David Ortiz

Dmitry Mikhailov

Andrew Hopkins

Joseph Hackett

Pool Access System Design Report

IST 311 - Group 2

The Pennsylvania State University

Table of Contents

[Introduction 3](#_Toc529191395)

[Use Case Diagram 4](#_Toc529191396)

[Use Case Specifications 5](#_Toc529191397)

[Class Diagram 9](#_Toc529191398)

[Class Responsibilities 10](#_Toc529191399)

[Activity Diagrams 13](#_Toc529191400)

[State Diagrams 14](#_Toc529191401)

[Deployment Diagram 14](#_Toc529191402)

[User Manual 14](#_Toc529191403)

# Introduction

# Use Case Diagram



## Use Case Specifications

Register Swimmer

**Functional Requirements**

The functional requirement of Register Swimmer is that costumers are added to and have their information stored in the program for later authentication at the pool.

**Nonfunctional Requirements**

* The input form will need to be clear and user-friendly while also being comprehensive
* Formatting should make input fields easy and obvious
* Error checking will need to occur before submitting data to the database
* Management will need to upload pictures from digital camera or scanner, the app will not include picture-taking software itself.

**Task Analysis**

1. A manager selects Register Customer from the navigational menu
2. The application displays a form with several input fields for general applicant information as well as radio button options for swimming ability.
3. The manager fills these out according to information supplied by the applicant
4. The manager selects the submit button
5. The application either displays a success message in the case of successful submission, or an error message addressing unfilled fields, faulty server connection, etc.

**Design Challenges:**

* The input form needs to be designed in a way that is simple, efficient, and easy for the user; the process should enhance the registration experience, not make it more difficult.
* The photo that is used for the profile will likely need to fit a pre-established dimension or the application will need to support photo cropping and positioning.
* The input form should provide sufficient error-checking before submission to minimize incorrect data (ages and zip codes need to be integers, swimming ability must be selected, etc.)

**Pre and post conditions:**

|  |  |
| --- | --- |
| Pre-conditions | Post-conditions |
| Applicant is verified by management as a member of the community | Success message is displayed |
| Applicant is in good-standing and has paid any necessary fees | Database stores and maintains membership data |
| Applicant consents to pool rules and agrees to uphold them  Applicant is physically at the management office | Member is able to check-in at the pool immediately following registration |
| Applicant’s picture has been taken or is on file |  |
| Manager successfully logs into the app |  |
| Connection to the database server is present |  |

Look-up (Search) Customer

**Functional Requirement:**

The functional requirement of Look-up Customer use case is that the application displays a searchable table of all registered pool customers that match the search criteria.

**Non-functional requirements:**

* Search table is updated instantly and constantly on key release
* Search results are selectable and needs to display buttons to check in/out and display customer profile, depending on actor type (operator or admin)

**Task Analysis:**

1. Pool Operator or Admin (the actor) select Search Customer from Main Menu
2. The screen displays a search bar which is used to locate specific customer by name, address, or phone number
3. The actor clicks on customer result and clicks Check-In
4. The system adds the customer checked in to the list of current swimmers

**Design Challenges:**

* Customer information needs to be accurate
* The application has to determine user’s role (operator or admin)
* Display check in or out appropriately depending on customer’s state
* Customer database has to be accurate and updated for new customers
* The workflow needs to be easy to follow to avoid confusion and mistakes
* Clickable search results needs to display data pertaining to selected customer only

**Pre- and Post-conditions:**

|  |  |
| --- | --- |
| Pre-conditions | Post-conditions |
| Actor role needs to be specified | Search accurately displayed results |
| Customer must be present in front desk | Actor was able to view customer profile |
| Customer must be registered | Actor successfully changed customer’s state |
| List of registered customers updated |  |

View Reports

**Task Analysis**

1. The actor (Pool Admin) selects ‘Reports’ from the Main Menu.
2. The application displays the list of supported report types.
3. The actor selects the required type of the report.
4. The application displays the form with the generated report based on the default parameters. The form also includes components for configuring the report and setting up other parameters (e.g., start date and end date).
5. The actor adjusts the parameters of the report.
6. The application displays the updated report.

**Functional Requirements**

1. Show the log of all visits by Customers during the specified period of time (two parameters: from date, to date). For each visit display the name of the Customer, the name of the Operator, date, check-in time, check-out time.

Allow sorting the report by any field: Customer’s name, Operator’s name, date, check-in time, check-out time.

1. Show the log of all customers who visited the pool during the specified period of time. For each customer display the number of visits, the average time, and the total time spent in the pool.

Allow sorting the report by any field: number of visits, the average time, and the total time spent in the pool.

1. Display the attendance numbers grouped by hour or by week-day, or both given the specified period of time. Information of minimum load, maximum load, and the average load must be included.

**Non-functional Requirements**

1. Any report must be generated within 2 seconds or less.
2. The reports must be properly formatted.
3. The width of the columns must be adjustable.

**Design Challenges**

* the size of the customer’s database is unknown. The optimal solution depends on the size of the database.
* the test database for past visits must be generated somehow.
* the choice of the visualization instruments is limited in core Java.

**Pre- and Post-conditions**

|  |  |
| --- | --- |
| Pre-conditions | Post-conditions |
| Actor (user) must be logged-in | Actor has not reported a problem with reading the reports. The reports are legible. |
| Application has a connection to the customer’s database | Actor can navigate between different types of reports and other options of the Main Menu |
| Application can read and write logs (implies the need for persistent storage) |  |

View Current Swimmers

**Functional Requirement:**

The functional requirement of View Current Swimmers is that the application shows a list of current swimmers on the front page of the Pool Access System.

**Non-functional Requirements:**

* Customer is added to the list when checked-in
* The name appears on the front page
* When the person is selected from the list, all of their information appears, including: Name, address, date of birth, any allergies or medical conditions, and a photo of the individual.

**Task Analysis:**

1. Pool member comes to the pool and checks in with the lifeguard.
2. Lifeguard uses the system to check the member in.
3. The screen displays the member’s name in the current swimmers list.
4. The lifeguard can click on the name, view all information, and check the individual out from there.

**Design Challenges:**

* Users information must be accurate and already in the database.
* Search function must work correctly for lifeguard to check-in resident.
* Lifeguards must have a way to ensure people are removed when members forget to check out.

**Pre and post conditions:**

|  |  |
| --- | --- |
| Pre-conditions | Post-conditions |
| Member is registered in the system | Member checks out with the lifeguard when leaving |
| Member’s privileges are active | Members in and out time are noted in attendance log |
| Member comes to the pool and checks in |  |

# C:\Users\ortizda\Downloads\diagram_v3.jpgClass Diagram

## Class Responsibilities

**Controller Class List**

|  |  |
| --- | --- |
| Class | Description |
| LoginController | LoginControlelr coordinates user input on the LogInUI and data within EmployeeRepository to determine if user is authenticated. When authenticated, an Employee object is passed to the MainMenu which will be used to determine access rights. |
| MainMenuController | MainMenuController acts as a nexus between the four main controllers and listens to user input from the MainMenuUI to determine which controllers to activate. It also allows users to logout and return to the loginUI. |
| SearchController | SearchController takes care of validating and searching for a registered Swimmer from the pool’s SwimmrRepository |
| RegisterController | The register controller monitors the registerUI, validates data on submission, and sends data to proper classes to create a new swimmer in the swimmer repository. |
| ViewPoolController | ViewPoolController creates the VewPoolUI and populates it with list of Swimmer objects currently contained in the Pool object. ViewPoolController listens to user input on the ViewPoolUI and is informed when to check-out swimmers. In turn, it removes swimmers from the list in the Pool object and then updates the ViewPoolUI with the new list. |
| ReportsController | ReportsController takes care of validating and creating a customized report from the pool’s repositories. |

**View Class List**

|  |  |
| --- | --- |
| Class | Description |
| LogInUI | LogInUI contains textfields for username and password submit, a submit button, and label to display input errors. Inputs are read by the LogInController to determine errors. |
| MainMenuUI | MainMenuUI contains the four use case buttons and a log out button. Availability of the buttons is informed by the MainMenuController and determined by access rights in an Employee object. |
| SearchUI | SearchUI takes displays fields that can be filled out before performing the query for a Swimmer |
| SearchResultUI | SearchResultUI displays and parses the result from the SearchController |
| RegisterUI | RegisterUI displays the fields necessary for an employee to register a new swimmer. This display can only be accessed with proper employee access rights. |
| ViewPoolUI | ViewPoolUI displays a list of Swimmer objects passed to it by the ViewPoolController. Essential swimmer information is displayed as well as a ‘check-out’ button for each swimmer. Check-out will inform the ViewPoolController to remove the Swimmer from the pool. |
| ReportsUI | ReportsUI holds the fields which constructs the query for the report |
| ViewReportUI | ViewReportUI parses and displays the report from the ReportController |

**Model Class List**

|  |  |
| --- | --- |
| Class | Description |
| Person (*interface*) | An interface which makes sure Employee and Swimmer classes have a getFirstName(), getLastName(), and getFullName() method. |
| Swimmer (abstract) | Swimmer is abstract and contains all necessary information regarding pool members necessary for look-up, identification, and pool safety. |
| Adult | A Swimmer subclass, responsible for adult swimmers. These have phone numbers for emergency contact and lookup |
| Youth | A Swimmer subclass, responsible for child swimmers. These have an Adult guardian and a Skill level. |
| Visit | Visit documents individual occurrences of when a swimmer checked into and out of the pool. |
| SwimmerStatistic | SwimmerStatistic keeps working calculations concerning statistics of each swimmer’s overall pool usage. Total visits, average visit time, and total visit time are tracked for each swimmer. |
| PeriodStatistic | PeriodStatistics keeps working calculations concerning the basic statistics of aggregate pool usage. The hours of maximum and minimum visitation are calculated for each day as well as the daily average. |
| Skill | Skill documents the overall swimming proficiency of the swimmer as well as any tests they have passed or special requirements for safety given their age or swimming ability. |
| Address | Contains attributes with different aspects of an address and enables searching by address as well as name. |
| Pool | Pool is an aggregator and encapsulator of SwimmerRepository, EmployeeRepository, and VisitRepository for easy access. |
| SwimmerRepository | Responsible for management of a list of Swimmers, enabling CRUD operations and handling its storage. |
| EmployeeRepository | Responsible for management of a list of Employees, enabling CRUD operations and handling its storage. |
| VisitRepository | Responsible for management of a list of Visits, enabling CRUD operations and handling its storage. |
| Role (Enum) | Role is of type Enum, and its constants are LOOKUP, CHECKIN,, CHECKOUT, VIEW\_REPORTS, CREATE\_PROFILE, EDIT\_PROFILE. |
| Employee (abstract) | Employee is abstract and contains all employee information such as firstName, lastName, empID, and credentials of type Credential |
| Admin | Admin stores any employees with elevated privileges. It stores empID, firstName, lastName, role, and a Credential object |
| Operator | Operator stores any employees with regular privileges. It stores empID, firstName, lastName, role, a Credential object, and an Admin object |
| Credential | Credential stores the Employee’s credential values (empID and password). It implements interface Login for verification |
| Login (*interface*) | Login is an interface with a sole method: verifyLogin. This returns true or false |
| Report | Report is subclass of AbstractReport and superclass of all other reports. It contains the start and end dates that will define the period of the report as well as tabular data in the form of the abstractreport attribute. |
| AbstractReport | AbstractReport is a java class used to contain data in tabular formats that may be easily abstracted into tables of the view class. It is extended into Report in our project. |
| ReportSearch | ReportSearch extends Report and provides a list of swimmers matching specific criteria entered by the user. The criteria may be confined to a particular attribute (name or DOB) or it may be a general search through all swimmer attributes. |
| ReportVisits | ReportVisits extends Report and provides check-in/check-out data given a specific start and end date. It may be applied to a particular swimmer or include all swimmers. |
| ReportSwimmer | ReportSwimmer extends Report and provides average visitation time, total visitation time, and visit count data given a specific start and end date. |
| ReportPeriods | ReportPeriods extends Report and provides daily maximum and minimum visitation data as well as daily average data given a specific start and end date. |
| ReportActiveSwimmers | ReportActiveSwimmers extends Report and provides data tallying the number of swimmers active at the pool in hourly increments given a specific start and end date. |
| Status (abstract) | StatusType is abstract with Permanent, Temp, and Pending as subclasses. It makes sure each type of Status has a toggle and generate comment function. |
| StatusChange (*Interface*) | An interface used to force the implementation of isPermitted() and generateComment() in sub classes regarding user status. |
| Permanent | Connected to swimmer to mark them as a permanently registered members of the pool. Extends status interface. |
| Temp | Used to mark a temporary status as a member of the pool. Includes variables to note expiration date and remaining days on their temporary status. |
| Pending | Status used to define a swimmer who has a pending membership. Implements the Status interface. |
| Contact | The contact class contains the name, phone number, and relationship of an emergency contact supplied on a swimmer’s profile. |

# Activity Diagrams

# C:\Users\ortizda\Downloads\Activity Diagram-3.jpg

# State Diagrams

# Deployment Diagram

# User Manual