Analysis and Design Document for the PROJECT of course TDDC32

Laurent POUYET

Rémi GUILLARD

remgu983@student.liu.se

laupo171@student.liu.se

[TDDC32] – [Project] Laundry booking system

Group A6

|  |  |  |
| --- | --- | --- |
| Revision | Date | Comment |
| 1.0 | 2012/02/29 | First version of document. |
| 1.1 | 2012/03/01 | Add of state, activity and sequences diagrams |
| 1.2 | 2012/03/25 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Summary

This document is the “Analysis and Design Document” for the project “Laundry Booking System” in the context of the Project part for the Course [TDDC32] – “Design and implementation of a software module in Java”. This project is a booking system for a laundry; it will consist in 2 parts: a client and a server. The server will handle a database and answer to the client requests. The client will handle the booking part and the calendar display. The user will be able to book a machine with the help of the GUI client, and then manage and administrate all his booking, when he is authenticated.

The system will be described through different diagrams :

* Class diagrams which represent the architecture of the system.
* A class description which will explain each class.
* Use cases diagrams for the explanation of some actions performed by the user.
* Interaction diagrams for non-trivial use cases.
* State and activity diagrams for the understanding of the behavior of the software

And then the test planning where we plan how to test our requirements.

Content

[Summary 2](#_Toc318456615)

[Document Conventions 4](#_Toc318456616)

[Class Diagram 5](#_Toc318456617)

[The client architecture 5](#_Toc318456618)

[The server architecture 6](#_Toc318456619)

[Class Description 7](#_Toc318456620)

[NetworkPackage 7](#_Toc318456621)

[Client 7](#_Toc318456622)

[Server 7](#_Toc318456623)

[Use case diagrams 8](#_Toc318456624)

[Interaction diagrams 10](#_Toc318456625)

[State and activity diagrams 12](#_Toc318456626)

[Testplanning 14](#_Toc318456627)

# Document Conventions

DB represents the database.

GUI represents the graphical interface.

“user” will be used to describe the customer.

“client” will be used to describe the client part of the software.

“server” will be used to describe the server part, with the database connection.

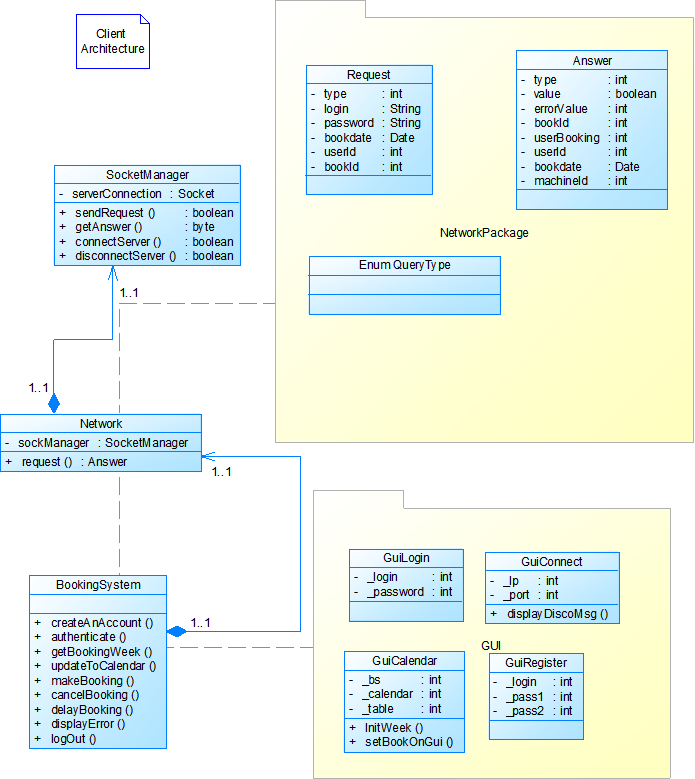
“Booking” will represent a booking for a washing machine.

“BS” represents the booking system.

“Network” represents the network part(Socket, protocol, etc …)

# Class Diagram

## The client architecture



## The server architecture



# Class Description

## NetworkPackage

* Request: it’s a class which simplify the protocol. As we send an object, we can get back another object on the other side of the socket. This class is just a container.
* Answer: same goal as Request. This class is just a container.
* queryType: It’s an enum to know the type of the request

## Client

* SocketManager: This class is used for the establishment of the connection between the server and the client. It’s also in charge of the sending and receiving of data
* Network: This class is in charge of the socketManager and will send the request to the server
* BookingSystem: This is the core of the application, this class is performing all the simple actions the user asks.
* Calendar: This class will handle the calendar data, to be sent to the display
* GUI: these classes will be the different windows.

## Server

* SocketManager: will handle all the sockets from different clients. Get the request and send the answers.
* Network: This class is in charge of the socketManager and will send the request to the server
* bookSystem: This is the core of the server. Will handle all the requests from the users and will perform all verification with the help of the database.
* databaseManager: will handle all type of request to the DB. Just need the parameters to get the needed data.

# Use case diagrams

Use case 1: creating an account and log out.

A user launches the client application. The client attempts to connect to the server. When the connection is established, the client display the login form and a button “Create an account”.

The user clicks on the button, and enters a login, a password and a confirmation of his password. The client sends the request of account creation, and if the login is available, the server returns a confirmation. If there is an error, the user has to do it again and to change the login.

Then the user can login by typing his login and password. The client sent the login request to the server, to check if the credentials are good. If it’s the case, the server sends back a confirmation. If there is an error, the user has to try again.

When the client gets the confirmation, the calendar is displayed with the availabilities of the washing machines and the booking of the user. To log out, the user can click on the button “Log out” to log out.

Use case 2: making a booking.

The user launches the client application and authenticates by typing his credentials. When the server sends back the confirmation of authentication, the booking screen is displayed.

The user can select an available slot by clicking on the wanted slot on the calendar. The client displays a window with the data of the slot. If a washing machine is available, the user can click on “Confirm the booking” to make a booking. A request of booking is send to the server and if the booking is possible, it returns a confirmation. The client displays a window of confirmation. If not the booking is not possible, the server send back an error.

Use case 3: Cancelling or delaying a booking.

The user launches the client application and authenticates by typing his credentials. When the server sends back the confirmation of authentication, the booking screen is displayed.

The user can cancel a booking by clicking on it and then clicking on “Cancel a booking”. He has to confirm his choice to cancel the booking. If not, the booking isn’t modified. If the user confirms the cancel, a request is sent to the server and the server deletes the booking. A confirmation of cancelling is sent to the client.

# Interaction diagrams





# State and activity diagrams





# Testplanning