|  |  |  |
| --- | --- | --- |
| PPSP | | **BHH Solutions** |
|  | Cinema Reservation System  Requirements Specification  Version 1.0 | |

PERT, GANT, CPA

Write about waterfall, with Project Plan.

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 03/03/2011 | 1.0 | First release to customer | Michael Bowman |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Software Requirement Specification 3](#_Toc287008641)

[1. Introduction 3](#_Toc287008642)

[1.1 Purpose 3](#_Toc287008643)

[1.2 Scope 3](#_Toc287008644)

[1.3 Definitions, Acronyms and Abbreviations 3](#_Toc287008645)

[2. Overview Description 3](#_Toc287008646)

[2.1 Project plan Gantt chart 0](#_Toc287008647)

[2.2 Class diagram of the proposed system 0](#_Toc287008648)

[3. Specific Requirements 0](#_Toc287008649)

[3.1 Functionality 0](#_Toc287008650)

[4. Design 2](#_Toc287008651)

[5. Pseudo Code 5](#_Toc287008652)

[6. System Features 7](#_Toc287008653)

[Use case diagram 7](#_Toc287008654)

[State Transition Diagram 9](#_Toc287008655)

# Software Requirement Specification

## Introduction

The purpose of this document is to outline the design the cinema reservation system.

The design is based on the requirements detailed in the requirements specification.

This document incorporates the following elements:

* Specification requirements
* Design
* System features

An extensive use of UML (Unified Modelling Language) is made to describe the different artefacts of the system and portray implementation guidelines for the development stage.

* 1. Purpose

The purpose of this project is to provide the three-screen cinema company a system allowing the operators to perform functions which are required. The software will allow customers to book screenings in advance via an operator for peek time showings. The software will allow the customer to select from different categories of seating from the range of films available. Payment for the chosen seats either by credit/debit cards are made full at the time of booking.

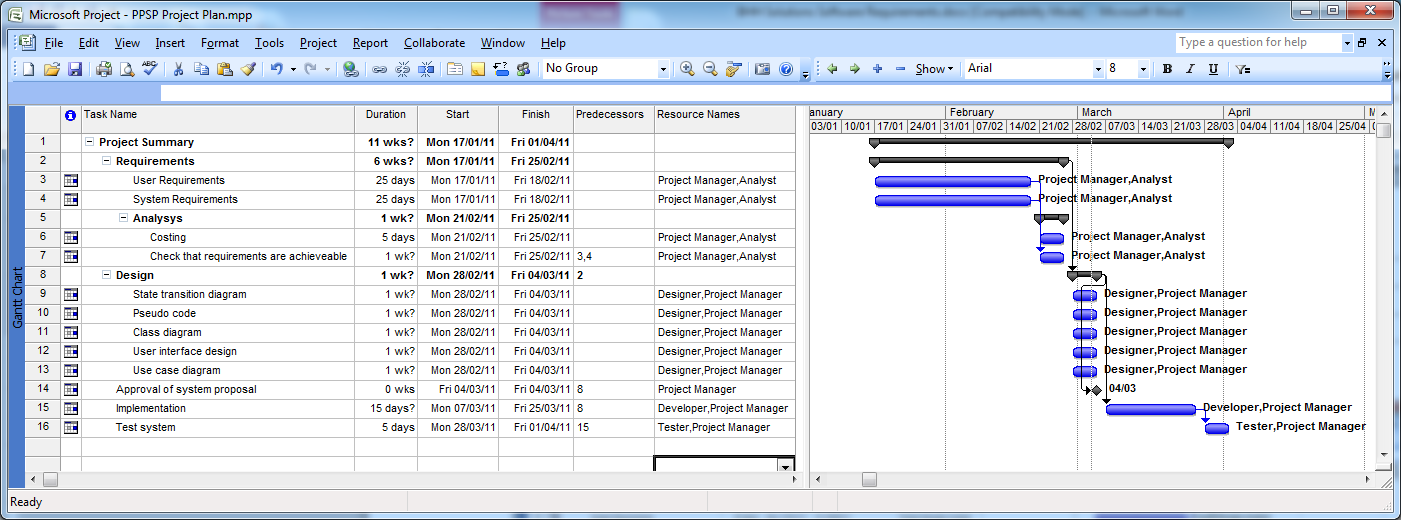
* 1. Scope

Create a new cinema reservation system for a three-screen cinema complex replacing the old paper based system, allowing customers to book screenings for films via an operator. The software will be run on computers for the operators to use. The software is for use inside the cinema complex. The software will show all three-screens and what films are being shown. The seats available for a screen showing can be displayed upon request. Bookings can be made on the available seats; the prices vary depending on the seat type. Payment can be taken via debit/credit card not by phone.

* 1. Definitions, Acronyms and Abbreviations

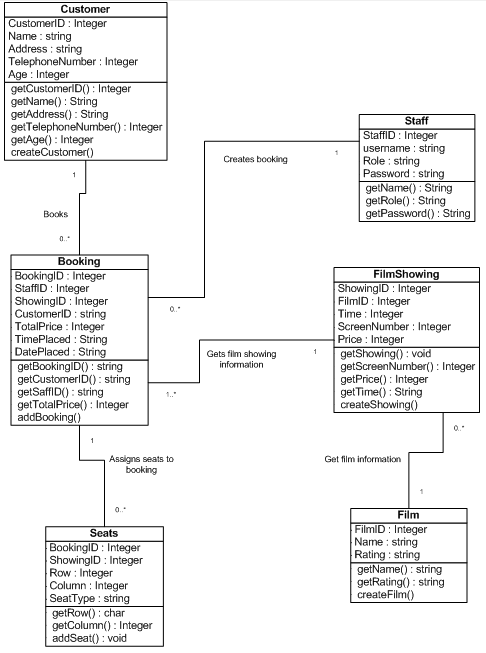
|  |  |
| --- | --- |
| **Term** | **Description** |
| UML | Unified Modelling Language |

1. Overview Description

2.1 Project plan Gantt chart**Figure C.1**

The project plan shown in figure C.1 shows the schedule of the project with detail of each part of the project stage showing which resources are working on the project at each given time.

### 2.2 Class diagram of the proposed system



**Figure C.2**

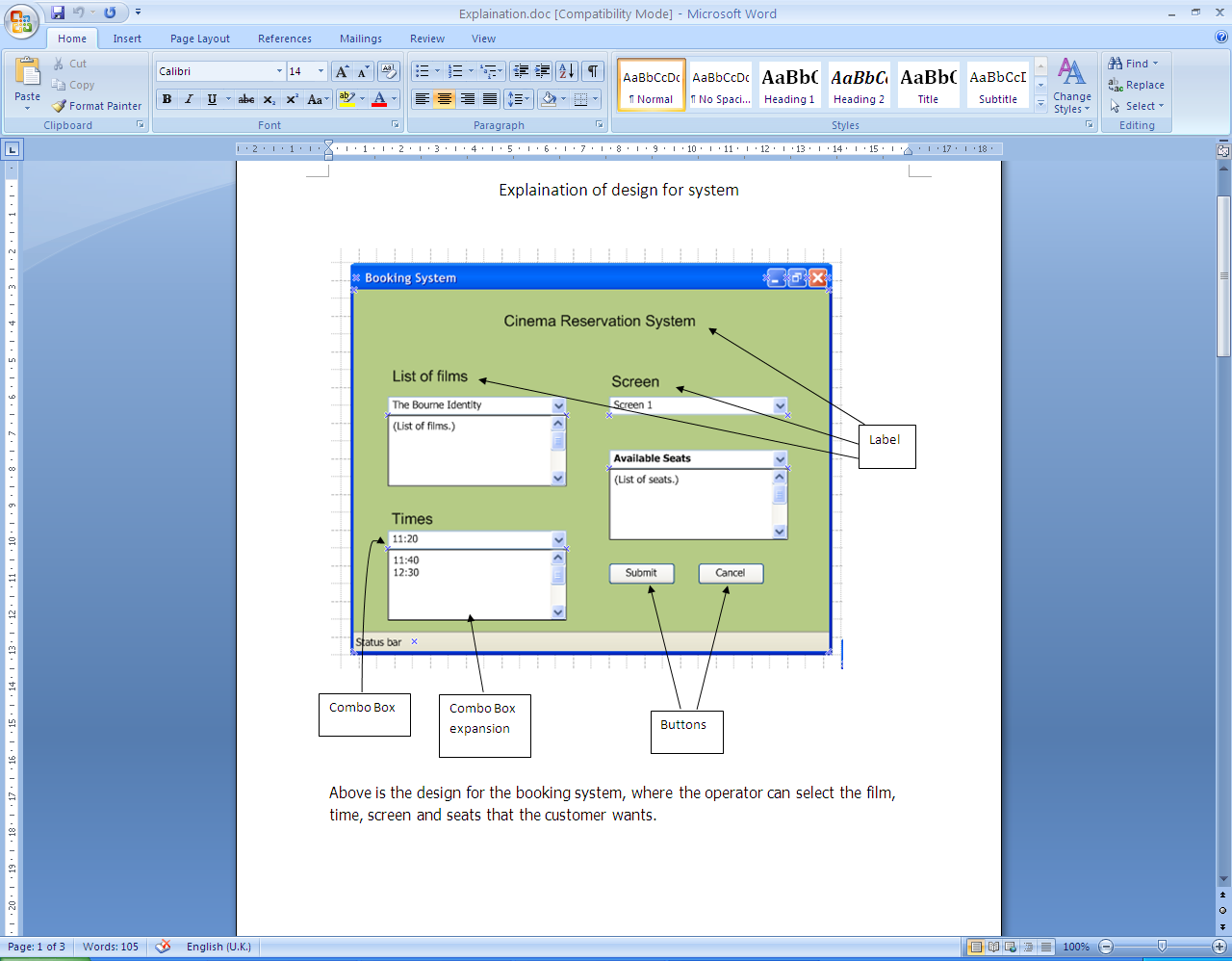
The class diagram shown in figure C.2 shows an overview of the main entities of the software and their relations. All key concepts are detailed in the following sections.

1. Specific Requirements
   1. Functionality
      1. Log operator onto the system \*Important\*
         * All operators must login to the system with a username and password to access the system.
      2. Record the operators bookings
         * Record a log of the bookings by each operator.
      3. Book seats in advance
         * Collect the customer’s details, first name and last name, telephone number.
         * Select a film showing before selecting the seats.
         * Book multiple seats in one booking.
         * Peak times are Saturdays and Sundays between 12-4 and 6-11.
      4. Cancel Bookings
         * Customer’s booking can be cancelled via request of the operator.
      5. View which seats are available
         * Operators can view which seats are available upon request.
         * Operator can see which seats are available for when placing the booking for the customer.
      6. Different prices for premium seats
         * Rows H and I are premium seats.
      7. Able to set pricing dynamically for the cost of premium seats based on percentage of the cost of the standard seating.
      8. Store film information
         * Film name and age rating for each film.
      9. Check customer is of correct age for viewing the film.
   2. **Non-Functional**
      1. Personal information not to be kept longer than necessary
         * Customers credit card information not to be kept longer than necessary 1 year limit.
   3. **Usability**
      1. The operator must be logged into the system to be able to perform any tasks on the system.
      2. Lock the system so only one operator can access one showing at a time.
         * Stop double booking of seats.
   4. **Performance**
      1. No more than 2 minutes to complete a booking.
      2. Show screens and seats less than 3 seconds.
   5. **Design constraints**
      1. Software’s colour scheme to be red white and green if possible
         * These are the cinema’s colour scheme.
      2. Easy to use simple steps to complete a booking
   6. **Licensing Requirements**
      1. **???**
   7. **Legal, copyright and others**
      1. **????**
   8. **Relationship between requirements**

This table shows the relationship between all the requirements showing the importance of them and if there are dependent on each other.

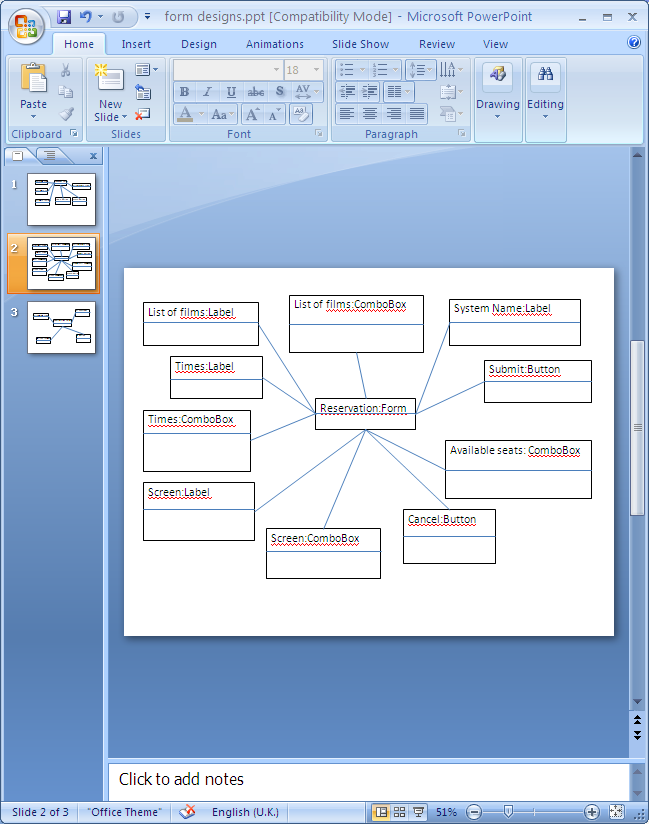
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| REQ ID | 3.1.1 | 3.1.2 | 3.1.3 | 3.1.4 | 3.1.5 | 3.1.6 | 3.1.7 | 3.1.8 | 3.1.9 | 3.2.1 | 3.3.1 | 3.3.2 | 3.4.1 | 3.4.2 | 3.5.1 | 3.5.2 |
| 3.1.1 | 1 | D | N | N | N | N | N | N | N | N | R | N | N | N | N | N |
| 3.1.2 | D | 2 | R | N | N | N | N | N | N | N | R | R | N | N | N | R |
| 3.1.3 | N | D | 1 | D | D | R | R | D | D | R | R | R | R | D | N | R |
| 3.1.4 | N | R | D | 2 | N | N | N | N | N | N | R | N | N | N | N | N |
| 3.1.5 | N | N | D | D | 2 | N | N | D | N | N | R | R | R | D | N | R |
| 3.1.6 | N | N | R | N | N | 2 | D | N | N | N | N | N | N | N | N | N |
| 3.1.7 | N | N | D | N | N | R | 2 | N | N | N | R | N | N | N | N | R |
| 3.1.8 | N | N | D | N | R | N | N | 2 | R | N | R | R | R | R | N | R |
| 3.1.9 | N | N | R | N | N | N | N | D | 2 | N | R | N | R | N | N | R |
| 3.2.1 | R | R | D | R | R | N | N | N | R | 3 | R | N | N | N | N | R |
| 3.3.1 | D | R | R | R | R | N | R | N | R | R | 1 | R | N | N | N | R |
| 3.3.2 | D | R | D | R | R | N | N | R | R | N | R | 2 | R | R | N | R |
| 3.4.1 | R | R | D | N | D | R | R | R | R | R | R | R | 3 | R | N | R |
| 3.4.2 | N | N | R | N | D | N | N | N | N | N | R | R | R | 3 | N | R |
| 3.5.1 | N | N | N | N | N | N | N | N | N | N | N | N | N | N | 3 | N |
| 3.5.2 | R | R | D | N | R | R | R | D | R | N | R | R | R | R | N | 2 |

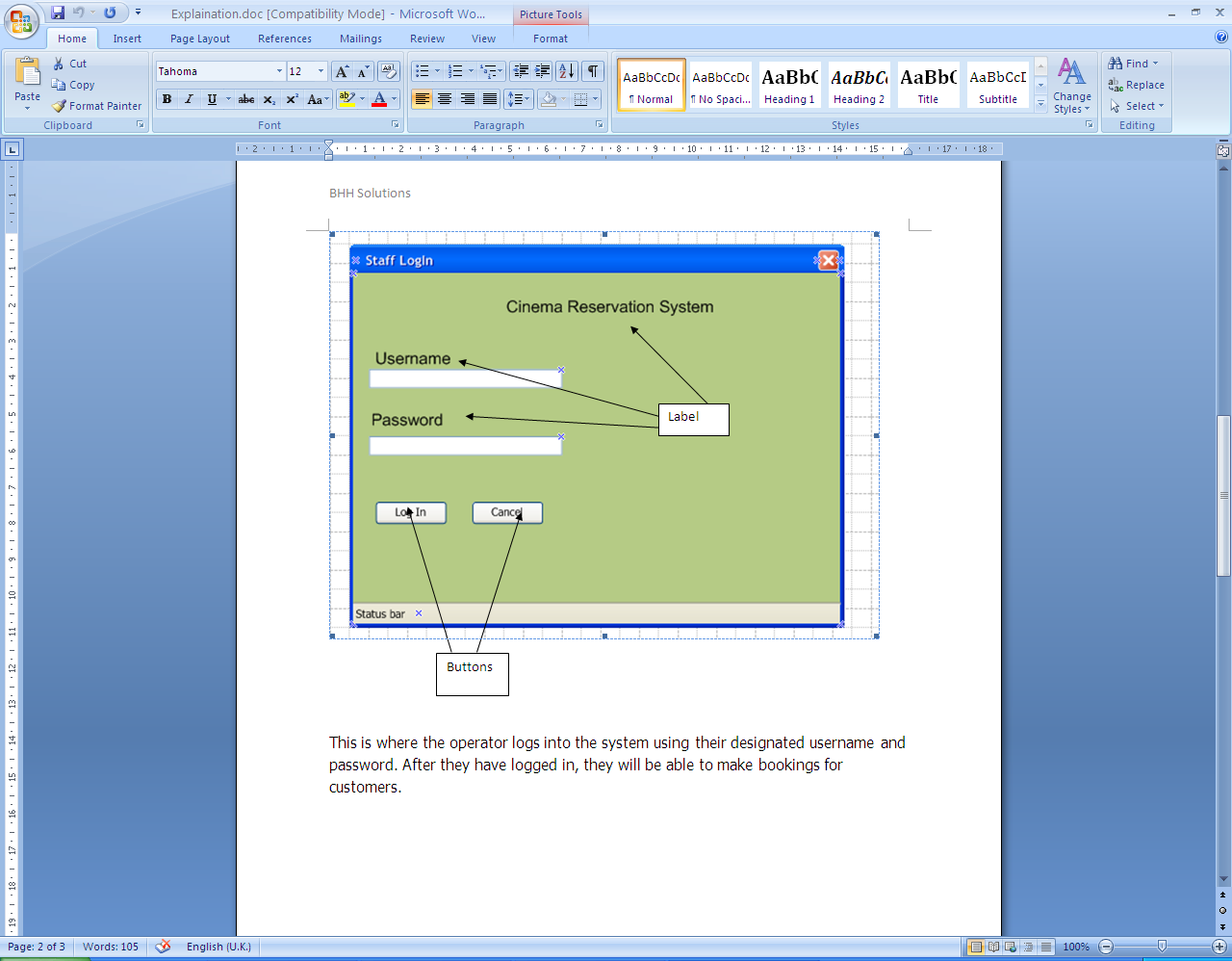
R = Relationship D = Dependency N = Not Related  
1 = Very important 2 = important 3 = not a major requirement

1. Design  
   

**Figure D.1**

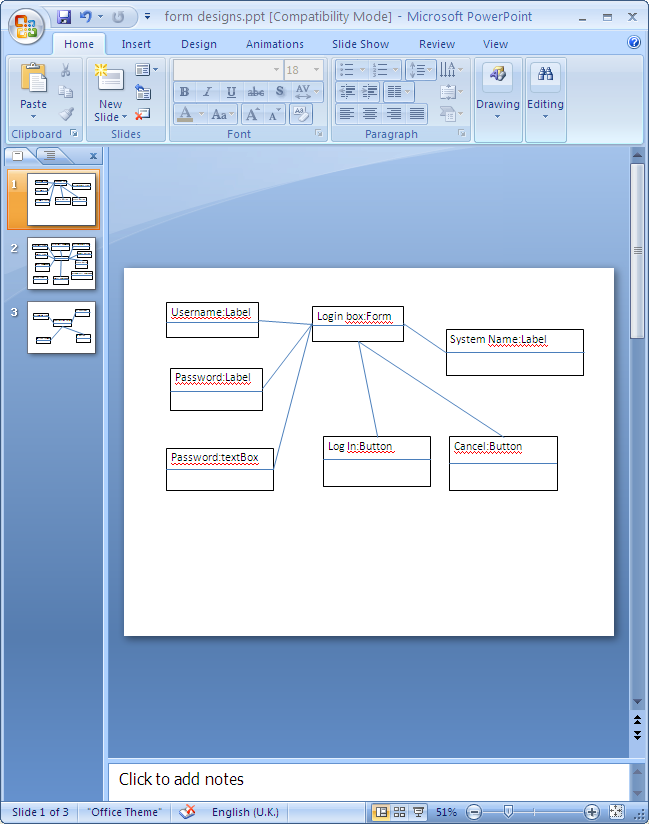
Figure D.1 shows the design for the booking system, where the operator can select the film time, screen and seats that the customer wants for the reservation.

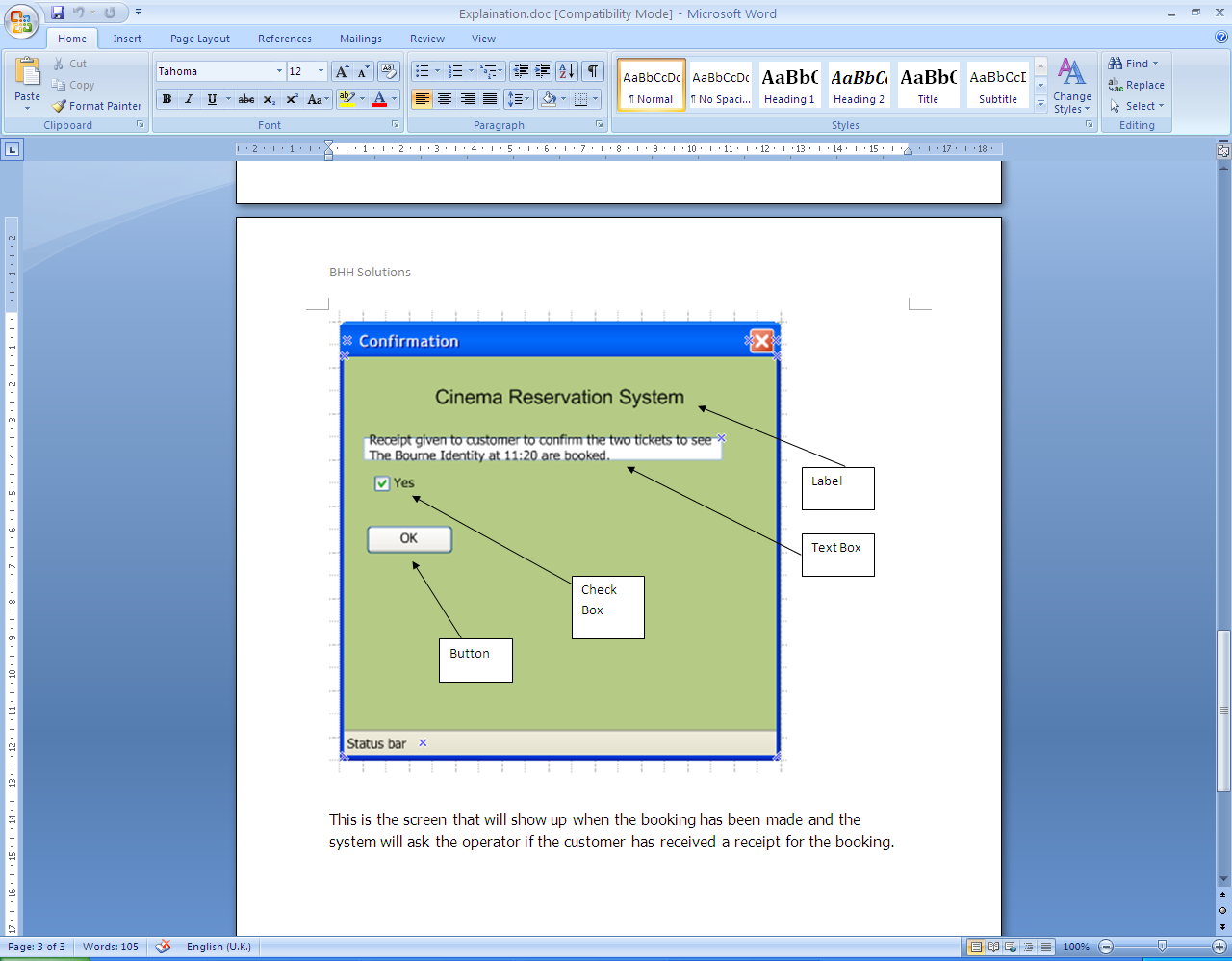




**Figure D.2**

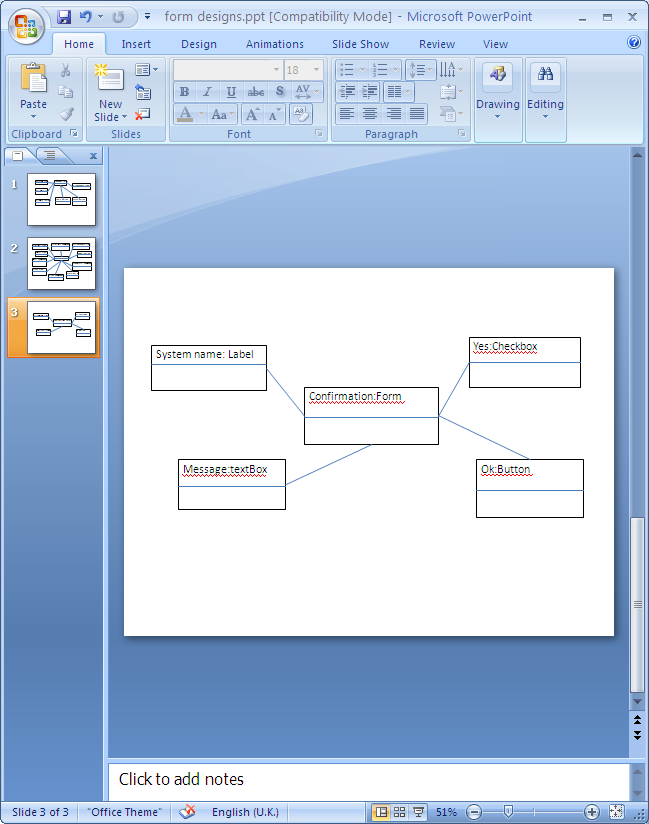
Figure D.2 is the main screen which shows where the operator logs into the system using their designated username and password. After they have logged in, they will be able to make reservations for the customer.





**Figure D.3**

Figure D.3 shows the reservations information and checking that the customer has received the receipt/tickets for the reservation.



1. Pseudo Code

func addBooking(Integer seats, Integer showingID, Integer customerAge, Integer CustomerID)

if customerAge < filmRating then

write “Customer is not old enough to view this film”

return 0

end if

seatID = readInput

Time = readInput

Film = readInput

customerID = CustomerID

ShowingID = readInput

NumberOfSeats = seats

TimePlaced = currentTime

DatePlaced = currentDate

while seatID ! between 0 and 144 then

write “Error not a valid seat. Please input seat again”

seatID = readInput

end while

for 0 to bookingsLength

if currentShowingID == showingID and seatID == currentSeatID then

write “Error seat already booked. Please input seat again”

seatID = readInput

end if

end for

ShowingID = showingID

if seatID between 112 and 144

totalPrice = totalPrice + (priceOfSeat \* pemiumPercentage)

else

totalPrice = totalPrice + priceOfSeat

end if

end func

func setPrice(Integer price)

Price = price

return 1

end func

func staffLogin(String username, String password)

if password == Password and username == Username then

Return 1

else

write "Incorrect Login"

return 0

end if

end func

func viewAvailableSeats(Integer ShowingID)

Integer trigger = 0 // variable to check whether current seat has been marked as not free

for 0 to 144

for 0 to bookingsLength

if currentSeatID == currentBookingSeatID and currentBookingShowingID == Showing ID then

write “seat” + row + column “is not free”

trigger = 1 //mark seat in current loop as not free

exit loop

end if

if trigger == 0 then //if trigger is 0 then the seat was not marked as booked

//therefore it is free

write “seat” + row + column “is free”

end if

end for

trigger = 0 // reset trigger to 0 for next loop

end for

end func

func cancelBooking(Integer bookingID)

for 0 to bookingsLength

if currentBookingID == bookingID then

seatID = null

time = null

film = null

customerID = null

showingID = null

numberOfSeats = null

timePlaced = null

datePlaced = null

totalPrice = null

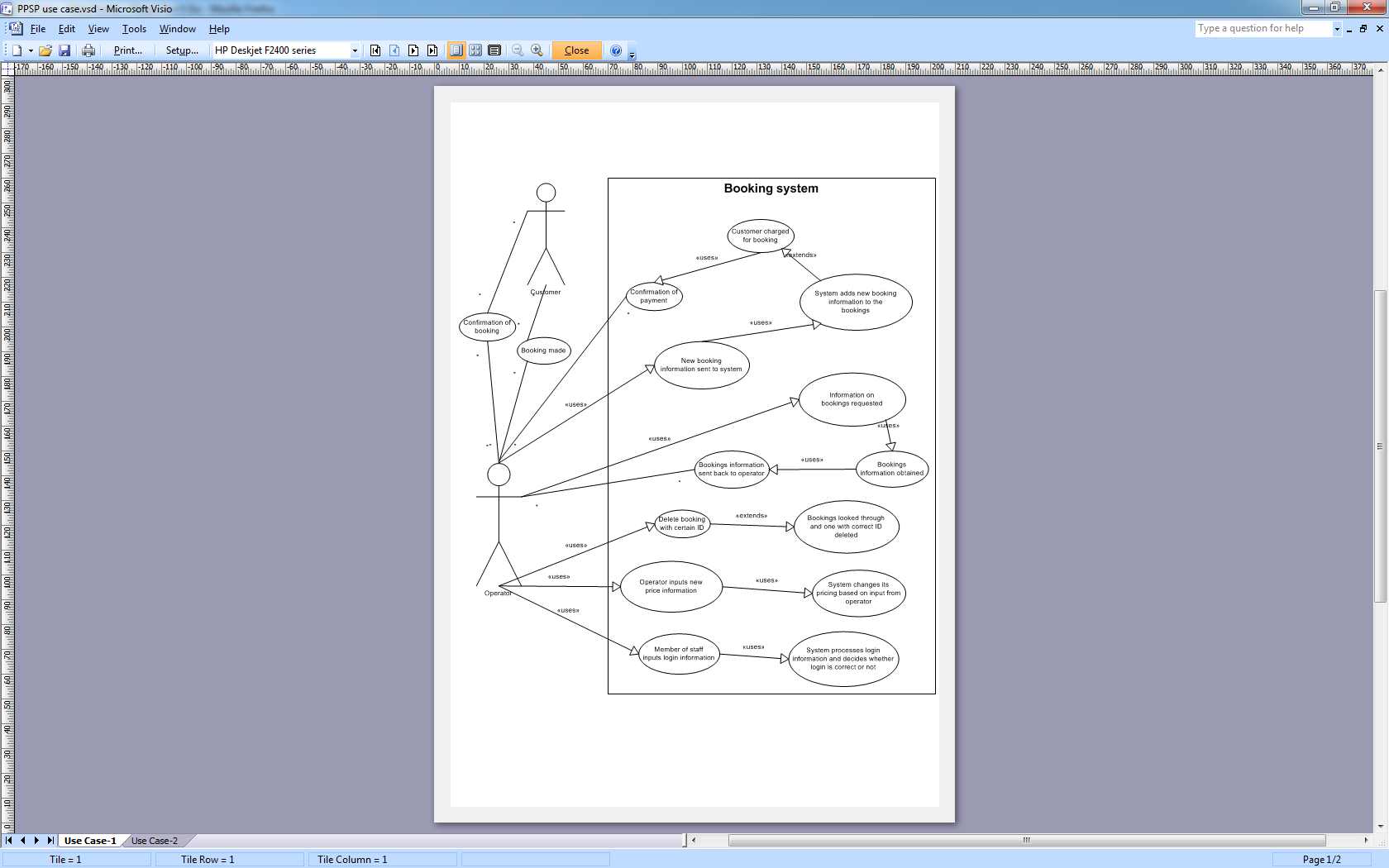
end if

end for

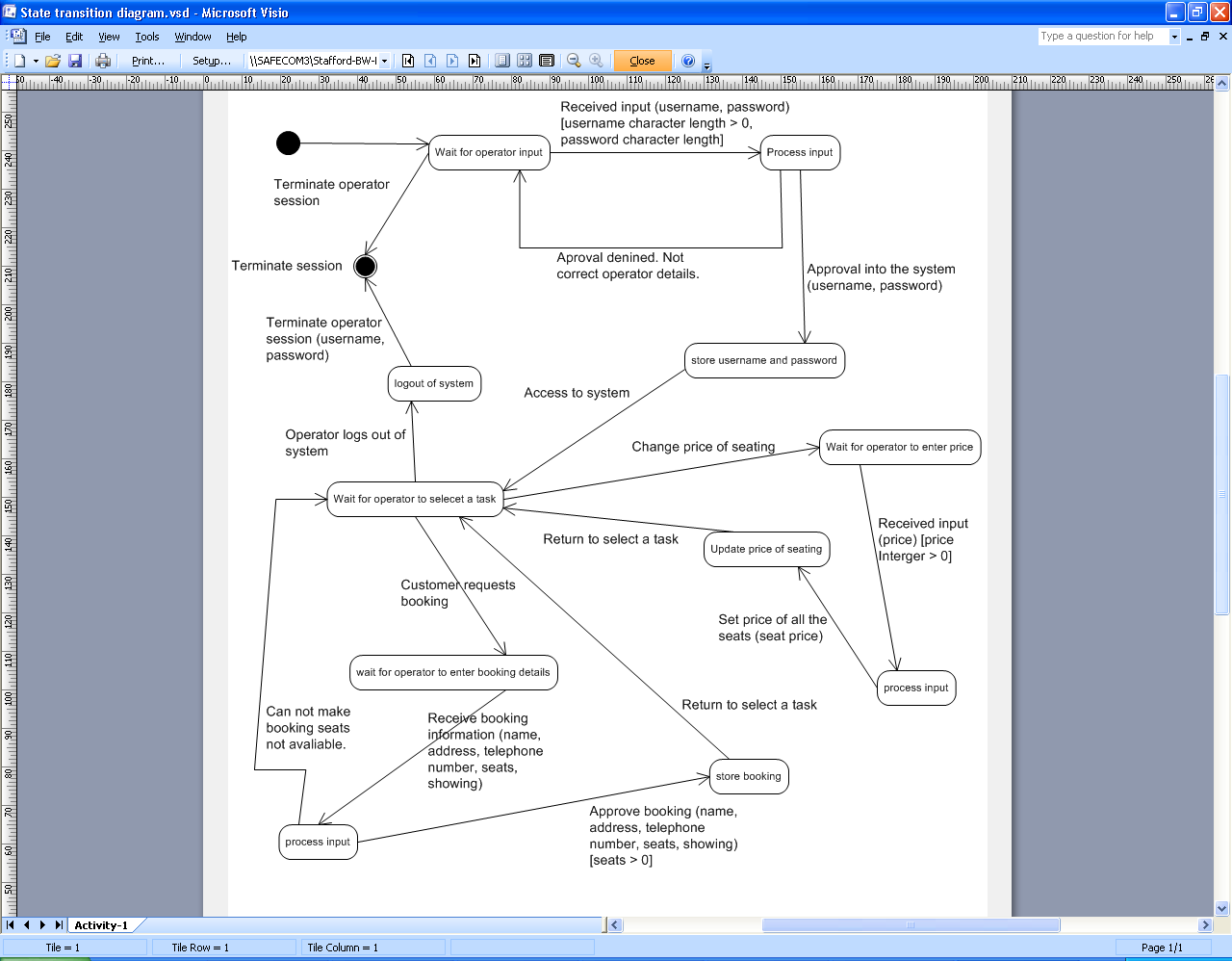
end func

1. System Features

## Use case diagram



Above is a Use Case Diagram, it shows the actors involved in the system as well as the system and the tasks which go on inside the system such as a booking being input to the system.

State Transition Diagram

Above is a State Transition diagram, it shows the various states the proposed system can be in as it performs its operations such as bookings, waiting for an operator to select a task etc.