TICKET RESERVATION SYSTEM

REQUIREMENTS DEFINITION

Version *1.0*

08/20/2013

VERSION HISTORY

Version #	Implemented By	Revision Date	Reason
1.0	*Ivan Ivanov*	*08/20/13*	Initial Requirements Definition draft
1.1	Stoyan Stoyanov	*08/20/13*	Extra Requirements and minor changes
1.2	Ivaylo Mladenov	08/20/2013	Some changes and added requirements.
1.3	Ivan Ivanov	08/20/2013	

TABLE OF CONTENTS

1. Introduction

1. Purpose

The purpose of this document is to define the functional and non-functional requirements of the ticket reservation system.

2. Business Requirements Overview

1. The ticket reservation system is a web application running on a dedicated server. The application will serve for real-time reservation and sales of tickets.
2. Users will be able simultaneously to reserve tickets

3. Non-Functional Requirements

1. Hardware Requirements

No specific hardware requirements

2. Software Requirements

1. The system will support five tickets for reservation

3. Performance Requirements

1. Users shall simultaneously reserve tickets

4. Interface Requirements

1. Users shall be able to connect to the system through web browser, or http requests.

4. FUNCTIONAL REQUIREMENTS

1. The system will be web based
2. The reservation process shall be synchronized i.e. no two users can reserve the same place at the same time.
3. The user, who first initiates the reservation transaction, shall reserve the place.
4. Every ticket shall be button-like and clickable
5. The reserved places shall be different color
6. The system shall notify the user in case of failed reservation
7. Every user should enter his name before requesting a ticket
8. Every ticket slot should be available again after the clear button is pressed

5. TECHNICAL REQUIREMENTS

5.1 APPLICATION REQUIREMENTS

1. The system will run on Apache Tomcat version 6.0 or bigger
2. The system shall be developed using JavaEE
3. JSF framework shall be used
4. All data required for the normal functioning of the system will be stored in server memory
5. Already registered ticket shouldn't be reserved again
6. The system will be thread safe i.e. no two threads will write to the same resources simultaneously
7. The system will provide notification messages in case the requested ticket is reserved by another user
8. A thread will be assigned to every user reserving a ticket
9. For performance and memory purposes the available threads will be five(optional)
10. The user shall be able to decline purchase

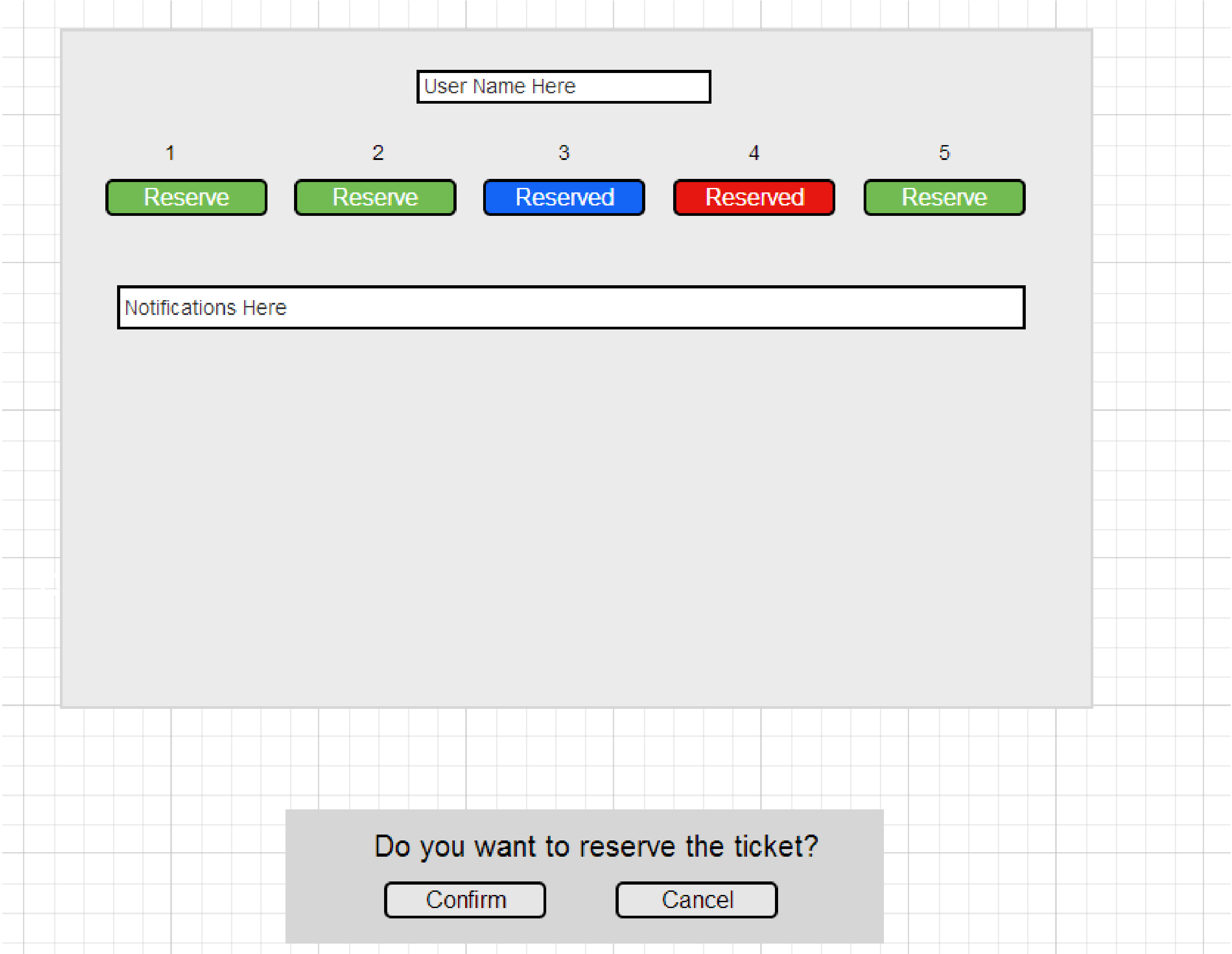
5.2 GRAPHICAL INTERFACE REQUIREMENTS

1.The interface shall be html based

2. The minor styling of the GUI will be done using CSS
3. Ticket reservation buttons should be 120x80 pixels
4. Clear button should be 100x70 pixels
5. Ticket button color will change depending on the status of the ticket:
   - red - payed
   - yellow - reserved
   - blue - reserved by current user
   - green - available
6. Input box for user's name shall be available on the main window
   - 6.1 Allowed number of characters will be 20
7. The notification messages will be 16 pixels in red color
8. Confirmation dialog will appear on user reservation

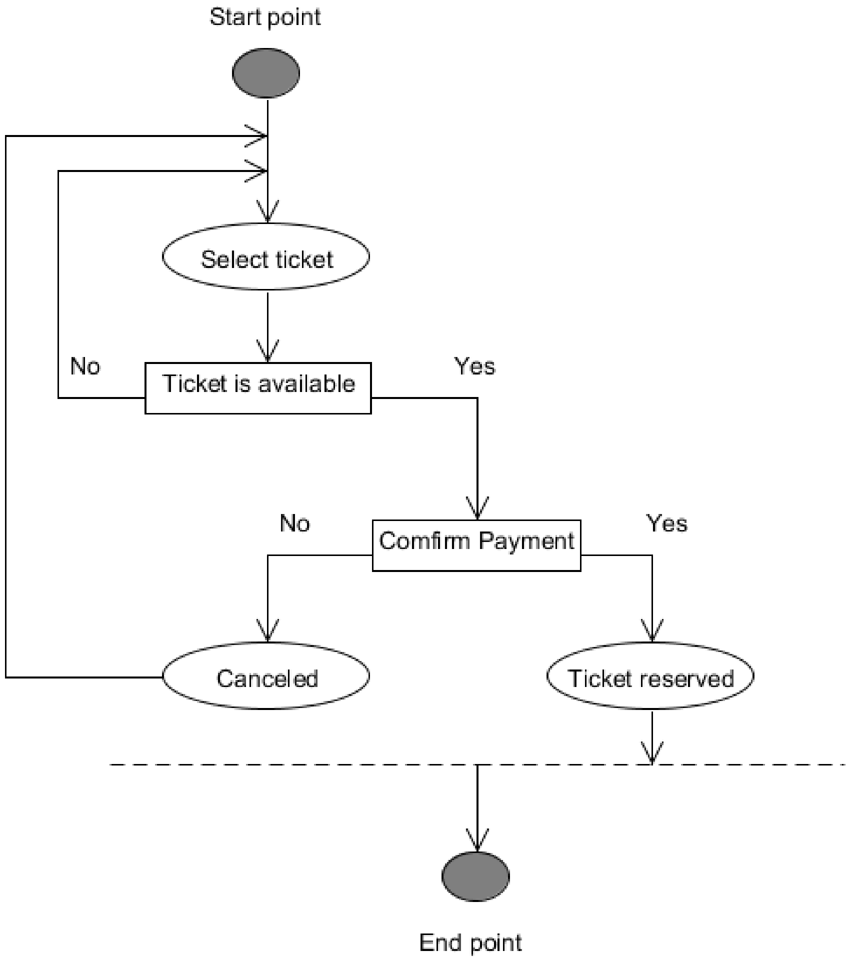
(note)

5.3 MOCKUP



Do you want to reserve the ticket?

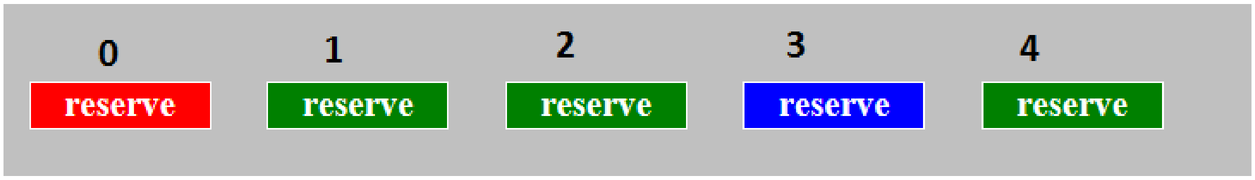
5.4 TICKET RESERVATION PROCESS DIAGRAM



The process starts with the user entering the web page. Then he can choose one of the five tickets. If he/she tries to click on ticket that is already taken an alert box will appear and ask for another choice. If this is not the case and the ticket is available, the user will be asked if the payment will take place or he/she would prefer to cancel the reservation. If reservation is canceled, the user will be redirected back so he/she can choose another ticket. If the payment takes place, a confirmation box appears for the successful booking.

5.5 LOGICAL EXPLANATION

There are 5 tickets available. Each ticket is taken when there is a user associated with it and free when there is no user associated with it. Each ticket has number.

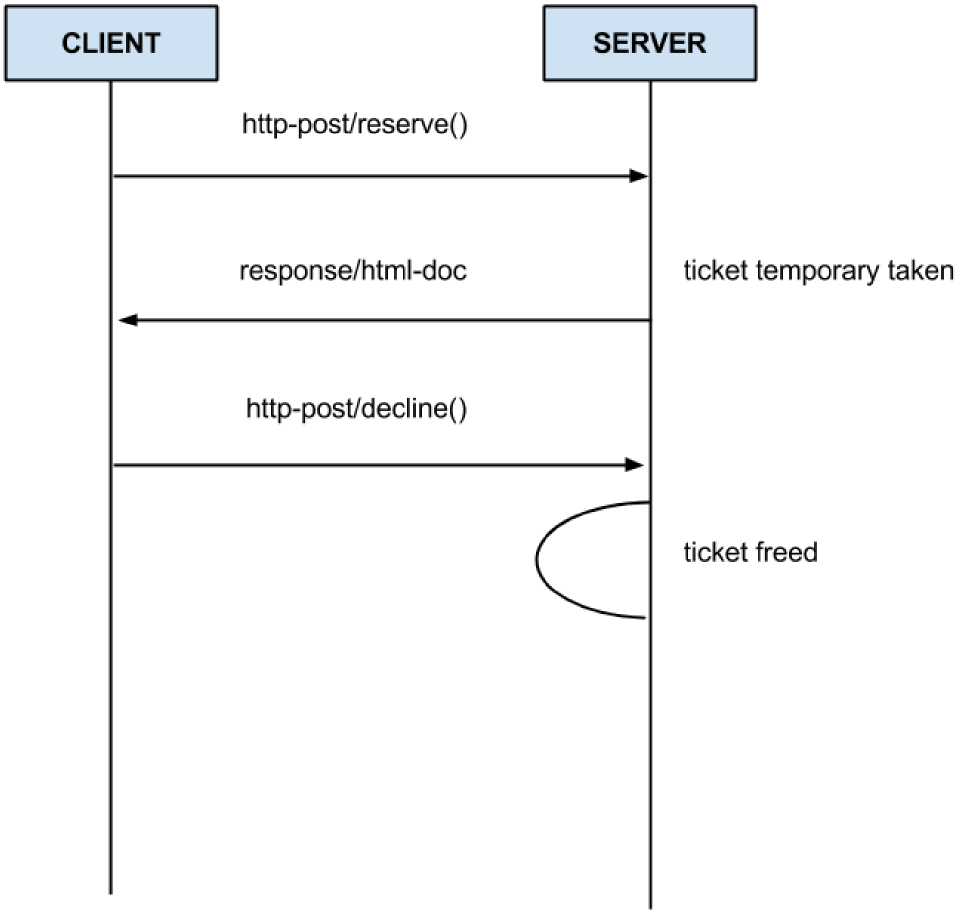


Tickets are stored in a hashmap as key value pairs. The key of the hashmap is the number of the ticket and the value of the hashmap is the name of the user. If the username value is null, then we assume the ticket is free. The application does not take into account separate user sessions. Users can not be recognized uniquely. The logic of the application is based on simple http communication between various users and the web server.

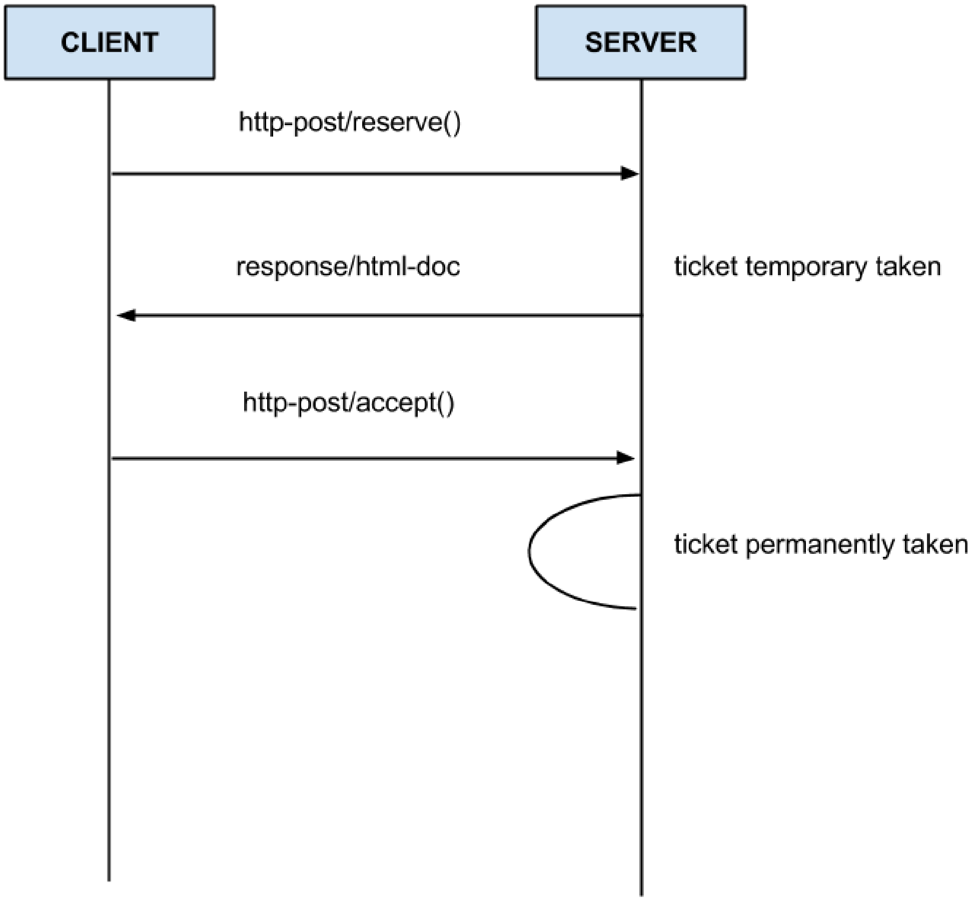
The first user who initiates reservation of the ticket, i.e. clicks on a button to reserve will take the ticket temporary. The the server will send him confirmation. During the confirmation process other users can not reserve the same ticket. If the user denies then the place is freed and other users can initiate reservation. If the user confirms reservation the place stays permanently reserved.

Here is a simple scheme of the communication between user's client machine and the server. The sequence starts when a user clicks *reserve* button.

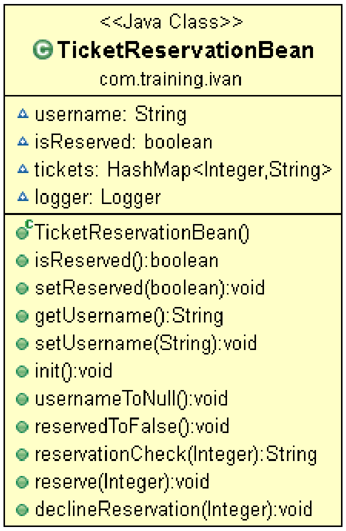
The following sequence diagram shows the "decline" ticket workflow



The following sequence diagram shows the "accept" ticket workflow



5.6 The following bean class is implemented to pragmatically complement the above presented logic.



<<Java Class>>
TicketReservationBean
com.training.ivan

- username: String
- isReserved: boolean
- tickets: HashMap<Integer,String>
- logger: Logger

- TicketReservationBean()
- isReserved():boolean
- setReserved(boolean):void
- getUsername():String
- setUsername(String):void
- init():void
- usernameToNull():void
- reservedToFalse():void
- reservationCheck(Integer):String
- reserve(Integer):void
- declineReservation(Integer):void

The following table summarizes the documents referenced in this document.

Document Name and Version	**Description**	**Location**
RS_Ticket_Reservation_System	*Functional and non-functional requirements development*	https://docs.google.com/document/d/1CycSR7cyeieO_1loqhkA8B82NhEhwU8kn1AhqsBZ3mk/edit