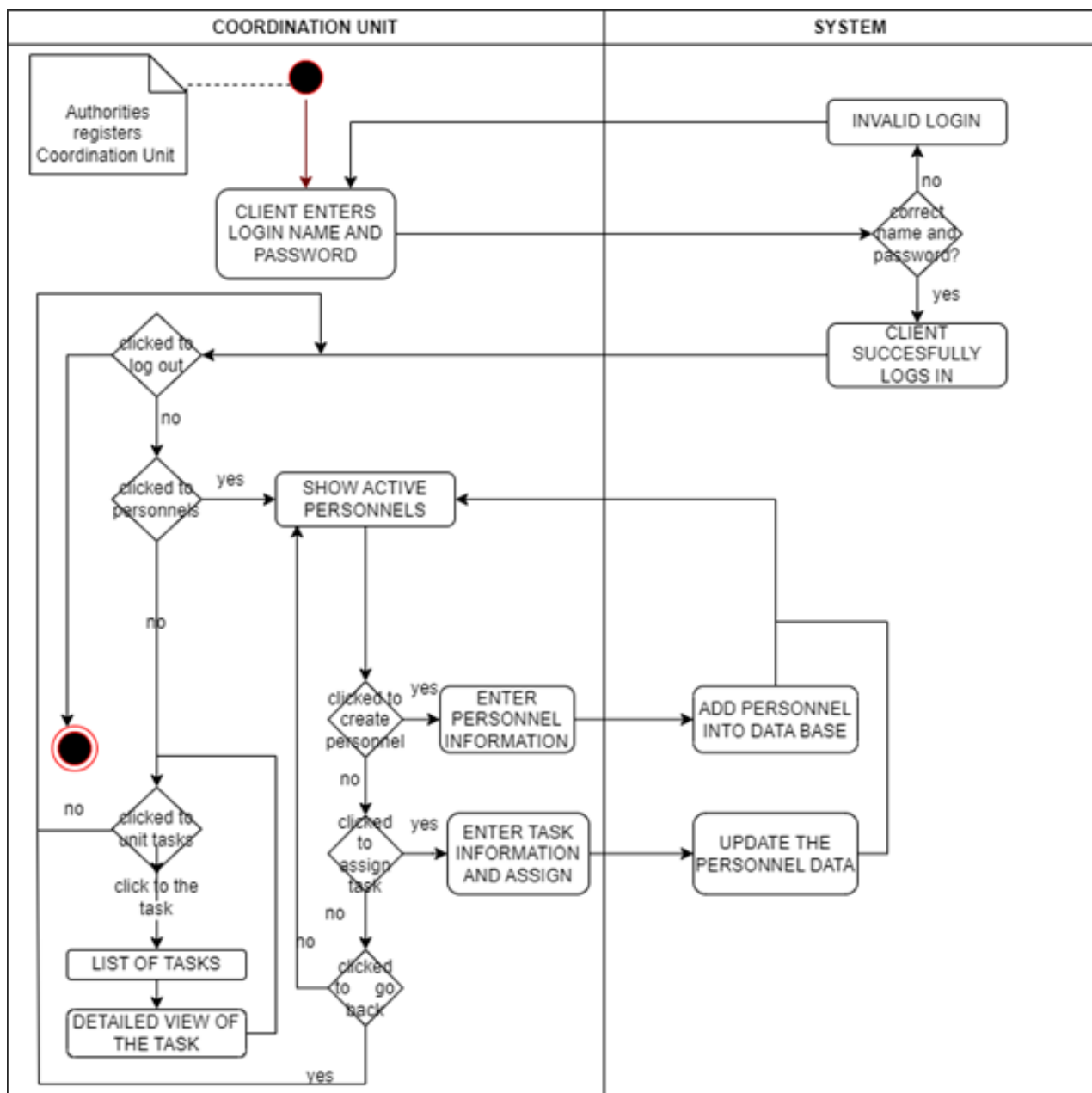
Personnel log into the system and the system checks login information from the personnel database. They can list the tasks and assign as solved or failed. On the system side, the data in the database will be updated and the system returns the status message. They can read personal information but they cannot edit. The system saves new information in the database. The personnel must be registered before by coordination unit.



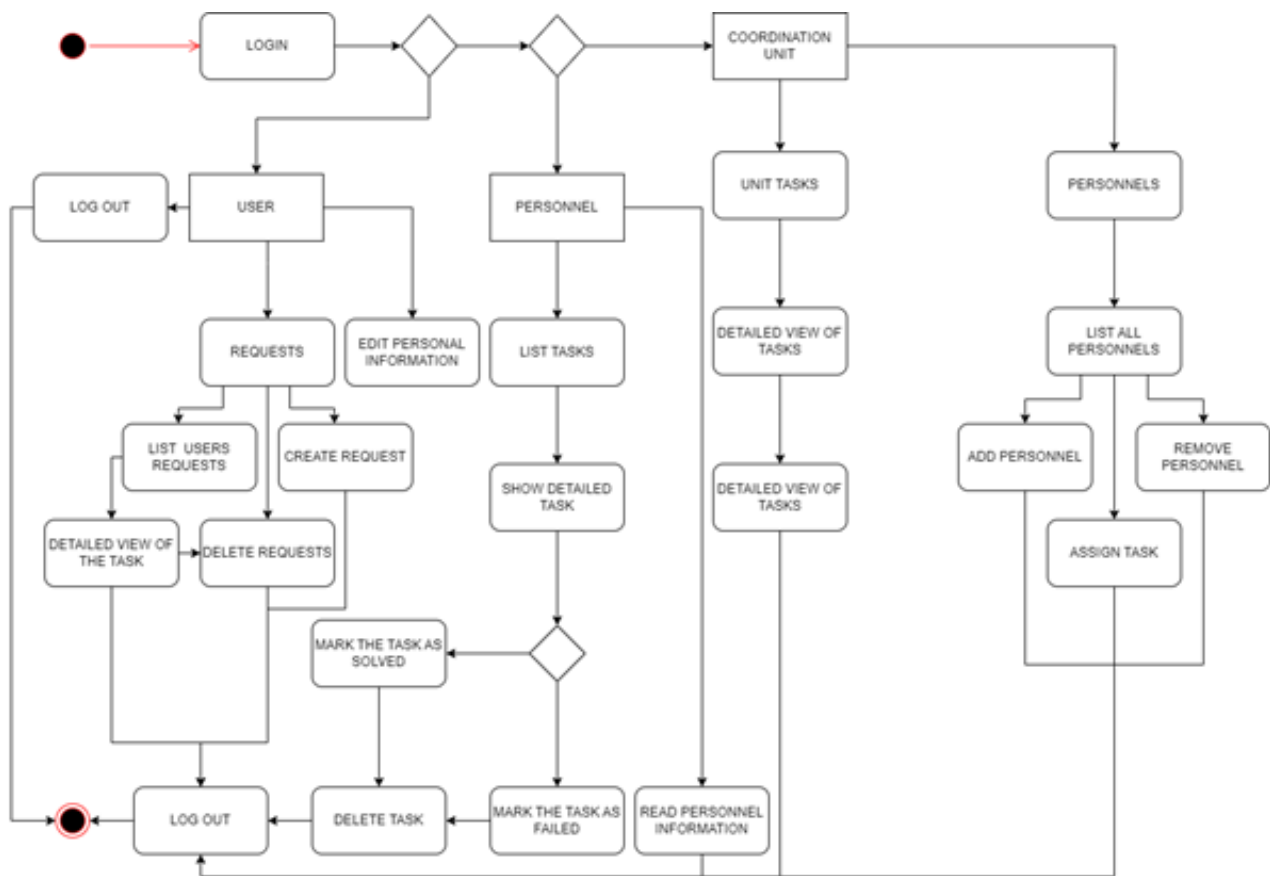
Coordination Unit Activity Diagram

Coordination units log into the system and the system checks login information from the coordination unit database. They can list the tasks.

They can list active personnel. They can create personnel and assign task to personnel. Those two action updates the personnel database on the system side. The coordination unit must be registered before by the authorities.

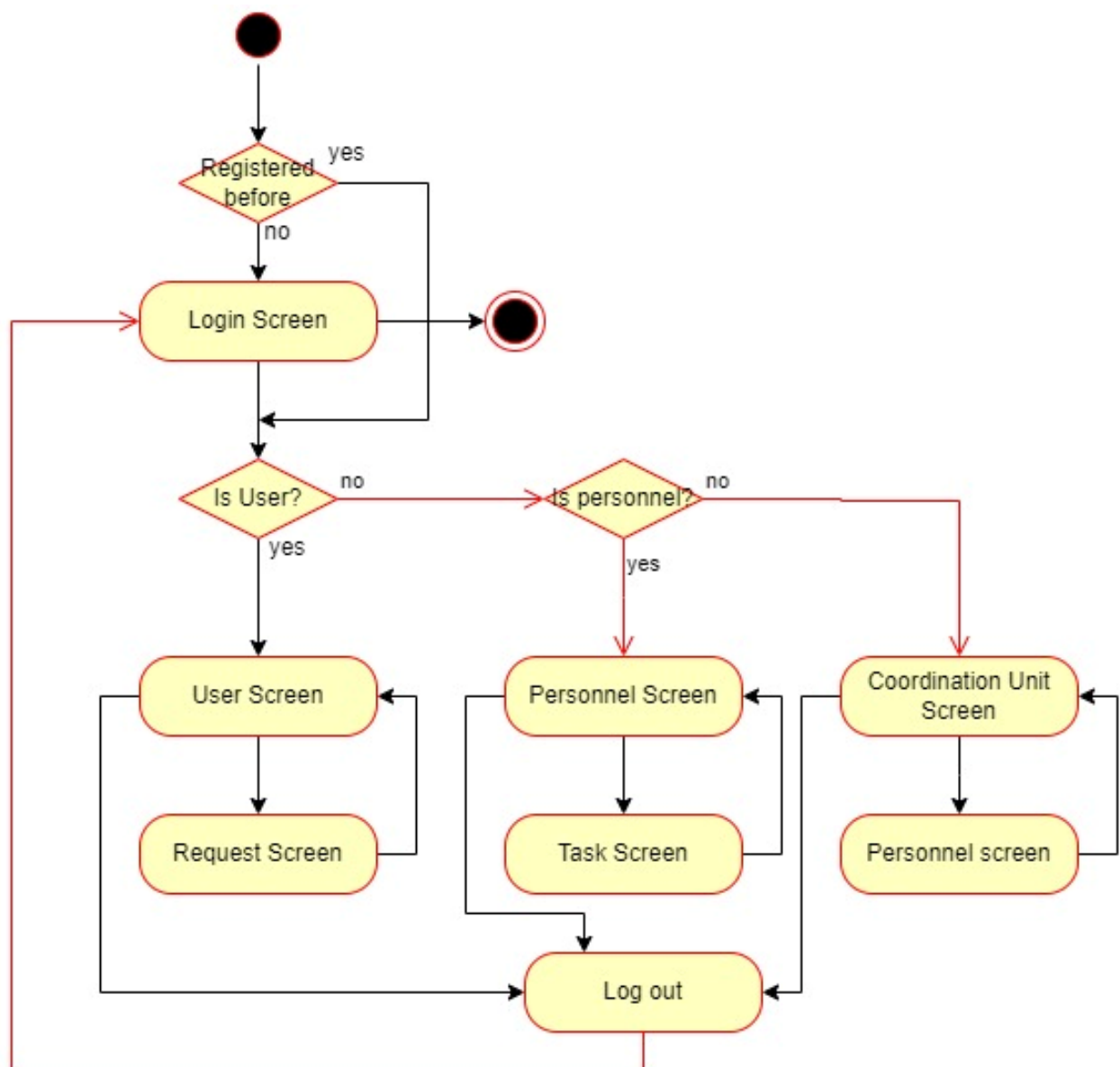


The general view of the activity diagrams are as follows,



STATE DIAGRAMS

Displayed screen has different states. If user is already registered, the appropriate user profile page will open up. User will have various options for his user profile. For all screens, log out button will changed displayed screen's state to login screen.



When a user sends a request, coordination unit will see it as an active request. Coordination unit will assign the task to a personnel and task's state will be "In progress". After that coordination unit will wait feedback from the personnel. After getting the feedback, the coordination unit will either approve the task and its state will be "Success" or task will be failed and its state will be "Active" again.



IMPLEMENTATION

USED TECHNOLOGIES:



JAVA

Disaster Coordination System written in Java.

Data Stored with postgresSQL. [PostgreSQL](#)



Java Swing library used for GUI. DCS has 14 windows. Those are:

Assign TaskWindow.java

CoordinationUnitMainMenuWindow.java

CoordinationUnitPersonnelWindow.java

CoordinationUnitRequests Window.java

CreatePersonnelWindow.java

CreateRequestWindow.java

LoginWindow.java

PersonnelMainMenuWindow.java

PersonnelTasksWindow.java

RegisterWindow.java

UpdateTaskWindow.java

UserEditPersonelInfoWindow.java

UserMainMenuWindow.java

UserRequestsWindow.java

Java SQL library used for the database. DCS has 3 table users, personnel, requests

id [PK] integer	task_id character varying	first_name character varying	last_name character varying	phone_number character varying	id_number character varying
---------------------------	-------------------------------------	--	---------------------------------------	--	---------------------------------------

Requests Table

id [PK] integer	description character varying	location character varying	status character varying	owner_id character varying	personnel_id character varying
---------------------------	---	--------------------------------------	------------------------------------	--------------------------------------	--

Personnel Table

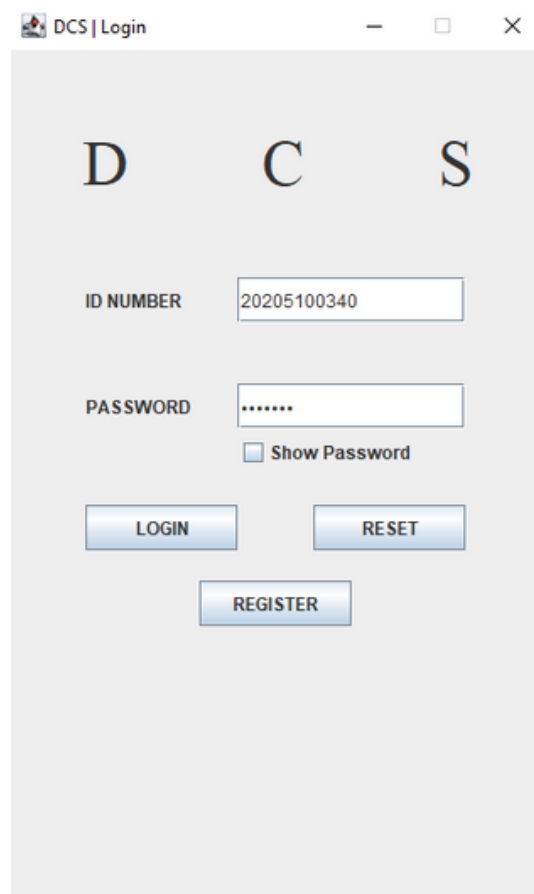
id [PK] integer	first_name character varying	last_name character varying	password character varying	phone_number character varying	email character varying	id_number character varying	is_personnel boolean	is_coordination_unit boolean
---------------------------	--	---------------------------------------	--------------------------------------	--	-----------------------------------	---------------------------------------	--------------------------------	--

Users Table

LOGIN WINDOW

The login window has two text fields, three buttons and one check box. The fields are ID NUMBER and PASSWORD field. The buttons are LOG IN, RESET and REGISTER BUTTON. The check box is SHOW PASSWORD.

ID number is Turkish Identity Number. The number is a unique number. Users can log in to the system with their ids and passwords which are stored in the database. They can be registered to the system by clicking register. They can clear text fields with the reset button. They can show the password by clicking the checkbox.

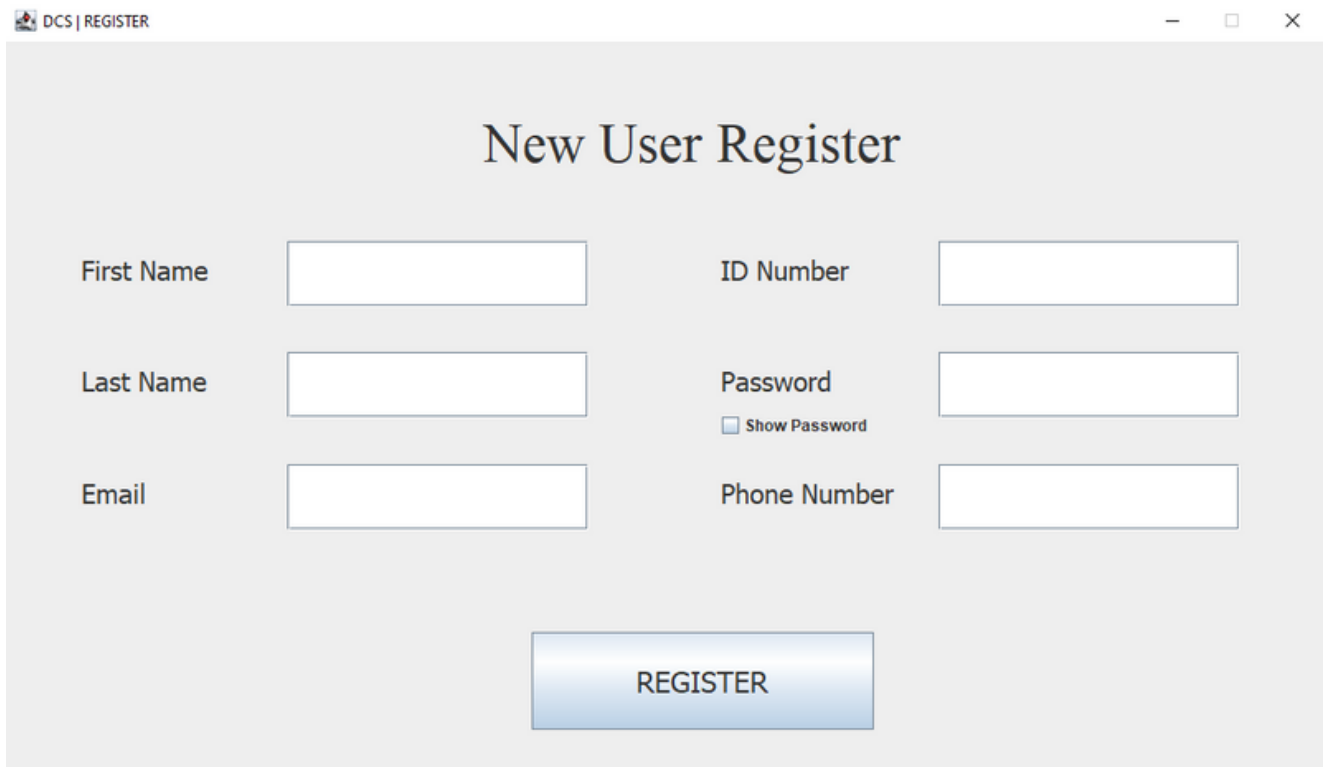


The screenshot shows a window titled "DCS | Login". At the top, the letters "D", "C", and "S" are displayed in a large, serif font. Below this, there are two text input fields. The first is labeled "ID NUMBER" and contains the text "20205100340". The second is labeled "PASSWORD" and contains seven dots. Below the password field is a checkbox labeled "Show Password". At the bottom, there are three buttons: "LOGIN", "RESET", and "REGISTER". The "LOGIN" and "RESET" buttons are positioned side-by-side, and the "REGISTER" button is centered below them.

REGISTER WINDOW

The register window has six text fields, one button and one checkbox. The fields are FIRST NAME, LAST NAME, EMAIL, ID NUMBER, PASSWORD, PHONE NUMBER. The button is REGISTER button. The checkbox is SHOW PASSWORD checkbox.

The user fills in the relevant fields. The system checks the id. If the database has the same id, the system returns an error message. Email, phone number, and id number are checked by regex. If there is an invalid pattern, the system warns the user. If all fields are correct, the system adds users to the users table from the DCS database.



DCS | REGISTER

New User Register

First Name	<input type="text"/>	ID Number	<input type="text"/>
Last Name	<input type="text"/>	Password	<input type="password"/>
		<input type="checkbox"/> Show Password	
Email	<input type="text"/>	Phone Number	<input type="text"/>

REGISTER

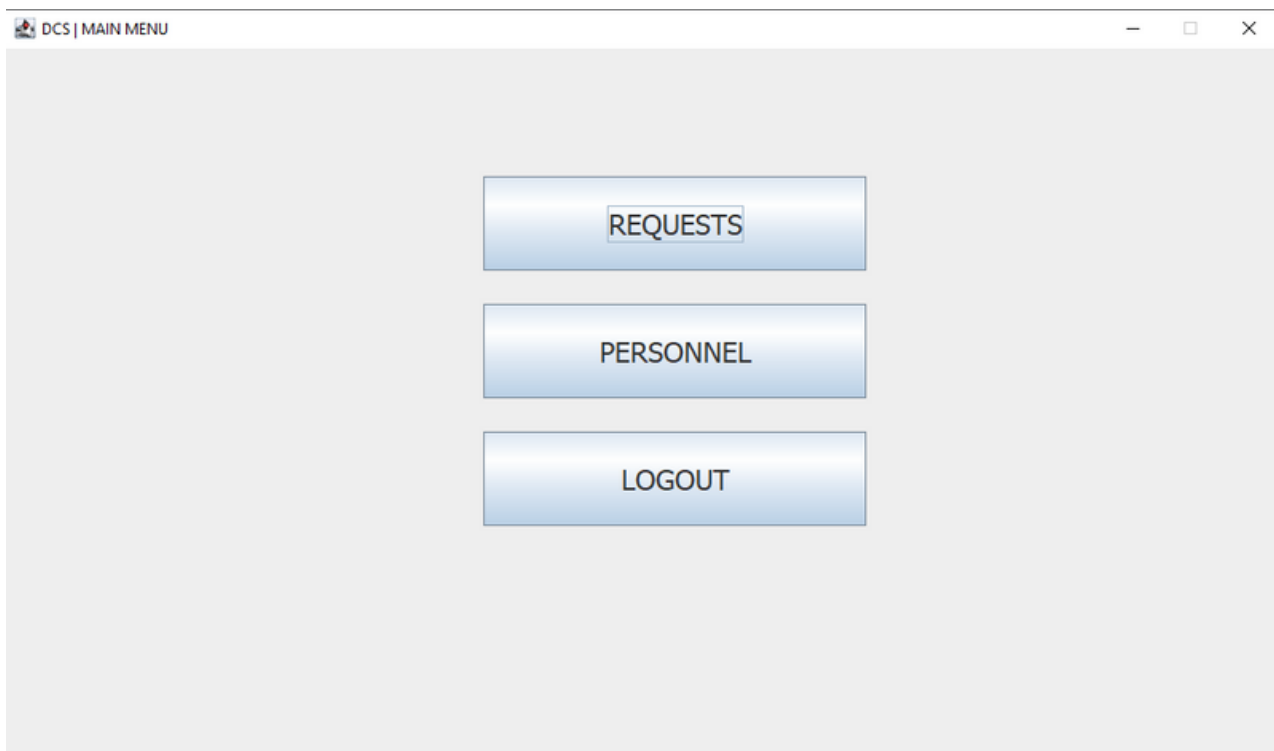
COORDINATION UNIT MAIN MENU WINDOW

The window has 3 buttons. Those are REQUESTS, PERSONNEL, LOG OUT.

The request button opens the requests window. The request window shows requests from users. The unit can do request operations in the requests window.

The personnel button opens the personnel window. The personnel window shows free personnel. The unit can do personnel operations in the personnel window.

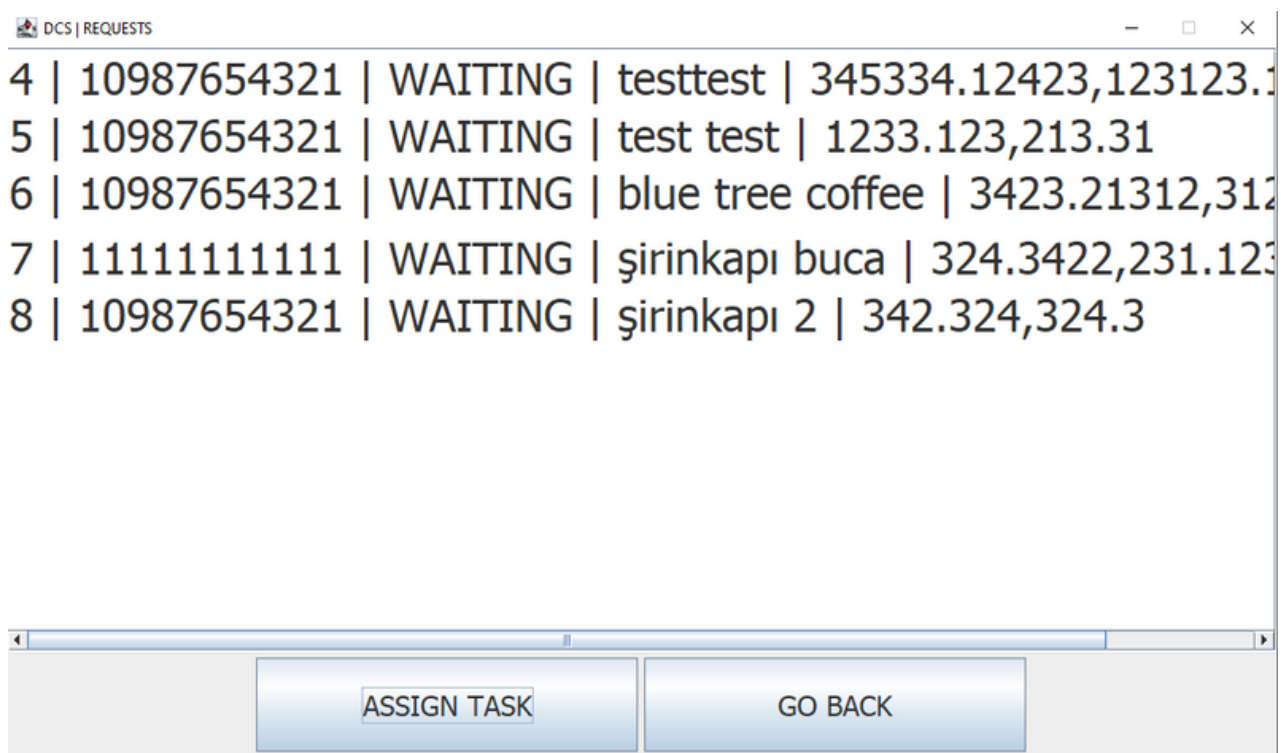
The log out window button quits DCS.



COORDINATION UNIT REQUESTS WINDOW

The window has two buttons and one panel to show a list. Those buttons are ASSIGN TASK and GO BACK.

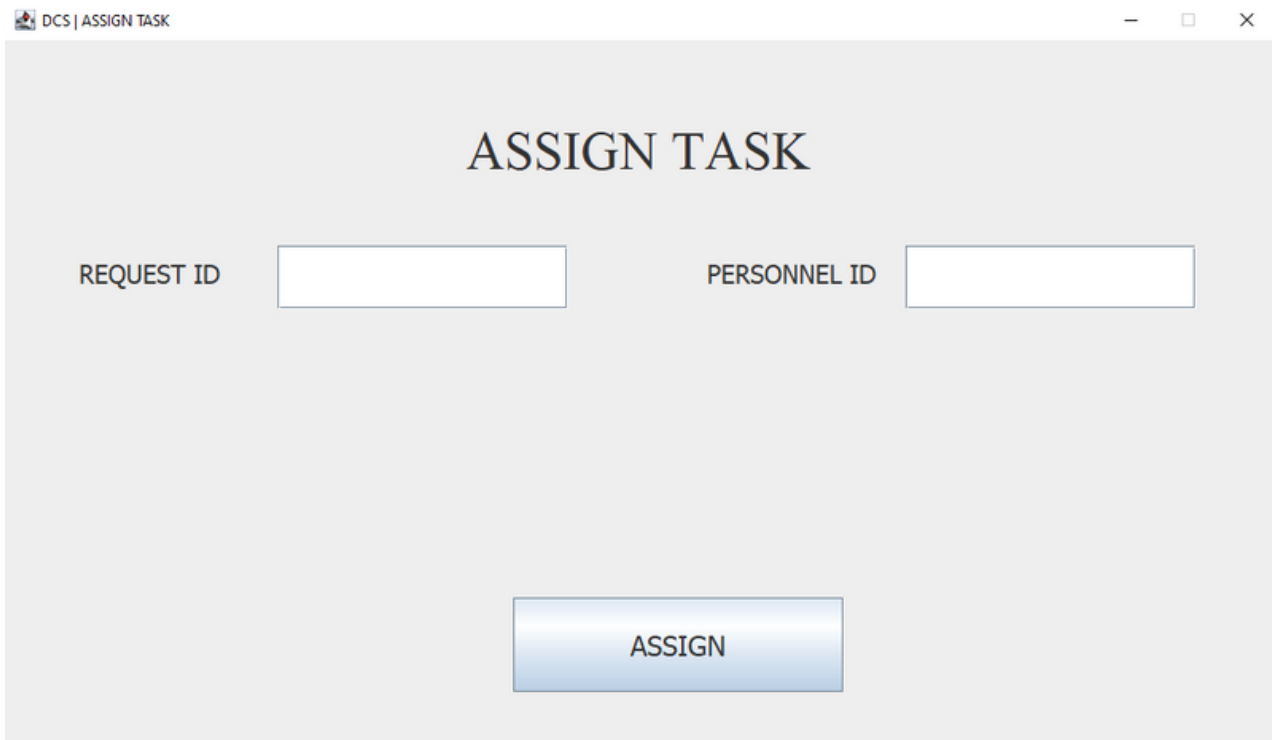
The panel lists and shows all unassigned requests with their unique request id, personal id, status, description and location. The unit notes request id and personnel id then assigns after clicking the assign task button. After all, they can click go back and close the window.



COORDINATION UNIT ASSIGN TASK WINDOW

The window has one button and two text fields. Those fields are REQUEST ID and PERSONNEL ID. The button is the ASSIGN button

Request ID field must be filled with the unique request id. The personnel ID field must be filled with the personnel id who will be assigned to the request.



DCS | ASSIGN TASK

ASSIGN TASK

REQUEST ID

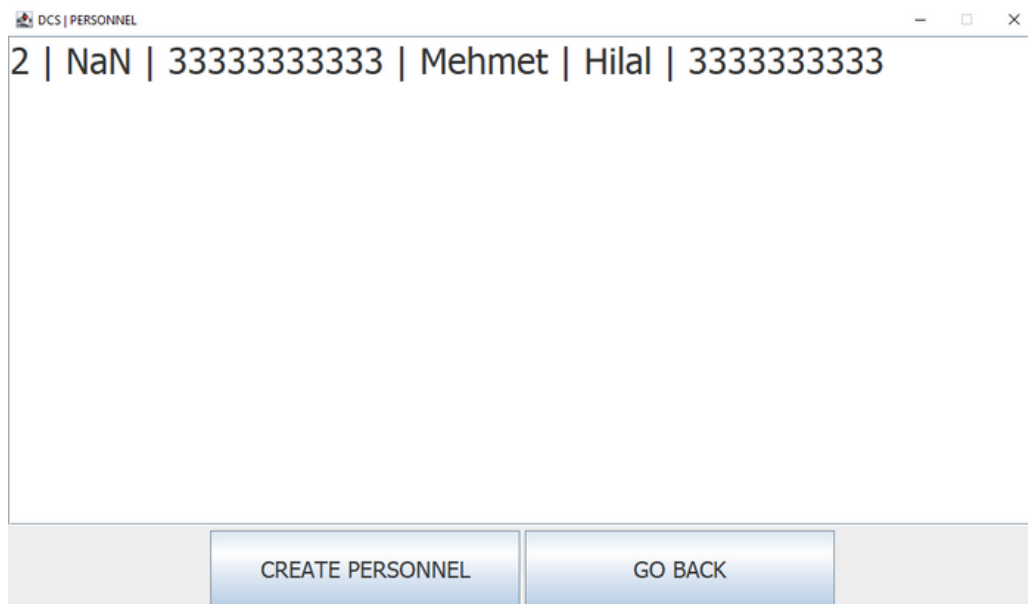
PERSONNEL ID

ASSIGN

COORDINATION UNIT PERSONNEL WINDOW

The window has 2 buttons. Those are CREATE PERSONNEL and GO BACK. Also there is a textfield that lists the personnels.

CREATE PERSONNEL button opens personnel window. Personnel window allows coordination units to add personnels with their ID number and phone number. GO BACK button opens the previous window. (Coordination Unit Main Window). The textfield displays all the personnels. Their personal info is displayed such as (ID, Task ID, ID Number, First Name, Last Name, Phone Number)

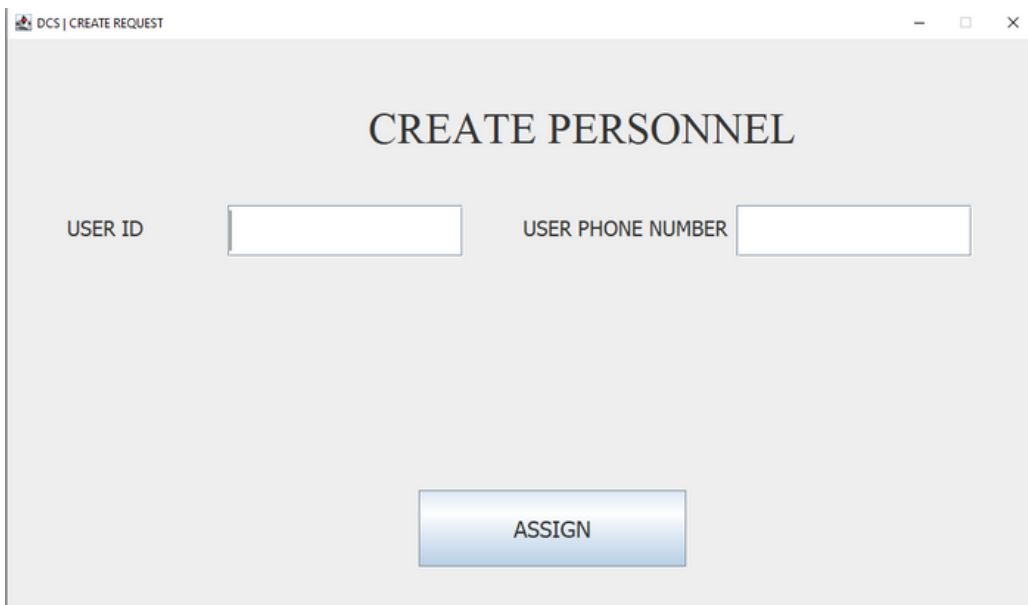


Screenshot: Personnel window

CREATE PERSONNEL WINDOW

The window has 2 textfields to add info and a button to save.

Coordination units have to enter User ID and Phone Number to create a new personnel.

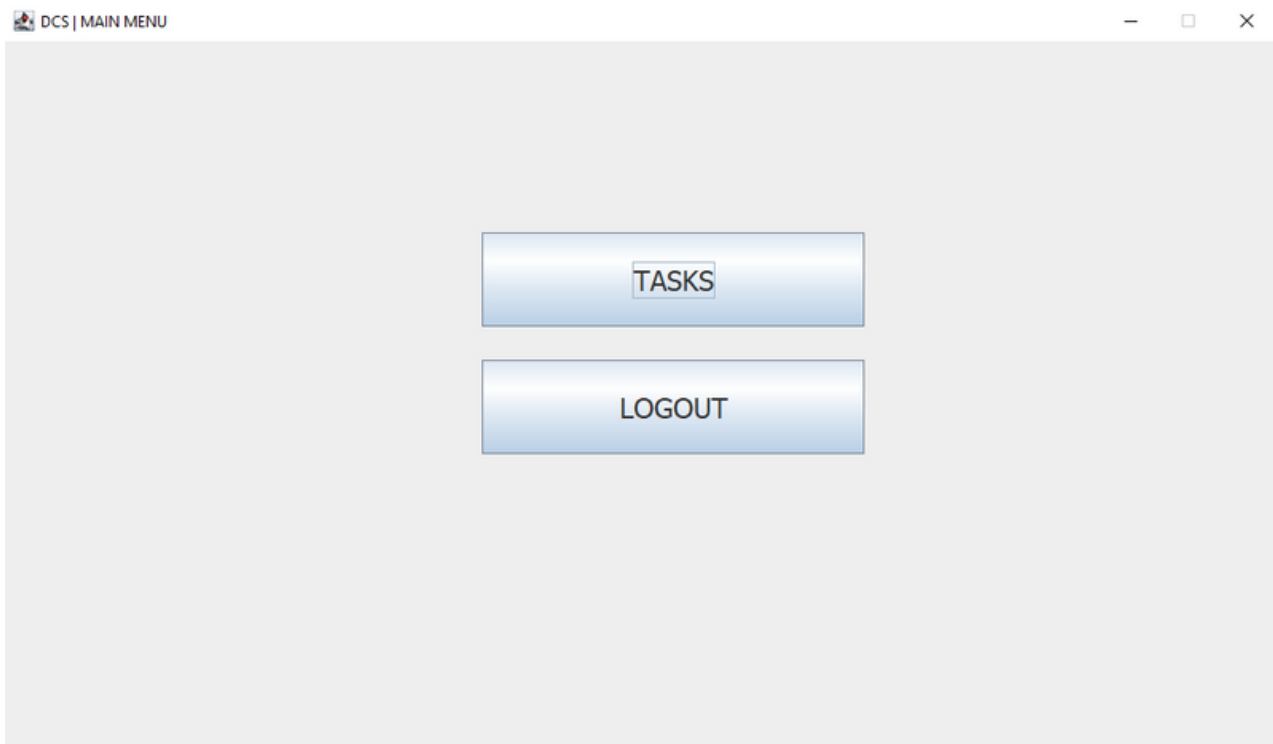
A screenshot of a software window titled "DCS | CREATE REQUEST". The window has a light gray background and a title bar with standard minimize, maximize, and close buttons. The main content area is titled "CREATE PERSONNEL" in a large, bold, black serif font. Below the title, there are two text input fields. The first field is preceded by the label "USER ID" in a black sans-serif font. The second field is preceded by the label "USER PHONE NUMBER" in a black sans-serif font. Both fields are empty and have a thin gray border. Below these fields, centered horizontally, is a blue button with a white border and the word "ASSIGN" in white, bold, sans-serif capital letters.

Screenshot: Create personnel

PERSONNEL MAIN MENU WINDOW

The window has 2 buttons. Those are TASKS, LOG OUT.

The tasks shows all tasks those assigned to the personnel. Log out ends the session and returns to login window.

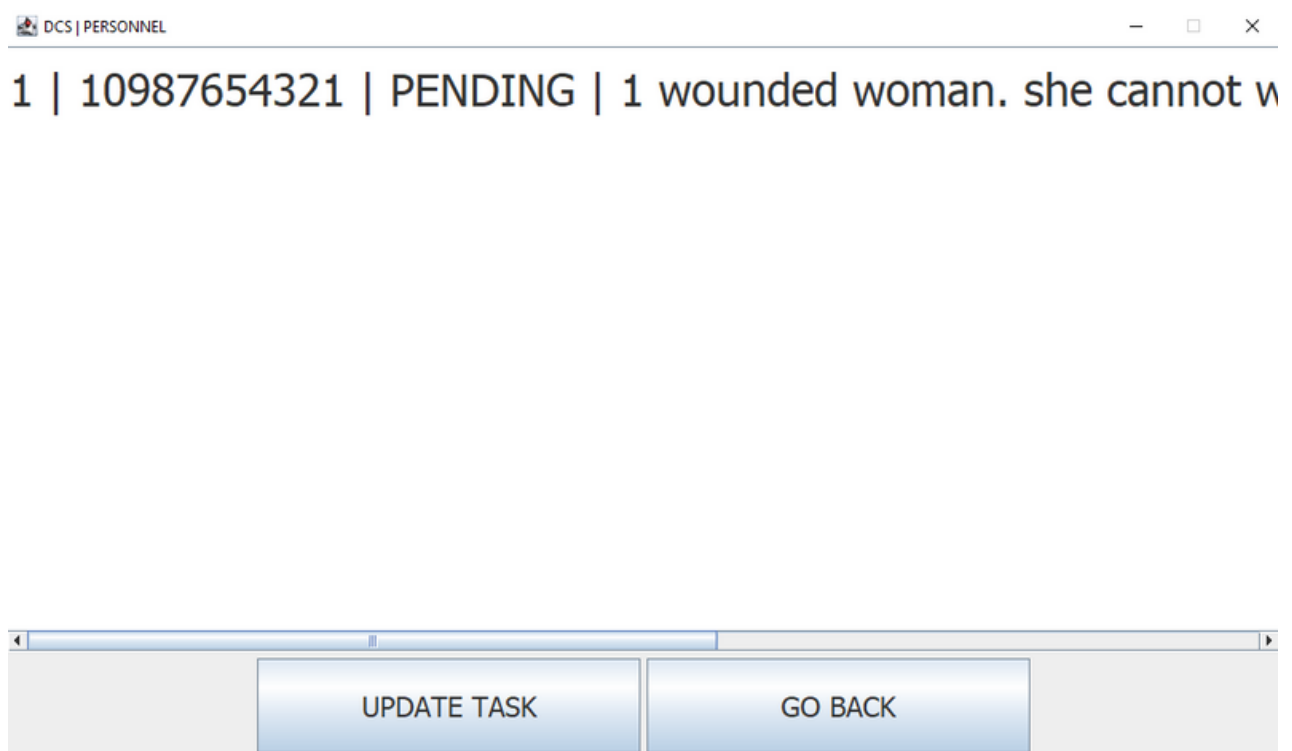


Screenshot: Personnel main menu

PERSONNEL TASKS WINDOW

The window has two buttons and one panel to show a list. Those buttons are UPDATE TASK and GO BACK.

The panel lists and shows all tasks with their unique request id, owner id, status, description and location. The personnel can update task status by clicking update task. After all, they can click go back and close the window.

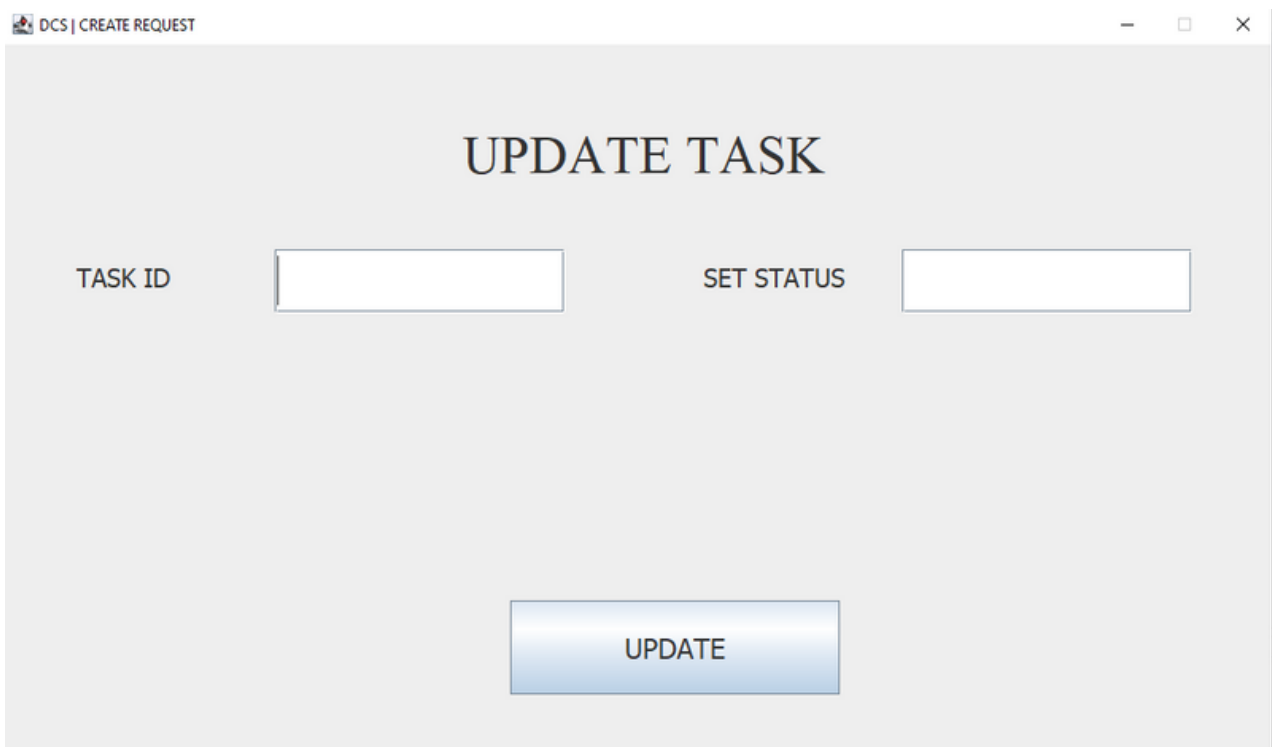


Screenshot: Personnel task window

PERSONNEL TASKS WINDOW

The window has one button and two text fields. Those fields are TASK ID and SET STATUS. The button is the UPDATE button

The task ID field must be filled with the unique task id. The set status field must be filled with "DONE" or "FAILED". When the status of the task is updated, it is processed in the database.



DCS | CREATE REQUEST

UPDATE TASK

TASK ID

SET STATUS

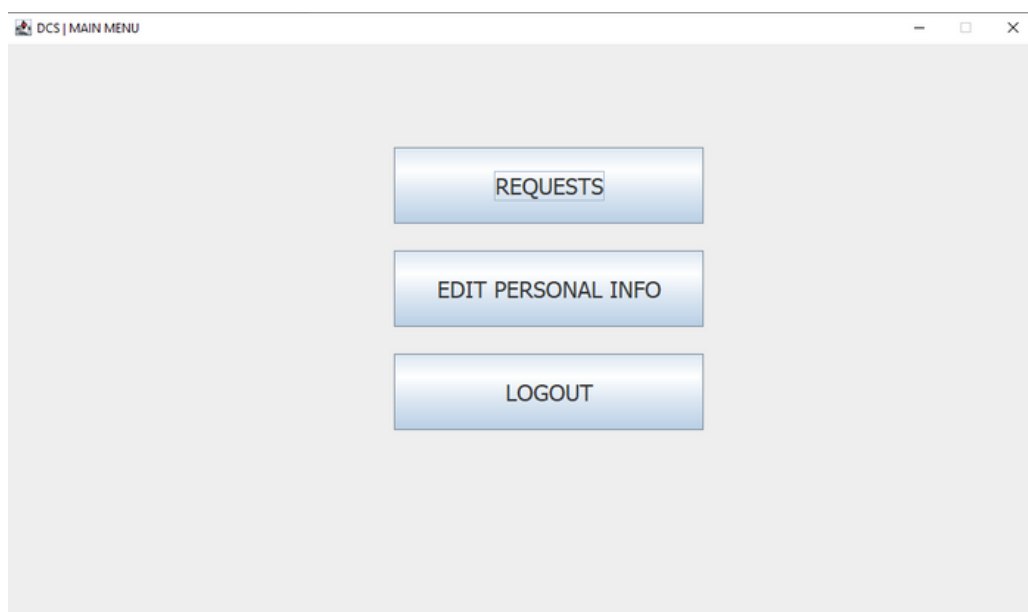
UPDATE

Screenshot: Task update window

USER MAIN MENU WINDOW

The window has 3 buttons: REQUESTS, EDIT PERSONAL INFO, LOGOUT.

REQUESTS button displays the current requests and their status.
EDIT PERSONAL INFO button allow the users to edit their email, password and phone number info.
LOGOUT returns back to login page.

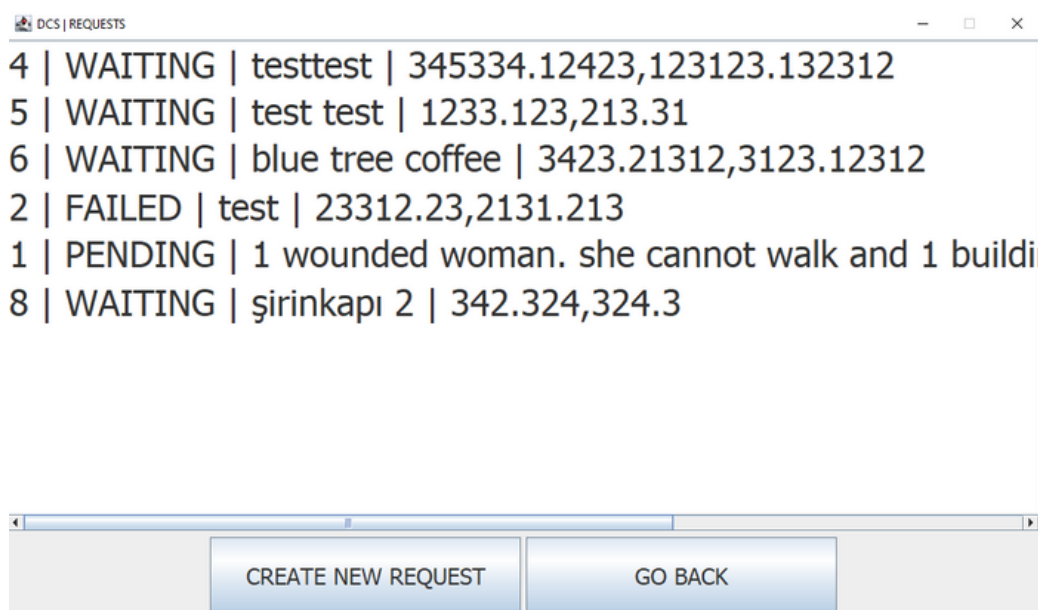


Screenshot: User main menu

USER REQUESTS WINDOW

The window has 2 buttons. Those are CREATE PERSONNEL and GO BACK. Also there is a textfield that lists the user requests.

CREATE NEW REQUEST button opens request window. Request window allows users to add requests with a description and coordinates (Latitude and Longitude).
GO BACK button opens the previous window. (User Main Window).
The textfield displays all the user's requests. Status is displayed as SUCCESS, FAILED, WAITING or PENDING. Also other details of requests are shown (Request ID, description, coordinates).

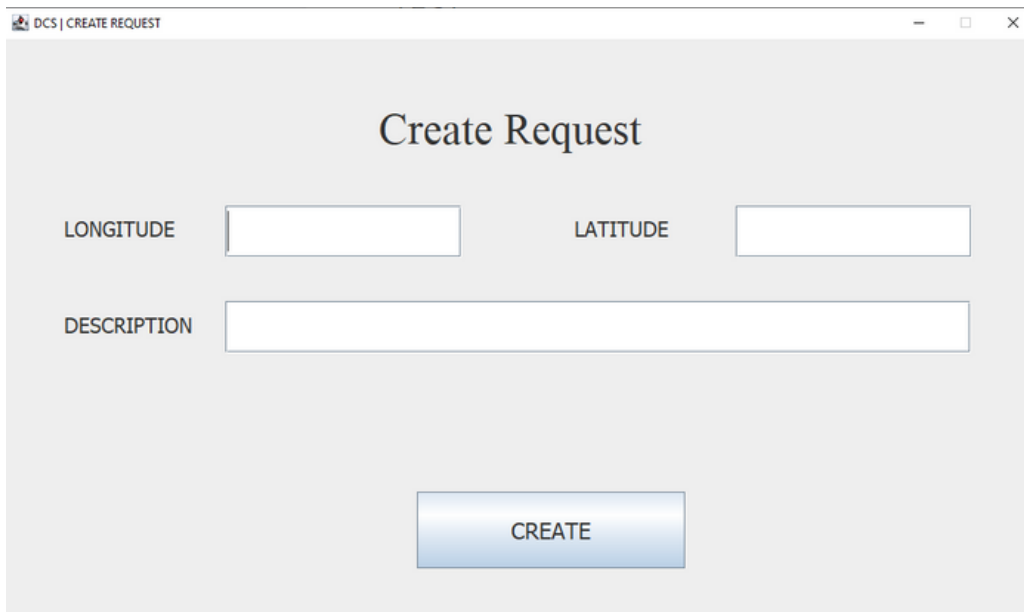


Screenshot: User requests menu

USER CREATE REQUEST WINDOW

The window has 3 textfields to add info and a button to create the request.

User has 2 separate fields to add location details (longitude and latitude). It is planned to get location info automatically from the device in the future. Another textfield is to add a description of the request. Clicking the button will save the request and send it to the system.

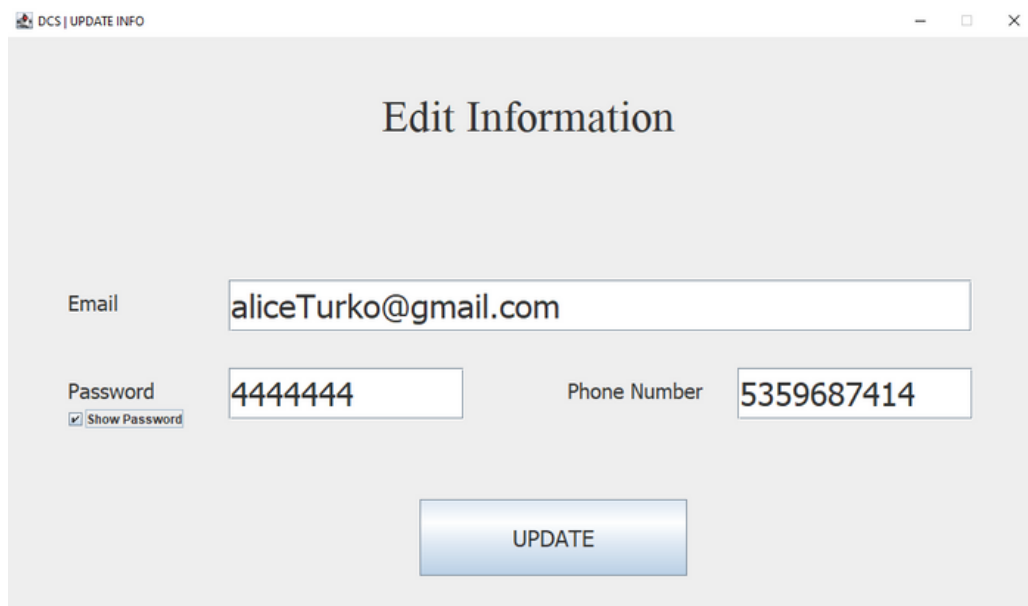
A screenshot of a software window titled "DCS | CREATE REQUEST". The window has a light gray background and a title bar with standard window controls. The main content area is titled "Create Request" in a large, dark font. Below the title, there are three input fields: "LONGITUDE" and "LATITUDE" are side-by-side, each with a small rectangular text box, and "DESCRIPTION" is below them with a larger rectangular text box. At the bottom center of the window is a blue button with the word "CREATE" in white capital letters.

Screenshot: Create request menu

USER EDIT INFORMATION WINDOW

The window has 3 textfields to edit info and a button to save the changes.

User has 3 fields to edit personal info (Email, Password and Phone Number). Clicking the button will save the changes and send it to the system.



DCS | UPDATE INFO

Edit Information

Email

Password ☒ Show Password

Phone Number

Screenshot: Edit info menu

SEMINAR PATTERNS

The patterns that were implemented in the project: Singleton and The Builder pattern.

Currently, in the implementation of the project, our team used two main approaches, one of them is the singleton pattern and the other one is the builder pattern.

The singleton pattern is mainly used in the database class (DB). The implementation utilizes a function "getInstance()" and a static reference to a database instance to implement the mechanism.

In the implementation, all of the references to the static database variable is controlled by the getInstance() method.

The other approach our team used in the project is the builder pattern. The builder pattern is used for building a PersonalInfo object, the personal info object is partially built by its properties in a PersonalInfoBuilder instance.

The future of disaster coordination systems holds great promise as advancements in technology and data-driven approaches continue to reshape emergency response efforts. One significant aspect of future work lies in the integration of artificial intelligence (AI) and machine learning algorithms into these systems. AI can analyze vast amounts of data in real-time, enabling early detection of potential disasters, predicting their impact, and suggesting optimal response strategies. This predictive capability can enhance preparedness and facilitate proactive decision-making by identifying vulnerable areas, prioritizing resource allocation, and optimizing evacuation routes. Furthermore, the integration of Internet of Things (IoT) devices can provide real-time data from sensors placed in disaster-prone areas, allowing for faster and more accurate situational awareness. Additionally, the utilization of drones and autonomous vehicles can revolutionize disaster response by providing aerial surveillance, delivering supplies to inaccessible areas, and conducting search and rescue missions. Collaborative platforms and social media integration can empower communities to actively participate in disaster response efforts, sharing information, and coordinating grassroots initiatives. Overall, the future work of disaster coordination systems holds the potential to harness technology, big data, and community engagement to enhance preparedness, response, and recovery in the face of disasters. We aim to pioneer these systems with DCS..

Disaster coordination systems play a pivotal role in safeguarding communities and minimizing the devastating impact of natural and man-made disasters. These systems serve as a vital lifeline for effective emergency response, facilitating the coordination and collaboration between various stakeholders such as government agencies, first responders, non-governmental organizations, and community members. By establishing clear communication channels, sharing critical information, and mobilizing necessary resources, disaster coordination systems enable swift and well-organized response efforts. They ensure timely warnings and evacuations, enhance search and rescue operations, and facilitate the delivery of essential supplies and services to affected areas. Moreover, these systems facilitate the sharing of best practices and lessons learned, promoting continuous improvement in disaster preparedness and response strategies. The importance of disaster coordination systems cannot be overstated, as they are instrumental in saving lives, reducing damage, and fostering resilience in the face of adversity.

BY

Yusuf Gassaloğlu
2020510034

Berkay Dinç
202051003

Alperen Aydın
2021510008