David Ortiz

Dmitry Mikhailov

Andrew Hopkins

Joseph Hackett

Pool Access System Project Proposal

IST 311 - Group 2

The Pennsylvania State University

# Table of Contents

|  |  |
| --- | --- |
| Introduction | Page 3 |
| Scenario catalog | Page 4 |
| Use case diagram | Page 9 |
| Use case specifications | Page 10 |
| User stories | Page 14 |
| Product backlog | Page 15 |
| Sprints | Page 17 |
| Class diagram: specification | Page 23 |
| Class responsibilities | Page 24 |
| Sequence diagrams | Page 27 |

# Introduction

The Pool Access System (PAS)’s overarching goal is to eliminate the need for physical passes that provide access to a pool and physical sign in sheets to track attendance. The final product will allow pool attendants to check members in simply by looking them up. Checking in and out will automatically track attendance numbers, reducing a large amount of administrative work for attendants. The development period is broken up into five sprints, each lasting 2 weeks.

Sprint #1 focuses around developing the functionality to allow member information to be registered into the system. This includes proper error checking functionality, as well as developing proper access levels to be used.

Sprint #2 focuses on additional registration details such as updating a member’s registration. It also focuses on creating proper JSON files to support the systems member registries.

Sprint #3 focuses on creating the necessary search functionality for the system to allow attendants to look up members, and begins the process of developing the “view” or User Interface by creating proper panels to allow search results and member info to be displayed.

Sprint #4 focuses on finishing up all functionality to allow members to access the pool. This sprint will include creating the “current swimmers” section of the application and all UI functionality necessary.

Sprint #5 will focus on the proper logging of pool attendance numbers, updates to the members registration, etc. This functionality is implemented last because it relies on proper new member registration, and checking in and out of members to function correctly.

We believe that following the development strategy and schedule listed below in detail will allow PAS to be developed at a quick and consistent speed.

# Scenario Catalog

## Register Swimmer

* ***Scenario Title*** – New customer wants to sign up for pool privileges
* ***Actor*** – Mike and Susan, parents of two children
* ***Setting*** – in the front desk at pool location
* ***Scenario Goal*** – Sign up members of family for access to pool while storing information for later use.
* ***Scenario Narrative*** – Mike and Susan’s family enjoys swimming. They would like to go to the pool on a regular basis without needing to go through a lengthy process of checking in and giving information such as the kid’s swimming abilities, ages, and other important data. They need identifying information saved into their profile so that they do not have to repeatedly give it to the front desk staff every time they come in. After a successful account is set up, the management system will store their names, address, pictures, children’s skill levels, if kids can swim alone or only with an adult, safety equipment needed to swim, etc. It will also keep track on the number of times they attend the pool, any improvements in skill, etc.
* ***Utility Analysis***
* ***Benefits***
  + Quick access to pool
  + No invasive questions every time they want to swim
  + Increased safety as staff knows kids’ swimming ability and adjusts supervision as needed
  + Keep track of kids’ skills as it improves
* ***Costs***
  + Application development
  + Storage space needs to be scaled accordingly to ensure reliability and availability
* ***Risks***
  + Make sure the service is worth signing up (does it save time?)
  + Ensure privacy across different customer accounts
  + Ensuring account data is updated accordingly so that it accurately represents family information
  + May feel like an invasion of privacy when first filling out

## Remove Swimmer

* ***Scenario Title*** – Existing customer wants to cancel their service
* ***Actor*** – Steve and Jen, parents of three children
* ***Setting*** – in the front desk at pool location
* ***Scenario Goal*** – Effectively cancel service and remove all personal and family information from all pool management systems.
* ***Scenario Narrative*** – Steve and Jen are moving and would like to close their account with this pool as they will not be visiting anymore. All personal data needs to be wiped from all systems in order to ensure membership is cancelled and privacy is honored. If they ever wish to visit this pool again, they will need to re-enter all their information again if they want to open their account.
* ***Utility Analysis***
* ***Benefits***
  + Membership is cancelled
  + Personal information is deleted from all systems
  + From the side of the application, storage space is saved after the family’s information is purged.
* ***Costs***
  + Questionnaire, waivers, and paperwork will need to be filled out every time in the future
  + No more tracking of kids’ skill through the years
  + Staff needs to be trained in order to successful handle the deletion of customer accounts from start to finish.
* ***Risks***
  + Customer may change their mind midway through cancelation
  + Make sure all personal data is erased
  + Make it as seamless as possible to prevent frustration

## Check in - Scenario I

* ***Scenario Title*** – a single visitor wants to enter the pool.
* ***Actor*** – Bob, a middle-aged man, the community resident.
* ***Setting*** – the swimming pool entrance, the front guard desk.
* ***Scenario Goal*** – check if the incoming person (Bob) is authorized to enter the pool.
* ***Scenario Narrative*** – Bob would like to swim in the open-air pool for the next hour or so. He is the community resident, paid the membership fees, and has the right to use the pool every day, on working hours. Bob would like to pass the controller’s desk without any delay. Bob does not have any documents or IDs with him; he wears only swim trunks and flip-flops.

The front guard (controller) is supposed to ask Bob’s name or address. The controller enters the name or address data into the system and selects the person’s profile from a short-list of search results. If the search results are empty, the controller checks for spelling. If the system returned too many results, the controller types-in additional qualifying info about the person (full address, name, date of birth if necessary).

Then the controller checks whether the selected profile corresponds to Bob by the photo, gender, age (date of birth if there is a doubt), address, and name. If Bob’s look matches his data in the system and the system indicates that Bob is authorized, the controller admits Bob to the pool. Otherwise, the controller doesn’t allow Bob to enter the pool. In such the case, the controller should explain the reason for denial and give instructions to Bob on how to get access to the pool if necessary.

* ***Utility Analysis***
  + *Benefits*
    - enhanced access control, unauthorized persons are not permitted even if they used to have a valid pass.
    - no need to manually produce a pass (fill in, print, cut, laminate).
  + *Costs*
    - initial setup of the database, including taking a photo of each authorized person;
    - time needed for a routine check of every incoming visitor;
    - a computer (laptop) for running the application.
  + *Risks*
    - fail to recognize a person by photo due to change in the look (e.g., a new beard);
    - delays in database synchronization (resulting in admission/non-admission errors);
    - errors in the database (e. g. misspelled address or name).

## Check in - Scenario II

* ***Scenario Title*** – a family of visitors wants to enter the pool.
* ***Actors*** – A family of four: Mike (age 37, a father), Suzan (age 32, a mother), Sam (age 9, son), Mary (age 5, daughter).
* ***Setting*** – the swimming pool entrance, the front guard desk.
* ***Scenario Goal*** – check if the incoming persons are authorized to enter the pool if there are any restrictions imposed on them
* ***Scenario Narrative*** – the family would like to swim in the open-air pool. There is no issue with authorization (each family member is authorized to use the pool). However, there are several additional restrictions imposed on children:
  + Sam is not allowed to swim alone despite being able to swim. He must be accompanied by at least one adult at all times. Only children aged at least 11 can swim alone given that they pass a swimming test;
  + Mary is not allowed to swim alone and must be accompanied by an adult too. However, due to her age (0 to 6), she must also wear a safety-vest or at least inflatable armbands.

Like in the scenario I, the family does not have any documents with them and want to pass the check-point as quickly as possible.

The front guard (controller) asks for the family’s address, enters the address into the system, and finds the profiles of Mike, Suzan, Sam, and Mary. The controller pays additional attention to the restrictions imposed on children. In this case, the guard must check whether Mary has inflatable armbands. If the baby has no armbands, the controller must pass over a safety-vest and insist on wearing it. Also, the controller checks the age of very young visitors. Babies up to 3 years old must wear swim diapers. Babies without swim diapers are not permitted to the pool.

* ***Utility Analysis***
  + *Benefits*
    - improved safety. The application minimizes the risks of child drowning;
    - improved hygiene around the pool due to the obligatory wearing of swim diapers by toddlers.
  + *Costs*
    - time needed to check every member of the group, including children;
    - a computer (laptop) for running the application;
  + *Risks*
    - fail to recognize a child (photos become outdated very fast);
    - delays in the database synchronization;
    - errors in the database.

## Review Pool Attendance

* ***Scenario Title*** – Manager and HOA board review pool attendance numbers
* ***Actor*** – Claire, Community Manager, and 5 members of the HOA board
* ***Setting*** – a meeting at the managing agents office.
* ***Scenario******Goal*** – To review the pool attendance numbers and see if any adjustments are warranted.
* ***Scenario******Narrative*** – Claire and the board members would like to review pool attendance numbers from the past year. Some of the board members feel that they are spending too much money to staff the pool during hours of little to no usage. On the other hand, the lifeguards have been complaining that there are too many people at the pool on the weekends for them to keep an eye on the water, enforce pool rules, and keep the are clean. The pool is open from 10am-8pm daily.
* ***Utility******Analysis*** –
  + ***Benefits***:
    - The Group can review attendance records by the hour and may find that there is not a sufficient number of people to justify opening the pool before noon.
    - The also realize that peak hours are all day on the weekends so they request to have the pool staffed with an extra guard.
  + ***Costs***:
    - The cost of the software.
    - Additional labor costs are possible if they have a net increase in working hours.
  + ***Risks***:
    - Pool management company may not be willing to adjust contract at the mid-point of the season.
    - A small vocal minority of residents may not appreciate the change in hours.

# 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 |  |

# User Stories

|  |  |
| --- | --- |
| User Story # | User Story |
| 1.0 | As a Pool operator (PO) I need to find a customer in the database so that I know that the customer is registered. The system displays the list of the registered customers (Search Form). |
| 1.1 | Extends 1.0. PO searches the database by typing-in a part of the user’s name, address, or a phone number into the search field. The system displays the list of top matches (Search Form). |
|  | PO can see detailed information about one customer. The system displays the form with the detailed information about the customer (Detailed Form), including a photo, full name, date of birth, allergies, emergency contact information, and swimming ability. |
|  | PO can check-in a customer by clicking the respective button either on the Search Form or on the Detailed Form. In turn, the system logs the event and adds the customer to the list of current swimmers. |
|  | Pool Administrator (PA) can add a new customer to the database of registered pool customers. The system displays the Detailed Form. PA has to fill in all required fields and the system has to validate the form before storing the new record in the database. |
|  | PA can edit all customer-related information. The system displays the Detailed Form in the edit mode and updates the record only after successful validation. |
|  | PO (or PA) can view the current swimmers in the pool. The system displays information on all swimmers who have checked-in but have not checked-out in a special form (Status Form). |
|  | PO can check-out a single customer either on Status Form, Search Form, or Detailed Form. The system logs the event and eliminates the customer from the list of current swimmers. |
|  | In the Reports section, PA (or PO) can see the list of all Visits during the specified period of time. The system displays the report (Visits Report) according to the specified Start & End dates. |
|  | In the Reports section, PA (or PO) can see the list of all Customers during the specified period of time. The system displays the report (Customers Report) according to the specified Start & End dates. |
|  | In the Reports section, PA (or PO) can see the attendance statistics for the specified period of time. The system displays the report (Attendance Report) according to the specified Start & End dates. PA has to select the statistic he is interested in (minimum, average, or maximum load; total count). The system shows the report as a table, where columns represent days of the week, and rows represent hours of the day. Also, the table has aggregates for each row and each column. |

# Product Backlog

|  |  |  |
| --- | --- | --- |
| Backlog # | User Story # | Backlog Item Description |
| 1. | 1. | Query database with search term/s according to the Search Form data filled out |
| 2. | 1. | Parse customer(s) account matching search query with photo, name, age and phone |
| 3. | 1. | Show relevant customer profile on customer selected |
| 4. | 2. | Display results from query |
| 5. | 2. | Query database for specific customer information stored in account |
| 6. | 2. | Parse query results into Detailed Form with customer’s photo, full name, date of birth, allergies, emergency contact information, and swimming ability. |
| 7. | 3. | Allow “Check-in” functionality from Search Form and Detailed Form |
| 8. | 3. | Write check-in event into system logs |
| 9. | 3. | Update customer in current swimmers list |
| 10. | 4. | “Add new” customer functionality depending on user privilege level using Detailed Form |
| 11. | 4. | Capture store customer information and store into database |
| 12. | 4. | Validate form completion and data |
| 13. | 5. | Enable Detailed Form to be editable depending on user privilege level |
| 14. | 5. | Capture new data from user |
| 15. | 5. | Update customer in database |
| 16. | 5. | Validate form completion and data |
| 17. | 6. | Query database for current swimmers form, which displays customers who are currently checked-in |
| 18. | 6. | Parse query results into Status Form |
| 19. | 6. | Give “refresh” capabilities for up-to-date results |
| 20. | 7. | Add “Check-out” functionality to Status Form, Search Form, or Detailed Form. |
| 21. | 7. | Write check-out event into system logs, tracking relevant information such as check-in time, check-out time, and number of swimmers in group |
| 22. | 7. | Delete customer from current swimmers list |
| 23. | 8 | Reports screen allows all users to build a Visits query with information such as customer or date and time range. |
| 24. | 8 | Parse Visits query into Visits Result given the customer and date and time range. |
| 25. | 9 | Current Swimmers option allows users to build a query specifying a start and end time to view swimmers in pool. |
| 26. | 9 | Parse query results into a Pool Report to view a given time range’s current swimmers number |
| 27. | 10 | Attendance option allows all users to build a query specifying a time range (start and end date/time), a selection of either minimum, average, maximum, or total (referring to current swimmers) |
| 28. | 10. | Parse query results into form Attendance Report, displaying days of week as column names, hours of day as rows, and totals. |

# Sprints

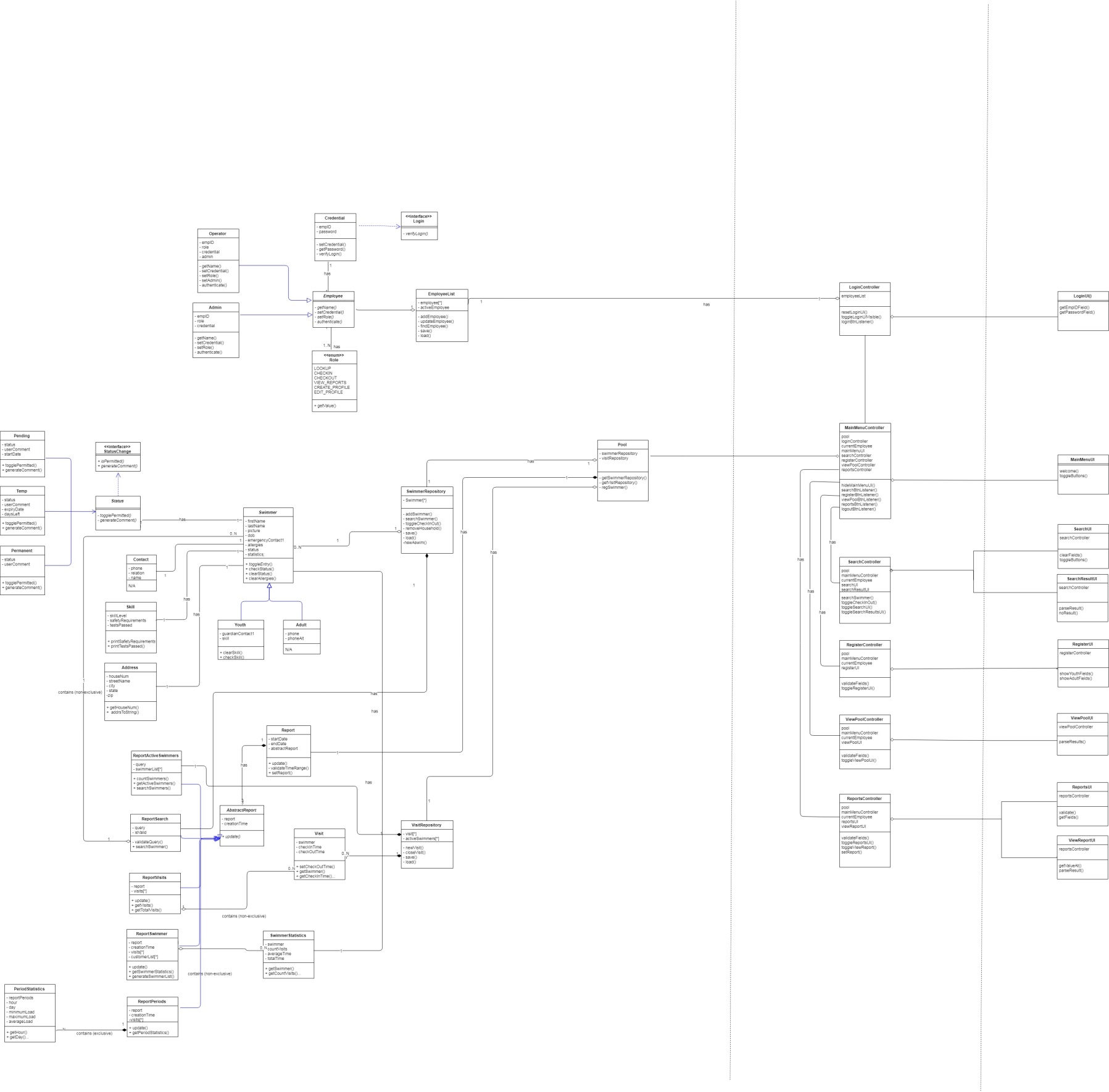
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint #1: from 2 OCT 2018 to 14 OCT 2018 | | | | | |
| Backlog # | **Task Description** | **Story Point Estimate** | **Story Point Actual** | **Status\*** | **Comments** |
| \* | Create sign-up form for username, user info, password, and access level | 2 |  |  |  |
| \* | Capture and store data from form in database | 1 |  |  |  |
| \* | Provide error checking on form submission: redundant username, invalid passwords, etc. | 1 |  |  |  |
| \* | Develop log-in panel to enter username and password, sign-up, or delete user | 1 |  |  |  |
| \* | Develop navigational panel to direct users to user case options | 1 |  |  |  |
| \* | Provide error checking on username/password entry; display error messages | 1 |  |  |  |
| \* | Develop delete user option that asks for confirmation with re-input of password | 1 |  |  |  |
| \* | Create necessary prototype files to simulate database (JSON files and customer images) | 1 |  |  |  |
| \* | Test username access and functionality | 1 |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint #2: from 15 OCT 2018 to 28 OCT 2018 | | | | | |
| Backlog # | **Task Description** | **Story Point Estimate** | **Story Point Actual** | **Status\*** | **Comments** |
| 10 | Develop input form logic, values, and design | 4 |  |  |  |
| 11 | Develop read and write functionality to JSON file | 1 |  |  |  |
| 12 | Add error checking to input form and try/catch blocks to write/read processing | 1 |  |  |  |
| 13 | Develop an option to recall the input form for existing customers | 1 |  |  |  |
| 14 | Form displays pre-existing data that can be deleted and re-entered | 1 |  |  |  |
| 15 | Override existing data with new data once update is submitted | 1 |  |  |  |
| 16 | Produce “success” message or suggest unfilled inputs or error diagnosis | 1 |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint #3: from 29 OCT 2018 to 4 NOV 2018 | | | | | |
| Backlog # | **Task Description** | **Story Point Estimate** | **Story Point Actual** | **Status\*** | **Comments** |
| 1 | Develop search criteria and algorithms | 1 |  |  |  |
| 2 | Build logic that filters customers from database and produces results according to search | 1 |  |  |  |
| 4 | Build display panel capable of showing data from search | 1 |  |  |  |
| 3 | Create functionality to click on customer in search result and open profile panel | 1 |  |  |  |
| 6 | Develop profile panel to organize and display customer info and picture | 2 |  |  |  |
| 5 | Feed in data on selected customer to profile panel | 1 |  |  |  |
| 7 | Add check-in status variable to customers and check-in buttons | 1 |  |  |  |
| 8 | Capture check-in times through running log | 1 |  |  |  |
| 9 | Create list of current swimmers with logic to add/remove given check-in status | 1 |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint #4: from 5 NOV 2018 to 18 NOV 2018 | | | | | |
| Backlog # | **Task Description** | **Story Point Estimate** | **Story Point Actual** | **Status\*** | **Comments** |
| 17 | Build swimmers’ panel to display thumbnails of checked-in customers | 2 |  |  |  |
| 18 | Use list of active swimmers to populate swimmers’ panel with data | 1 |  |  |  |
| 19 | Add ‘refresh’ functionality and add button to reload panel | 1 |  |  |  |
| 20 | Add ‘check-out’ functionality and add button to each thumbnail | 1 |  |  |  |
| 21 | Update check-in log to incorporate check-out events | 1 |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint #5: from 19 NOV 2018 to 9 DEC 2018 | | | | | |
| Backlog # | **Task Description** | **Story Point Estimate** | **Story Point Actual** | **Status\*** | **Comments** |
| 22 | Develop reports query panel to display different query option: Visits | 2 |  |  |  |
| 23 | Develop logic to parse Visits query into Visits Result given the customer and date and time range. | 1 |  |  |  |
| 24 | Build panel to accept information about Pool report (start and end time) | 2 |  |  |  |
| 25 | Develop logic to parse query results into a Pool Report to view a given time range’s current swimmers number | 1 |  |  |  |
| 26 | Build panel to construct attendance report. User should be able to enter a time range (start and end date/time), a selection of either minimum, average, maximum, or total (referring to current swimmers) | 2 |  |  |  |
| 27 | Develop logic to parse query results into Attendance Report, displaying days of week as column names, hours of day as rows, and totals. | 1 |  |  |  |
| \* | Test all queries and results for accurate numbers in all reports. | 1 |  |  |  |

Class diagram: specification

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. |

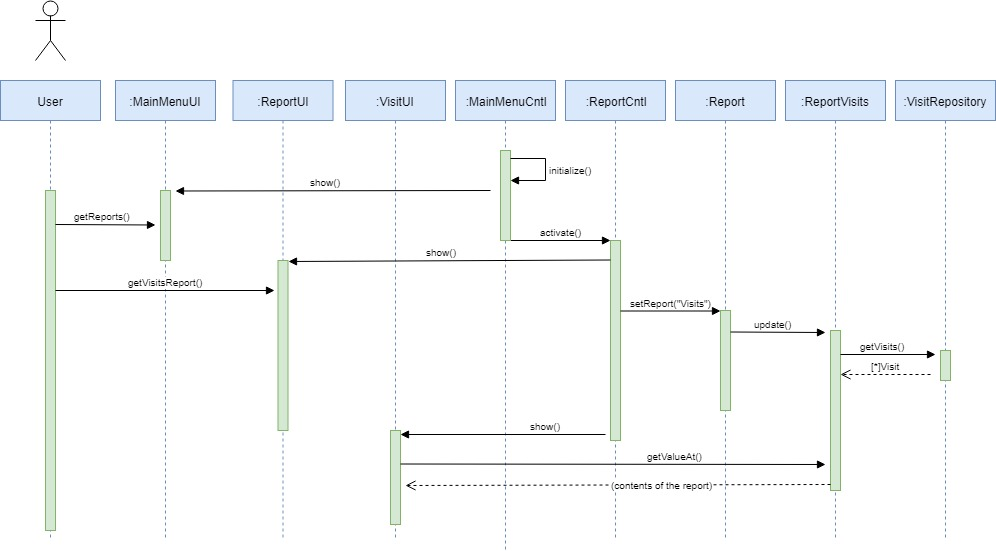
**View Class List**

|  |  |
| --- | --- |
| 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 |

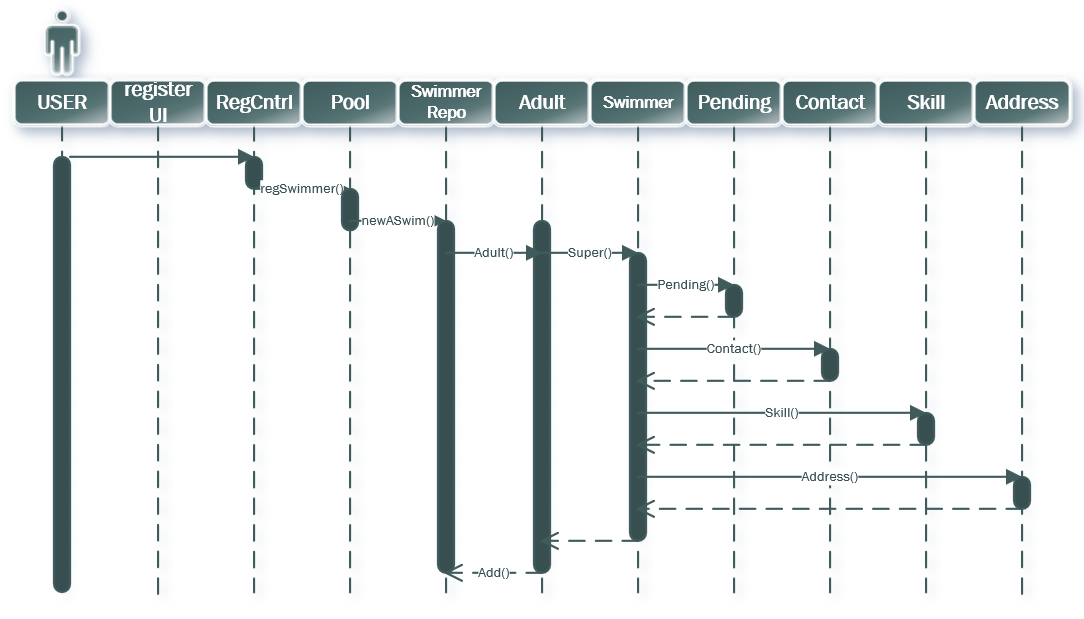
**Model Class List**

|  |  |
| --- | --- |
| 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. |

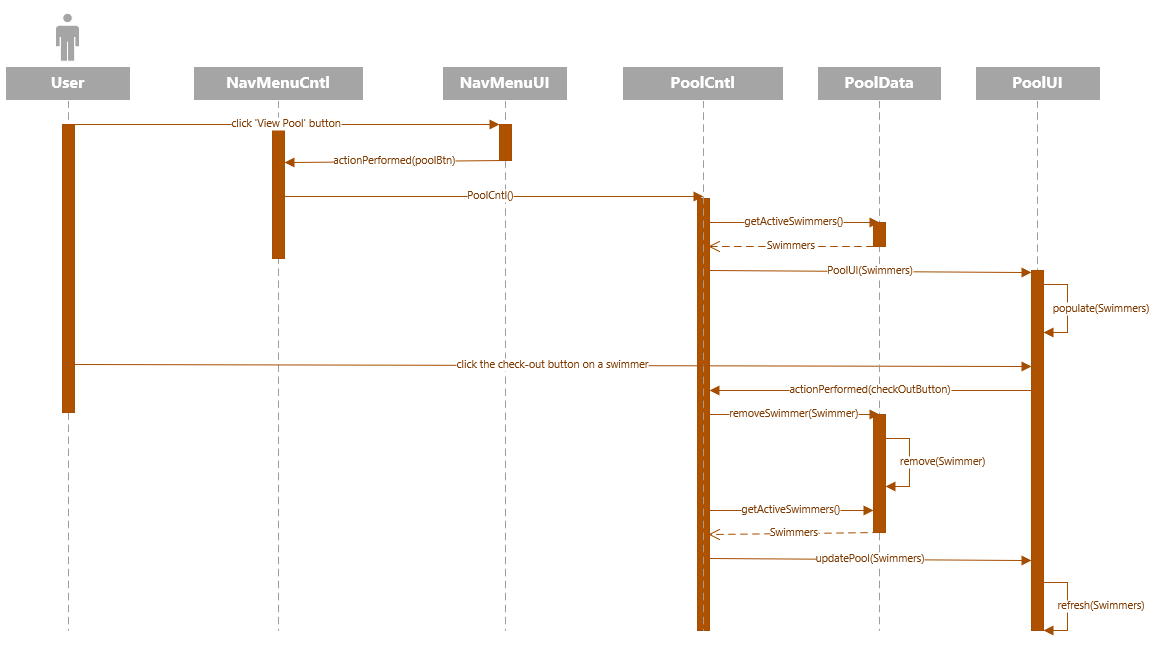
Sequence diagrams



View Report

Register Swimmer

Look-up (Search) Swimmer

View Pool