Requirements Engineering

BowlingSYS

Submitted By: Pawel Kaminski (t00225804)

Computing with Software Development

Date Submitted: 21/04/2023

**Table of Contents**

[1. Introduction/overview 4](#_Toc132980460)

[2. Functional Components 5](#_Toc132980461)

[3. User Requirements 6](#_Toc132980462)

[3.1. BowlingSYS will manage Lanes 6](#_Toc132980463)

[3.3. BowlingSYS will process Bookings 6](#_Toc132980464)

[3.4. BowlingSYS will perform administrative reporting 6](#_Toc132980465)

[4. System Requirements 7](#_Toc132980466)

[4.1. System Level Use Case Diagram 7](#_Toc132980467)

[4.2. Manage Lanes 8](#_Toc132980468)

[4.2.1. Add Lane 8](#_Toc132980469)

[4.2.2. Update Lane 10](#_Toc132980470)

[4.3. Manage Shoes 11](#_Toc132980471)

[4.3.1. Add Shoes 11](#_Toc132980472)

[4.3.2. Update Shoes 13](#_Toc132980473)

[4.4. Manage Booking 15](#_Toc132980474)

[4.4.1. Make Booking 15](#_Toc132980475)

[4.4.2. Cancel Booking 17](#_Toc132980476)

[4.4.3. Update Booking 19](#_Toc132980477)

[4.4.4. Check Arrival 21](#_Toc132980478)

[4.5. Manage Admin 23](#_Toc132980479)

[4.5.1. Yearly Revenue Analysis 23](#_Toc132980480)

[4.5.2. Yearly Booking Analysis 24](#_Toc132980481)

[5. System Model 25](#_Toc132980482)

[5.1. Level-0 DFD 25](#_Toc132980483)

[5.2. Level-1 DFD 26](#_Toc132980484)

[5.3. Level-2 DFD (Lane) 27](#_Toc132980485)

[5.4. Level-2 DFD (Shoes) 28](#_Toc132980486)

[5.5. Level-2 DFD (Booking) 29](#_Toc132980487)

[5.6. Level-2 DFD (Admin) 30](#_Toc132980488)

[6. Data Model (Class Diagram) 31](#_Toc132980489)

[6.1. Class Diagram 31](#_Toc132980490)

[6.2. Relational Schema 32](#_Toc132980491)

[6.3. Database Schema 33](#_Toc132980492)

[7. Conclusion 34](#_Toc132980493)

# Introduction/overview

This system will function as a bowling alley booking system. It will be able to add, update lanes,

add, and update shoes, add, update and check arrival of bookings and lastly provide administrative functions in form of calculating yearly bookings and revenue. The system will track all the bookings as well as the status of the lane and shoes and will calculate administrative functions.

# Functional Components

# User Requirements

## BowlingSYS will manage Lanes

* + 1. BowlingSYS will add a Lane
    2. BowlingSYS will remove a Lane
    3. BowlingSYS will update a Lane
  1. **BowlingSYS will manage Shoes**

3.2.1. BowlingSYS will add Shoes

3.2.2. BowlingSYS will remove Shoes

## BowlingSYS will process Bookings

* + 1. BowlingSYS will make a booking
    2. BowlingSYS will cancel a booking
    3. BowlingSYS will update a booking
    4. BowlingSYS will check arrival

## BowlingSYS will perform administrative reporting

* + 1. BowlingSYS will produce a yearly revenue analysis
    2. BowlingSYS will produce a yearly booking analysis

# System Requirements

The System Requires Lanes and Shoes to function before the Bookings.

## System Level Use Case Diagram

The following system level use case diagram illustrates the high-level system requirements.

Manager

Customer

Receptionist

## Manage Lanes

This module provides functions to add and remove Lanes from the System.

### Add Lane

This function adds a Lane to the System.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Lane | |
| **Use Case Id** | BS001 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds a Lane to this system. A Lane is identified by a Lane\_ID. | |
| **Preconditions** | There is no preconditions | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager invokes the start Lane function **.**  **Step 4:** Manager confirms lane to be added. | **Step 2:** The system assigns the next Lane Number.  **Step 3:** Display UI.  **Step 5:** Set Lane status to “Available”(“A”)  **Step 6:** Save data in Lane File:   * Lane\_ID * Status   **Step 7:** Display confirmation message  **Step 8:** Reset UI. |
| **Alternate Scenarios** | **Actor** | **System** |
|  |  |  |
| **Conclusions** | The Lane has been added to the System. | |
| **Post conditions** | This Lane can be reserved | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Lane

This function sets the status of the Lane to M (Maintenance), A (Available) and U (Unavailable) .

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Lane | |
| **Use Case Id** | BS002 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function updates a lane to M (Maintenance), A (Available) and U (Unavailable) . from this system. A Lane is identified by a Lane\_ID. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager invokes the remove Lane function**.**  **Step 4:** Manager selects the lane to be have status changed.  **Step 6:** Manager confirms the lane to have it’s status changed. | **Step 2:** Retrieve unoccupied lanes from the Lane files.  **Step 3:** Display UI.  **Step 5:** Ask for user confirmation  **Step 7:** Set status to Unavailable to the User’s request. (M , A or U).  **Step 8:** DisplayConfirmation message  **Step 9:** Reset UI |
| **Alternate Scenarios** | **Actor** | **System** |
|  |  |  |
| **Conclusions** | The Lane Status has been changed. | |
| **Post conditions** | The Lane can no longer be Booked. | |
| **Business Rules** | Lanes that are available cannot be Booked. | |
| **Implementation Constraints** |  | |

## Manage Shoes

### Add Shoes

This Function adds Shoes to the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Shoes | |
| **Use Case Id** | BS003 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds Shoes to this system. Shoes are identified by a  ShoesID. | |
| **Preconditions** | There is no preconditions | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager invokes the start Equipment function **.**  **Step 3:** Manager enters the required data.   * Size * Quantity | **Step 2:** Display UI.  **Step 4:** Validate data   * All fields must be entered * Quantity must be between 1 and 99.   **Step 5:** Create Shoes\_IDFor the quantity specified:   * Assign Next Shoes\_ID * Save data in Shoes Files * Shoes\_ID * Size * Status ‘A’   **Step 6:** Display Confirmation Message  **Step 7:** Reset UI. |
| **Alternate Scenarios** | **Actor** | **System** |
| **Invalid data Enter** |  | **Step 5:** Invalid data detected  **Step 6**: Display an **appropriate**  Error message  **Step 7:** Return to Step 4 |
| **Conclusions** | The Shoes have been added to the System. | |
| **Post conditions** | This pair of Shoes can be used. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Remove Shoes

This Function removes Shoes on the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Remove Shoes | |
| **Use Case Id** | BS004 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function removes Shoes on this system. | |
| **Preconditions** | At least one SlotNo row has to exist in total. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager invokes the updates shoes function **.**  **Step 4:** Manager selects SlotNo to be set as removed   * SlotNo   **Step 7:** Manager confirms the message. | **Step 2:** The system retrieves the available Shoes from Shoe files.  **Step 5:** Validate data   * SlotNo must be a number between 1-999 and exist in the database   **Step 8:** Ask for confirmation message.  **Step 9:** Update the selected row from the Shoes table that corresponds with the selected SlotNo.  **Step 10:** Display confirmation message  **Step 11:** Reset UI. |
| **Alternate Scenarios** | **Actor** | **System** |
|  |  |  |
| **Conclusions** | The Shoes have been Removed on the System. | |
| **Post conditions** | Shoes have been changed. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Booking

### Make Booking

This Function Make a Booking in the System.

Receptionist

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Make Booking | |
| **Use Case Id** | BS005 | |
| **Priority** | 1 | |
| **Source** | Receptionist | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** |  | |
| **Description** | This function makes a booking on the System. A booking is identified by a Booking\_ID. | |
| **Preconditions** | There has to be at least one available Lane and Shoes. | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System** |
|  | **Step 1:** The Receptionist invokes the start Booking function **.**  **Step 4:** The Receptionist enters the required data.   * Forename * Surename * Booking\_Date * Booking\_Time * Lane\_ID * NumofPeople   **Step 8:** Receptionist confirms the message. | **Step 2:**The System assigns the next Booking\_ID.  **Step 5:** Validate the data:   * All fields must be entered * Forename has to be all letters between 2 and 30 * Surname has to be all letter between 2 and 40 * Date has to be after SYSDATE.   **Step 6:** System Calculates Booking Cost, as well as Assigning an available lane by checking the data entered.  **Step 9:** Ask for confirmation message.  **Step 10:** Save data in Booking files and Set the status of the Lane to Unavailable ‘U’ and Shoes to Unavailable ‘U’ on the date specified to the booking.  **Step 9:** Display confirmation message  **Step 10:** Reset UI. |
| **Alternate Scenarios** | **Actor** | **System** |
| **Invalid data Enter** |  | **Step 6:** Invalid data detected  **Step 7**: Display an **appropriate**  Error message  **Step 8:** Return to Step 3 |
| **Conclusions** | The Booking has been added to the System. | |
| **Post conditions** | The Lane and Shoes are not available to book on the System | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Cancel Booking

This Function Cancels a Booking in the System.

Receptionist

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Cancel Booking | |
| **Use Case Id** | BS006 | |
| **Priority** | 1 | |
| **Source** | Receptionist | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** |  | |
| **Description** | This function Removes a booking on the System. | |
| **Preconditions** | There has to be a Booking on the System. | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System** |
|  | **Step 1:** The Receptionist invokes the cancel Booking function **.**  **Step 4:** The Receptionist sbooking to be removed.  **Step 6:** Receptionist confirmsthe message. | **Step 2:**The System retrieves all bookings from booking files.  **Step 3:** Display UI.  **Step 5:**Validate data   * All fields must be entered. * Booking\_ID searched must be a number between 1 and 9999   **Step 5:** Ask for confirmation message.  **Step 7:** Remove the selected row from the Booking table that corresponds with the selected Booking\_ID.  **Step 8:** Display confirmation message  **Step 9:** Reset UI. |
| **Alternate Scenarios** | **Actor** | **System** |
|  |  |  |
| **Conclusions** | The Booking has been removed from the System. | |
| **Post conditions** | The Lane and Shoes are now available to book on the System. | |
| **Business Rules** | A refund can be only issued if a booking is cancel 6 hours before booking time. | |
| **Implementation Constraints** |  | |

### Update Booking

This Function Updates a Booking in the System.

Receptionist

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Booking | |
| **Use Case Id** | BS007 | |
| **Priority** | 1 | |
| **Source** | Receptionist | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** |  | |
| **Description** | This function Updates a booking on the System. | |
| **Preconditions** | There has to be a Booking on the System. | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System** |
|  | **Step 1:** The Receptionist invokes the Update Booking function **.**  **Step 4:** The Receptionist selects booking to be updated.  **Step 5:** The Receptionist Modifies the required data.   * Forename * Surname * Booking\_Date * Booking\_Time * Lane\_ID * NumofPeople   **Step 9:** Receptionist confirmsthe message. | **Step 2:**The System retrieves all bookings from booking files.  **Step 3:** Display UI.  **Step 6:**Validate data   * All fields must be entered. * Forename has to be all letters between 2 and 30 * Surname has to be all letter between 2 and 40 * Date has to be after SYSDATE * Time has to 2 digits between 16-21 * NumofPeople has to be between 1-9   **Step 7:** System Calculates Booking Cost, as well as Assigning an available lane by checking the data entered.  **Step 8:** Ask for confirmation message.  **Step 10:** Save data in Booking files and Set the status of the Lane to Unavailable ‘U’ and Shoes to Unavailable ‘U’.  **Step 11:** Display confirmation message  **Step 12:** Reset UI. |
| **Alternate Scenarios** | **Actor** | **System** |
| No lanes available |  | **Step 7:** System Calculates Booking Cost, as well as Assigning an available lane by checking the data entered if none available it responds with an error message telling the user that no Lanes are available. |
| **Conclusions** | The Booking has been updated on the System. | |
| **Post conditions** | The Lane and Shoes are not available to book on the Day of the Booking. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Check Arrival

This Function Checks Arrival in the System.

Receptionist

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Check Arrival | |
| **Use Case Id** | BS008 | |
| **Priority** | 1 | |
| **Source** | Receptionist | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** |  | |
| **Description** | This function Changes the Booking Status of a Booking to “A – Arrived”. | |
| **Preconditions** | There has to be a Booking on the System. | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System** |
|  | **Step 1:** The Receptionist invokes the Check Arrival function **.**  **Step 4:** The Receptionist selects booking to be set as “A ” - Arrived and the sizes of the shoes required.  **Step 6:** Receptionist confirmsthe message. | **Step 2:** The System retrieves all available Shoe Sizes and makes them distinct.  **Step 3:** Display UI.  **Step 5:** Ask for confirmation message.  **Step 7:** Change Status of the booking to “Arrived”.  **Step 8:** Display confirmation message  **Step 9:** Reset UI. |
| **Alternate Scenarios** | **Actor** | **System** |
| No Lanes Available |  |  |
| **Conclusions** | The Booking status has been set as “Arrived”. | |
| **Post conditions** | The booking went through. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Admin

### Yearly Revenue Analysis

This Function analyses Yearly Revenue in the System.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Yearly Revenue Analysis | |
| **Use Case Id** | BS009 | |
| **Priority** | 3 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function analyses Yearly Revenue on the System. | |
| **Preconditions** | There has to be at least one booking. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Receptionist invokes the start Yearly Revenue Analysis function **.**  **Step 3:** The Receptionist selects the year of the analysis. | **Step 2:** The System retrieves booking files.  **Step 4:** The System calculates the yearly and monthly revenue from the files as well as placing the results on a graph.  **Step 5:** Display UI. |
| **Alternate Scenarios** | **Actor** | **System** |
|  |  |  |
| **Conclusions** | The System calculated yearly and monthly revenue. | |
| **Post conditions** | The Analysis of the yearly and monthly revenue is available to be used by the Manager. | |
| **Business Rules** |  | |
| **Implementation**  **Constraints** |  | |

### Yearly Booking Analysis

This Function analyses Yearly Booking in the System.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Yearly Booking Analysis | |
| **Use Case Id** | BS010 | |
| **Priority** | 3 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function analyses Yearly Revenue on the System. | |
| **Preconditions** | There has to be at least one booking. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Receptionist invokes the start Yearly Booking Analysis function **.**  **Step 3:** The Receptionist selects the year of the analysis. | **Step 2:** The System retrieves booking files.  **Step 4:** The System calculates the yearly and monthly Bookings from the files as well as placing the results on a graph.  **Step 5:** Display UI. |
| **Alternate Scenarios** | **Actor** | **System** |
|  |  |  |
| **Conclusions** | The System calculated number of people per month. | |
| **Post conditions** | The Analysis of the yearly booking is available to be used by the Manager. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

Bowling

System

Booking Details

Customer

Booking Details

## Level-1 DFD

Lane Details

Lane Files

D1

Manage Lanes

Manage Shoes

Shoe Details

Shoe Details

Lane Details

Shoe Files

D2

Lane Details

Shoe Details

Shoe Details

Lane Details

Manage Bookings

Admin Files

D4

Admin Details

Admin Details

Booking Details

Booking Details

Booking Details

Bookings Files

D3

Manage Admin

Booking Details

Booking Confirmation

Receptionist

## Level-2 DFD (Lane)

Add Lane

Lane Details

Lane Details

Update Lane

Lanes File

Lane Details

## Level-2 DFD (Shoes)

Add Shoes

Shoes Details

Update Shoes

Shoes Details

Shoes File

Shoes Details

## Level-2 DFD (Booking)

Lanes Details

Lanes File

Make a Booking

Lanes Details

Cancel a Booking

Booking Details

Booking Details

Booking Details

Bookings File

Booking Details

Booking Details

Booking Details

Booking Details

Booking Details

Update Booking

Check Arrival

Shoes Details

Shoes Details

Shoes File

## Level-2 DFD (Admin)

Bookings File

Yearly Booking Analysis

Yearly Revenue Analysis

Admin File

# Data Model (Class Diagram)

## Class Diagram

Shoes

Lane

**SlotNo:** int

**ShoeSize:** int

**BookingRef:** int

**Lane\_ID:** int **Lane\_Status:** String

(1..1)

(1..1)

(1..1)

(1..6)

Admin

(1..1)

**Booking\_Year:** int **Total\_Bookings:** int  
**Total\_Revenue**: int

Booking

**Booking\_ID:** int

**Lane\_ID:** int  
**Forename:** String **Surname:** String **Booking\_Date:** String **Booking\_Time:** int

**Booking\_Status:** String **NumOfShoes:** int

**Booking\_Cost:** int

(1..\*)

## Relational Schema

Relational schema for the data requirements - Using ***bracket notation***

**Booking**(Booking\_ID, Lane\_ID, Forename, Surname, Booking\_Date, Booking\_Time, Booking\_Status, NumOfShoes, Booking\_Cost)  
**Lane**(Lane\_ID, Lane\_Status)  
**Shoes**(SlotNo, ShoeSize, BookingRef)  
**Admin**(Booking\_Year, Total\_Bookings, Total\_Revenue)

## Database Schema

A definition of the database to be implemented.

This includes primary key, foreign key and other constraints to be implemented.

**Schema:** BowlingSYS   
   
**Relation Booking**   
Forename: Varchar2(20) NOT NULL  
Surname: Varchar2(30) NOT NULL  
Booking\_Date: Date NOT NULL  
Booking\_Time: Numeric2) NOT NULL  
Booking\_Status: Char (1) DEFAULT ‘S’

NumOfShoes: Numeric(1) NOT NULL

Booking\_Cost: Numeric(5, 2) NOT NULL  
Booking\_ID: Numeric(3)  
Lane\_ID: Numeric(2)  
**Primary Key:** Booking\_ID  
**Foreign Key:** Lane\_ID

**Relation Lane**Lane\_Status: Char(1) DEFAULT ‘U’

Lane\_ID: Num (2)

**Primary Key:** Lane\_ID

**Relation PigeonHoles**

ShoeSize: Numeric(2)

BookingRef: Numeric (3) DEFAULT NULL

SlotNo: Numeric(3)

**Primary Key:** SlotNo

**Relation Admin**Total\_Bookings: Numeric(5)

Total\_Revenue: Numeric(10)

Booking\_Year: Numeric (4)

**Primary Key:** Booking\_Year

# Conclusion

In conclusion this system provides the user with a fully functional bowling alley booking interface as well as keeping track of all the bookings and revenue.