PUSS214204

v. 0.1

TimeMate

Software Top Level Design Document

Group 2

Responsible: System Group

Authors: Developer Group

2021-02-11

Contents

1 Document History

Version	Date	Responsible	Description
0.1	2021-02-11	UG	Document created.

2 Introduction

This document describes the design of "TimeMate". The system is a further development of the system "BaseBlockSystem". "TimeMate" hosts various functions related to the creating of time reports, getting a view of reported time and more.

3 Reference Documents

1. Software Requirements Specification: TimeMate, v. 0.5, Doc. number: PUSS214201
2. Software Top Level Design Document: BaseBlockSystem, v 1.0, Doc. number: PUSS12004

4 Terminology

- **Java Server Page (JSP):** Server-side technology that enables the creation of dynamic views.
- **Servlets:** Java programs that run on the server side.
- **Java Beans:** A Java class that only contains set/get methods with private attributes.

5 Overview

This system is developed using Apache Tomcat, wherein the controllers (Java Servlets) manipulates and sends data to the views (JavaServer Pages).

5.1 Java Server Pages

Below are the pages used for "TimeMate" and what their functions are.

5.1.1 login.jsp

The view that is shown when a user that is not logged tries to access the system. This is the only page in the "TimeMate" system a user can get access to without having an account. The page consists of two fields, one for username and one for password, as well as an option to request a new password in case the user has forgotten their password. If the user chooses to reset their password, a pop up is shown where the user can enter their e-mail which the new password will be sent to.

5.1.2 index.jsp

This file contains the view that is shown when a user has logged in. From here, the user can access the menu which is dynamically updated depending on what group the user belong to. The index page contains overview information about TimeMate.

5.1.3 viewReport.jsp

This file contains the view that shows a collection of the users submitted reports. When being here, the user can select a submitted report to show more information about.

5.1.4 newReport.jsp

This file contains the view that represents the creation of a new time report.

5.1.5 editReport.jsp

This view is used by users that want to edit their previously reported time reports. When being here, the user can select a submitted report to edit.

5.1.6 summaryReport.jsp

This view represents a summary of a user's previously reported time reports.

5.1.7 signReport.jsp

This view is used by the project leaders for signing and unsigning time reports.

5.1.8 usermanagement.jsp

This file contains the view of the usermanagement page. The view is used by the administrator and the project leaders.

5.1.9 administration.jsp

This file contains the view of the administration page. Only the administrator has access to this view.

5.1.10 changepassword.jsp

This view is used by all users to change their password. The page contains three fields, one for the user's current password, and two for the new password (in order to confirm it).

5.2 Servlets

The files below are the controllers for the system.

5.2.1 LoginServlet

Communication link between the view login.jsp and the bean LoginBean.

5.2.2 PasswordChangerServlet

Servlet used when a user changes their password.

5.2.3 TimeReportServlet

Communication link between the views viewReport.jsp, newReport.jsp and summaryReport.jsp and the bean ReportBean.

5.2.4 TimeReportManagementServlet

Communication link between the views editReport.jsp and signReport.jsp and the bean ReportBean

5.2.5 UserManagementServlet

Communication link between the view usermanagement.jsp and the bean UserManagementBean.

5.3 Java Beans

The files below are the Java Beans that are used to send information from the server to the client view. When created, beans are stored in the current session and discarded when the session ends.

5.3.1 UserBean

Get/set class containing user information required to execute queries and updates related to the logged in user.

5.3.2 TimeReportBean

Get/set class containing a list of users and their time reports which is required to render the report.jsp view.

5.3.3 TimeReportManagementBean

Get/set class containing a list of signed time reports which is required to render signReport.jsp.

5.3.4 UserManagementBean

Get/set class containing a list of users(Excluding project leaders) and their roles which is required to render the usermanagement.jsp view.

5.4 Other classes

The classes below are never accessed by the user in any way, but are instead helper classes to the servlets.

5.4.1 MailHandler

Class used to send e-mails to users upon account creation and password changes.

5.4.2 PasswordHandler

Class used to encrypt passwords when needed.

5.4.3 Database

Class to establish connection between Tomcat and the database.

6 Database

In this project, there is one database that consists of the nine relations shown in Figure 1. Every relation has one primary key each, which are the following: *userID*, *projectID* and *reportID*.

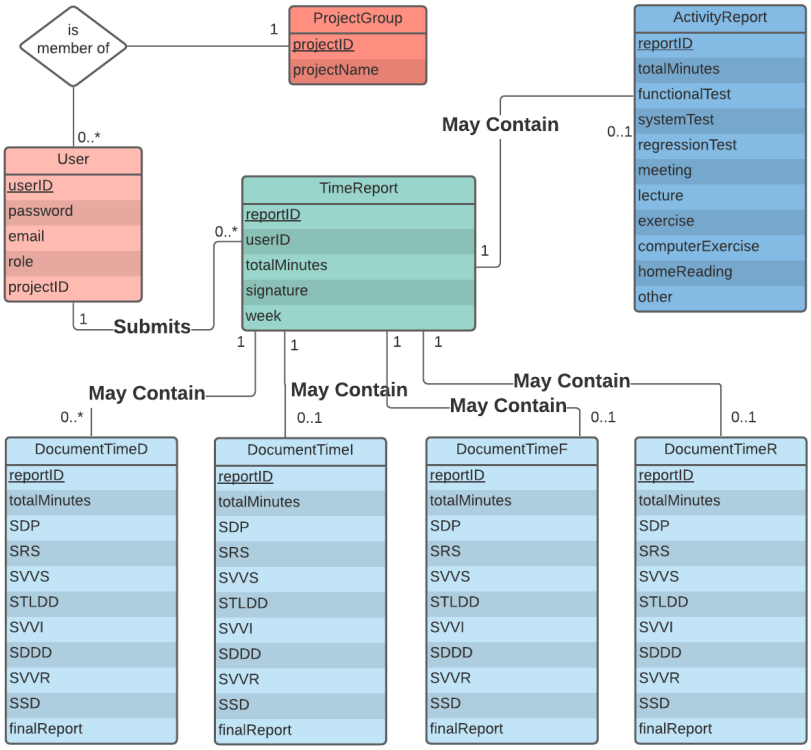

Figure 1: The database consists of the following relations: *User*, *ProjectGroup*, *TimeReport*, *ActivityReport* and *DocumentTimeD/I/F/R*. *Is member of* represents the relationship between User and ProjectGroup.

6.1 User Relation

The User relation contains all registered users together with user specific information such as password, email and information about whether a user belongs to a project group or not. A user belonging to a group may be assigned a role under the role attribute. The username attribute

is used as the primary key, making every username unique. The appearance of the table can be viewed in Figure 2.

```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| userID    | int         | NO   | PRI | NULL    |       |
| password  | varchar(50) | NO   |     | NULL    |       |
| email     | varchar(100)| YES  |     | NULL    |       |
| role      | varchar(30) | YES  |     | NULL    |       |
| projectID | int         | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Figure 2: A view of the User relation and its attributes.

A User table can be created using the following MySQL command:

```
mysql> CREATE TABLE User (
    -> userID Integer NOT NULL,
    -> password varChar(50) NOT NULL,
    -> email varChar(100),
    -> role varChar(30),
    -> projectID Integer,
    -> PRIMARY KEY(userID),
    -> FOREIGN KEY(projectID) REFERENCES ProjectGroup(projectID)
    -> ON UPDATE CASCADE ON DELETE SET NULL
    -> );
```

6.2 ProjectGroup Relation

The ProjectGroup relation contains all project group IDs and connects an ID to a project name. The projectID attribute is used as the primary key, enabling two or more project groups to be registered under the same name but still be distinguished from one another by their IDs. The appearance of the table can be viewed in Figure 3.

```
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| projectID   | int         | NO   | PRI | NULL    |       |
| projectName | varchar(30) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

Figure 3: A view of the ProjectGroup relation and its attributes.

A ProjectGroup table can be created using the following MySQL command:

```
mysql> CREATE TABLE ProjectGroup (
    -> projectID Integer NOT NULL,
```

```
-> projectName varChar(30) NOT NULL,
-> PRIMARY KEY(projectID)
-> );
```

6.3 TimeReport Relation

The time report to be filled out by each individual user on a weekly basis contains 55 columns, aiming to help the user specify the time spent on various activities. Four sections - *D (Development)*, *I (Informal review)*, *F (Formal review)*, *R (Rework, improvement or correction)*. However, there may be weeks where no time is spent on, e.g., formal reviews. In that case, a large number of columns would be wasted, as there will be no meaningful data added to the columns related to that section of the time report. The appearance of the table can be viewed in Figure 4.

```
+--------------+------+------+-----+---------+-------+
| Field        | Type | Null | Key | Default | Extra |
+--------------+------+------+-----+---------+-------+
| reportID     | int  | NO   | PRI | NULL    |       |
| userID       | int  | NO   | MUL | NULL    |       |
| totalMinutes | int  | YES  |     | NULL    |       |
| signature    | int  | YES  | MUL | NULL    |       |
| week         | int  | NO   |     | NULL    |       |
+--------------+------+------+-----+---------+-------+
```

Figure 4: A view of the TimeReport relation and its attributes.

A TimeReport table can be created using the following MySQL command:

```
mysql> CREATE TABLE TimeReport (
    -> reportID Integer NOT NULL,
    -> userID Integer NOT NULL,
    -> totalMinutes Integer,
    -> signature Integer,
    -> week Integer NOT NULL,
    -> PRIMARY KEY(reportID),
    -> FOREIGN KEY(userID) REFERENCES User(userID)
    -> ON UPDATE CASCADE ON DELETE CASCADE,
    -> FOREIGN KEY(signature) REFERENCES User(userID)
    -> ON UPDATE CASCADE ON DELETE SET NULL
    -> );
```

To avoid data redundancy, the time reports are being separated into several different relations. The reason for this is the fact that every single submitted time report is going to be represented in TimeReports:

6.4 ActivityReport Relation

ActivityReport is the relation concerning the general activity types. As can be seen in Figure 1, these include the unique primary key: reportID and other attributes such as homeReading, meeting and various tests. The appearance of the table can be viewed in Figure 5.

```
+------------------+------+------+-----+---------+-------+
| Field            | Type | Null | Key | Default | Extra |
+------------------+------+------+-----+---------+-------+
| reportID         | int  | NO   | PRI | NULL    |       |
| totalMinutes     | int  | YES  |     | NULL    |       |
| functionalTest   | int  | YES  |     | NULL    |       |
| systemTest       | int  | YES  |     | NULL    |       |
| regressionTest   | int  | YES  |     | NULL    |       |
| meeting          | int  | YES  |     | NULL    |       |
| lecture          | int  | YES  |     | NULL    |       |
| exercise         | int  | YES  |     | NULL    |       |
| computerExercise | int  | YES  |     | NULL    |       |
| homeReading      | int  | YES  |     | NULL    |       |
| other            | int  | YES  |     | NULL    |       |
+------------------+------+------+-----+---------+-------+
```

Figure 5: A view of the ActivityReport relation and its attributes.

An ActivityReport table can be created using the following MySQL command:

```
mysql> CREATE TABLE ActivityReport (
    -> reportID Integer NOT NULL,
    -> totalMinutes Integer,
    -> functionalTest Integer,
    -> systemTest Integer,
    -> regressionTest Integer,
    -> meeting Integer,
    -> lecture Integer,
    -> exercise Integer,
    -> computerExercise Integer,
    -> homeReading Integer,
    -> other Integer,
    -> PRIMARY KEY(reportID),
    -> FOREIGN KEY(reportID) REFERENCES TimeReport(reportID)
    -> ON UPDATE CASCADE ON DELETE CASCADE
    -> );
```

6.5 DocumentTimeD/I/F/R Relation

As mentioned earlier, there are four different activity types in the time documentation: *Development and documentation (D)*, *Informal review (I)*, *Formal review (F)* and *Rework, improvement or correction (R)*. All of these relations contain the same attributes mainly concerning the various documents. Besides these attributes, there is one attribute concerning the totalMinutes. The purpose of this attribute is to oversee the amount of minutes spent on each activity type. Since each document continuously is in a different stage, it was concluded to include different phases as activity types to ensure that the time reports are comprehensible. All of these tables can be created by using the same SQL statement. The appearance of each of these tables can be viewed in Figure 6.

Field	Type	Null	Key	Default	Extra
reportID	int	NO	PRI	NULL	
totalMinutes	int	YES		NULL	
SDP	int	YES		NULL	
SRS	int	YES		NULL	
SVVS	int	YES		NULL	
STLDD	int	YES		NULL	
SVVI	int	YES		NULL	
SDDD	int	YES		NULL	
SVVR	int	YES		NULL	
SSD	int	YES		NULL	
finalReport	int	YES		NULL	

Figure 6: A view of a DocumentTime relation and its attributes.

A DocumentTime table can be created using the following MySQL command:

```
mysql> CREATE TABLE TimeDevelopments (
    -> reportID Integer NOT NULL,
    -> totalMinutes Integer,
    -> SDP Integer,
    -> SRS Integer,
    -> SVVS Integer,
    -> STLDD Integer,
    -> SVVI Integer,
    -> SDDD Integer,
    -> SVVR Integer,
    -> SSD Integer,
    -> finalReport Integer,
    -> PRIMARY KEY(reportID),
    -> FOREIGN KEY(reportID) REFERENCES TimeReport(reportID)
    -> ON UPDATE CASCADE ON DELETE CASCADE );
```

7 Description of Classes

This section describes all classes and their public methods.

7.1 Database.java

7.1.1 addUser();

Adds a user to the database.

7.1.2 removeUser()

Removes a user from the database.

7.1.3 retrieveTimereports()

Retrieves a list of all time reports from the database.

7.1.4 viewTimereport()

Retrieves a specific time report.

7.1.5 signTimereports()

Changes all timereports according to user input.

7.1.6 viewUsers()

Retrieves a list of all users.

7.1.7 setProjectLeader()

Changes a users role to PG.

7.1.8 newTimereport()

Adds a new time report to the database..

7.1.9 updateTimereport()

Changes an existing unsigned time report according to user input.

7.1.10 deleteTimereport()

Removes an unsigned time report from the database.

7.1.11 setPassword()

Checks if the new password is valid. If it's valid, it replaces the old password. If not, the password remains unchanged.

7.1.12 getSummary()

Retrieves a summary of all reported time.

7.1.13 updateRole()

Changes a users role according to user input. Cannot change a users role to PG.

7.1.14 checkLogin()

Checks if the login credentials are correct.

7.1.15 getPassword()

Retrieves the users current password.

7.1.16 getEmail()

Retrieves the users email.

7.2 UserBean

7.2.1 populateBean()

Sets all attributes in the bean according to user input.

7.2.2 getUserName()

Gets the username contained in the bean.

7.2.3 getEmail()

Gets the email contained in the bean.

7.2.4 getRole()

Gets the role contained in the bean.

7.3 UserManagementBean

7.3.1 populateBean()

Sets all attributes in the bean according to the database.

7.3.2 getUserName()

Gets the username contained in the bean.

7.3.3 getUserList()

Gets the userlist contained in the bean.

7.3.4 getRole()

Gets the role contained in the bean.

7.4 timeReportBean

7.4.1 populateBean()

Sets all attributes in the bean according to user input.

7.4.2 getUsername()

Gets the username contained in the bean.

7.4.3 getWeek()

Gets the week of the time report contained in the bean.

7.4.4 getReportValues()

Gets the values of the time report contained in the bean.

7.5 timeReportManagementBean

7.5.1 populateBean()

Sets all attributes in the bean according to user input.

7.5.2 timeReportList

Gets a list of all time reports contained in the bean.

7.5.3 getUserName

Gets the username contained in the bean.

7.6 MailHandler

7.6.1 sendPasswordMail()

Sends an email containing a randomly generated password to a newly created user.

7.7 PasswordHandler

7.7.1 hashPassword()

Returns the hash of a given password.

7.7.2 generatePassword

Randomly generates a new password for a user.

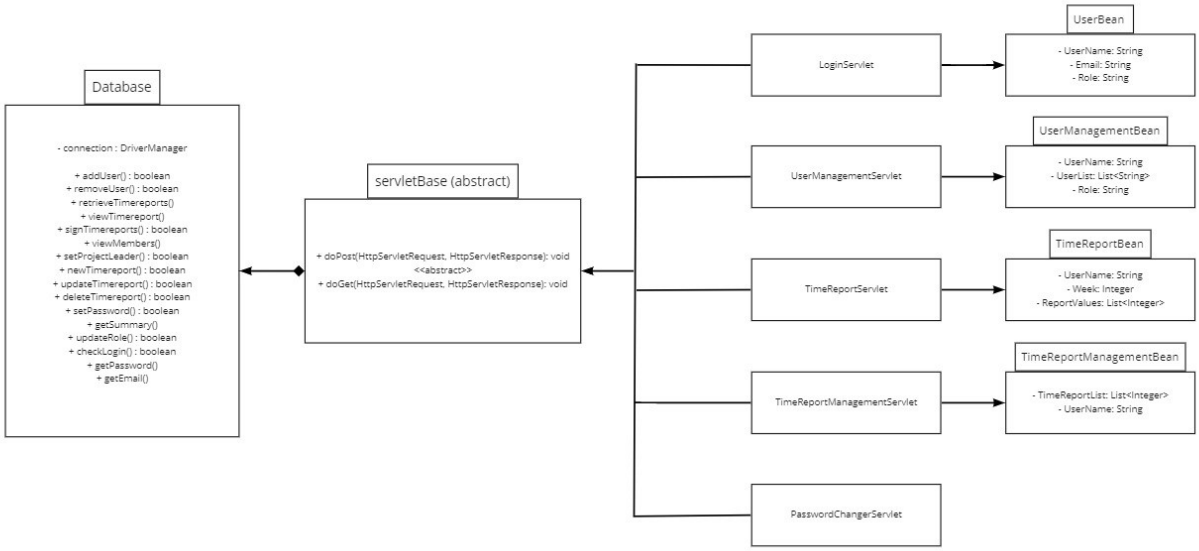

Figure 7: Diagram containing an overview of all classes

8 Sequence Diagrams

8.1 LoginServlet

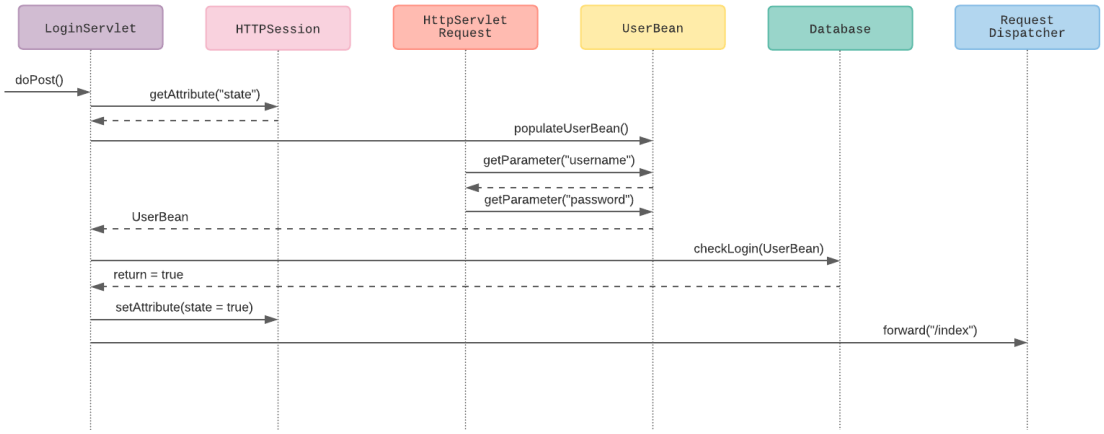

Figure 8: Sequence diagram for a successful login.

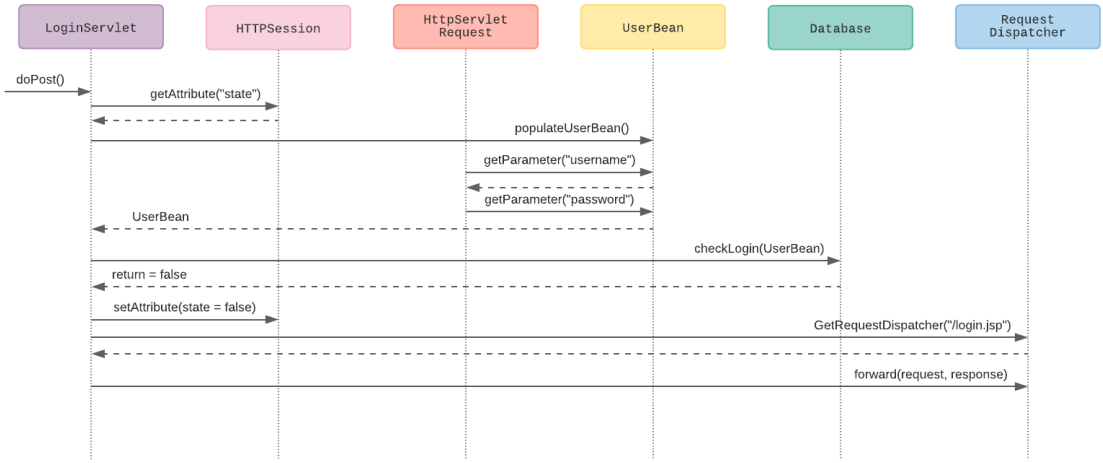

Figure 9: Sequence diagram for a failed attempt to login.

8.2 UserManagementServlet

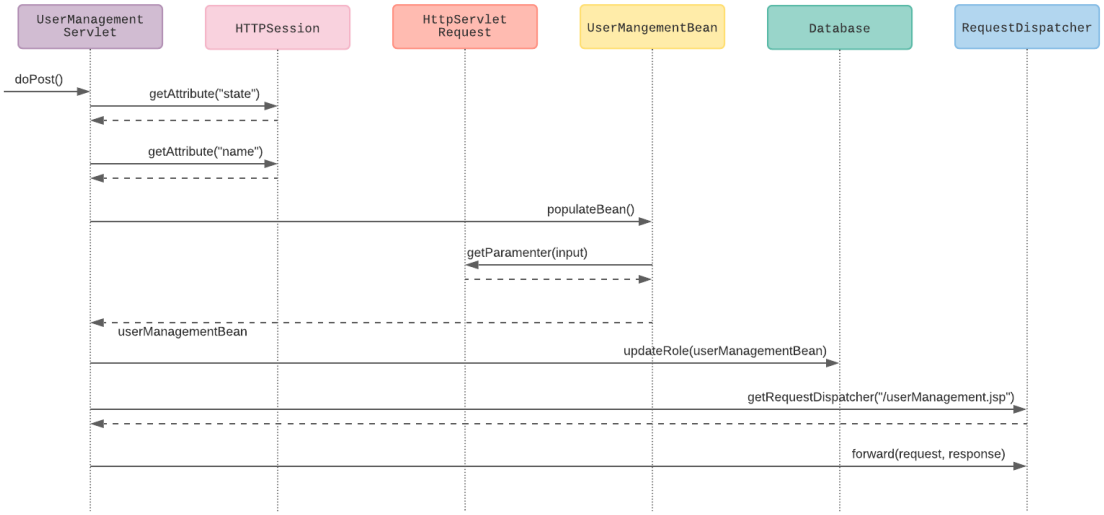

Figure 10: Sequence diagram for the process of updating the role of a project member.

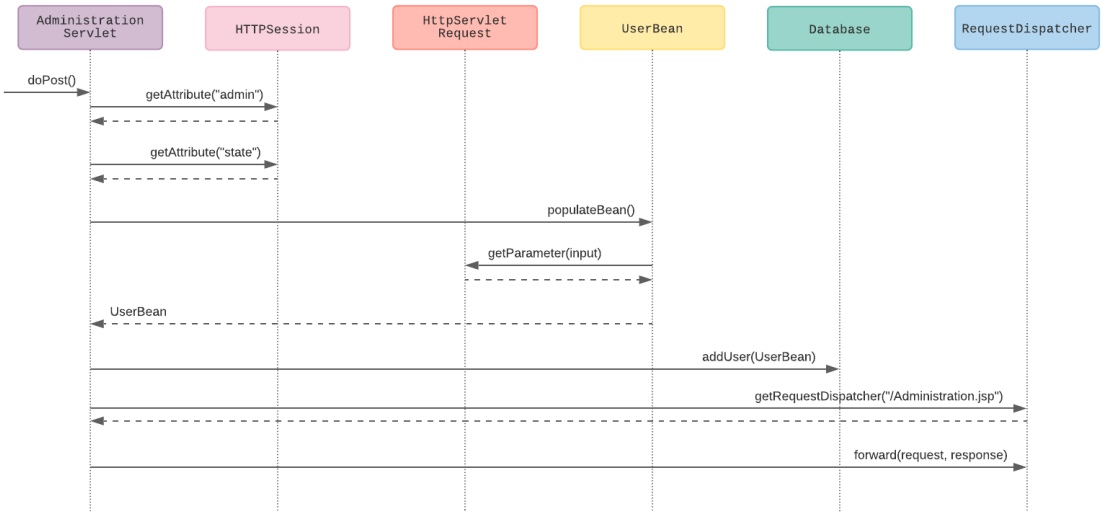

Figure 11: Sequence diagram for adding a user by administrator.

8.3 PasswordChangerServlet

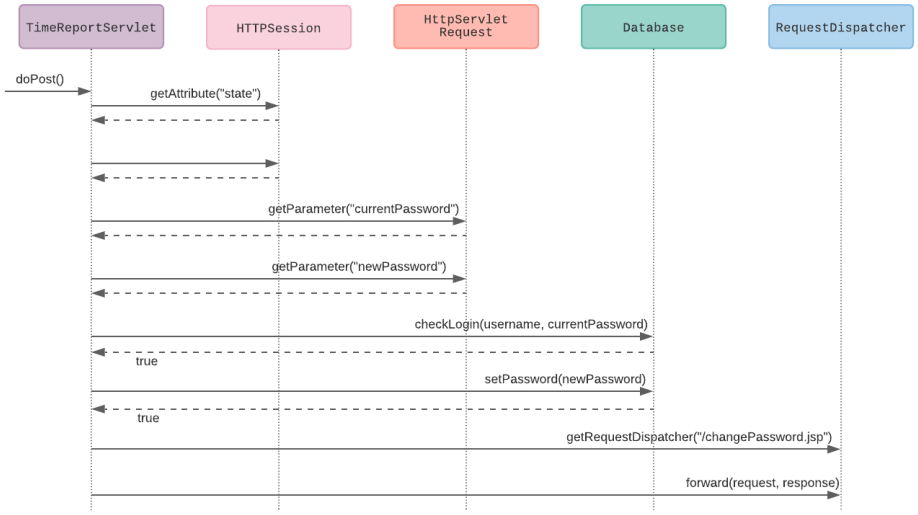

Figure 12: Sequence diagram for a successful password change.

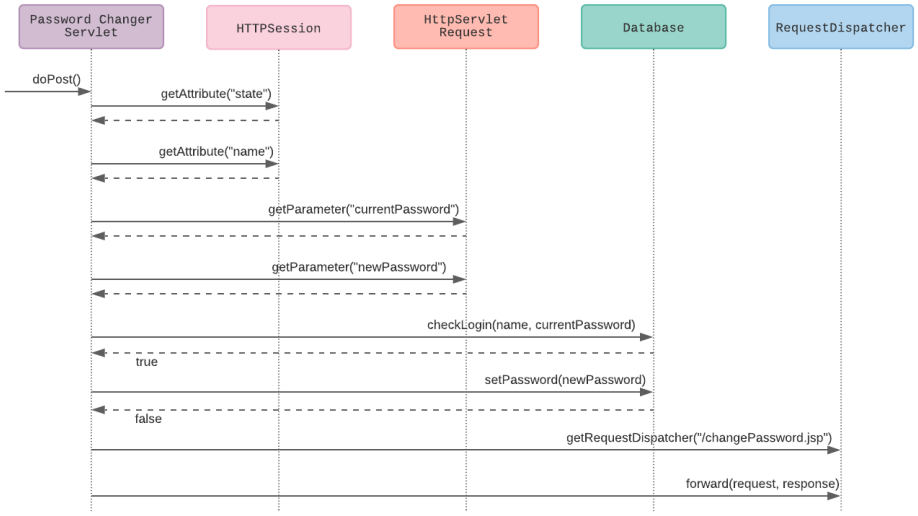

Figure 13: Sequence diagram for a failed attempt to change password.

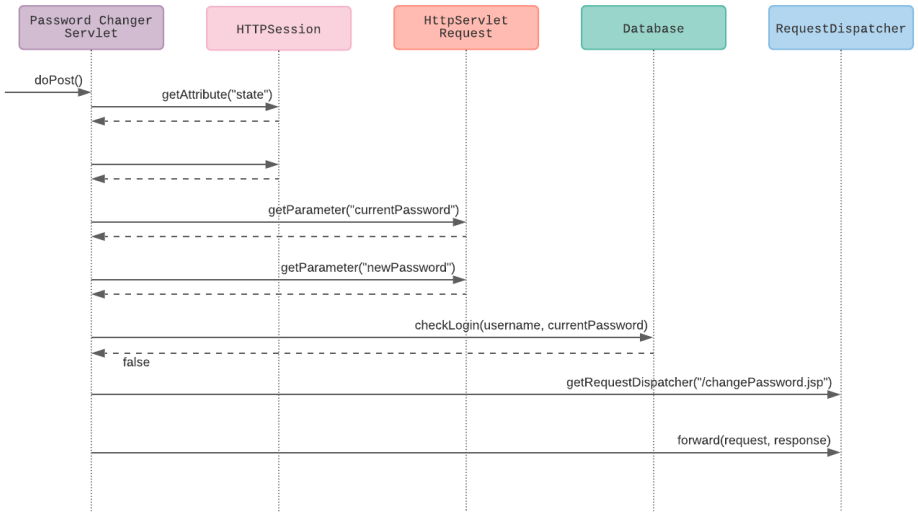

Figure 14: Sequence diagram for a failed attempt to change password due to incorrect user input.

8.4 TimeReportServlet

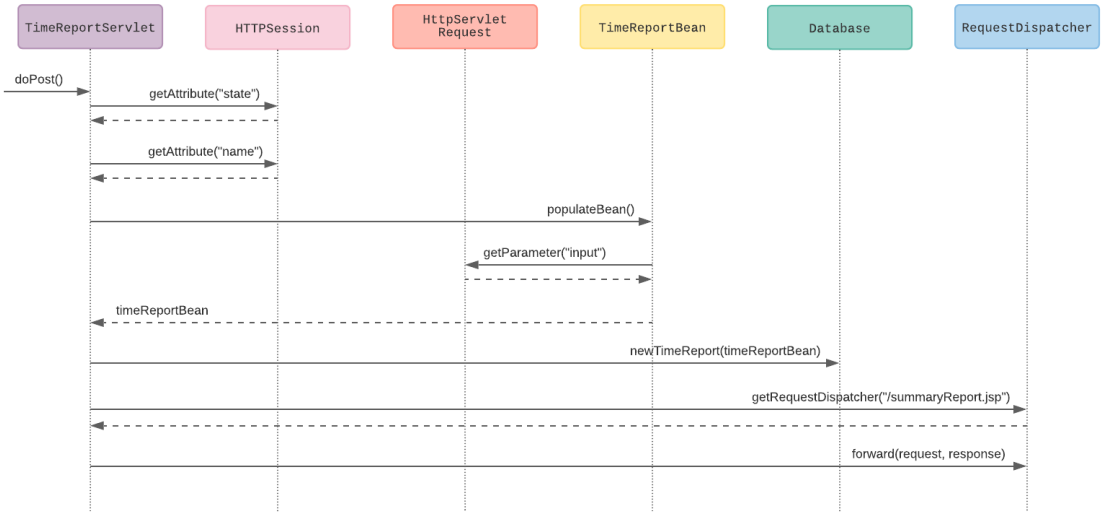

Figure 15: Sequence diagram for adding a new Time Report.

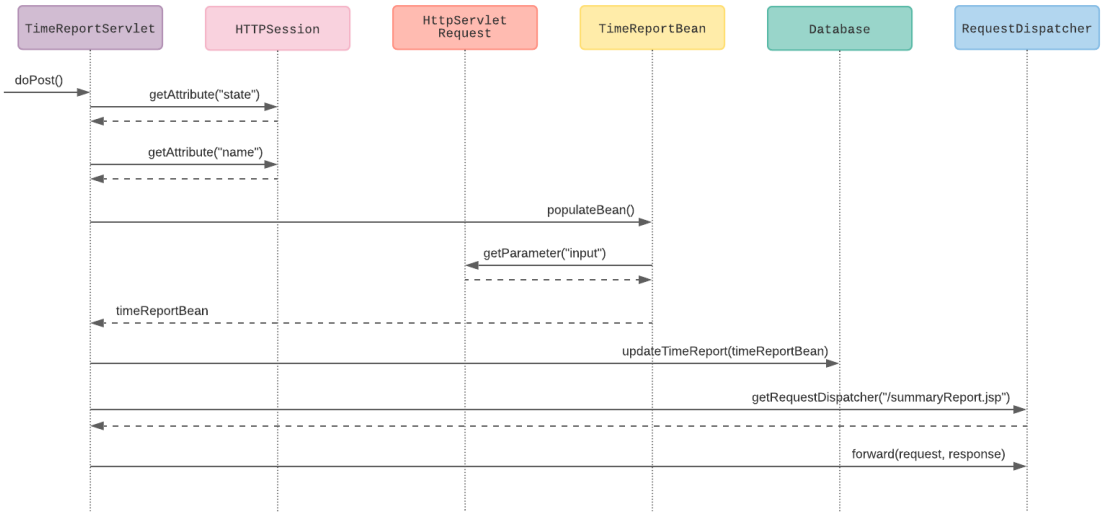

Figure 16: Sequence diagram for a editing a Time Report.

8.5 TimeReportManagementServlet

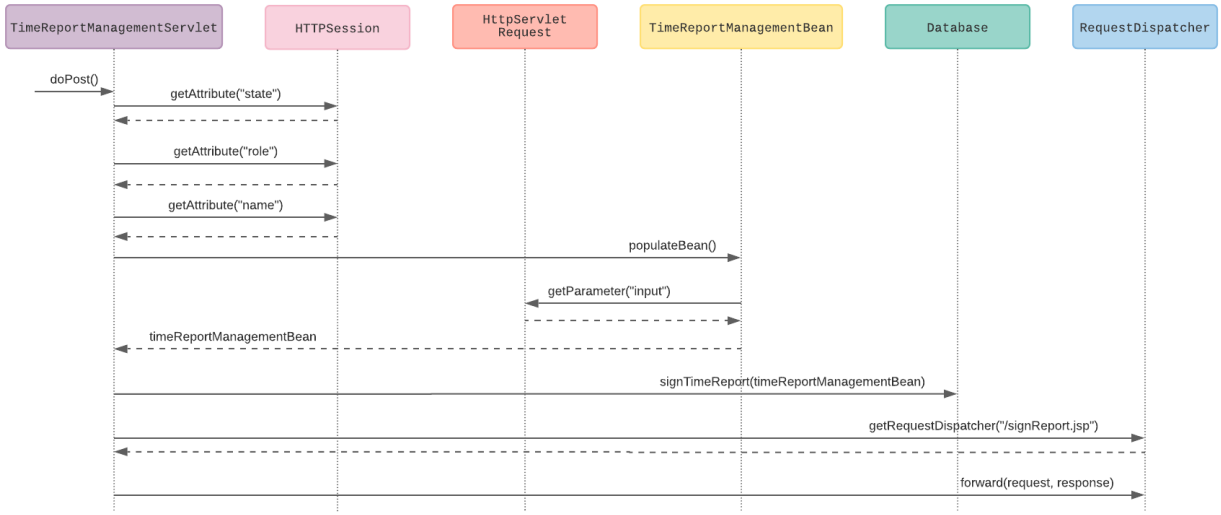

Figure 17: Signing/unsigning a Time Report. This can only be done by the project leader.