Requirements Analysis Document

Employee Scheduling System
CSCI 4711 Software Engineering
Fall 2016
Augusta University
Augusta, GA
Date: 24 October 2016
Version 3

Team Members

Chris Gonsalves
Matt Tennis
Connor Williams
Ryan Mahoney

Abstract

This document contains the requirements, analysis and design artifacts for the Employee Scheduling System (ESS) software system. ESS is a personnel scheduling system that facilitates the employee submission and subsequent supervisor approval or denial of time off requests.

The rest of this document is structured as follows: Chapter 1 contains the introduction. This chapter presents a brief description of the system. Chapter 2 outlines the functional requirements of the system. In addition, Chapter 2 contains use case diagrams and use case descriptions for all use cases involved in ESS. Chapter 3 illustrates key GUI screen mockups for the Employee Scheduling System. Chapter 4 contains graphical representations of class design and object interactions. Chapter 5 depicts a decomposition of the ESS subsystems, while chapter 6 is the appendix.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope of System

The Employee Scheduling System (ESS) is a system used to provide simple and efficient means for an employee to request time off and for appointed supervisors to administrate, approve, or deny those requests. ESS has an internal database with authorized users and their password hashes. Employees can submit requests for time off, which are stored in the database. Supervisors are then able to see the contents of the time off requests, the employee that initiated it, and the reason for the request. Once the Supervisor responds to a request, it is removed from the Supervisor's queue and the database. The system includes secure login, logout functionality in addition to the primary scheduling applications.

1.2 Overview of Document

The rest of the document is structured as follows: Chapter 2 outlines the functional requirements of the system, then the use case diagram. Individual detailed use case descriptions are then listed. Chapter 3 depicts several individual user interface mockups. Chapter 4 displays the class design and object interactions. Chapter 5 shows the ESS subsystem decomposition. Finally, chapter 6 is an appendix of initial implementation of some entity objects.

2 REQUIREMENTS OF SYSTEM

2.1 FUNCTIONAL REQUIREMENTS

- **Login –** All users, **Employees** and **Supervisors**, must supply valid login credentials (EmployeeID and password) to be authorized to access and use the system. Upon doing so, the user will have created a session with ESS, where a user can modify database contents through normal usage. Valid login will direct the user to his or her appropriate activity based on the user's class.
    - **InvalidLogin** – Handle invalid credentials, out-of-scope characters, and exploitation attempts. Returns control to user after job.
- **Logoff –** All users must have clear and immediate access to a Logoff button in order to gracefully and securely close the connection with ESS. Resources allocated to a user session must be terminated in an orderly fashion as to eliminate potential software bugs. Every form or interface must have a clearly marked Logoff button.
- **TimeOffRequest – Employees** must be able to supply a time off request in the Time Off Request form. Employee will select dates via the calendar GUI. Radio buttons enable the **Employee** to indicate the reason (and weight) of his or her request. The user can then submit or logout from that form. Then the system sends time off request to the database.
- **RequestResponse – Supervisors** must be able to view the time off requests that have been submitted in a scroll box queue. The queue will have highlighted regions that correspond to the reason (or weight) supplied by the user's time off request. The **Supervisor** can then approve, deny, or logout from this window. Approvals and denials modify database contents and update the queue, while logout will terminate the session gracefully.
    - **ApproveRequest** – Approves Employee request and updates relevant database data.
    - **DenyRequest** – Denies Employee request and updates relevant database data

2.2 USE CASE DIAGRAM

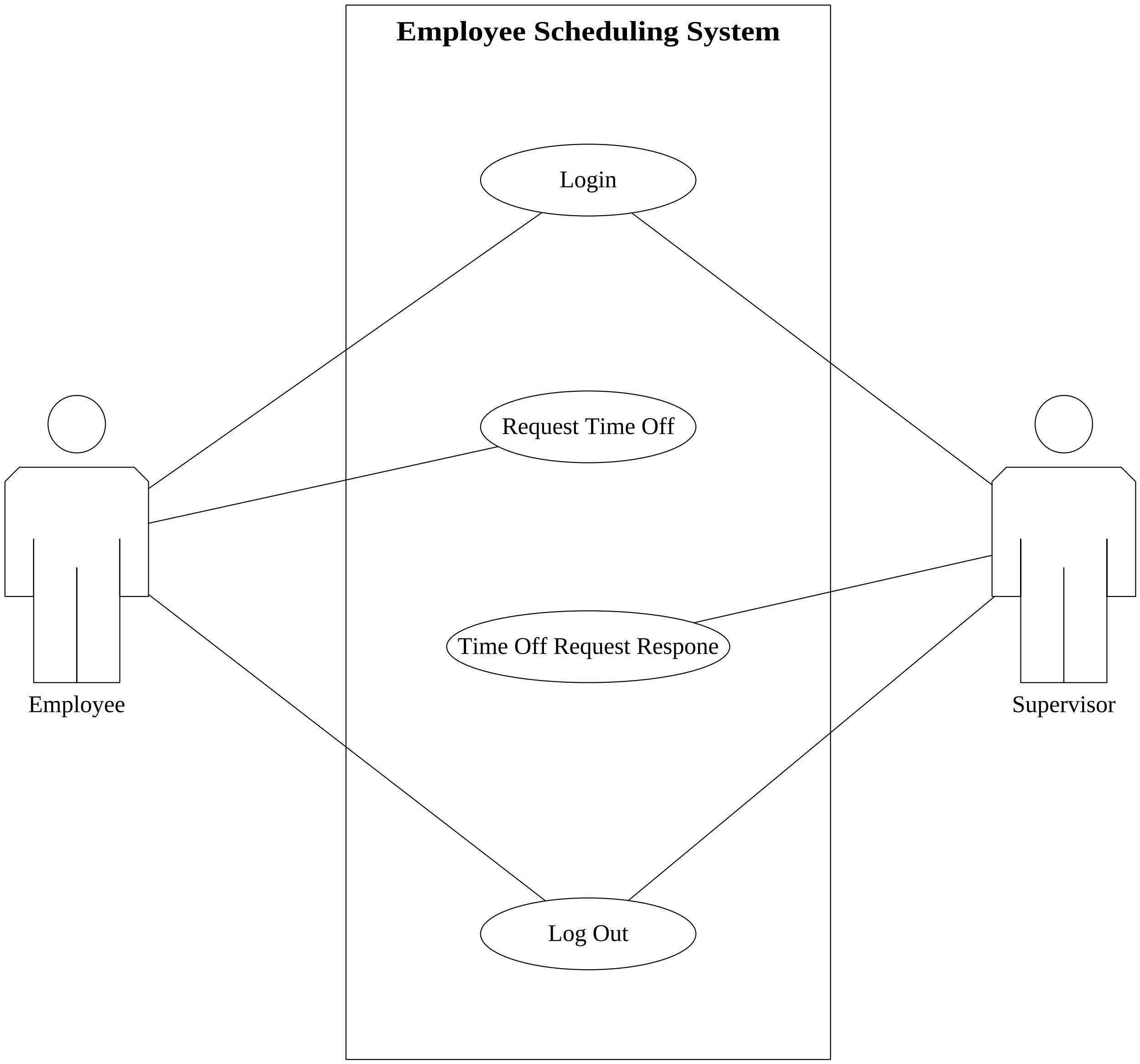


Figure 2.1 use case diagram for ESS

2.3 Use Case Descriptions

Use case name	Login
Participating actors	Initiated by Employee or Supervisor
Flow of events	1. Employee enters their user ID in User ID field and Password in Password field. 2. **ESS responds by authenticating the entered user ID and Password**. **The ESS will distinguish if User is Employee or Supervisor via SQL query. The Time Off Request interface will open for Employee and the Supervisor Menu interface will open for Supervisor.**
Entry condition	
Exit condition	User ID and password are authenticated via SQL query.
Security requirements	The password must be hashed at all times. The dialogue boxes that handle username and password must be shielded against code execution and SQL injections. Password policy must be used to eliminate malicious input. Windows shortcut-key exploits must be disabled to avoid accessing a shell or forcing an exploit. Only <ENTER> will be recognized for acknowledgement of the message/dialogue box.

Figure 2.2: Login: valid login

Use case name	Login
Participating actors	Initiated by Employee or Supervisor**.**
Flow of events	1. User supplies invalid credentials to the login interface. **2. System handles the input, returning a user-specific error in a pop-up message/dialog box. A dialogue box pops up to alert User of invalid login.** 3. The user must acknowledge the button in the dialog/box in order to proceed. 4. 4. **System returns the user to the login page, where the user is then able to try to enter valid credentials once more.**
Entry condition	
Exit condition	The user acknowledges the invalid entry.
Security requirements	The password must be hashed at all times. The dialogue boxes that handle username and password must be shielded against code execution and SQL injections. Password policy must be used to eliminate malicious input. Windows shortcut-key exploits must be disabled to avoid accessing a shell or forcing an exploit. Only <ENTER> will be recognized for acknowledgement of the message/dialogue box.

Figure 2.3: Login: invalid login

Use case name	Logout
Participating actors	Initiated by Employee or Supervisor
Flow of events	1. Employee presses the logout button on the Time Off Request interface or Supervisor presses logout button on Supervisor Menu Form, Time Off Request interface, or Request Response interface. 2. **System closes any open form (Supervisor Dashboard, Time Off Request interface, or Response Request interface) and returns user to the login interface.**
Entry condition	Employee or Supervisor is logged in to the ESS system.
Exit condition	Employee or Supervisor is logged out and returned to the login interface.
Security requirements	Resources allocated to the session must be terminated properly to ensure there are no bugs in the software.

Figure 2.4: Logout

Use case name	RequestResponse
Participating actors	Initiated by Supervisor
Flow of events	1. Supervisor selects the appropriate request from Time Off Request queue on the Request Response interface and clicks the Approve button. 2. **ESS updates Time Off Request status field in database with "Approved".**
Entry condition	The Supervisor selects Time Off Request Response button from Supervisor Menu Form.
Exit condition	Time Off Request status field in database with "Approved".
Security requirements	All responses are tracked by User ID ensuring that no unauthorized individuals are able to surreptitiously gain access to a request.

Figure 2.5: RequestResponse: Approve

Use case name	RequestResponse
Participating actors	Initiated by Supervisor
Flow of events	1. Supervisor selects the appropriate request from Time Off Request queue on the Request Response interface and clicks the Deny button. 2. **ESS updates Time Off Request status field in database with "Denied".**
Entry condition	The Supervisor selects Time Off Request Response button from Supervisor Menu Form.
Exit condition	Time Off Request status field in database with "Denied".
Security requirements	All responses are tracked by User ID ensuring that no unauthorized individuals are able to surreptitiously gain access to a request.

Figure 2.6: RequestResponse: Deny

Use case name	TimeOffRequest
Participating actors	Initiated by Supervisor or Employee
Flow of events	1. The Employee or Supervisor select a date, time and a reason per request. 2. **2. ESS receives the form and pushes following fields to the database tables: Employee/Supervisor name, request date, request time, and request reason.**
Entry condition	The Supervisor selects Request Time Off button from Supervisor Menu Form.
Exit condition	The employee's time off request is reflected in the appropriate employee and supervisor queues.
Security requirements	

Figure 2.7: TimeOffRequest

2.4 Requirements Analysis

This page intentionally left blank

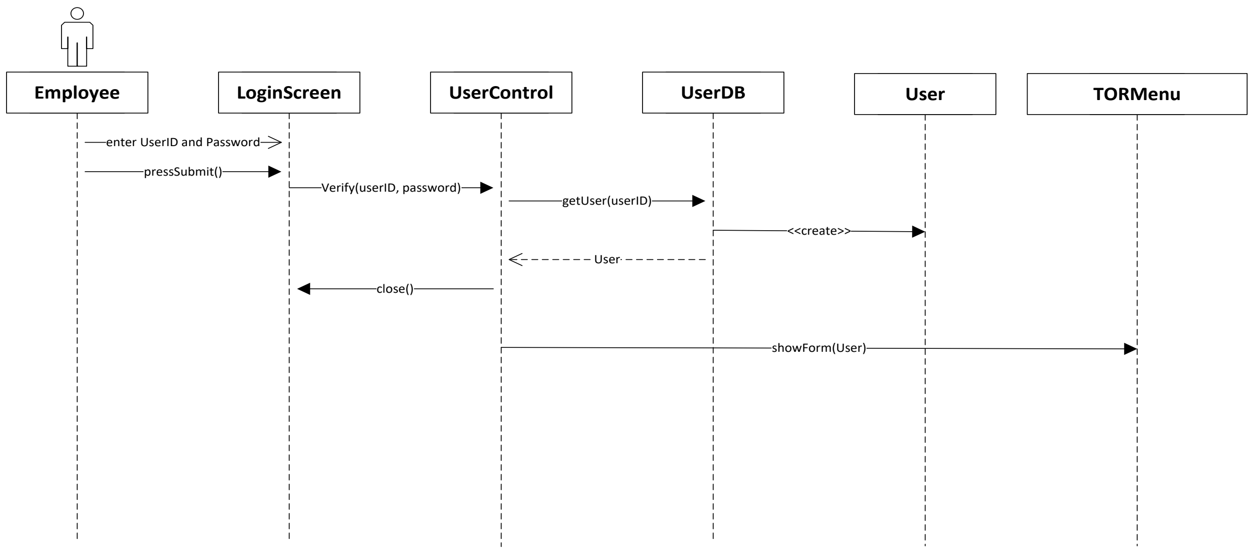


Figure 2.8: EmployeeLogin (non-supervisor) sequence

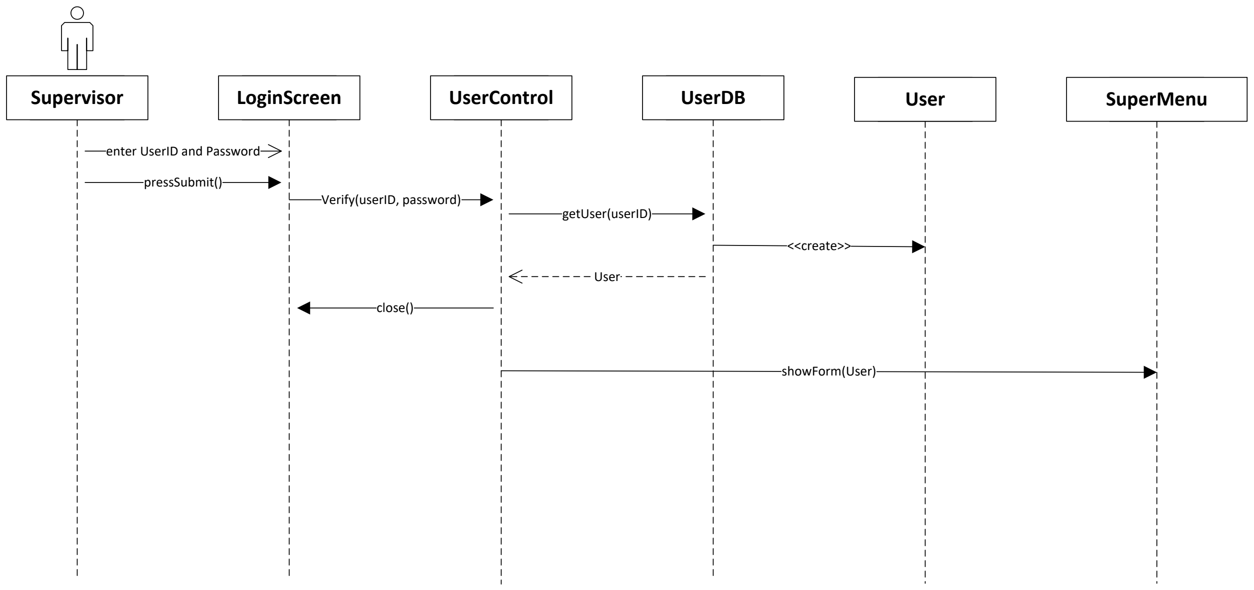


Figure 2.9: EmployeeLogin (supervisor) sequence

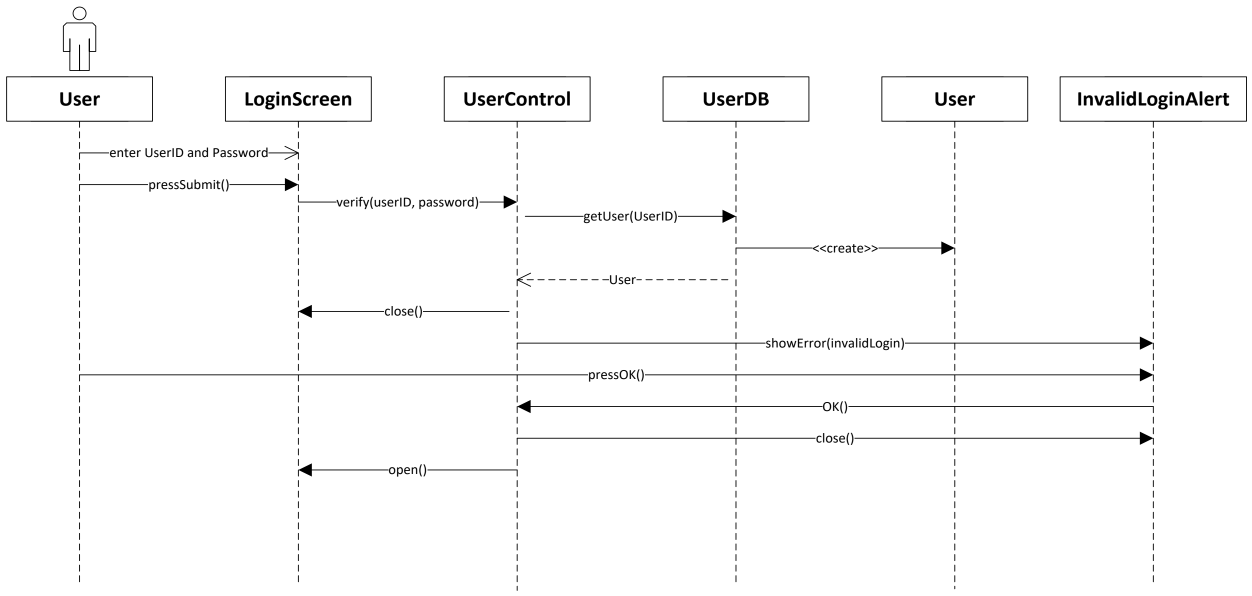


Figure 2.10: InvalidLogin sequence

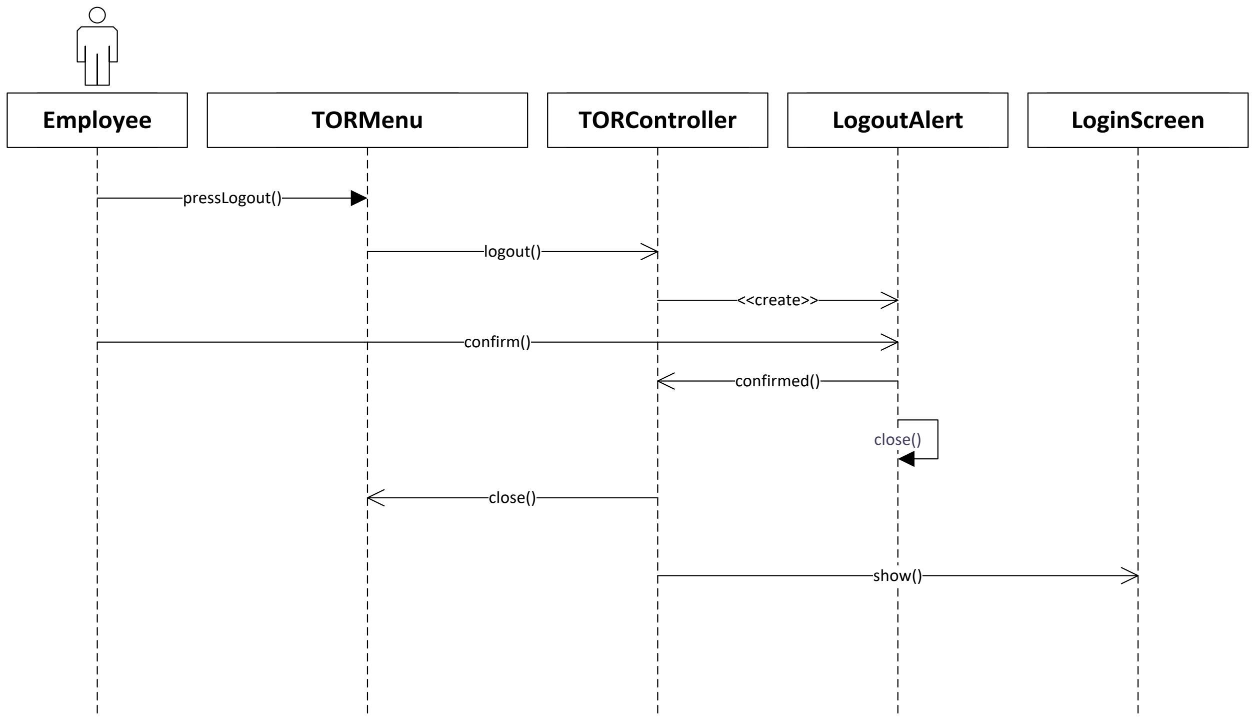


Figure 2.11: TORLogout sequence

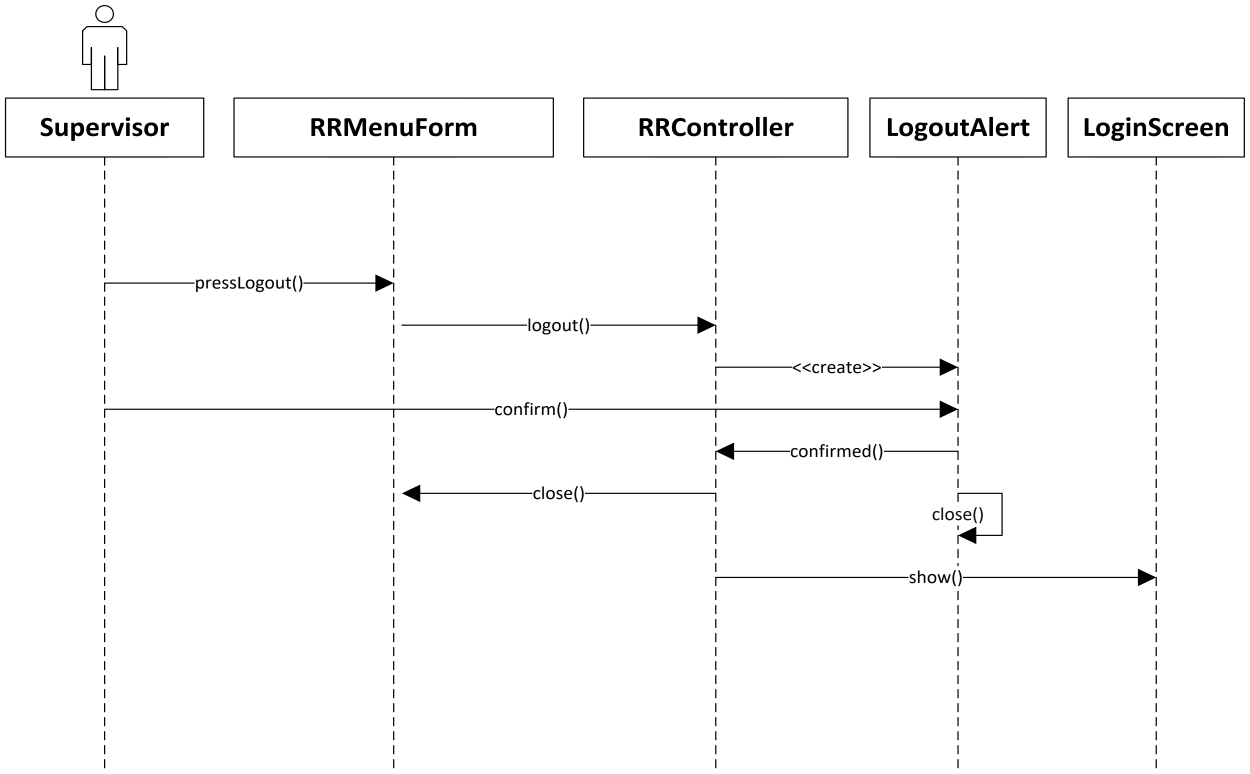


Figure 2.12: RRLogout sequence

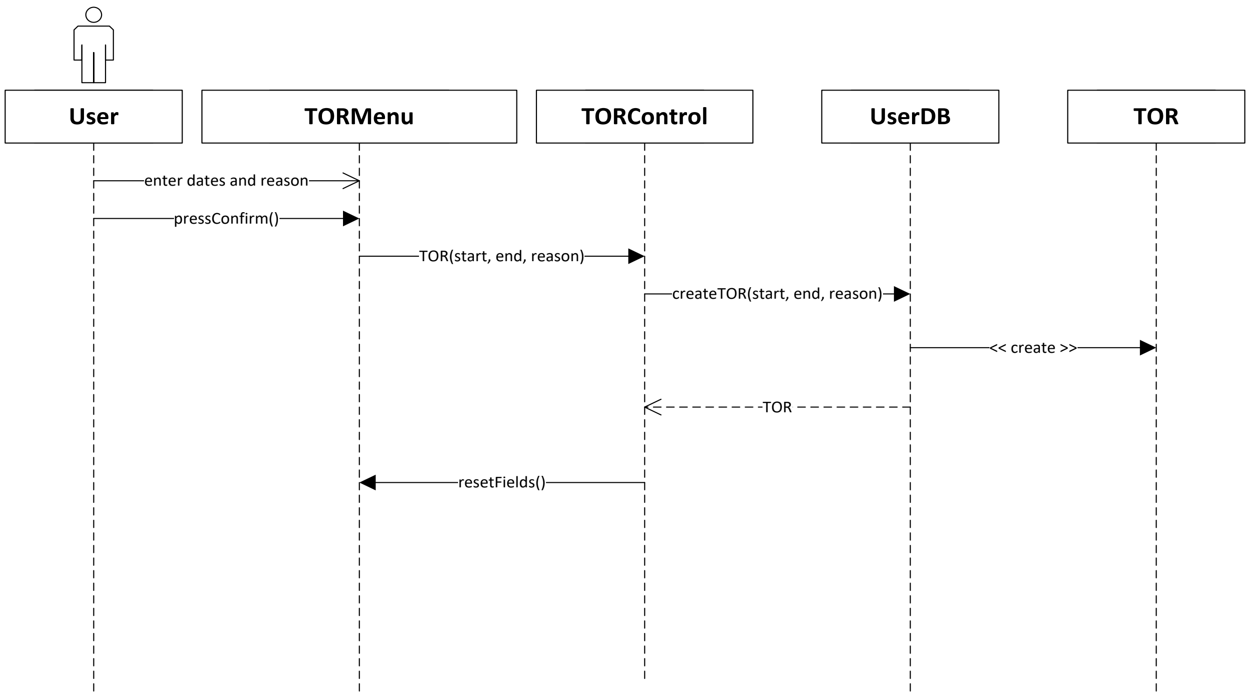


Figure 2.13: TOR sequence

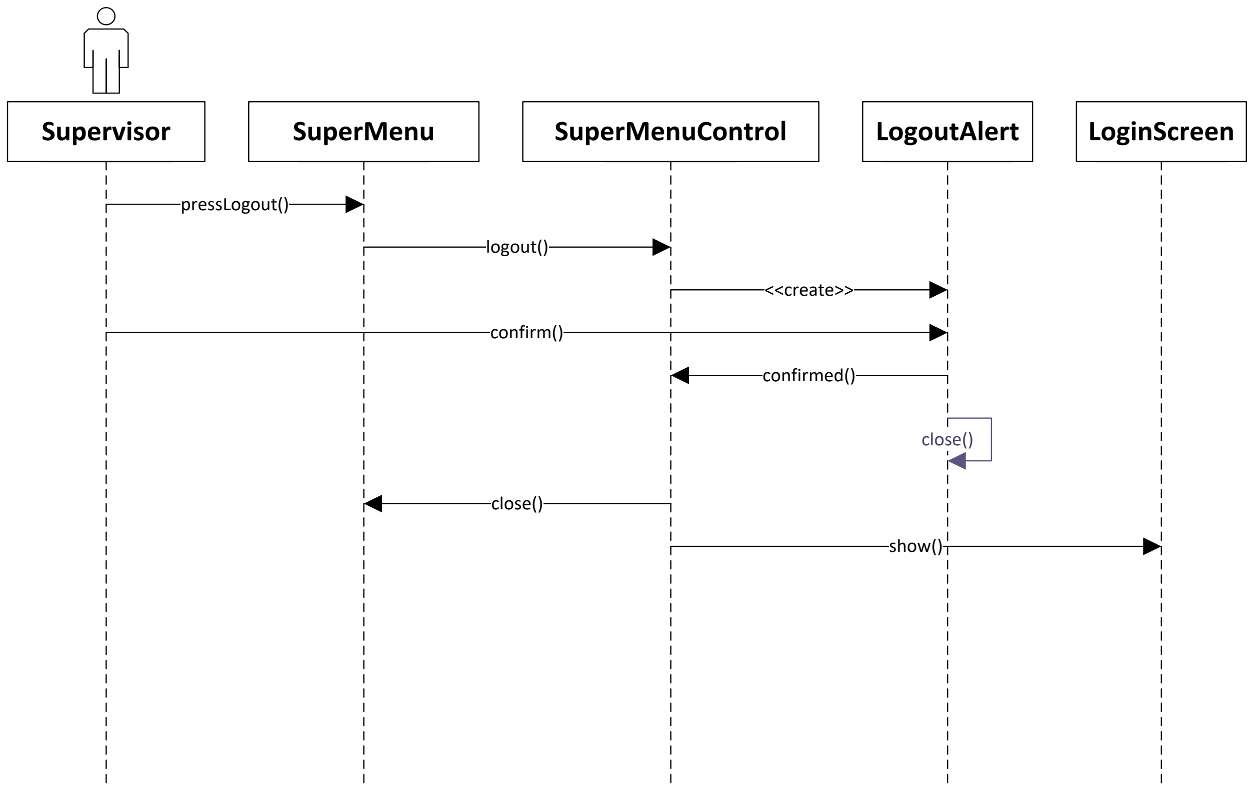


Figure 2.14: SuperMenuLogout sequence

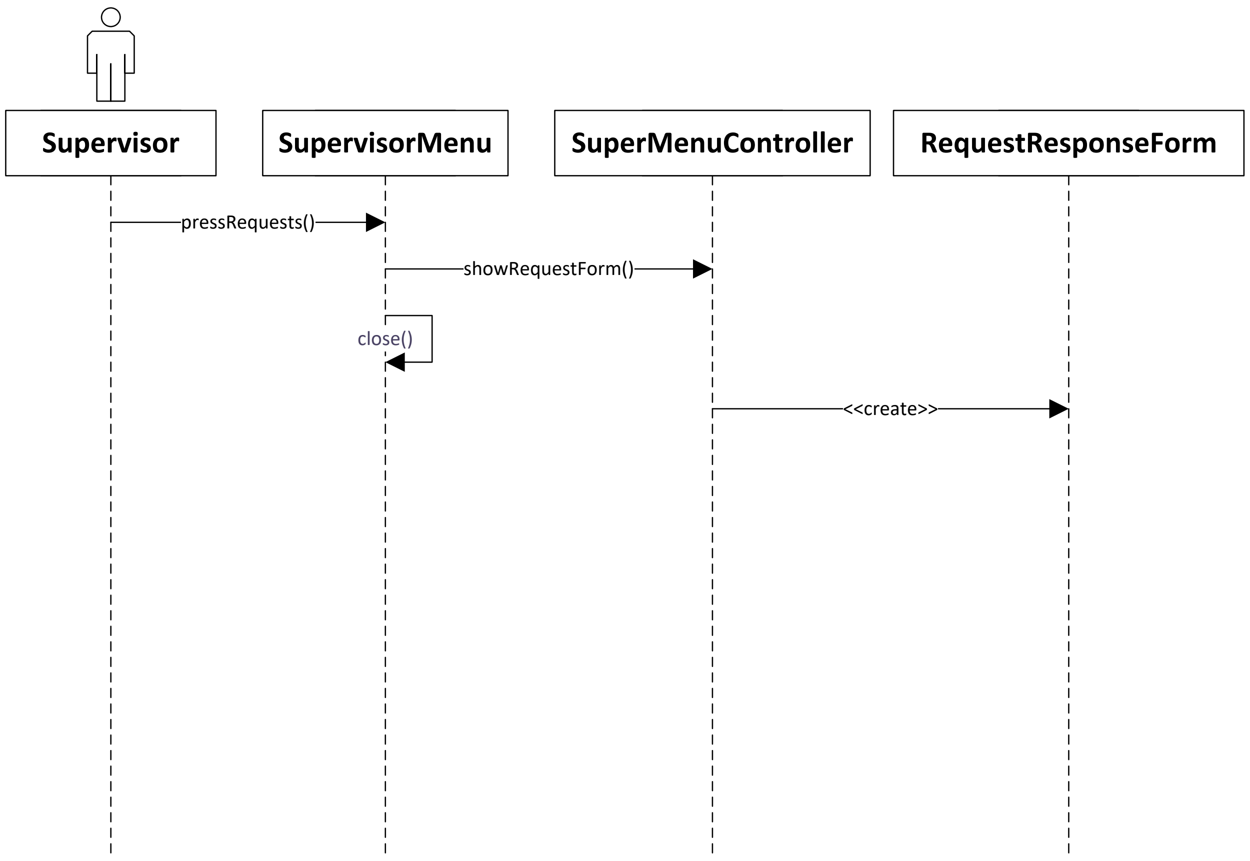


Figure 2.15: SupervisorMenu_RRForm sequence

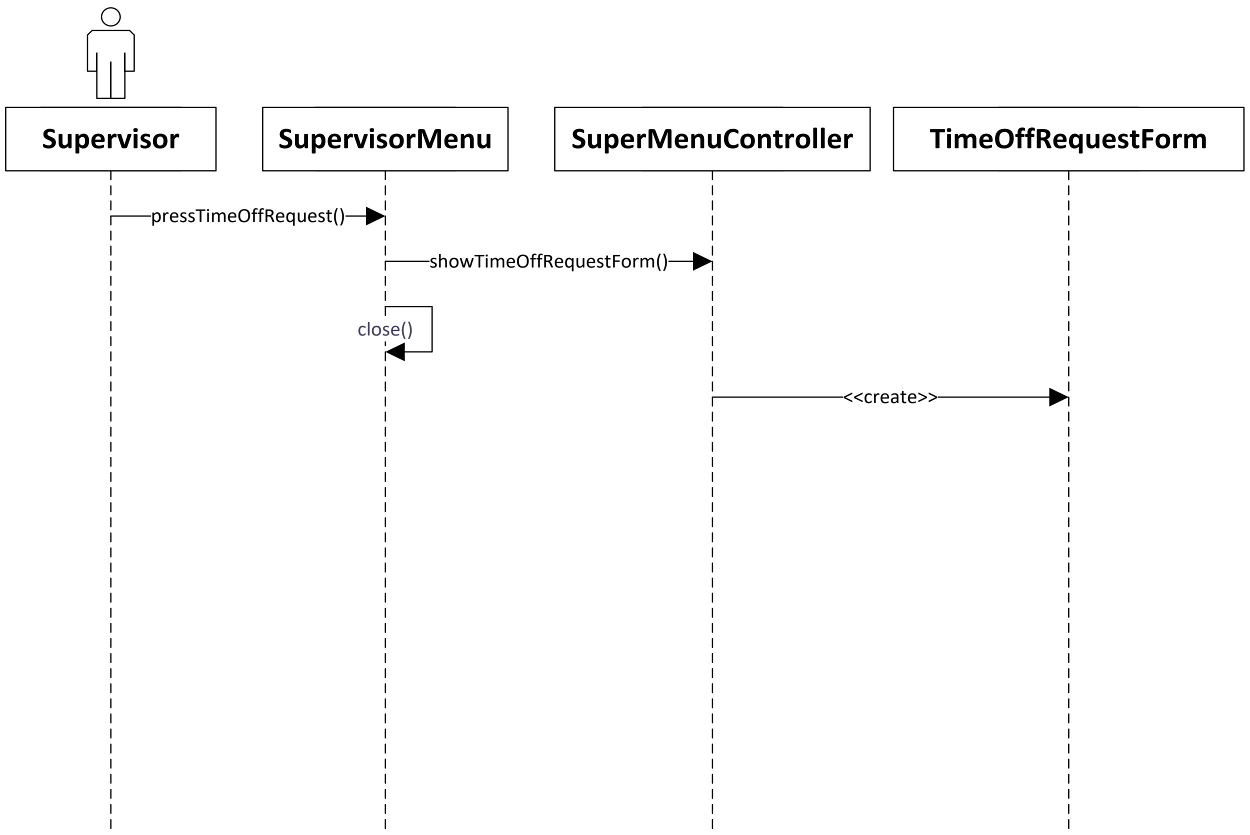


Figure 2.16: SupervisorMenu_TOR sequence

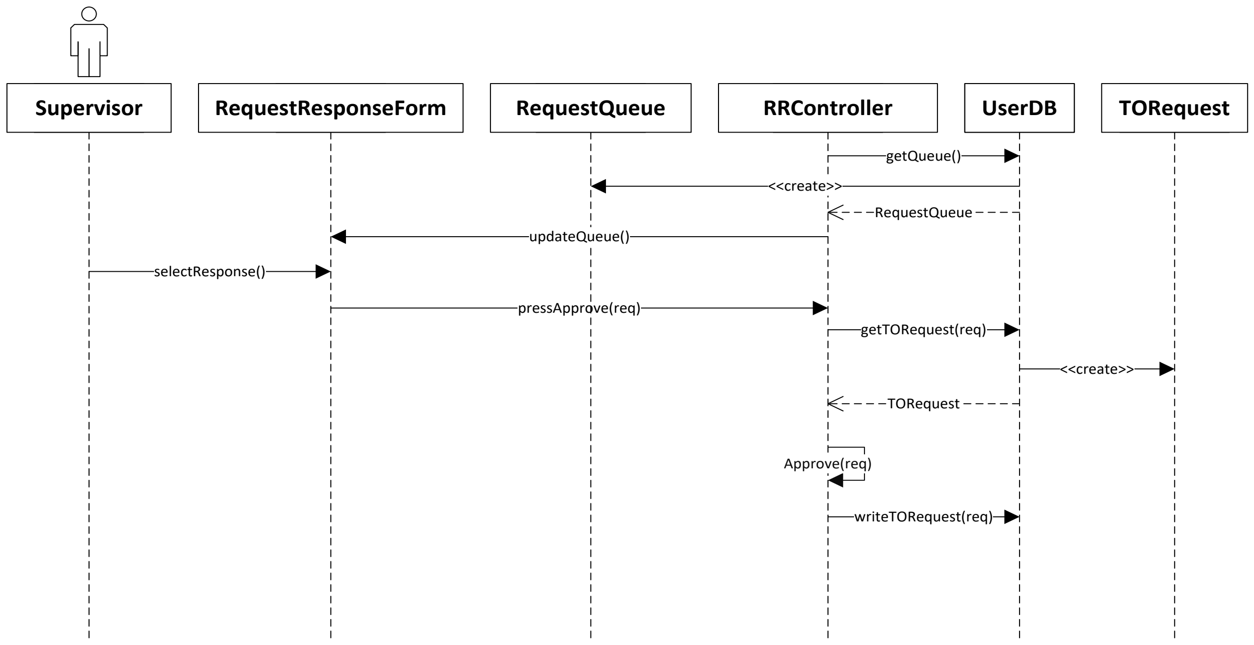


Figure 2.17: RRApprove sequence

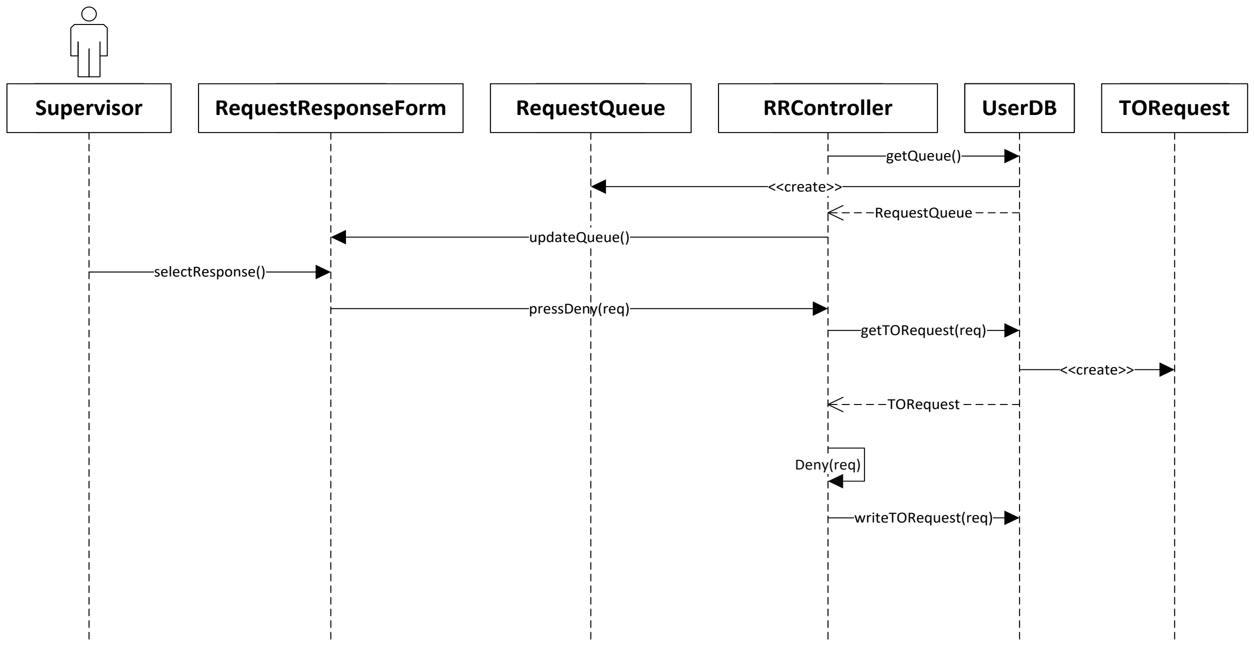


Figure 2.18: RRDeny sequence

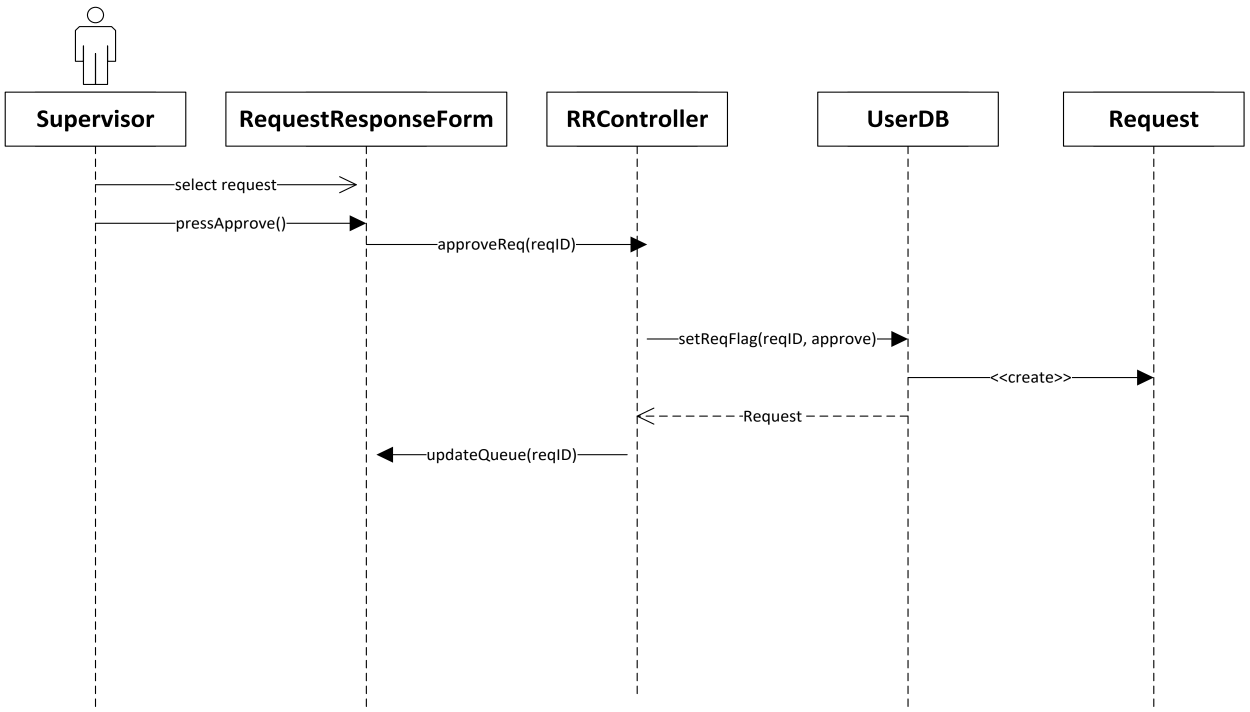
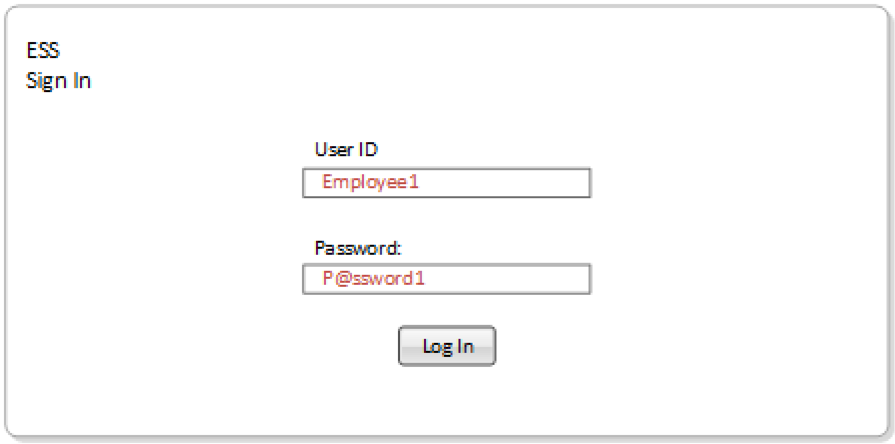


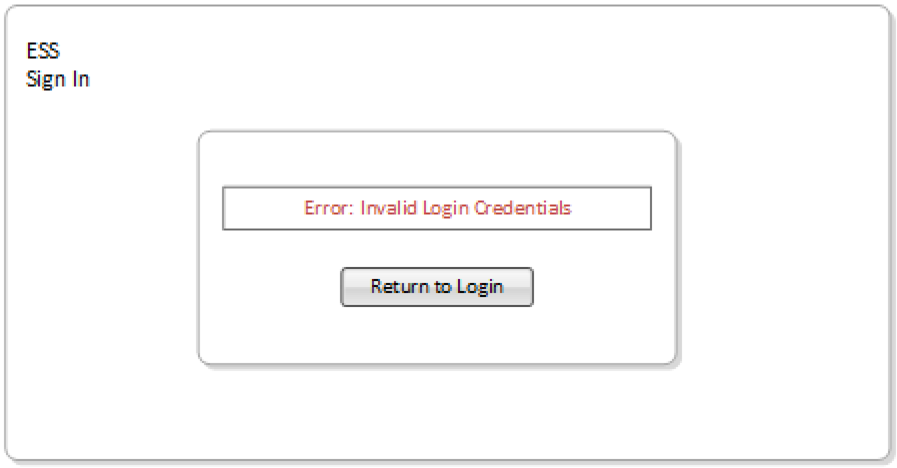
Figure 2.19: TimeOffResponseApprove sequence

3 USER INTERFACE MOCKUPS

3.1 Login

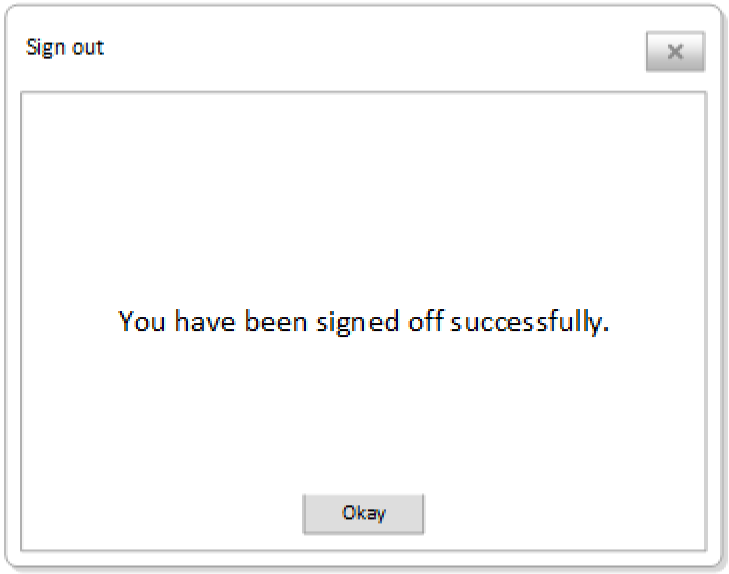


3.2 InvalidLogin



Error: Invalid Login Credentials

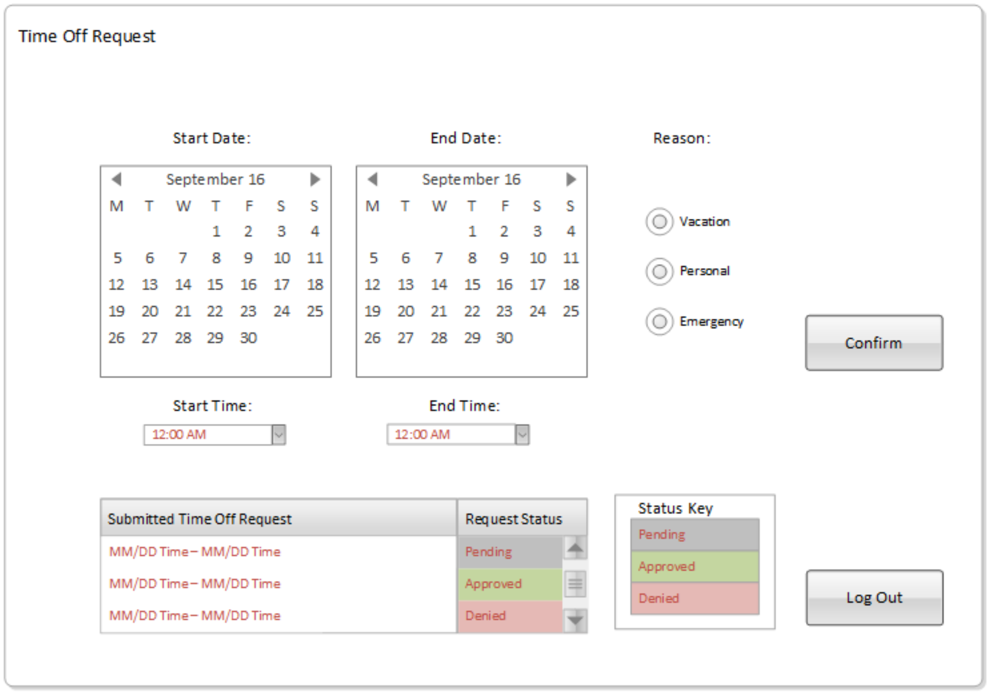
3.3 Logout



Sign out

You have been signed off successfully.

3.4 TimeOffRequest



Time Off Request

Start Date:

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

September 16

End Date:

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

September 16

Reason:

- Vacation
- Personal
- Emergency

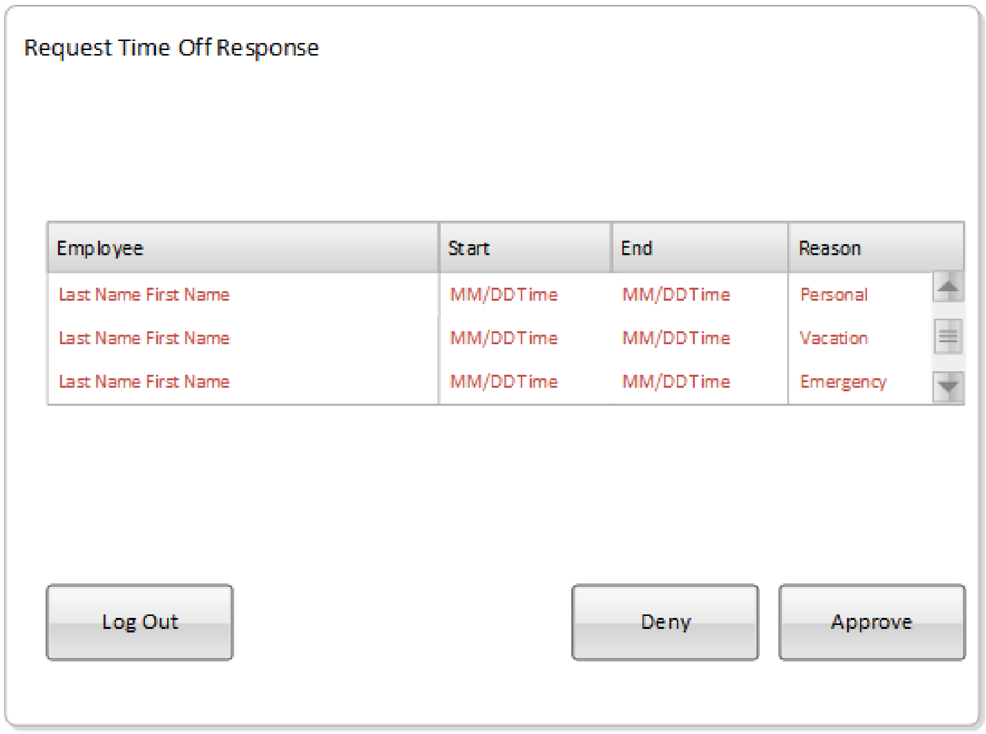
Start Time: 12:00 AM

End Time: 12:00 AM

Submitted Time Off Request	Request Status
MM/DD Time – MM/DD Time	Pending
MM/DD Time – MM/DD Time	Approved
MM/DD Time – MM/DD Time	Denied

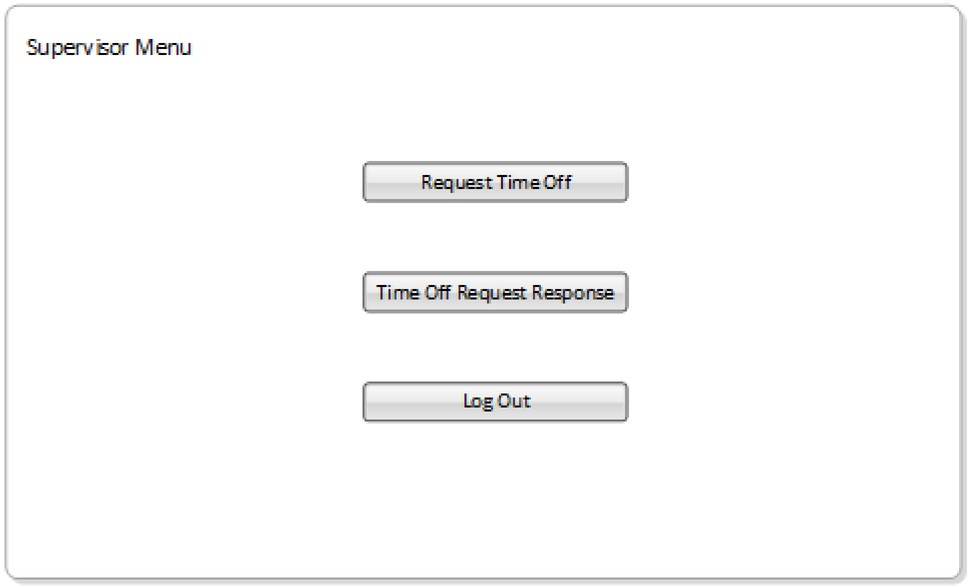
Status Key
- Pending
- Approved
- Denied

3.5 RequestResponse



Employee	Start	End	Reason
Last Name First Name	MM/DDTime	MM/DDTime	Personal
Last Name First Name	MM/DDTime	MM/DDTime	Vacation
Last Name First Name	MM/DDTime	MM/DDTime	Emergency

3.6 SupervisorMenu



4 OBJECT DESIGN

4.1 OBJECT INTERACTION

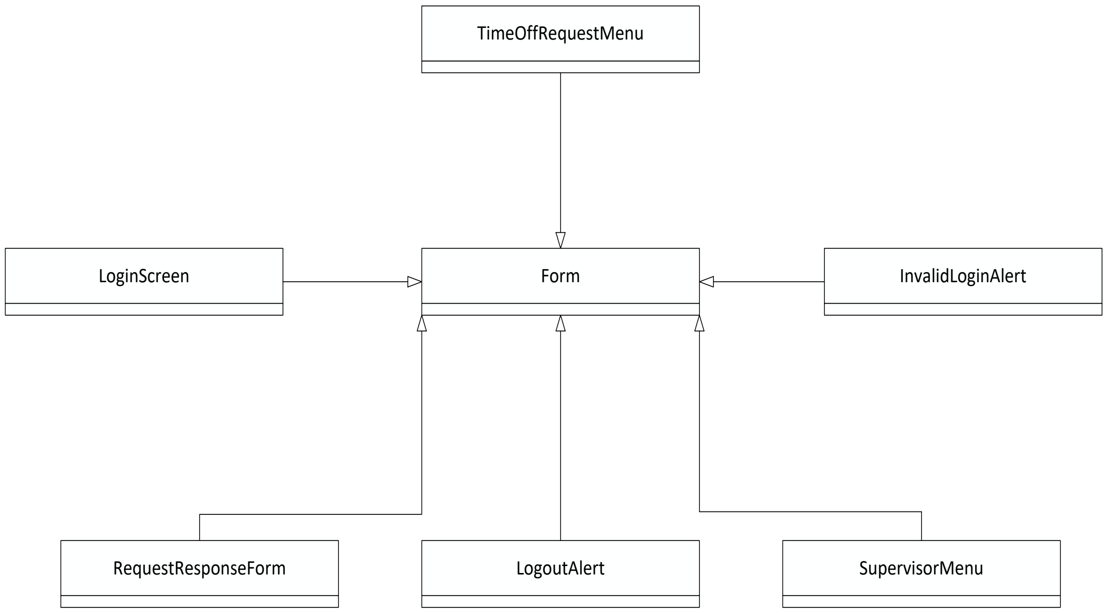


Figure 4.1: Class Diagram: Boundary

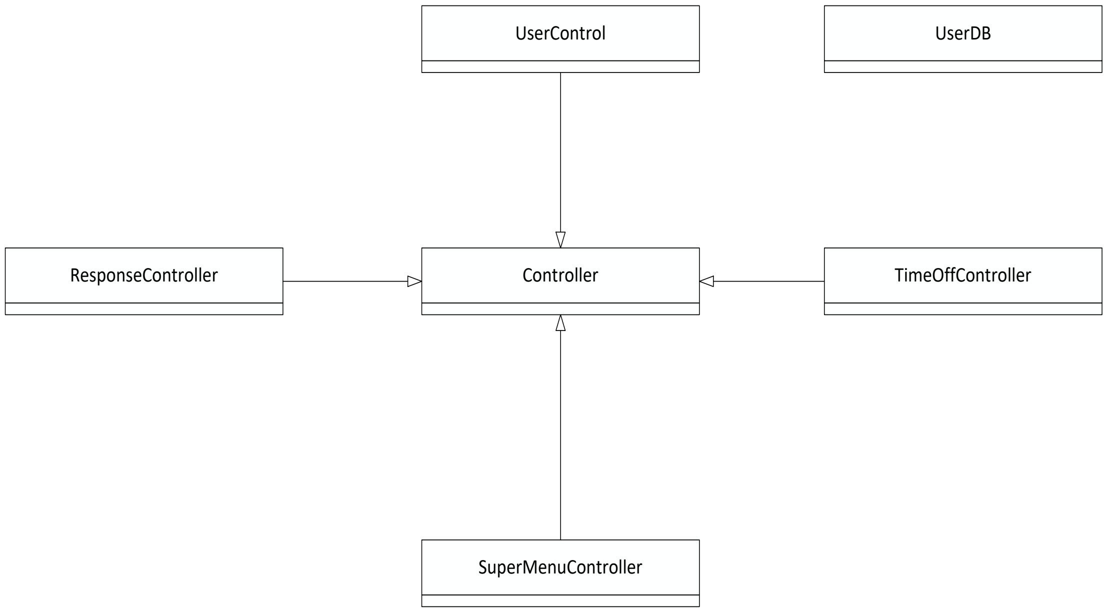


Figure 4.2: Class Diagram: Control

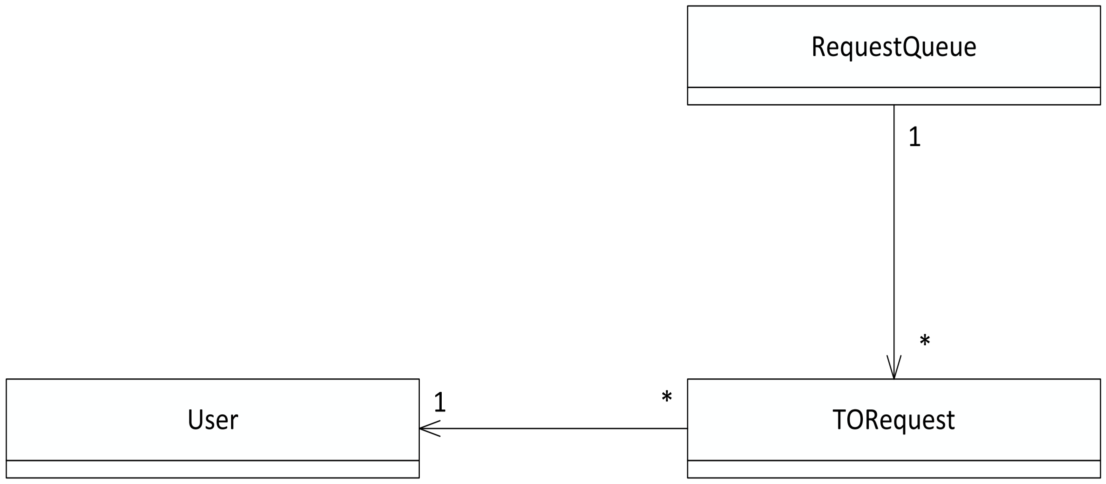


Figure 4.3: Class Diagram: Entity

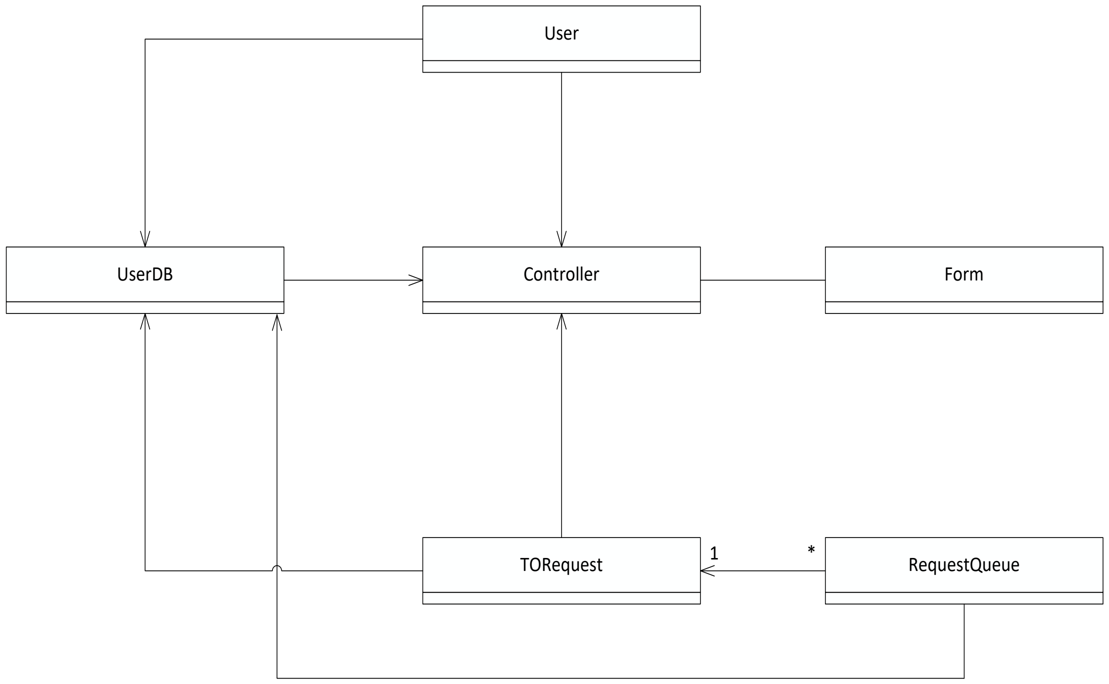


Figure 4.4: Class Diagram

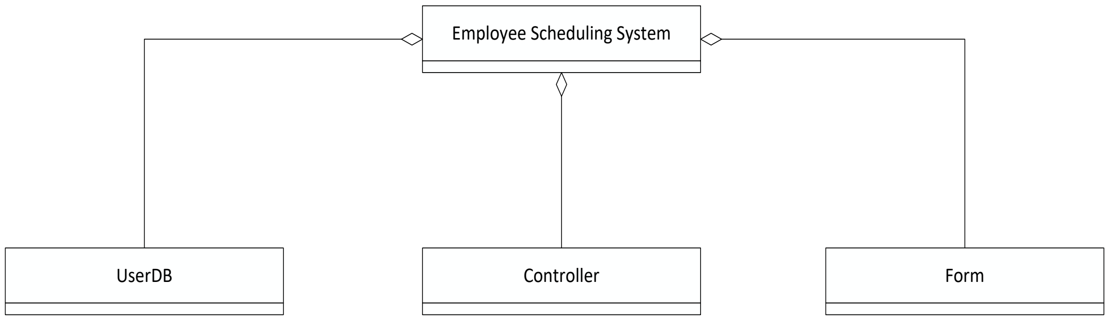


Figure 4.5: Class Diagram

4.2 Detailed Class Design

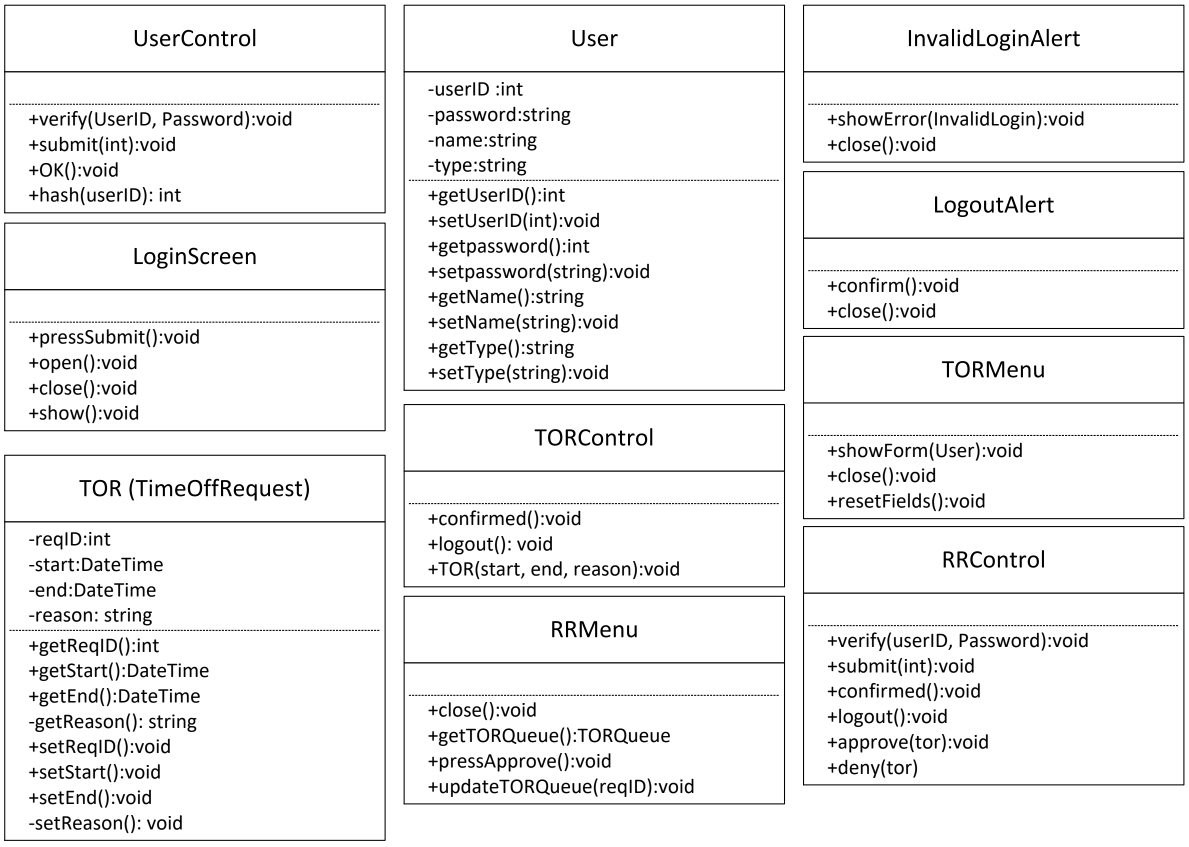


Figure 4.6: Class Diagram

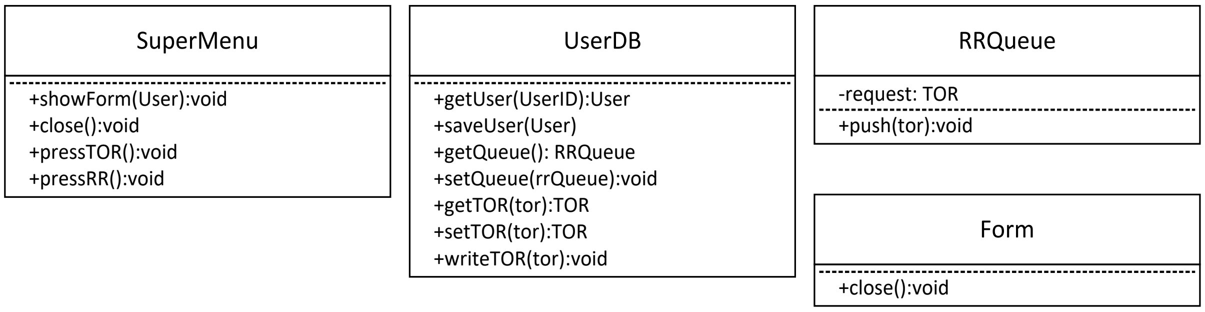


Figure 4.7: Class Diagram

4.3 STATECHART DIAGRAMS

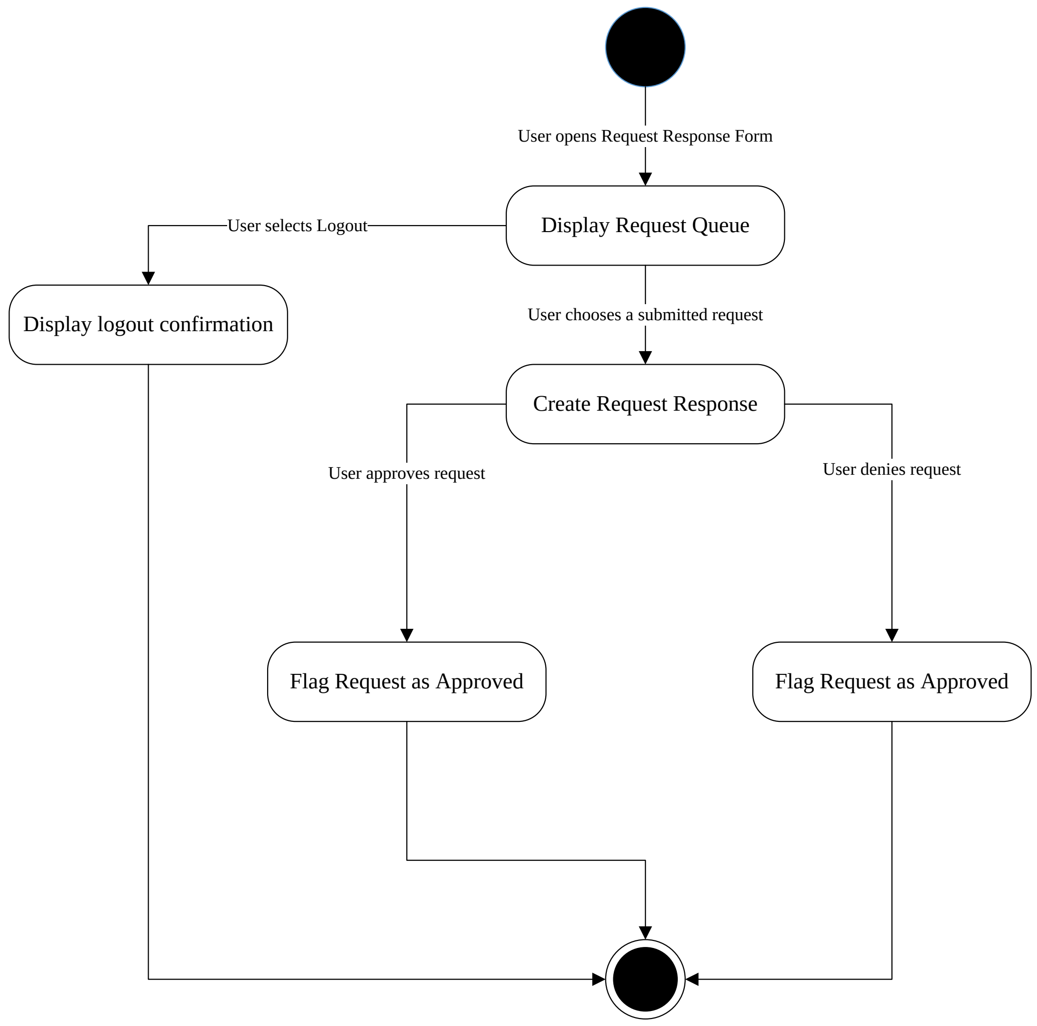


Figure 4.8: Request Response Controller statechart

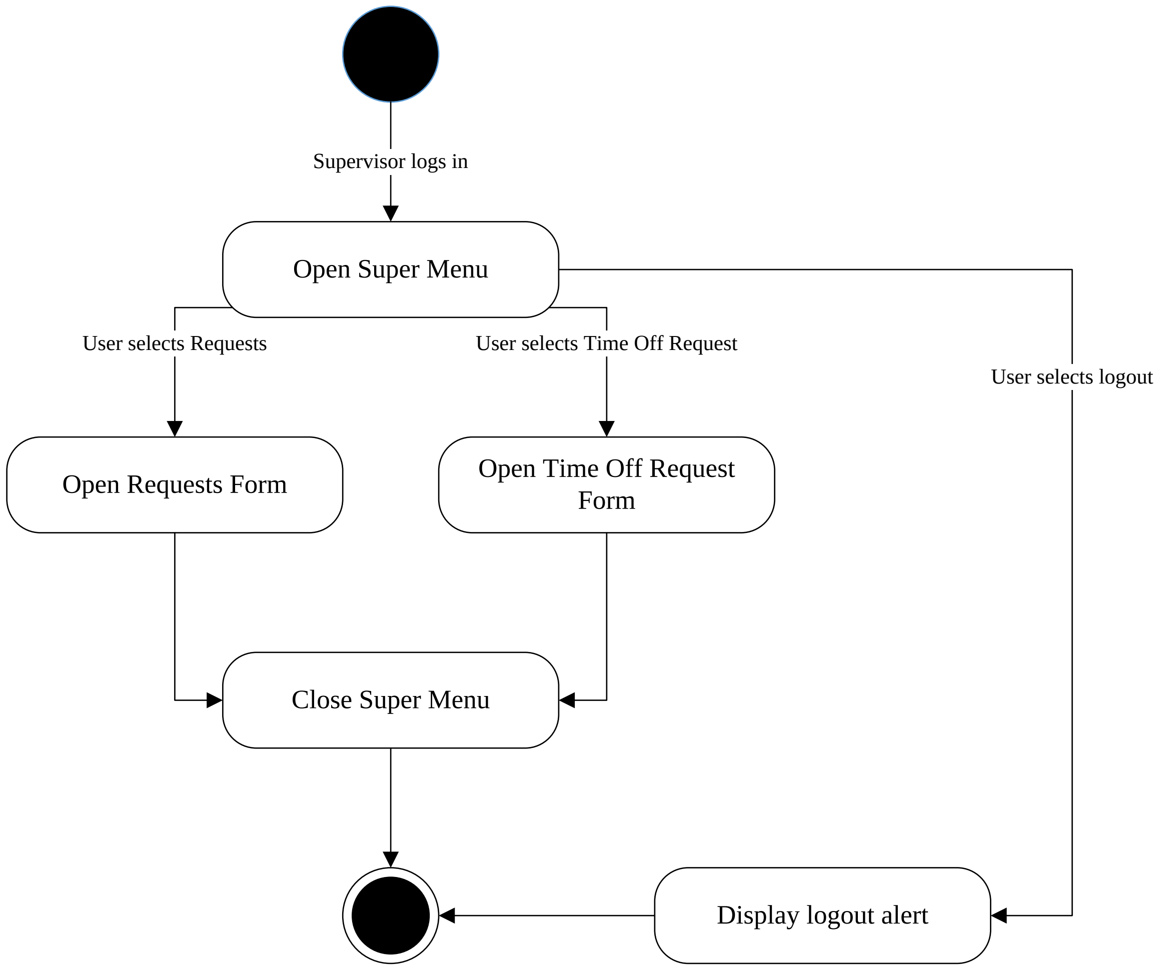


Figure 4.9: Supervisor Menu Controller statechart

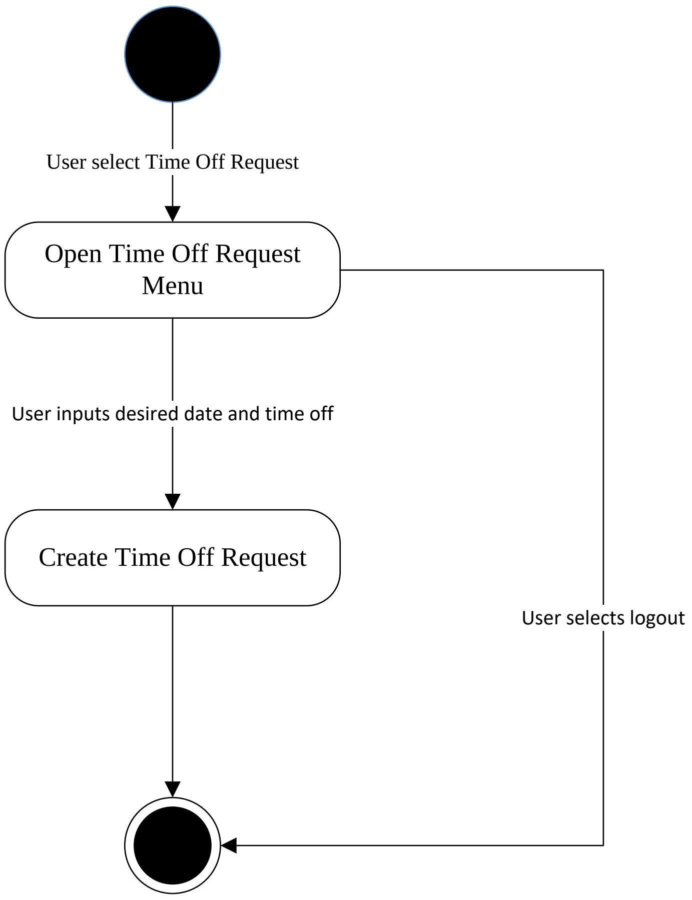

User select Time Off Request
Open Time Off Request Menu
User inputs desired date and time off
Create Time Off Request
User selects logout

Figure 4.10: Time Off Request Controller statechart

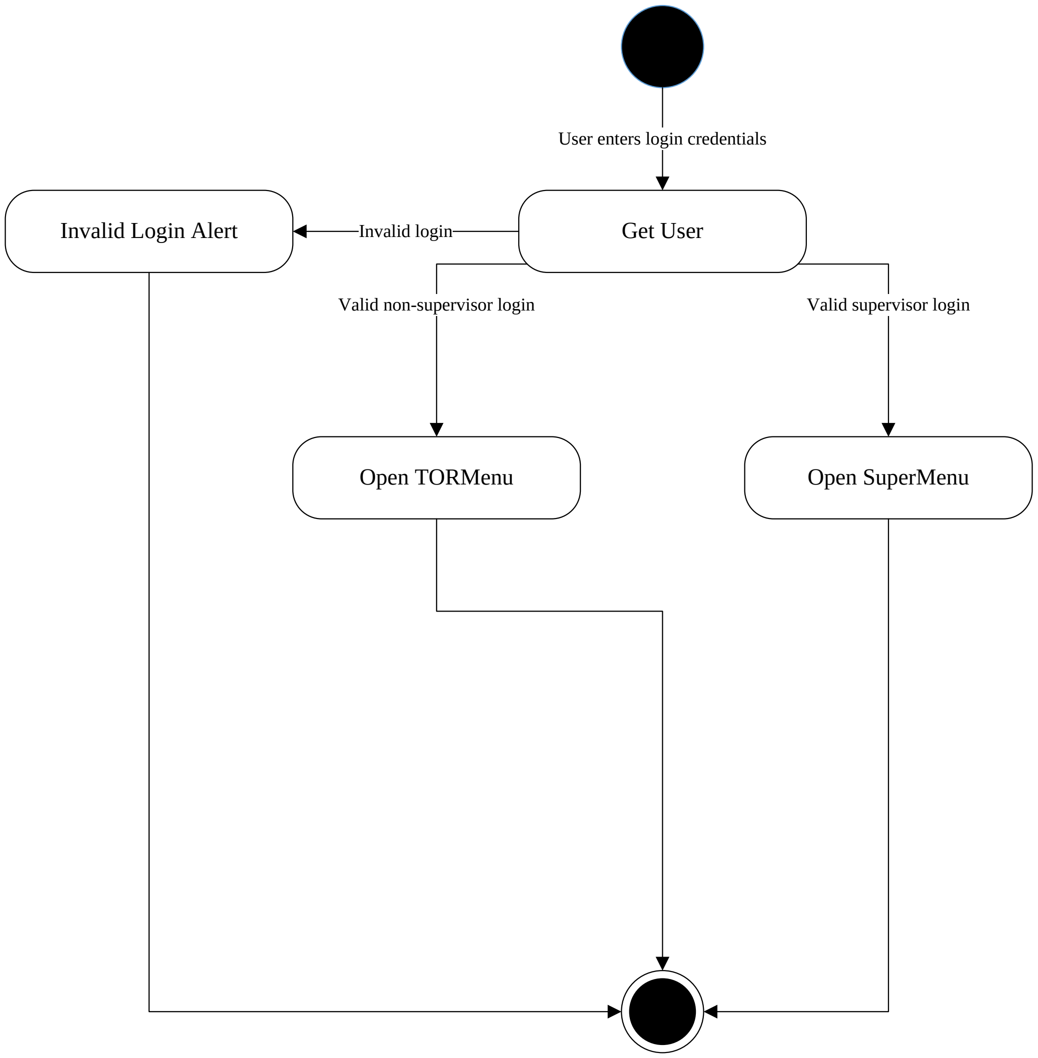


Figure 4.11: User Controller statechart

5 SYSTEM DESIGN

5.1 Subsystem Decomposition

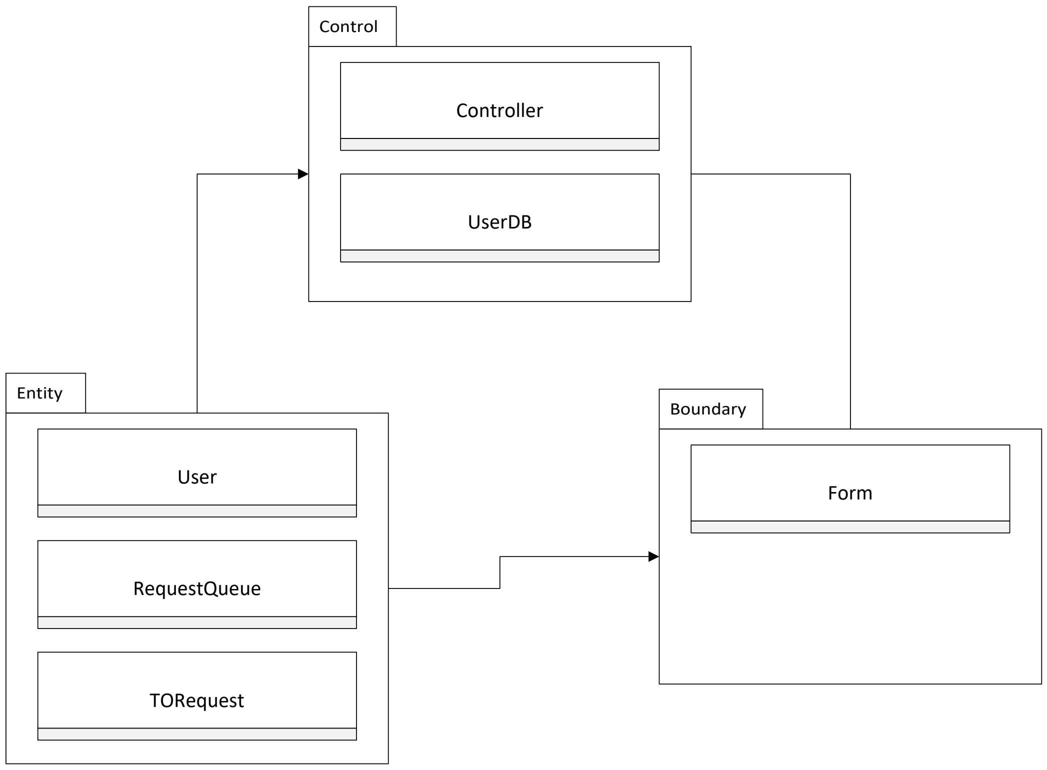


Figure 5.1: Subsystem

6 APPENDIX

6.1 Appendix A – Source Code

```
public class TORequest
{
        private int reqID;
        private DateTime start;
        private DateTime end;
        private string reason;

        public int getReqID()
        {
                return this.reqID;
        }

        public void setReqID(int value)
        {
                this.reqID=value;
        }

        public DateTime getStart()
        {
                return this.start;
        }

        public void setStart(DateTime value)
        {
                this.start=value;
        }

        public DateTime getEnd()
        {
                return this.end;
        }

        public void setEnd(DateTime value)
        {
                this.end=value;
        }

        public string getReason()
        {
                return this.reason;
        }

        public void setReason(string value)
        {
                this.reason=value;
        }
}
```

Figure 6.1: TORequest source

```
public class User
{
    private int userID;
    private string password;
    private string name;
    private string type;

    public int getUserID()
    {
        return this.userID;
    }

    public void setUserID(int value)
    {
        this.userID = value;
    }

    public string getPassword()
    {
        return this.password;
    }

    public void setPassword(string value)
    {
        this.password = value;
    }

    public string getName()
    {
        return this.name;
    }

    public void setName(string value)
    {
        this.name = value;
    }

    public string getType()
    {
            return this.type;
    }

    public void setType(string value)
    {
            this.type = value;
    }
}
```

Figure 6.2: User source code

REVISION HISTORY:

Version #, Section #: Item Modified

Version 2, All Sections: Corrected formatting
Version 2, Section 1.1, Overview of System: Added system overview
Version 2, Section 2.1, Functional Requirement: Logoff
Version 2, Section 2.4, Analysis Requirements: Added sequence diagrams

Version 3, Section 2.2, Use Case Diagram: Updated use case diagram
Version 3, Section 2.3, Use Case Descriptions: Edited multiple use case descriptions
Version 3, Section 2.4, Analysis Requirements: Edited multiple sequence diagrams
Version 3, Section 4.1, Object Relationship: Added Object Interaction Diagrams
Version 3, Section 4.2, Detailed Class Design: Added Class Design Diagrams