DOKUZ EYLÜL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

CME 2210
Object Oriented Analysis and Design

CEMETREE

by
Boran Bereketli – 2022510105
Bedirhan Yenilmez – 2022510159
Alperen Dönmez – 2022510113
Ramazan Denli - 2022510111

IZMIR
01.05.2024

CHAPTER ONE

INTRODUCTION

Cemeteries have a problem of being unorganized and hard to navigate. Some graves are even unknown. This project aims to solve this by providing a way to store and manage people's and cemetery's information. This is a cemetery management and navigation system designed to help finding where people are buried to and get information about them.

The system is designed to store cemetery information and buried people's information and assist finding and choosing a cemetery. It will be able to connect people, find where their relatives are buried to and indicate their location.

The information about buried people (name, surname, birthdate, death date, relatives, message, death cause etc.) is stored in a data structure and can be saved and loaded from a file. The records of visitors(name, surname, date etc.) can be kept by the system.

The people can be filtered by their name, surname, age, cause of death. Some statistics such as number of people who died of the same cause, recent deaths, oldest people, top death causes can be provided according to the user's requests.

This software can be utilized by city managements or countries.

CHAPTER TWO

REQUIREMENTS

2.1 External Interfaces

- Users and administrators can interact with the system.

- Text files are used to read/write data.

2.2 Functions

Functions in Cemetree class:

- void addPerson(Person person): This function adds a new person to graph and to the cemetery where they are buried.

- void removePerson(string id): Removes the person from graph and the cemetery where they are buried.

- Person getPerson(string id): Returns the person with given id.

- void updatePerson(Person person): Updates the person's informations.

- List<Person> searchPeopleByFilter(Person filter): Returns all people conforming the filter.

- List<Person> searchPeopleByDate(Date startDate, Date endDate): Returns people who died between startDate and endDate.

- List<Person> searchRelatives(int generationDifference): Returns the dead relative informations in the given interval.

- void printPeople(ArrayList<Person> people): Prints people in a formatted string.

Functions in Cemetery class:

- boolean isFull(): Returns if the cemetery is full.

- ArrayList<Person> getVisitorList(): Returns visitor list of the cemetery.

- ArrayList<Person> getVisitors(Person person): Returns the visitors of the selected person.

- void printVisitorList(): Prints the visitor list in a formatted string.

Functions in Address class:

- String toString(): Return the address in string format.

Functions in Person class

- Getter and setters of name, surname, id, sex, birthDate, motherId, fatherId, dead, admin, deathDate, cemetery, deathCause

2.3 Performance Requirements

- Relatives are expected to be found in less than 20 seconds.

- Visitor list search must take less than 10 seconds.

- The program is expected to search for people in less than 15 seconds.

2.4 Logical Database/File System Requirements

- The program facilitates reading and writing operations from a text file.

2.5 Design Constraints

- Cemeteries are designed to include 20000 people at most.
- Distant relatives cannot be searched.
- Only administrators can add or remove dead people.

2.6 Software System Quality Attributes

- HashMap and Graph data structures are used to lower wait times.
- Software must be reliable, maintainable, efficient, and testable.
- Software mustn’t crash under any circumstances and give error messages.

2.7 Object Oriented Models

- 2.7.1 Analysis Class Model (Static Model)

- Cemetree class is used to store all people.
- Cemetery class stores all information about the cemetery.

- Person class is used to represent a person.

- Address class stores an address.

- 2.7.2 Analysis Collaborations (Dynamic Model)

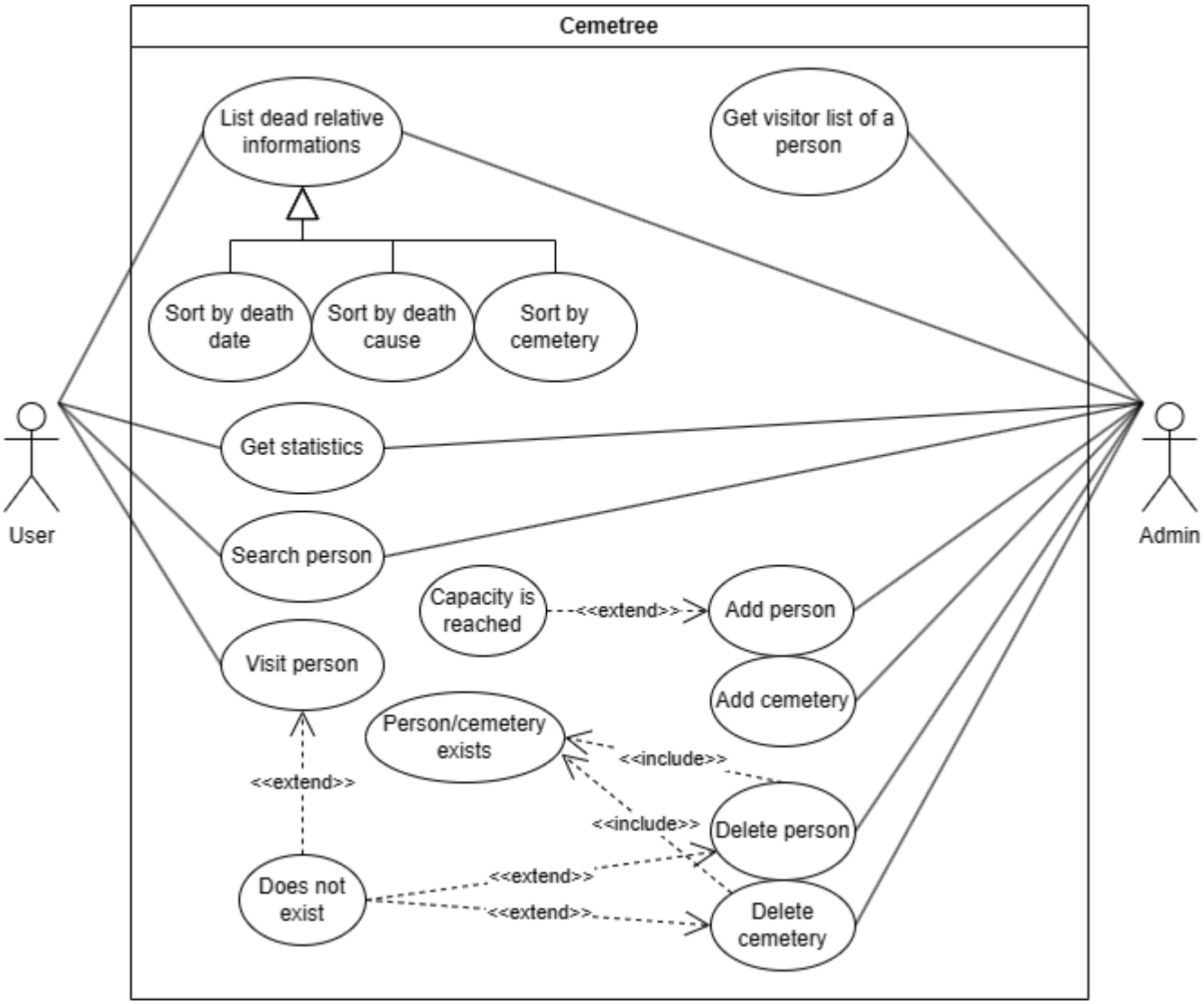
- Cemetree is the main class and stores all the cemeteries and all people with connections.

- All people’s information is manipulated from the Cemetree class.

CHAPTER THREE

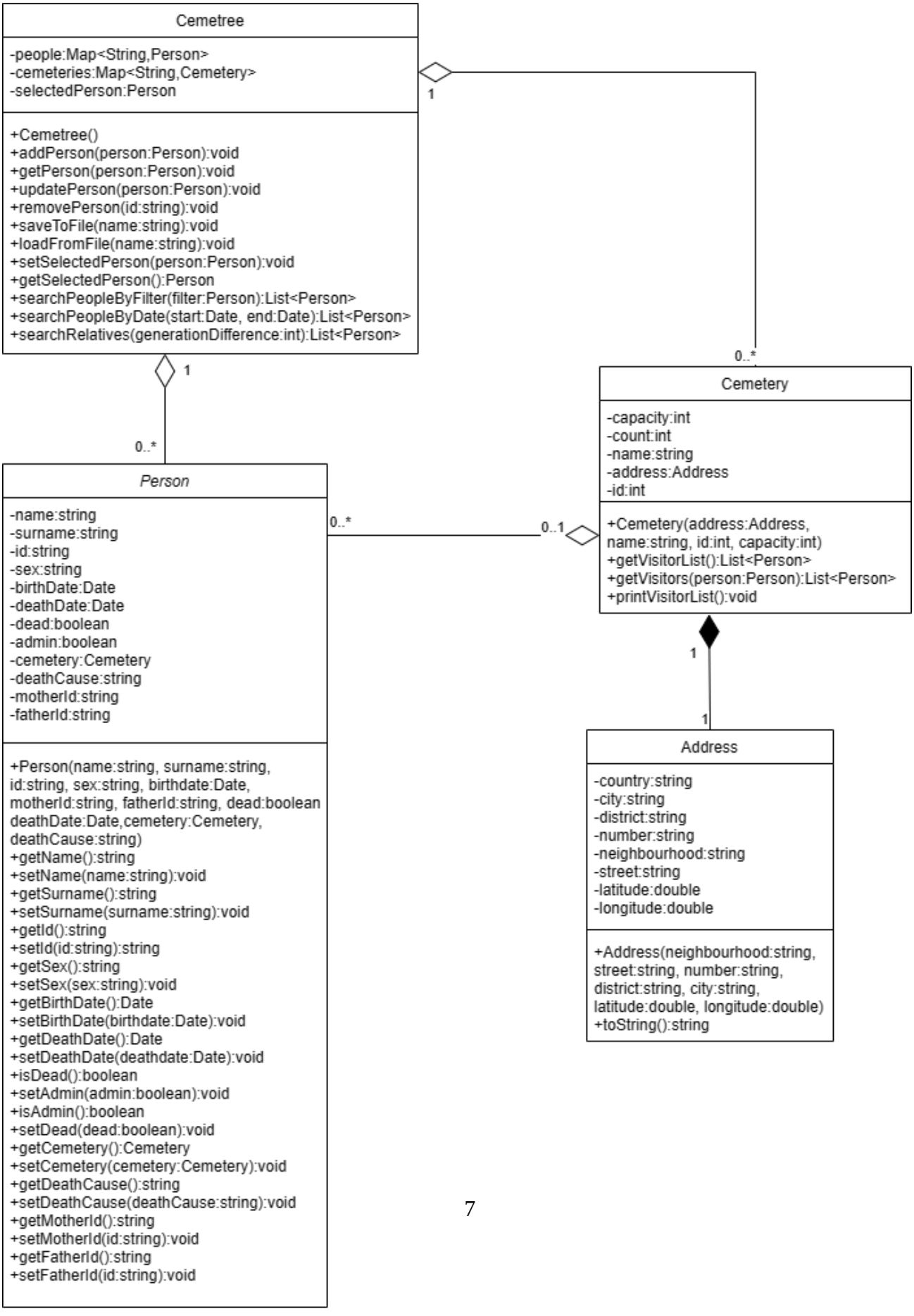
UML DIAGRAMS

USE CASE DIAGRAM



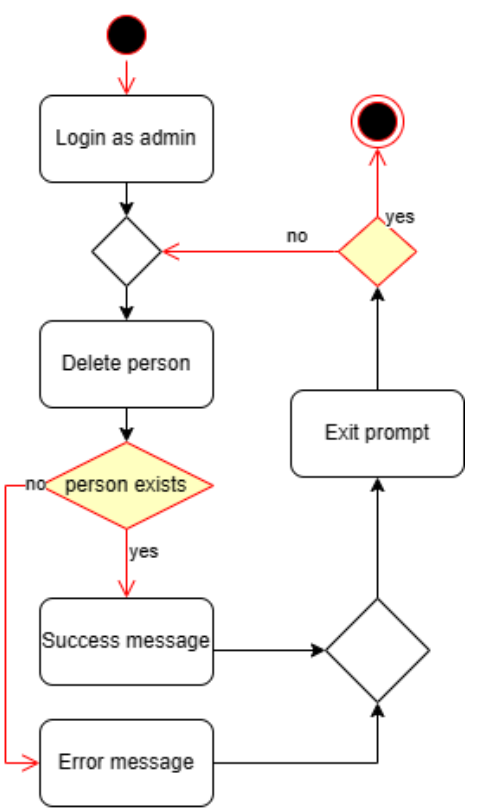
This diagram depicts the general operations the user and admin actors can use. While the user can list relatives, view statistics, search people; the admin can additionally view the visitor records, add people, add cemeteries, delete people and delete cemeteries. Various limits and conditions are also shown like the cemetery capacity and the existence of the person or cemeteries.

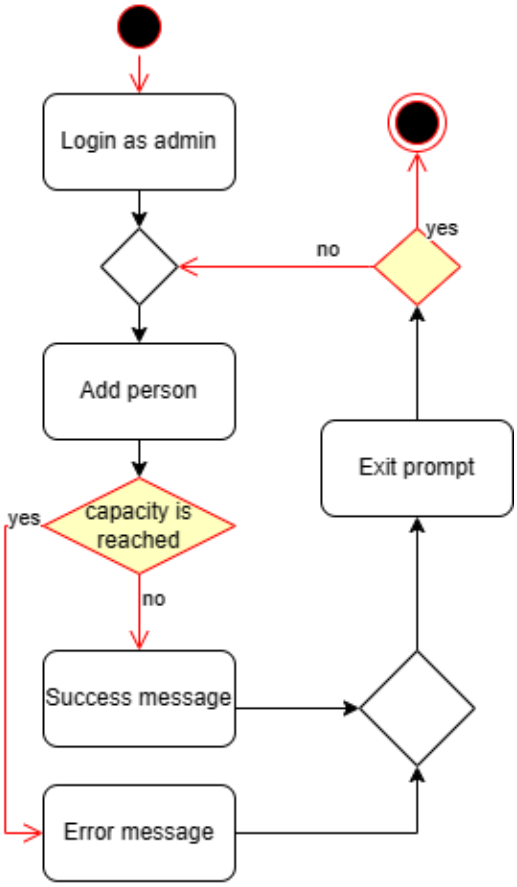
CLASS DIAGRAM

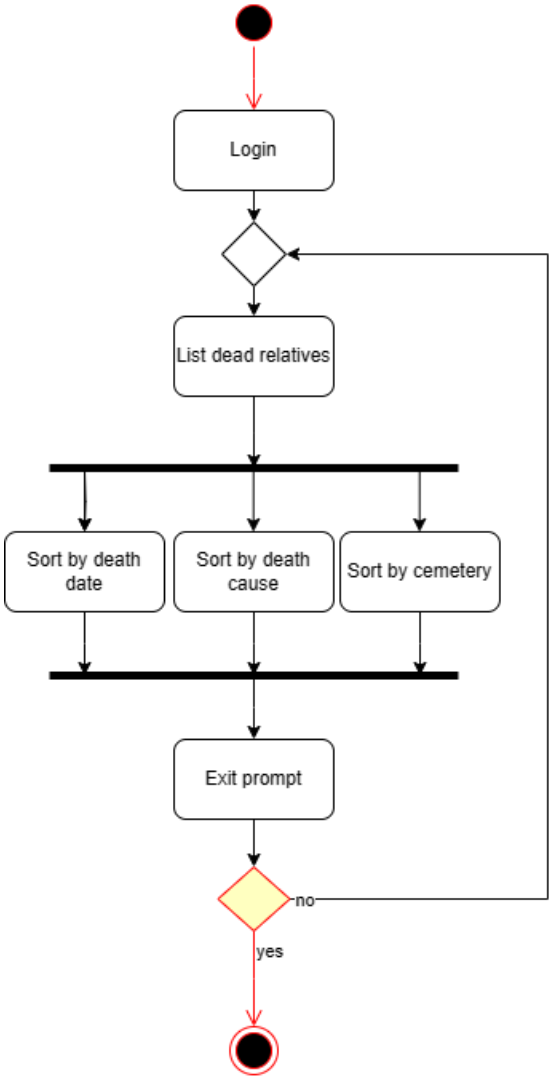


The class structure of the program is shown in the diagram above. The *Cemetree* class contains all people and cemeteries. Zero or more cemeteries and people can be contained in the class. It is the class where all objects are managed from. The other classes hold the data of their respective types. *Person* and *Cemetery* classes hold person and cemetery data. Each cemetery has exactly one address.

ACTIVITY DIAGRAM







Each activity needs login as the first action.

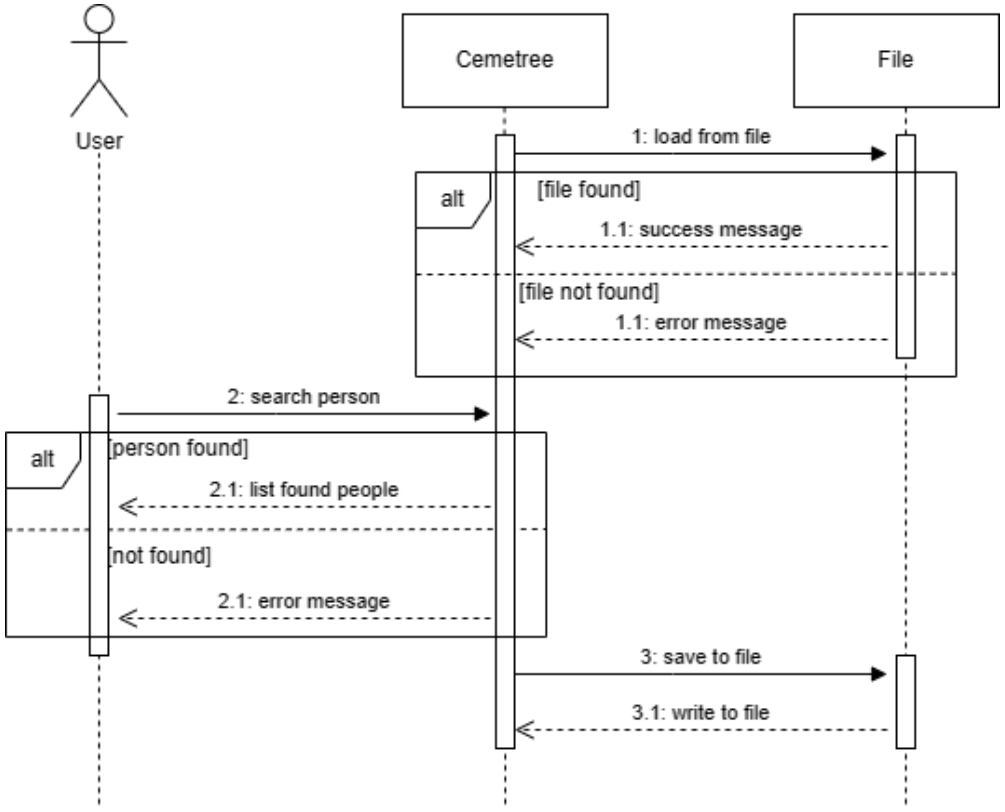
The deletion of people checks the existence of the person, deletes the person and shows a message to the user.

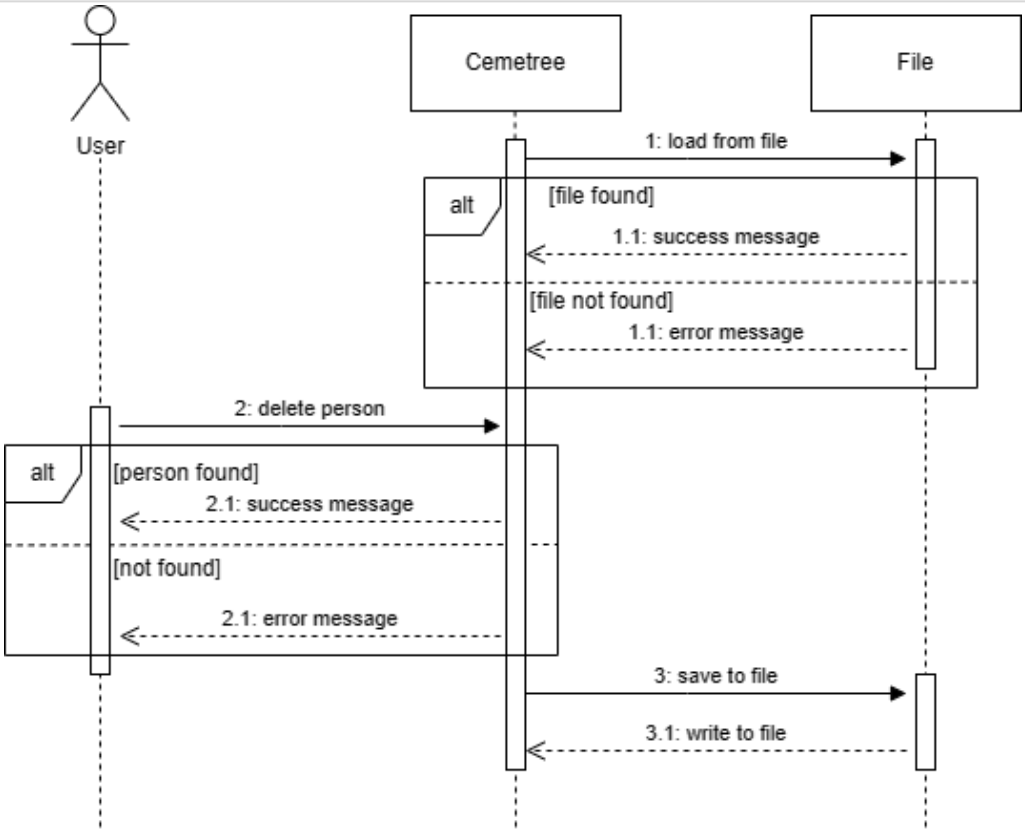
The addition of people checks the cemetery capacity, adds the person and shows a message to the user.

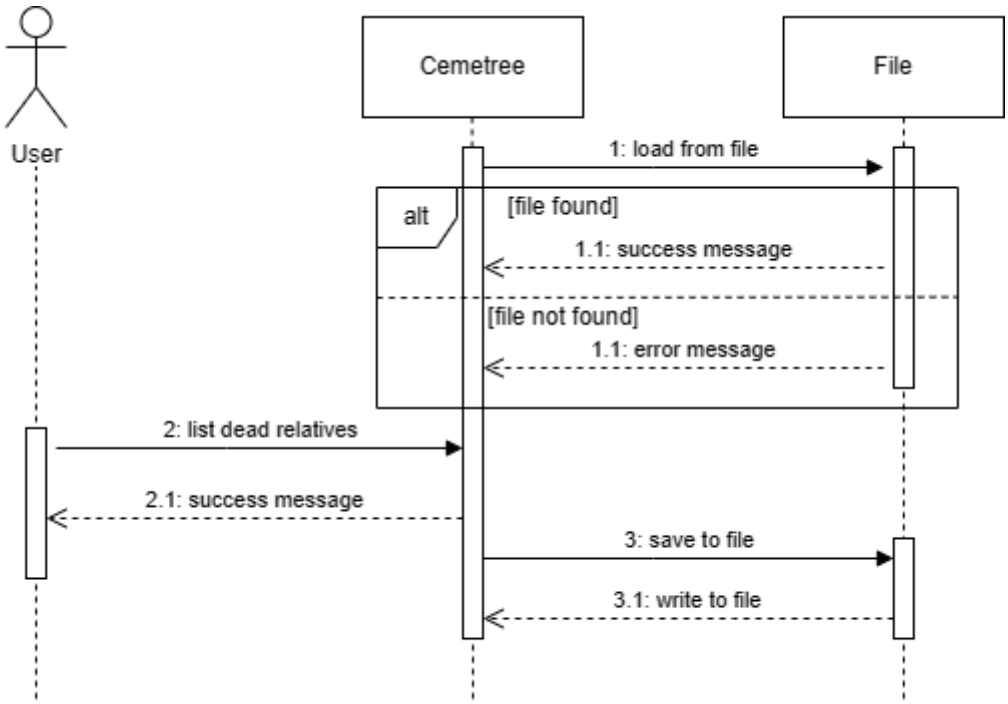
The results of the list function can be sorted by death date, death cause or cemetery.

An exit prompt is shown after all actions.

SEQUENCE DIAGRAM







Each diagram has loading from the file as the first action. It returns a message after the file is read.

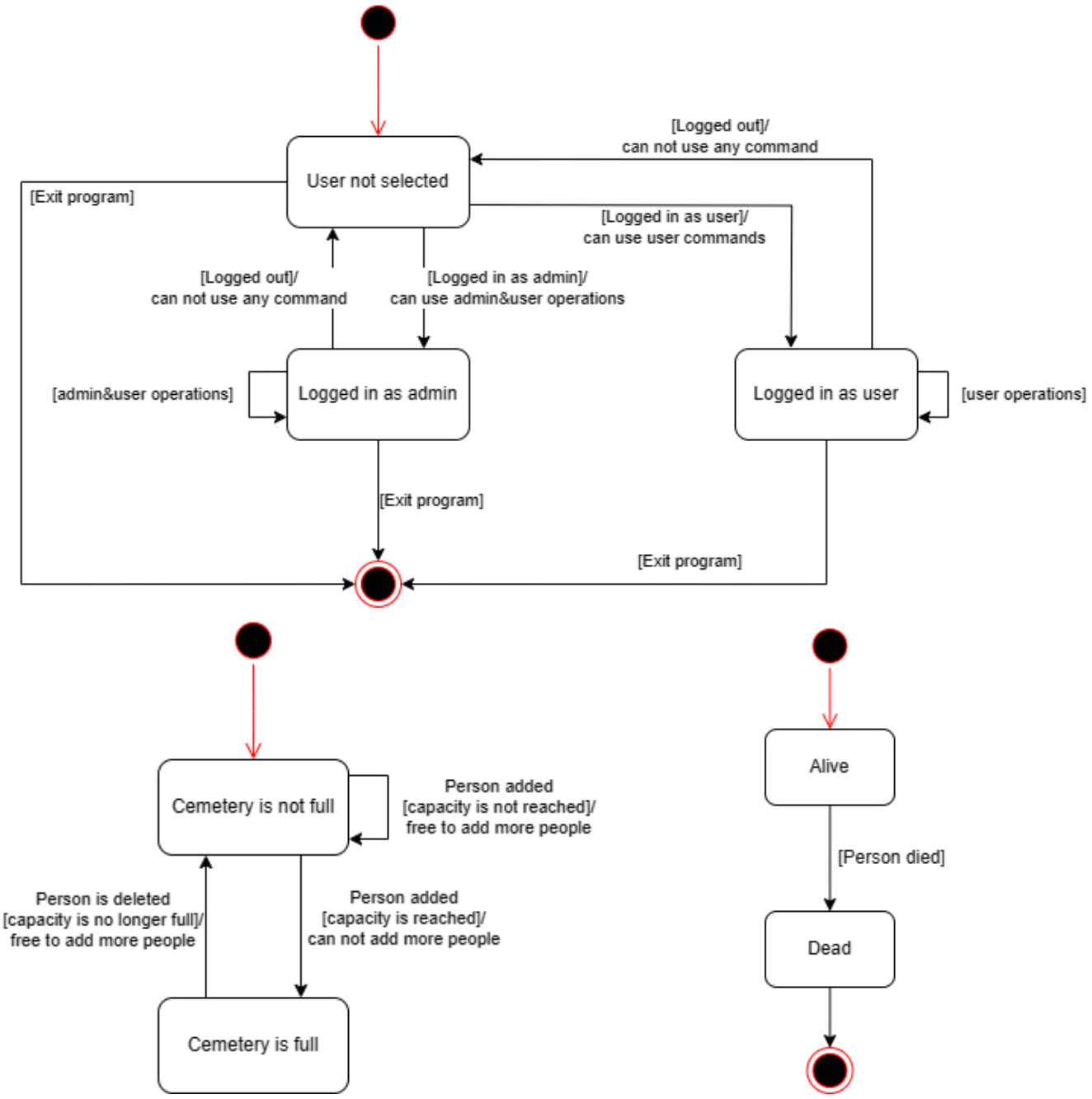
The first diagram shows the search operation. It shows the list of the matched people.

The second shows the delete operation. It checks if the person exists and returns a message according to the result.

The last diagram depicts the list dead relatives function.

All diagrams save the changes to file and show a message according to the result as the last action.

STATE DIAGRAM



The first diagram shows the login states. The user can login either as user or as admin. User and admin have their respective operations they can use. They can exit the program or log out at any time.

The second diagram shows the cemetery capacity states. It is either full or not full. People can be added when it is not full.

The last diagram shows the state of the person.