VOS TECHNOLOGY

# Fit-U-In System Requirements


Clients:

Rob McGarry

Shawn Trickett

Team Members:

Zachary Lazzara

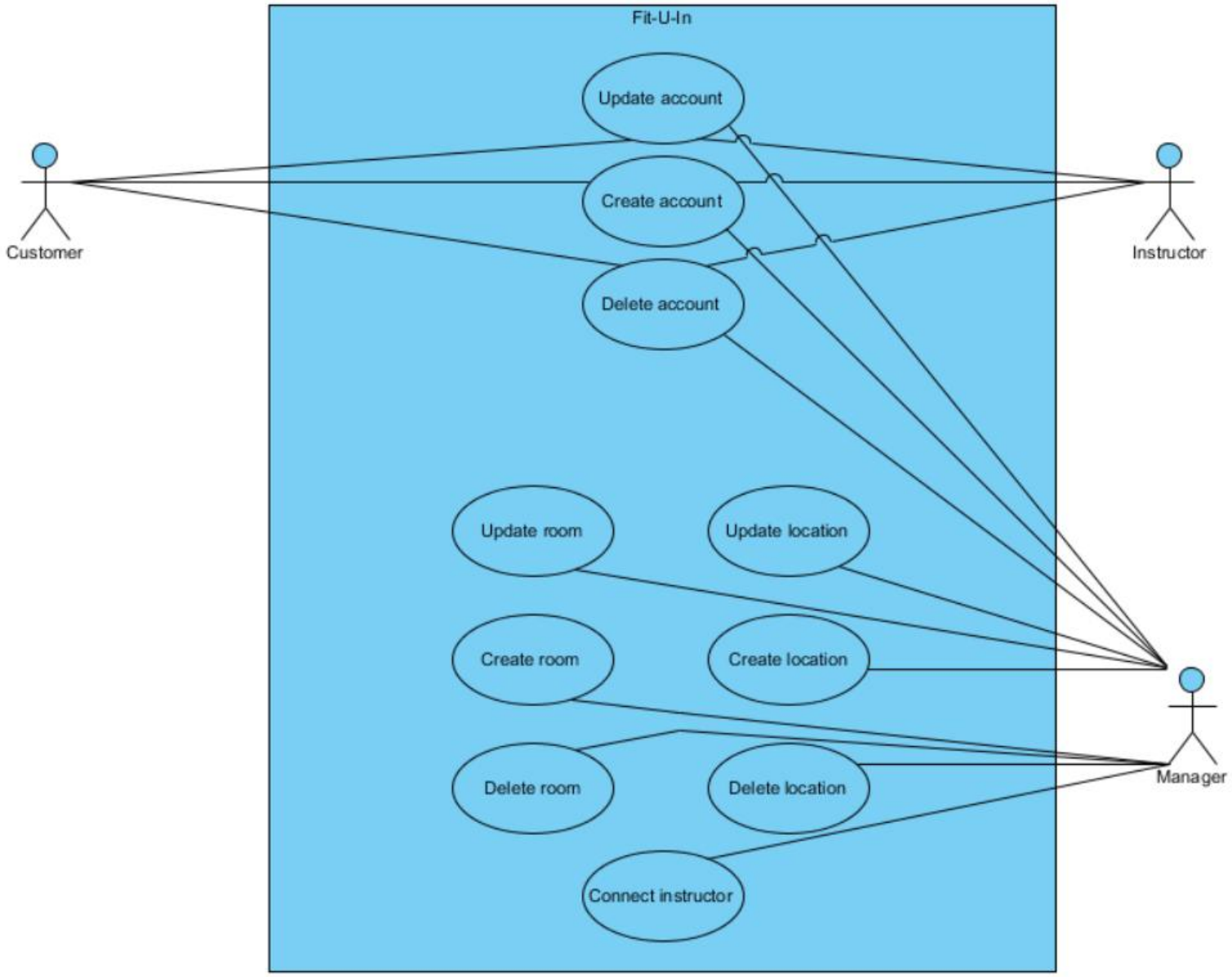
Ryan McCallum

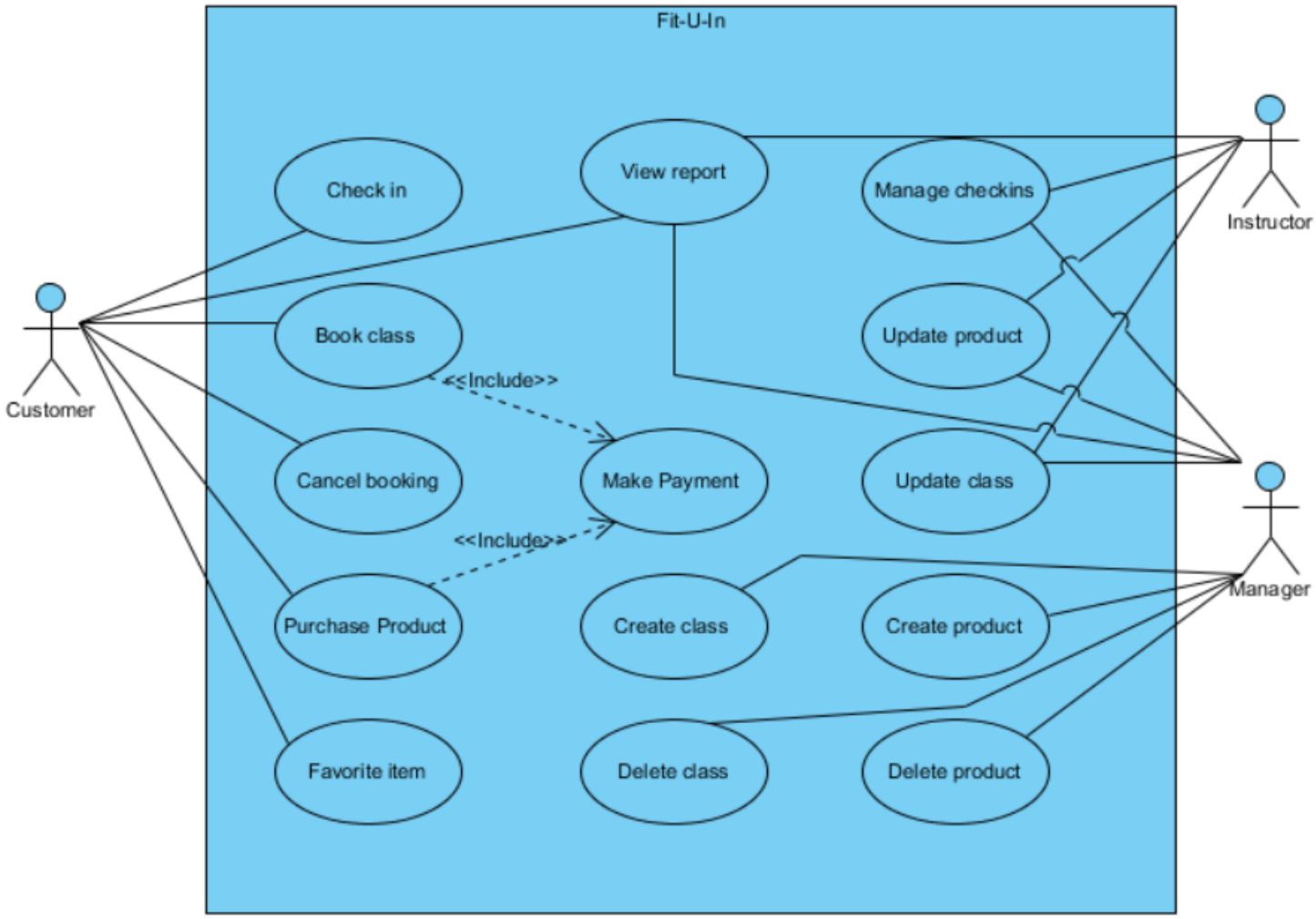
Kevin Rice

Tyler Templeton

# Table of Contents

## Use Case Diagrams






## Use Case Scenarios

| | |
|---|---|
| Use Case name: | Update account |
| Scenario: | Update account details |
| Triggering event: | User wants to update account |
| Brief description: | The customer and instructor can update their profile and the manager can update the studio profile |
| Actors: | Customer, instructor and manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager, customer, instructor and Fit-U-In |
| Preconditions: | Account exists |
| Postconditions: | Account details updated |
| Flow of events: | 1. Navigates to profile<br>2. SYSTEM Displays profile information<br>3. Update and save information<br>4. SYSTEM Save changed information to the database and displays profile with the changes |

| Use Case name: | Create account |
|---|---|
| Scenario: | Create a new account |
| Triggering event: | User wants to create an account |
| Brief description: | The customer and instructor can create a personal account and the manager can create the studio account |
| Actors: | Customer, instructor and manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager, customer, instructor and Fit-U-In |
| Preconditions: | Account does not exist |
| Postconditions: | Account created |
| Flow of events: | 1. Select create a new account<br>2. Fill out account details and save<br>2.a.<br>1. if<br>1.1. Retype password or email do not match original<br>1.2. SYSTEM Prompt the user that the email or passwords do not match<br>1.3. Update the password or email filed<br>end if<br>2.b.<br>1. if Important field left blank<br>1.1. SYSTEM Prompt the user to update the missing field<br>1.2. User updates missing field<br>end if<br>3. SYSTEM Creates the account in the database and informs the user<br>4. SYSTEM Moves to the home screen |

| Use Case name: | Delete account |
|---|---|
| Scenario: | Delete an account |
| Triggering event: | User wants to delete an account |
| Brief description: | The customer and instructor can delete their personal account and the manager can delete the studio account |
| Actors: | Customer, instructor and manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager, customer, instructor and Fit-U-In |
| Preconditions: | Account exist |
| Postconditions: | Account Deleted |
| Flow of events: | 1. Navigate to profile<br>2. Select delete account option<br>3. SYSTEM Prompt user to re-enter password<br>4. Re-enters password<br>4.a.<br>1. if The entered password is not correct<br>1.1. SYSTEM Prompts the user to re-enter the password<br>1.2. Re-enters the password |

| | |
|---|---|
| | end if<br>5. SYSTEM The users account is deleted from the database |

| | |
|---|---|
| Use Case name: | Update room |
| Scenario: | Update a room details |
| Triggering event: | User wants update a room's details |
| Brief description: | The manager updates the room's details |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Room exists |
| Postconditions: | Room's details updated |
| Flow of events: | 1. Navigate to rooms<br>2. SYSTEM Display rooms<br>3. Select room from list<br>4. SYSTEM Display room details<br>5. Select update room<br>6. Update details and save<br>7. SYSTEM Update database<br>8. SYSTEM Display room details with changes |

| | |
|---|---|
| Use Case name: | Create room |
| Scenario: | Create a new room |
| Triggering event: | User wants up create a new room |
| Brief description: | The manager creates a new room to host classes |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Room does not exist |
| Postconditions: | Room is created |
| Flow of events: | 1. Navigates to rooms<br>2. Selects add room<br>3. Inputs and saves room details<br>3.a.<br>1. if Important field left blank<br>1.1. SYSTEM Prompt the user to update the missing field<br>1.2. User updates missing field<br>end if<br>4. SYSTEM Save room to database<br>5. SYSTEM Show the new room profile |

| | |
|---|---|
| Use Case name: | Delete room |
| Scenario: | Delete a room |
| Triggering event: | User wants to delete a room |

| | |
|---|---|
| Brief description: | The manager deletes a room |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Room exists |
| Postconditions: | Room deleted from database |
| Flow of events: | 1. Navigate to rooms<br>2. SYSTEM Display rooms<br>3. Select room<br>4. SYSTEM Display room details<br>5. Select delete room<br>6. Confirm delete room<br>6.a.<br>1. if Yes<br>1.1. SYSTEM Delete room from database<br>1.2. SYSTEM Display deletion confirmation<br>end if |

| | |
|---|---|
| Use Case name: | Update location |
| Scenario: | Update a location details |
| Triggering event: | User wants update a location's details |
| Brief description: | The manager updates the location's details |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Location exists |
| Postconditions: | Location's details updated |
| Flow of events: | 1. Navigate to locations<br>2. SYSTEM Display locations<br>3. Select location from list<br>4. SYSTEM Display location details<br>5. Select update location<br>6. Update details and save<br>7. SYSTEM Update database<br>8. SYSTEM Display location details with changes |

| | |
|---|---|
| Use Case name: | Create location |
| Scenario: | Create a new location |
| Triggering event: | User wants up create a new location |
| Brief description: | The manager creates a new location to contain rooms |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Location does not exist |

| | |
|---|---|
| Postconditions: | Location is created |
| Flow of events: | 1. Navigates to locations<br>2. Selects add location<br>3. Inputs and saves location details<br>3.a.<br>1. if Important field left blank<br>1.1. SYSTEM Prompt the user to update the missing field<br>1.2. User updates missing field<br>end if<br>4. SYSTEM Save location to database<br>5. SYSTEM Show the new locations details |

| | |
|---|---|
| Use Case name: | Delete location |
| Scenario: | Delete a location |
| Triggering event: | User wants to delete a location |
| Brief description: | The manager deletes a location |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Location exists |
| Postconditions: | Location deleted from database |
| Flow of events: | 1. Navigate to locations<br>2. SYSTEM Display locations<br>3. Select location<br>4. SYSTEM Display location details<br>5. Select delete location<br>6. Confirm delete location<br>6.a.<br>1. if Yes<br>1.1. SYSTEM Delete location from database<br>1.2. SYSTEM Display deletion confirmation<br>end if |

| | |
|---|---|
| Use Case name: | Connect instructor |
| Scenario: | Connects the studio and an instructor |
| Triggering event: | User wants to connect the studio and an instructor |
| Brief description: | The manager finds the instructor by searching for their user email/email or with a direct link provide by the instructor. Then connects the accounts |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, instructor, manager and Fit-U-In |
| Preconditions: | Studio and instructor must exist |
| Postconditions: | Studio and instructor are connected |
| Flow of events: | 1. Find instructor via |

| | |
|---|---|
| | 1.1. Email search or Name search<br>1.1.a.<br>1. Input email or name<br>2. if There is an instructor with matching email or name<br>2.1. SYSTEM Display matching instructor<br>3. else<br>3.1. SYSTEM Prompt that there are no matching instructors<br>end if<br>1.2. Direct link<br>2. SYSTEM Display instructor<br>3. Connect the instructor<br>4. SYSTEM Create link in database |

| | |
|---|---|
| Use Case name: | Check in |
| Scenario: | Check into a class |
| Triggering event: | User checks into a class |
| Brief description: | The Customer is prompted by geolocation/time or chooses to check in manually. They then select the class and check in. |
| Actors: | Customer |
| Related Use Cases: | |
| Stakeholders: | Studio, customer and Fit-U-In |
| Preconditions: | Customer must be booked in a class |
| Postconditions: | Customer is checked into the class |
| Flow of events: | 1. User is prompted to check in<br>1.1. SYSTEM Geolocation<br>1.1.a.<br>1. if User is near the class location and the time is near the class start time<br>1.1. SYSTEM User will be promoted to check in<br>end if<br>1.2. User decides to check in without being promoted<br>2. SYSTEM The logical check in class will be displayed with the option to see other classes they have booked<br>2.a.<br>1. if user choose to view other classes<br>1.1. SYSTEM A list of other classes will be displayed<br>1.2. Select the choose class<br>end if<br>3. User asked to check in<br>3.a.<br>1. if user checks in<br>1.1. SYSTEM User is changed to checked in on the class database<br>end if |

| | |
|---|---|
| Use Case name: | Book class |

| Scenario: | Book a class |
|---|---|
| Triggering event: | User wants to book a class |
| Brief description: | The user wants to book a class, they have to select the class and then pay with a pass or payment. |
| Actors: | Customer |
| Related Use Cases: | Make payment |
| Stakeholders: | Studio, customer and Fit-U-In |
| Preconditions: | User and class exists |
| Postconditions: | User is booked into a class |
| Flow of events: | 1. Navigate to class from class search or favorites list<br>1.1. Class search<br>1.1.1. Search by criteria to find a class<br>1.1.2. SYSTEM Display class details<br>1.2. Favorites list<br>1.2.1. Select the class from the list of favorites classes<br>1.2.2. SYSTEM Display class details<br>2. Select pass or payment<br>2.a.<br>1. if Pass<br>1.1. SYSTEM Display list of passes owned by the user that can be used to pay for the class<br>1.2. Select a pass and confirm<br>1.3. SYSTEM Pass count reduced by 1<br>1.4. if Pass count reduced to 0<br>1.4.1. SYSTEM pass delete from the users account<br>end if<br>2. else if payment<br>2.1. SYSTEM User will be promoted to pay<br>2.2. if Payment not successfully<br>2.2.1. SYSTEM User promoted to contact instructor or manager to inquire about alternative payments<br>end if<br>end if<br>3. SYSTEM User added to the class list as booked<br>4. SYSTEM Display new class details |

| Use Case name: | Cancel booking |
|---|---|
| Scenario: | Cancel a booking |
| Triggering event: | User wants to cancel a booking |
| Brief description: | Customer wants to cancel a booking, they must select which, they will be shown if they will get a refund and will have to confirm |
| Actors: | Customer |
| Related Use Cases: | |
| Stakeholders: | Studio, customer and Fit-U-In |
| Preconditions: | User is booked into a class |
| Postconditions: | User status is changed to dropped |

| | |
|---|---|
| Flow of events: | 1. Navigates to the booked class<br>2. Selects cancel booking<br>3. SYSTEM determine if class is refundable<br>3.a.<br>1. if yes<br>1.1. SYSTEM Promoted the user if they are sure they wish to cancel and that they will be refunded<br>2. else<br>2.1. SYSTEM Promoted the user if they are sure they wish to cancel and that they will not be refunded<br>end if<br>4. SYSTEM Wait for user response if they wish to cancel<br>4.a.<br>1. if Yes<br>1.1. Users status is changed to dropped<br>1.2. if refundable<br>1.2.1. SYSTEM An appropriate pass will be added to the users account<br>end if<br>end if |

| | |
|---|---|
| Use Case name: | Purchase product |
| Scenario: | Purchase a product |
| Triggering event: | User want to purchase a product |
| Brief description: | Customer wants a product, they will have to select the product, add it to a cart, then pay for the cart. A product can be any consumable, service or pass the studio wants to sell |
| Actors: | Customer |
| Related Use Cases: | Make payment |
| Stakeholders: | Studio, customer and Fit-U-In |
| Preconditions: | Product and user exist |
| Postconditions: | The product will be added to the users account, if applicable a note will be added to the appropriate class list |
| Flow of events: | 1. while User not in cart<br>1.1. Navigate to a product<br>1.1.1. Search for product<br>1.1.2. See product on quick by<br>1.1.3. See product on favorites<br>1.2. SYSTEM Display product details<br>1.3. Choose to purchase products<br>1.3.a.<br>1. if User chooses to purchase<br>1.1. SYSTEM added the item to the cart<br>1.2. SYSTEM Prompt the user if they wish to keep shopping or go to cart<br>end if<br>end while<br>2. Navigate to cart |

| | |
|---|---|
| | 3. SYSTEM Display cart details<br>4. SYSTEM User given the option to check out or change their order<br>4.a.<br>1. if Changes the order<br>1.1. SYSTEM the page will be refreshed with the changes<br>2. else if User checks out<br>2.1. SYSTEM User will be promoted to pay with PayPal<br>2.2. if Payment successfully<br>2.2.1. SYSTEM Products will be added to the users account, if the products are related to a specific class that will be noted on the class list<br>2.2.2. SYSTEM Receipt will be generating, added to the customer reports and shown to the user<br>2.3. else if Payment failed<br>2.3.1. SYSTEM User told payment failed and returned to the cart screen<br>end if<br>end if |

| | |
|---|---|
| Use Case name: | Favorite item |
| Scenario: | Favourite an item |
| Triggering event: | User wants to favorite an item |
| Brief description: | If the customer finds an item that they want to favourite, they select add to favorites in the item details. An item can be almost anything, location, studio, class, product etc.. |
| Actors: | Customer |
| Related Use Cases: | |
| Stakeholders: | Studio, customer and Fit-U-In |
| Preconditions: | Item and customer exist |
| Postconditions: | Items added to favorites list |
| Flow of events: | 1. Navigate to an item<br>2. SYSTEM Display item details<br>3. Select add item to favorites<br>4. SYSTEM Item added to the user's favorites list<br>5. SYSTEM Display the corresponding favorites list |

| | |
|---|---|
| Use Case name: | View report |
| Scenario: | View a report |
| Triggering event: | User wants to view a report |
| Brief description: | The customer, manager or instructor wish to view a report, they select a report from the list appropriate to their account type and it gets generated |
| Actors: | Customer, manager and Instructor |
| Related Use Cases: | |
| Stakeholders: | Studio, manager, instructor, Customer and Fit-U-In |
| Preconditions: | Customer, manager or instructor exist |
| Postconditions: | Report is displayed |
| Flow of events: | 1. Navigate to reports<br>2. SYSTEM Display reports page with search, categories and common reports |

| | |
|---|---|
| | 3. Select a report<br>4. SYSTEM Gather details for, create and display report |

| | |
|---|---|
| Use Case name: | Make payment |
| Scenario: | Make a payment |
| Triggering event: | User clicks pay with PayPal |
| Brief description: | The system will pass the user to the PayPal API then respond according to the results |
| Actors: | Customer |
| Related Use Cases: | |
| Stakeholders: | Studio, PayPal, Customer and Fit-U-In |
| Preconditions: | Payment requested |
| Postconditions: | Send a message back on the payments success |
| Flow of events: | 1. SYSTEM Call PayPal API with customer details<br>2. SYSTEM Wait for response<br>2.a.<br>1. if Success response<br>1.1. SYSTEM Report success<br>2. else if Failure response<br>2.1. SYSTEM Report failure<br>3. else if PayPal error<br>3.1. SYSTEM Report PayPal error<br>end if |

| | |
|---|---|
| Use Case name: | Manual check-in |
| Scenario: | Manual check-in management |
| Triggering event: | User wants to change a customers check in status manually |
| Brief description: | The manager or instructor can change which users have booked, checked in or dropped a classes manually |
| Actors: | Manager and instructor |
| Related Use Cases: | |
| Stakeholders: | Studio, manager, instructor, Customer and Fit-U-In |
| Preconditions: | User and class exists |
| Postconditions: | Appropriate change has taken place |
| Flow of events: | 1. Navigate to classes<br>2. SYSTEM Display list of classes in logical order<br>3. Select the class<br>4. SYSTEM Class displayed with booked customers and their status listed<br>5. Choose action<br>5.1. Manual booking<br>5.1.a.<br>1. Search for a user via name or email<br>2. Book the user to the class<br>3. SYSTEM Add the user to the class list database as booked<br>5.2. Manual drop |

| | |
|---|---|
| | 5.2.a.<br>1. Select user<br>2. Select drop from class<br>3. SYSTEM check if class will be refunded<br>4. if Yes<br>4.1. SYSTEM An appropriate pass will be added to the users account<br>end if<br>5. SYSTEM Change user's status to dropped<br>5.3. Manual check<br>5.3.a.<br>1. Select user<br>2. Change status to checked in<br>3. SYSTEM User status is updated to checked in<br>6. SYSTEM Page updates to reflect changes |

| | |
|---|---|
| Use Case name: | Update product |
| Scenario: | Update a product |
| Triggering event: | User wants update a product |
| Brief description: | The manager or instructor updates the product's details |
| Actors: | Manager and instructor |
| Related Use Cases: | |
| Stakeholders: | Studio, manager, instructor and Fit-U-In |
| Preconditions: | Product exists |
| Postconditions: | Product's details updated |
| Flow of events: | 1. Navigate to products<br>2. SYSTEM Display products<br>3. Select product from list<br>4. SYSTEM Display product details<br>5. Select update product<br>6. Update details and save<br>7. SYSTEM Update database<br>8. SYSTEM Display product details with changes |

| | |
|---|---|
| Use Case name: | Create product |
| Scenario: | Create a new product |
| Triggering event: | User wants up create a new product |
| Brief description: | The manager creates a new product |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Product does not exist |
| Postconditions: | Product is created |
| Flow of events: | 1. Navigates to products<br>2. Selects add product |

| | |
|---|---|
| | 3. Inputs and saves location details<br>3.a.<br>1. if Important field left blank<br>1.1. SYSTEM Prompt the user to update the missing field<br>1.2. User updates missing field<br>end if<br>4. SYSTEM Save product to database<br>5. SYSTEM Show the new products details |

| | |
|---|---|
| Use Case name: | Delete product |
| Scenario: | Delete a product |
| Triggering event: | User wants to delete a product |
| Brief description: | The manager deletes a product |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Product exists |
| Postconditions: | Product deleted from database |
| Flow of events: | 1. Navigate to products<br>2. SYSTEM Display products<br>3. Select product<br>4. SYSTEM Display product details<br>5. Select delete product<br>6. Confirm delete product<br>6.a.<br>1. if Yes<br>1.1. SYSTEM Delete product from database<br>1.2. SYSTEM Display deletion confirmation<br>end if |

| | |
|---|---|
| Use Case name: | Update class |
| Scenario: | Update a class |
| Triggering event: | User wants update a class |
| Brief description: | The manager or instructor updates the class' details |
| Actors: | Manager and instructor |
| Related Use Cases: | |
| Stakeholders: | Studio, manager, instructor and Fit-U-In |
| Preconditions: | Class exists |
| Postconditions: | Class's details updated |
| Flow of events: | 1. Navigate to classes<br>2. SYSTEM Display classes<br>3. Select class from list<br>4. SYSTEM Display class details<br>5. Select update class |

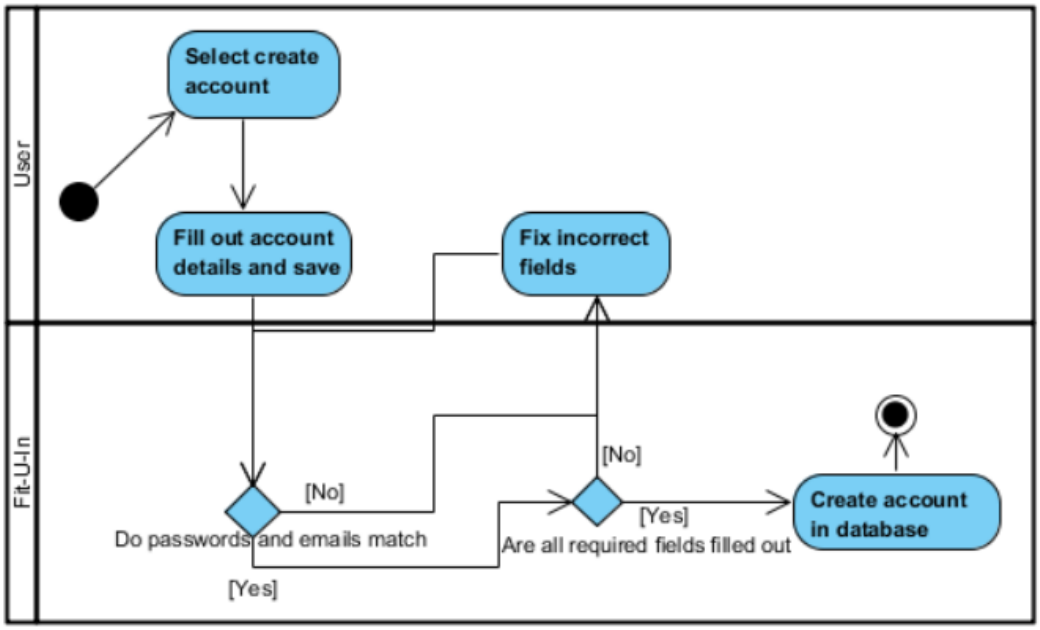
| | |
|---|---|
| | 6. Update details and save<br>7. SYSTEM Update database<br>8. SYSTEM Display class details with changes |

| | |
|---|---|
| Use Case name: | Create class |
| Scenario: | Create a new class |
| Triggering event: | User wants up create a new class |
| Brief description: | The manager creates a new class |
| Actors: | Manager |
| Related use case: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Class does not exist |
| Postconditions: | Class is created |
| Flow of events: | 1. Navigates to classes<br>2. Selects add class<br>3. Inputs and saves class details<br>3.a.<br>1. if Important field left blank<br>1.1. SYSTEM Prompt the user to update the missing field<br>1.2. User updates missing field<br>end if<br>4. SYSTEM Save class to database<br>5. SYSTEM Show the new class' details |

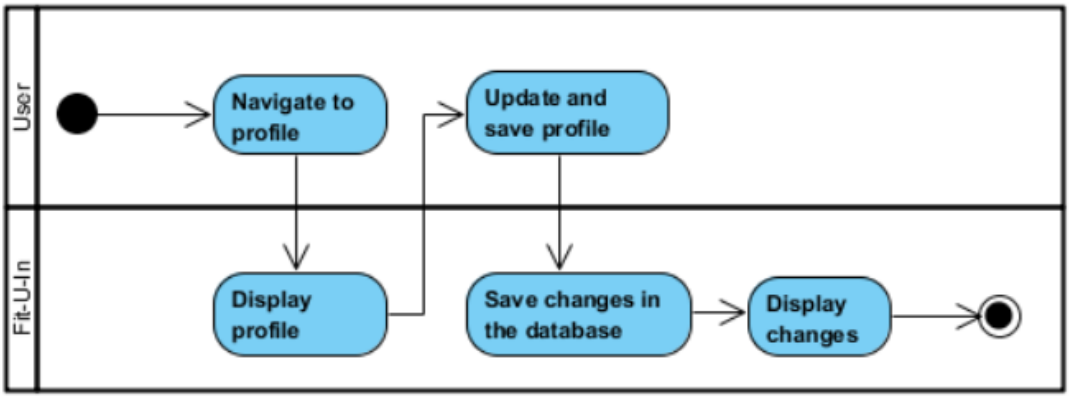
| | |
|---|---|
| Use Case name: | Delete class |
| Scenario: | Delete a class |
| Triggering event: | User wants to delete a class |
| Brief description: | The manager deletes a class |
| Actors: | Manager |
| Related Use Cases: | |
| Stakeholders: | Studio, manager and Fit-U-In |
| Preconditions: | Class exists |
| Postconditions: | Class deleted from database |
| Flow of events: | 1. Navigate to classes<br>2. SYSTEM Display classes<br>3. Select class<br>4. SYSTEM Display class details<br>5. Select delete product<br>6. Confirm delete product<br>6.a.<br>1. if Yes<br>1.1. SYSTEM Delete class from database<br>1.2. SYSTEM Display deletion confirmation<br>end if |

# Activity Diagrams

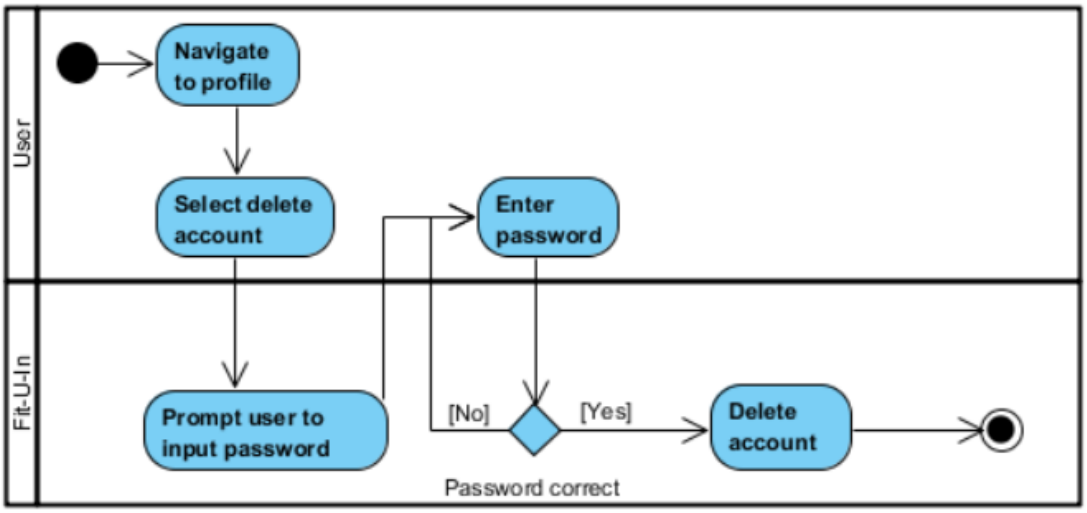
## Create account




## Update account

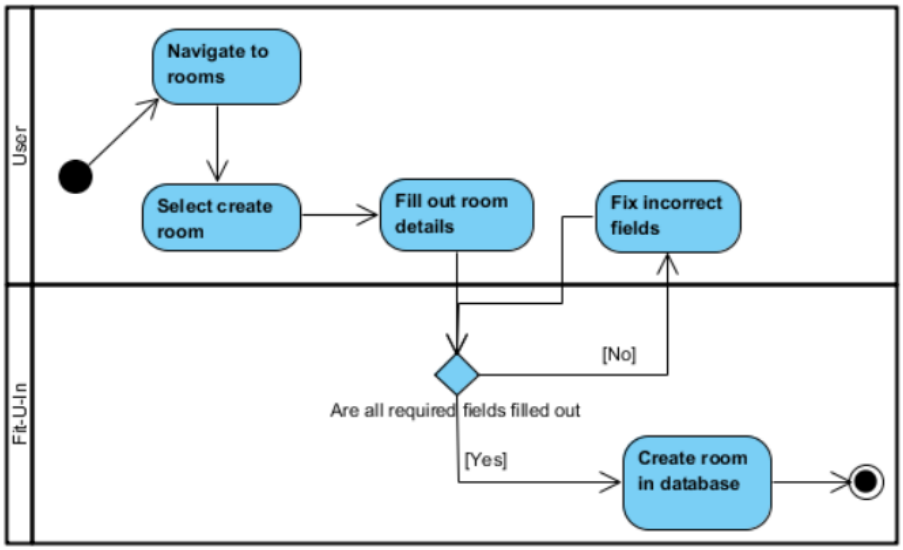


## Delete account

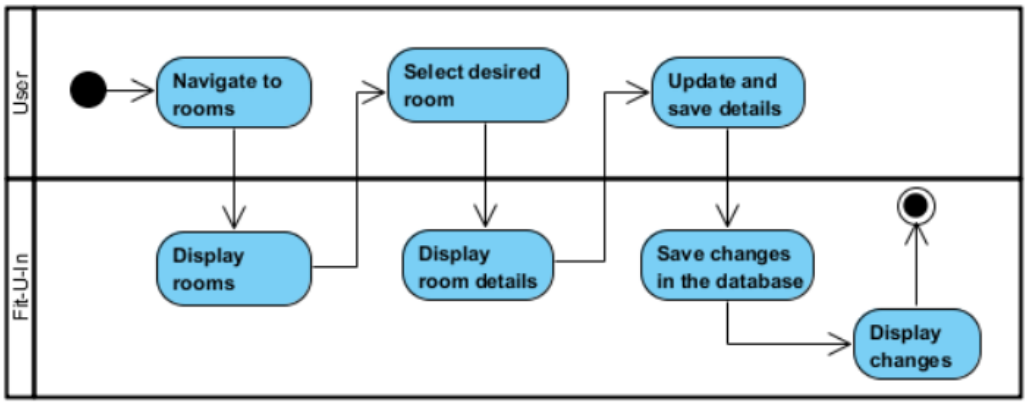



The variations of create, update and delete for rooms, locations, products and classes all follow the same process. So only the rooms variation ill be show

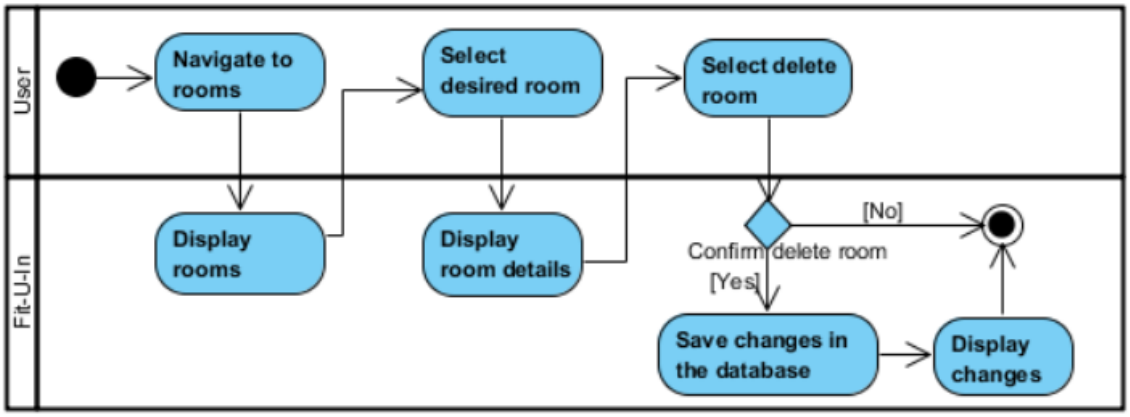
## Create room



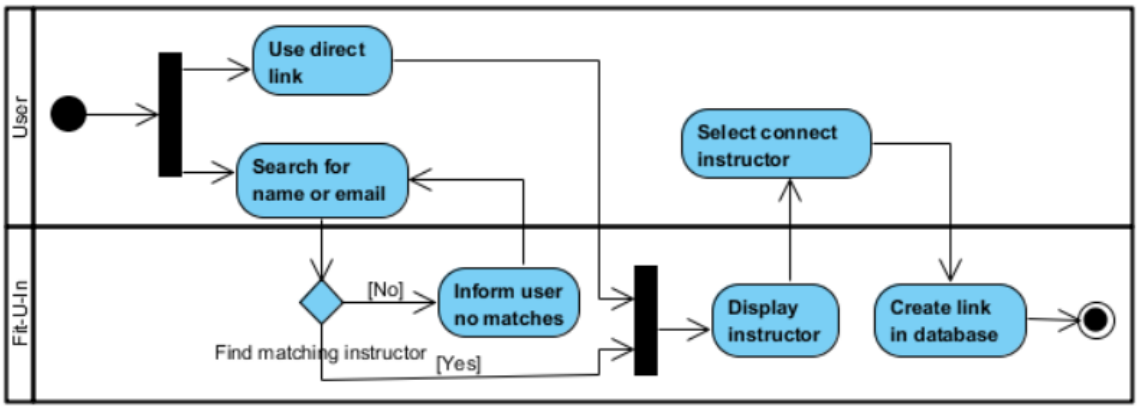
## Update room



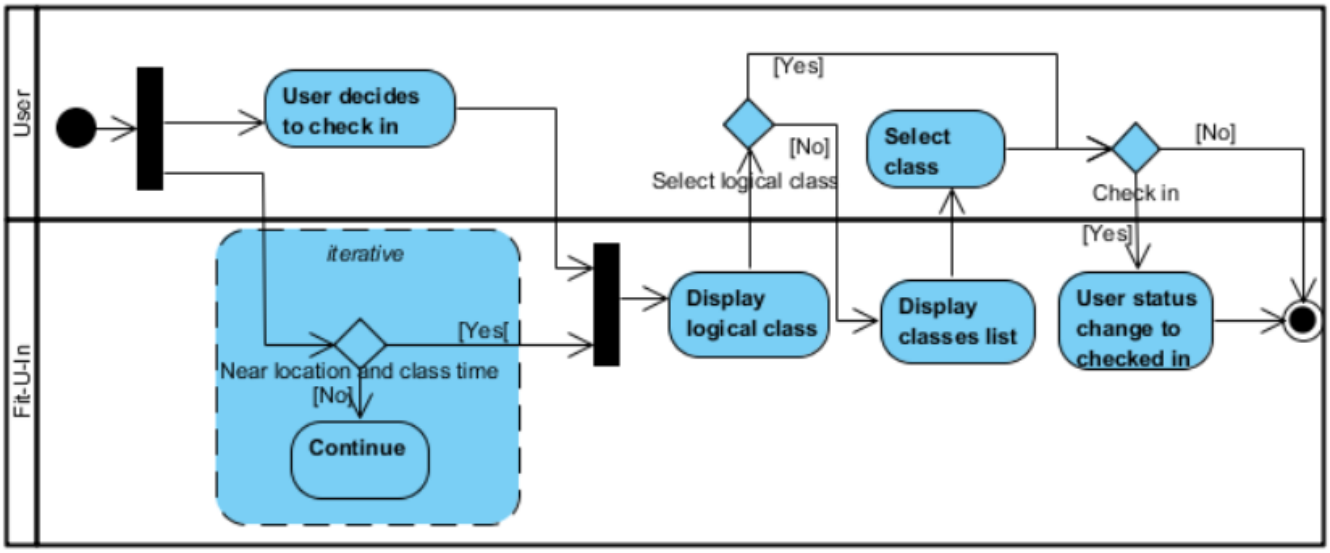

## Delete room



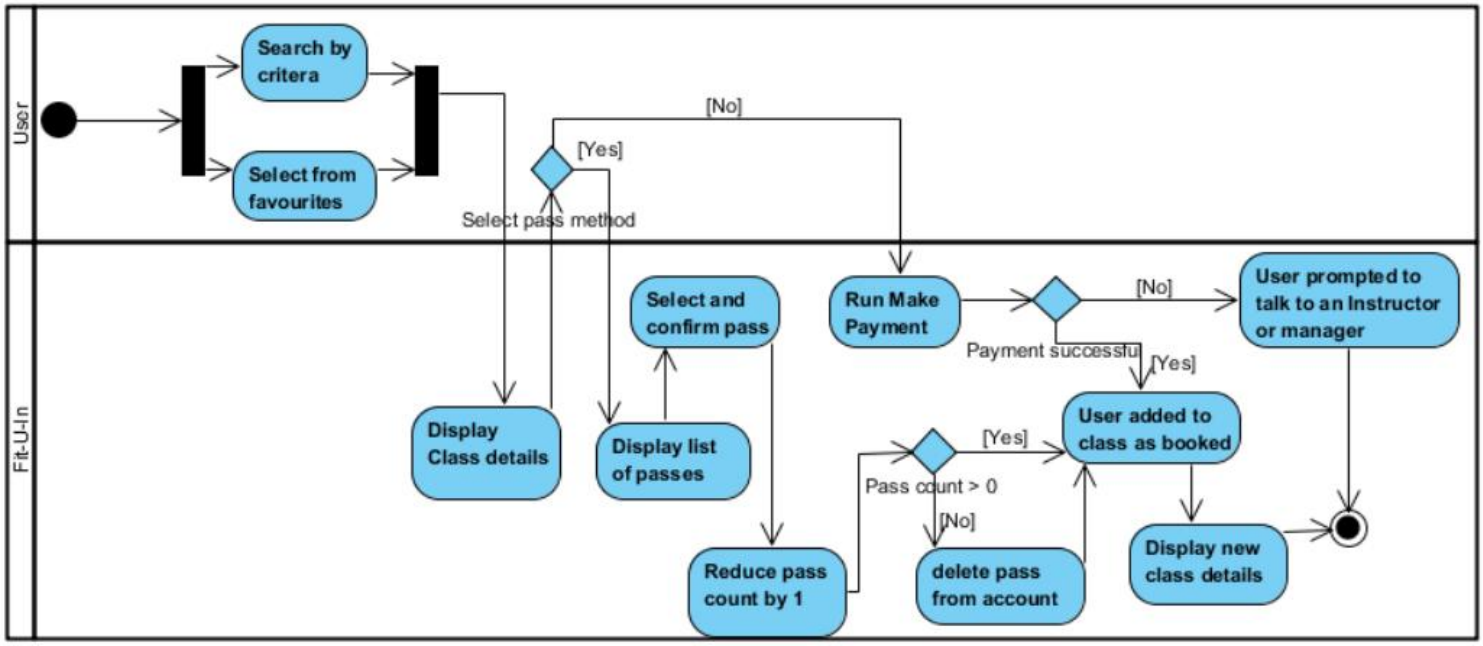

## Connect instructor



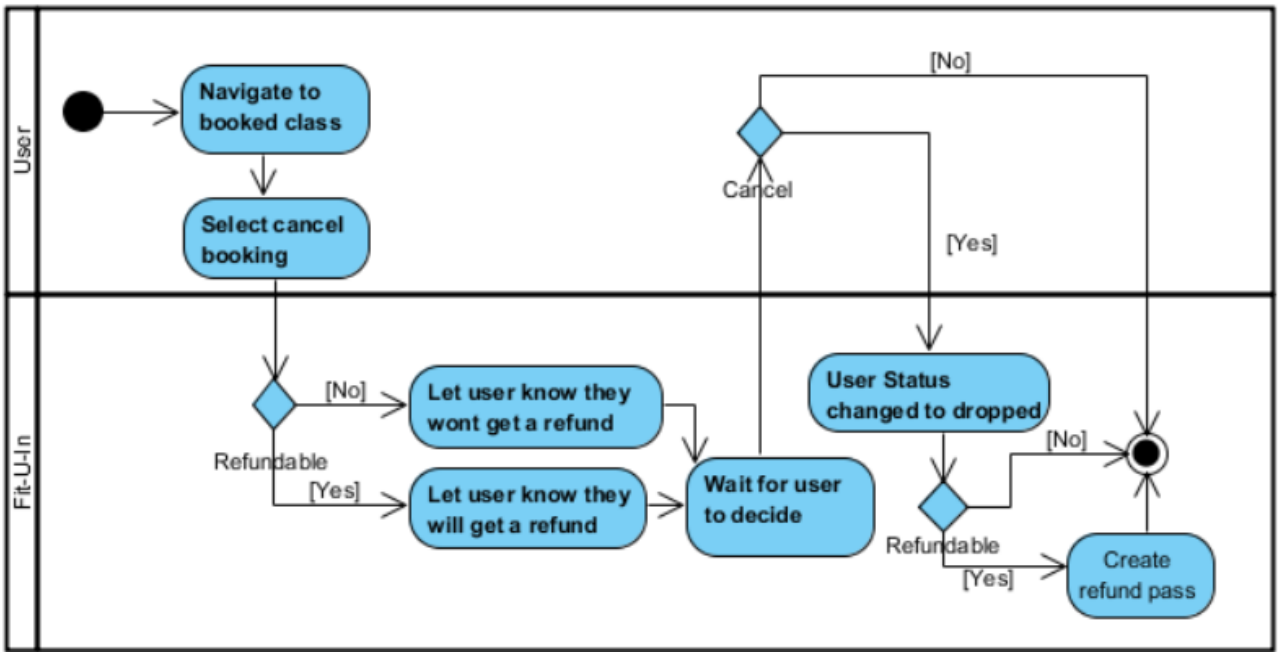

## Check in


User
Fit-U-In
User decides to check in
iterative
Near location and class time
[Yes]
[No]
Continue
Select logical class
[Yes]
[No]
Display logical class
Select class
Display classes list
Check in
[No]
[Yes]
User status change to checked in

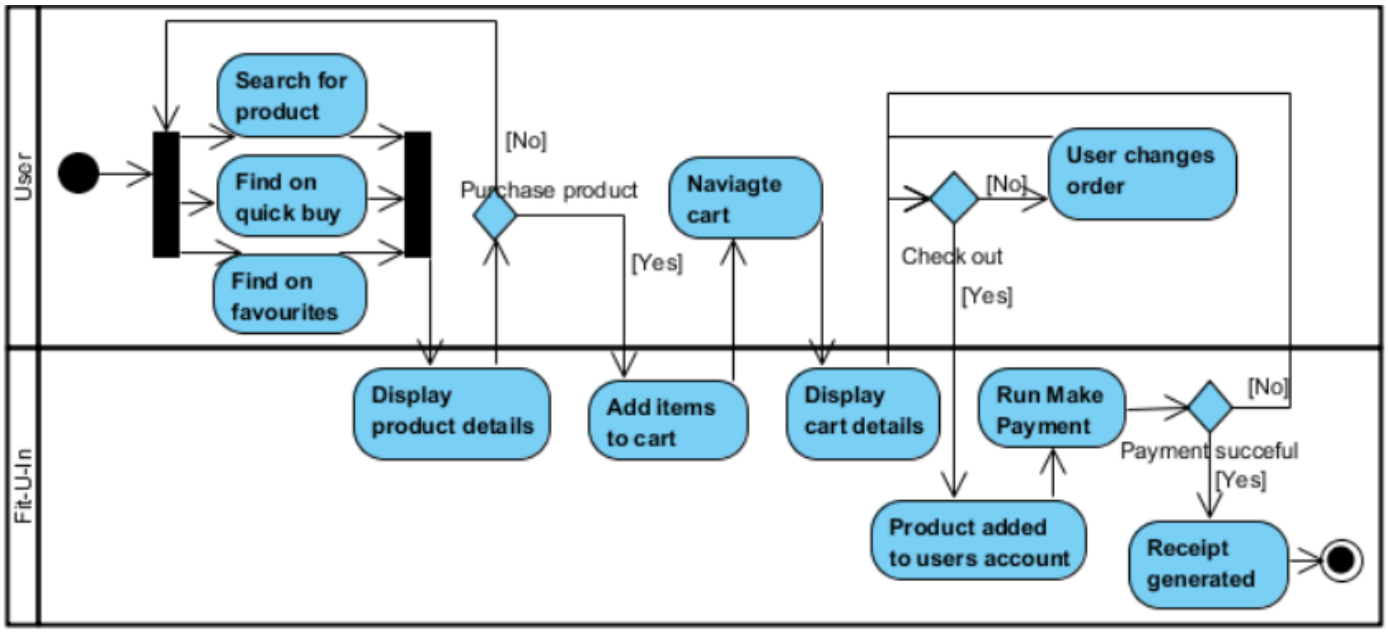
## Book class


User
Fit-U-In
Search by critera
Select from favourites
Select pass method
[Yes]
[No]
Display Class details
Display list of passes
Select and confirm pass
Reduce pass count by 1
Pass count > 0
[Yes]
[No]
delete pass from account
Run Make Payment
Payment successfu
[No]
[Yes]
User prompted to talk to an Instructor or manager
User added to class as booked
Display new class details

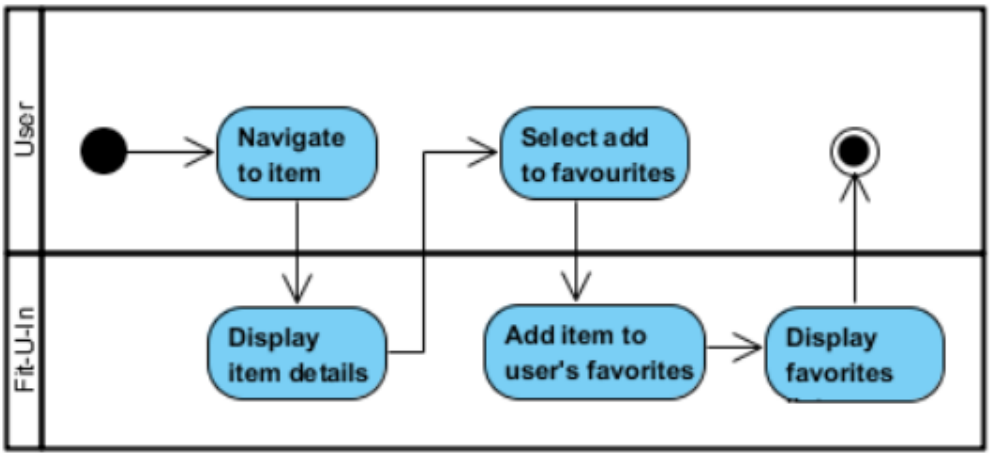
## Cancel booking



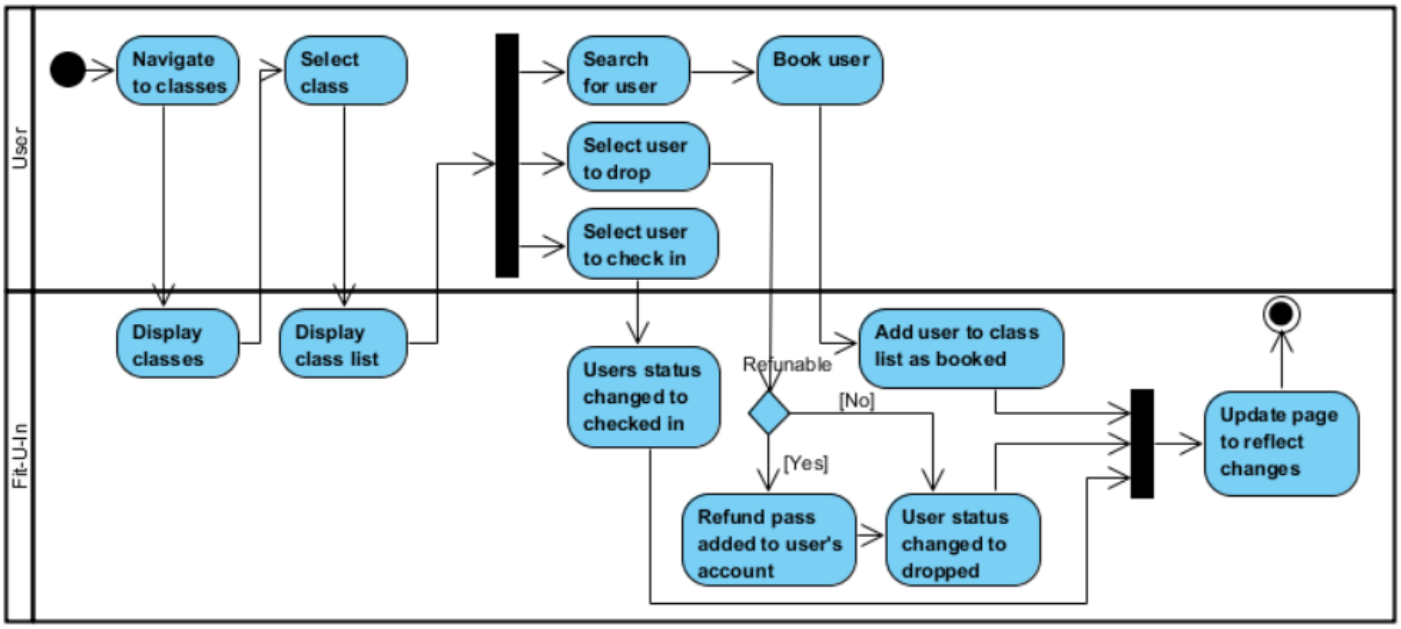

## Purchase product



## Favorite item

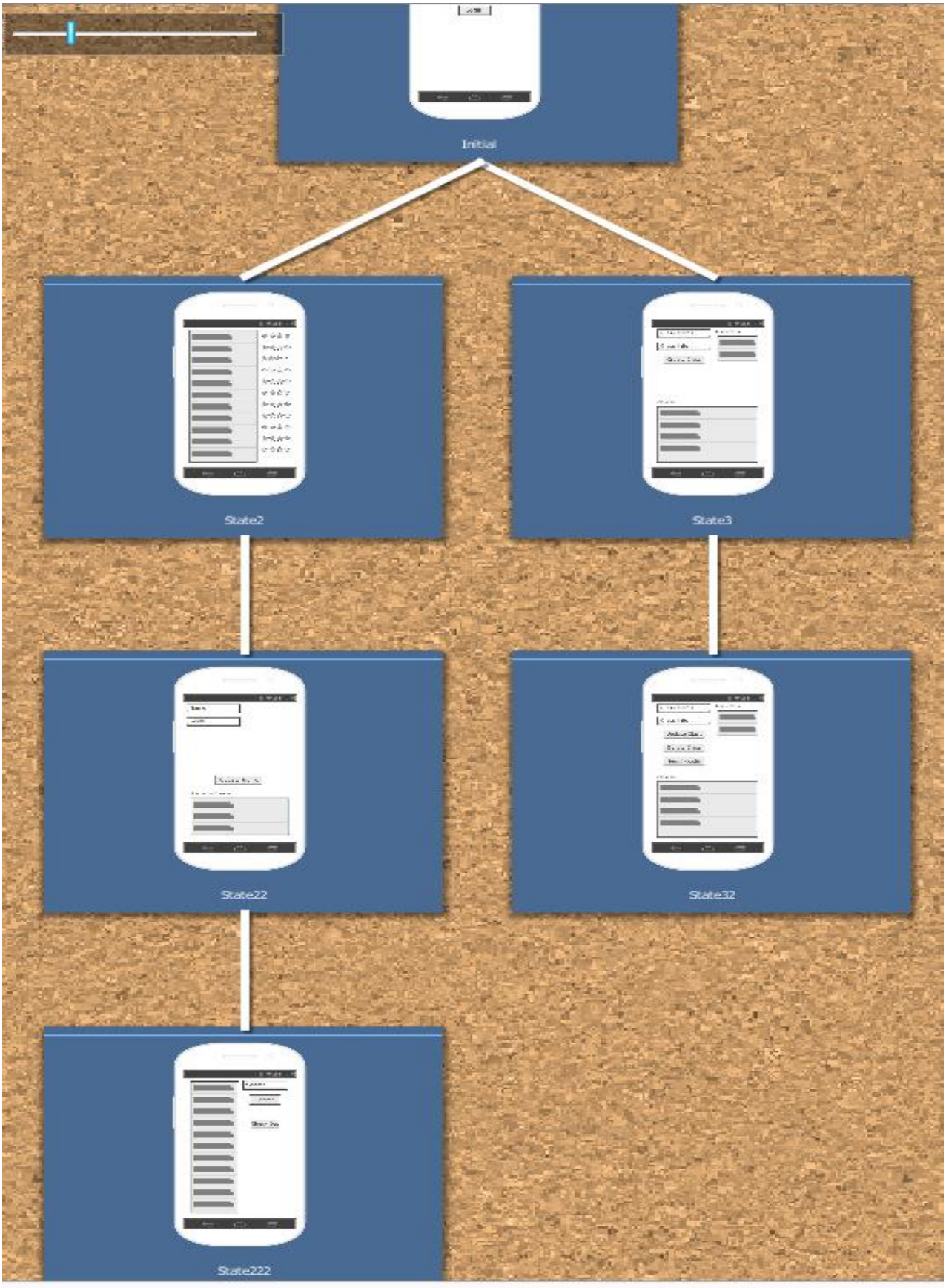



## Manage check-in

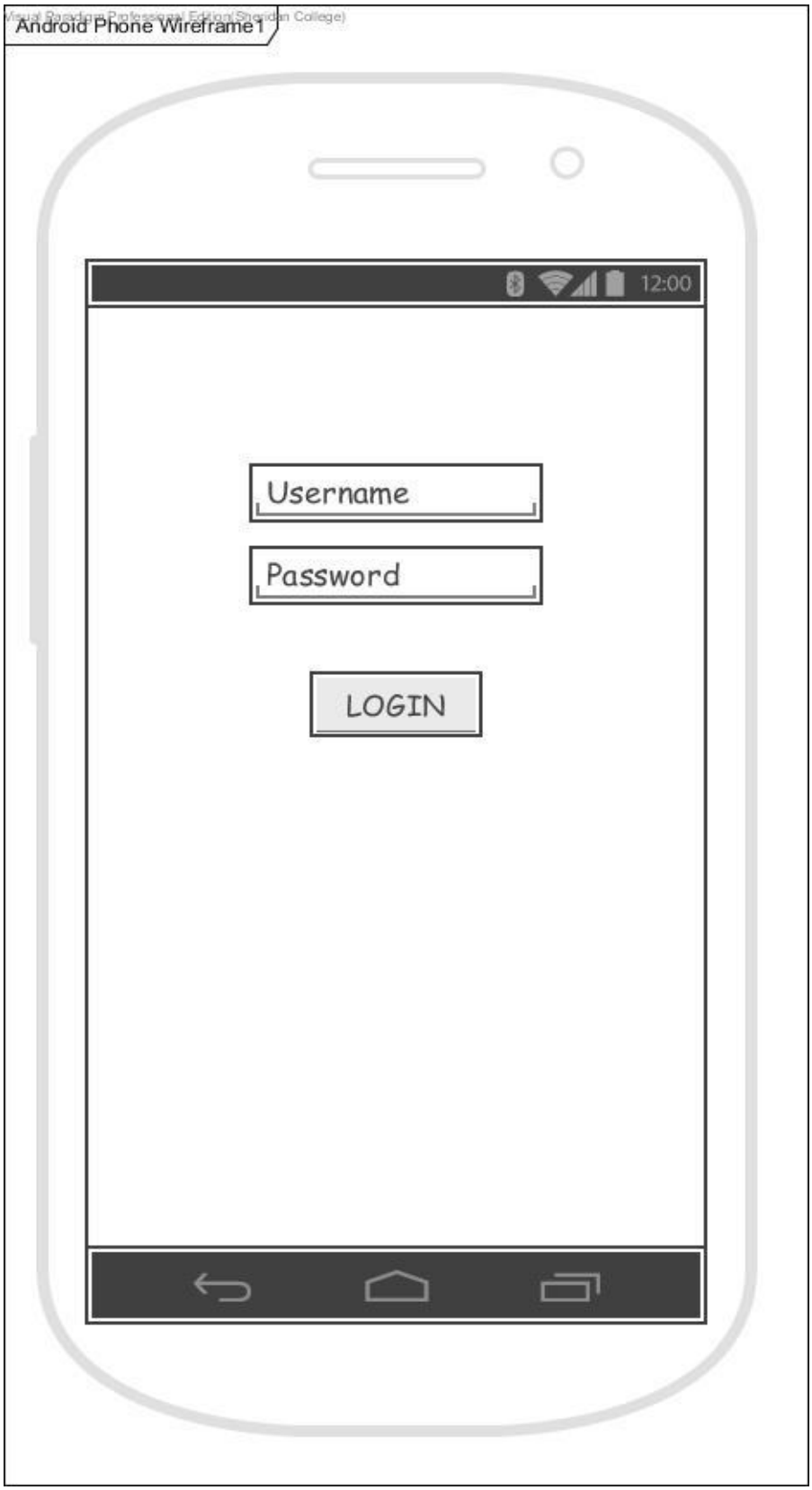



# Wireframes

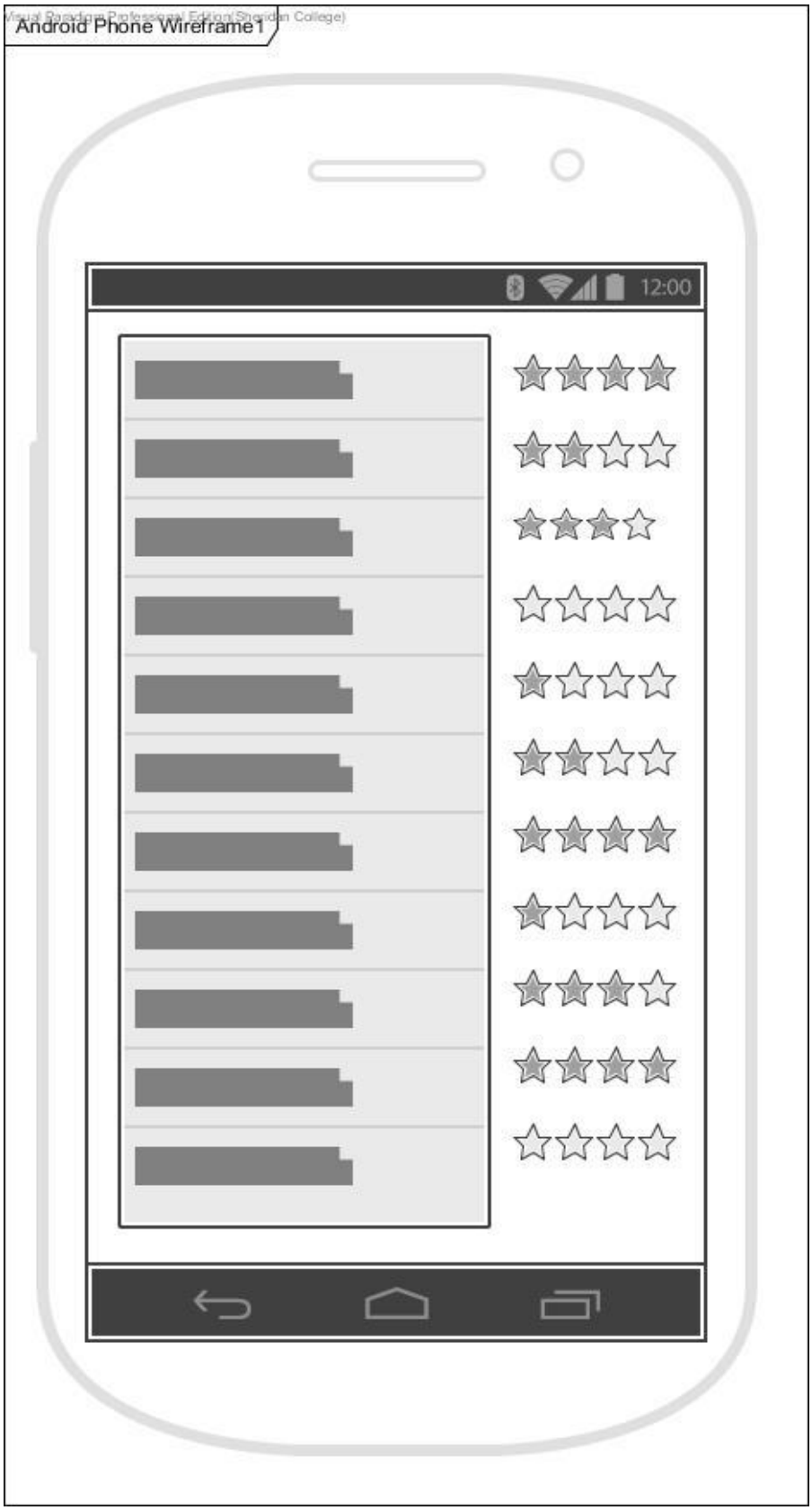
## Overview


Initial
State2
State3
State22
State32
State222

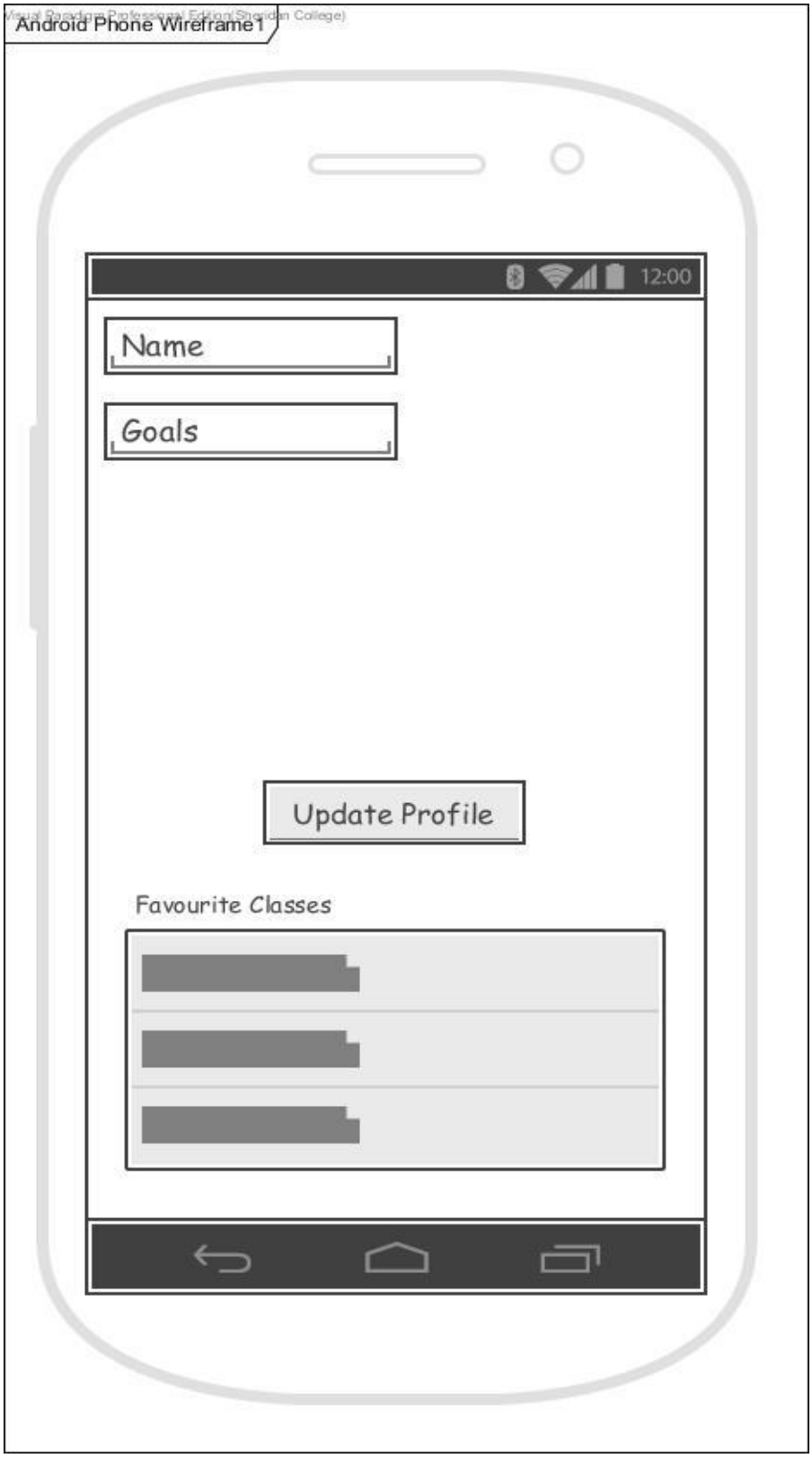
## Initial State



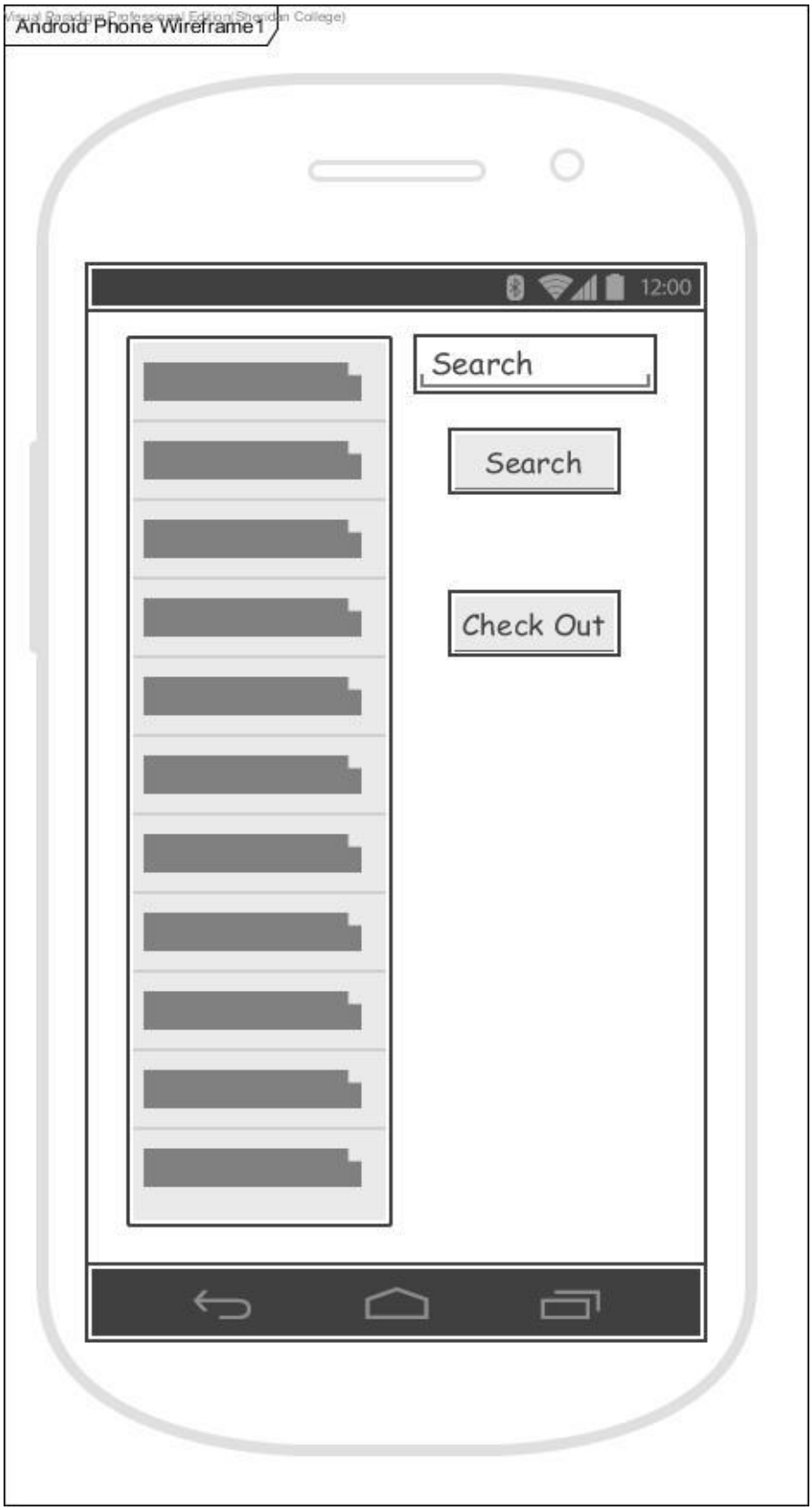
## State 2



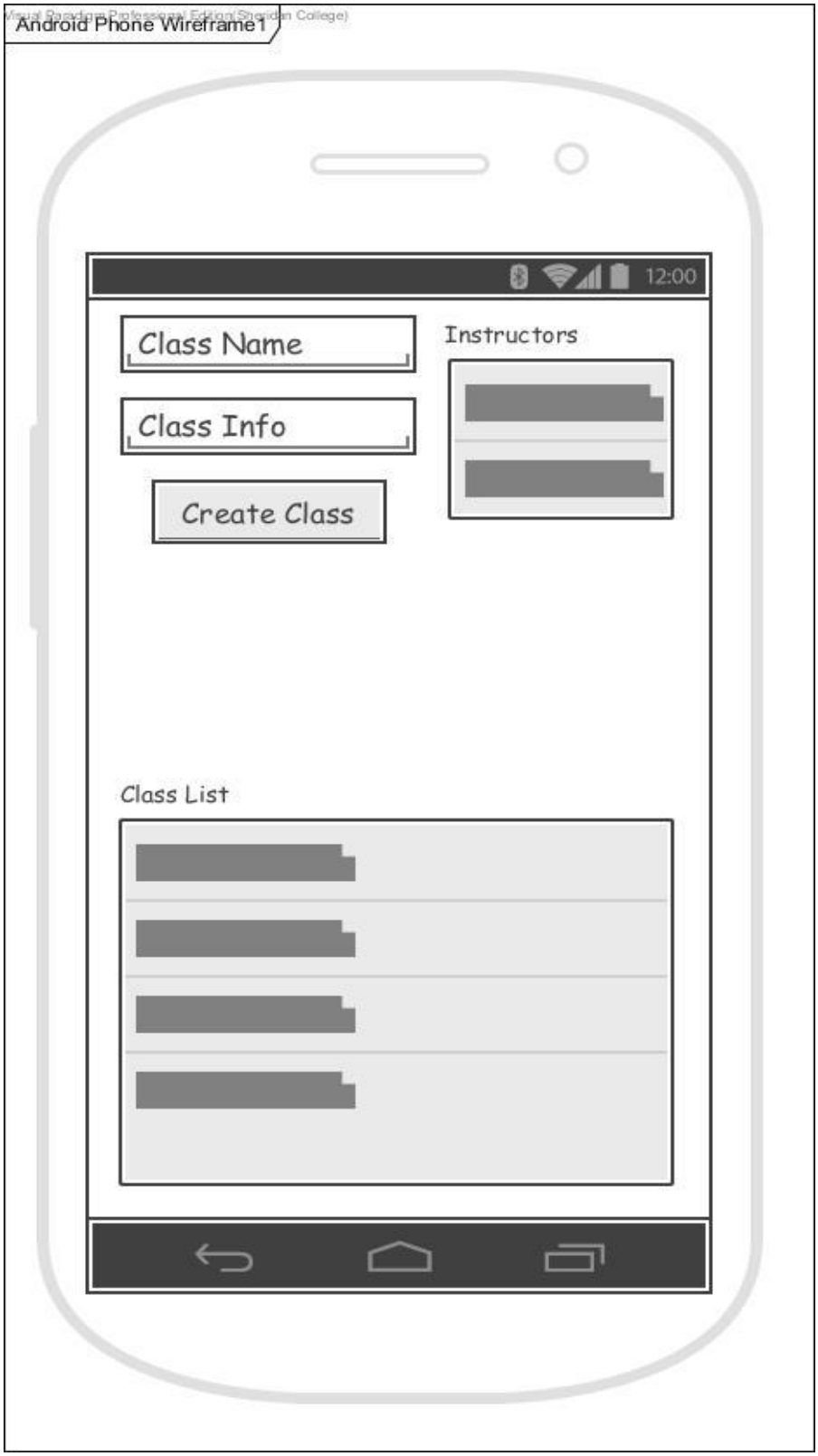
## State 22


Android Phone Wireframe 1
12:00
Name
Goals
Update Profile
Favourite Classes

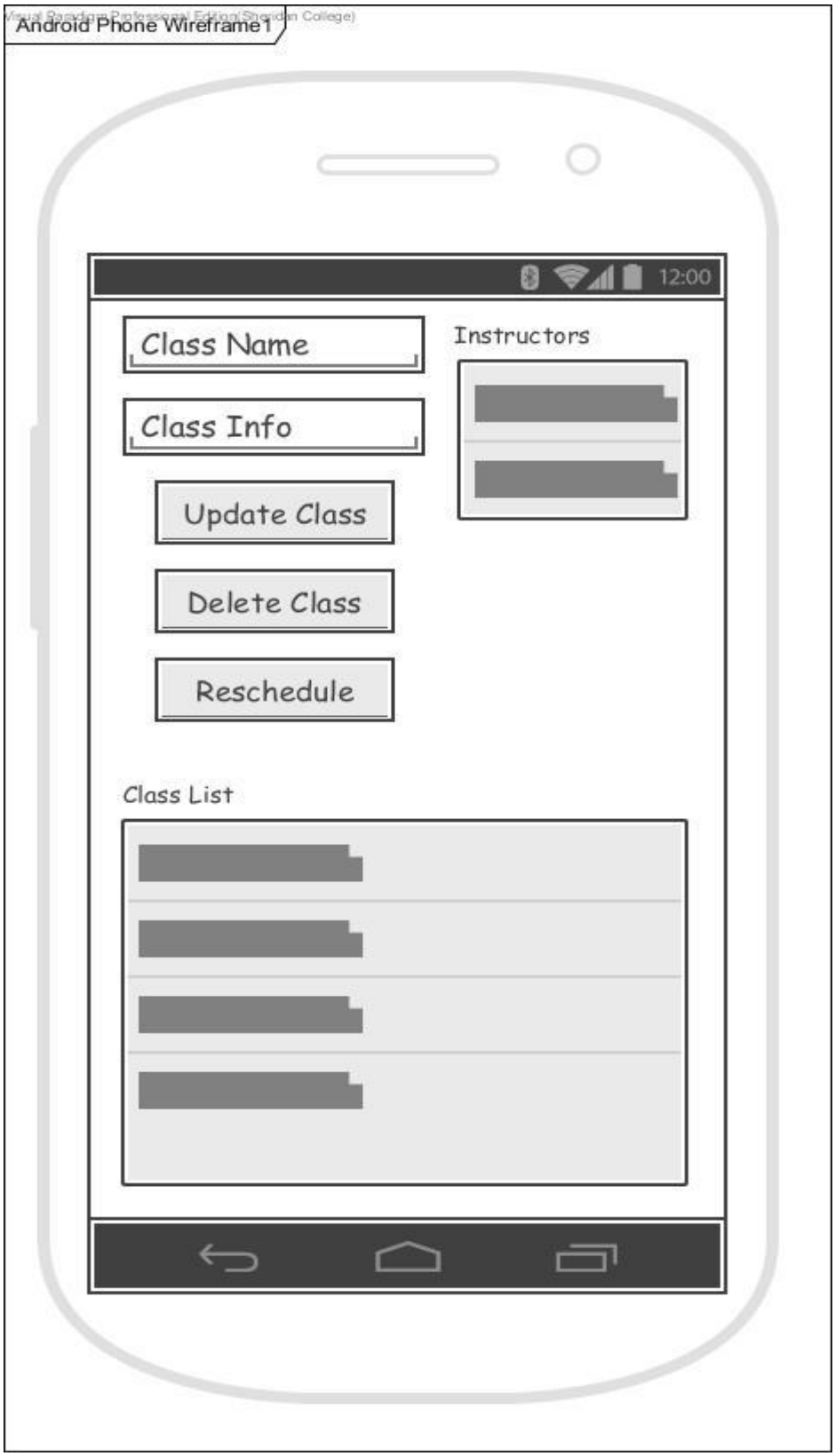
## State 222



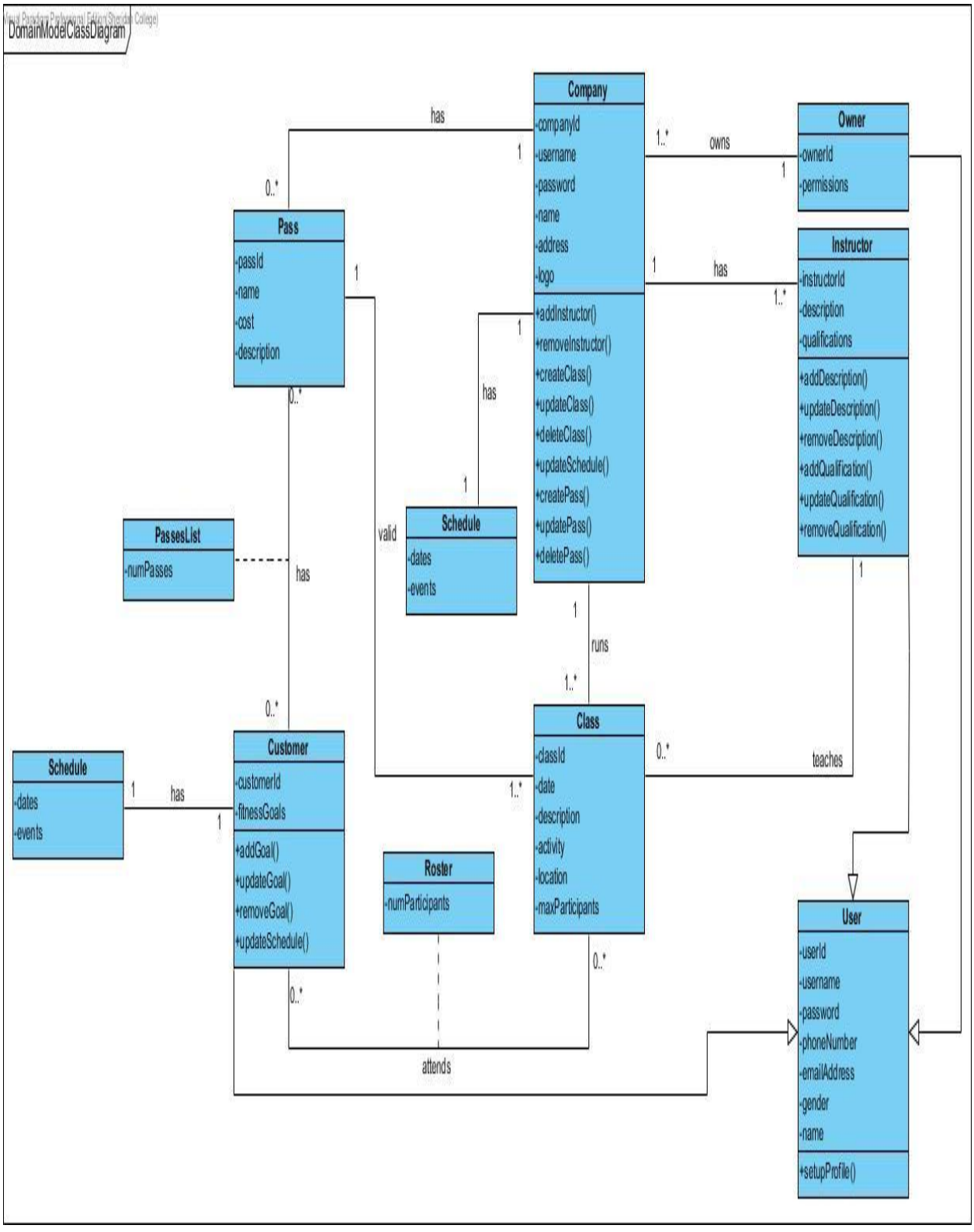
## State 3


Android Phone Wireframe1
12:00
Class Name
Instructors
Class Info
Create Class
Class List

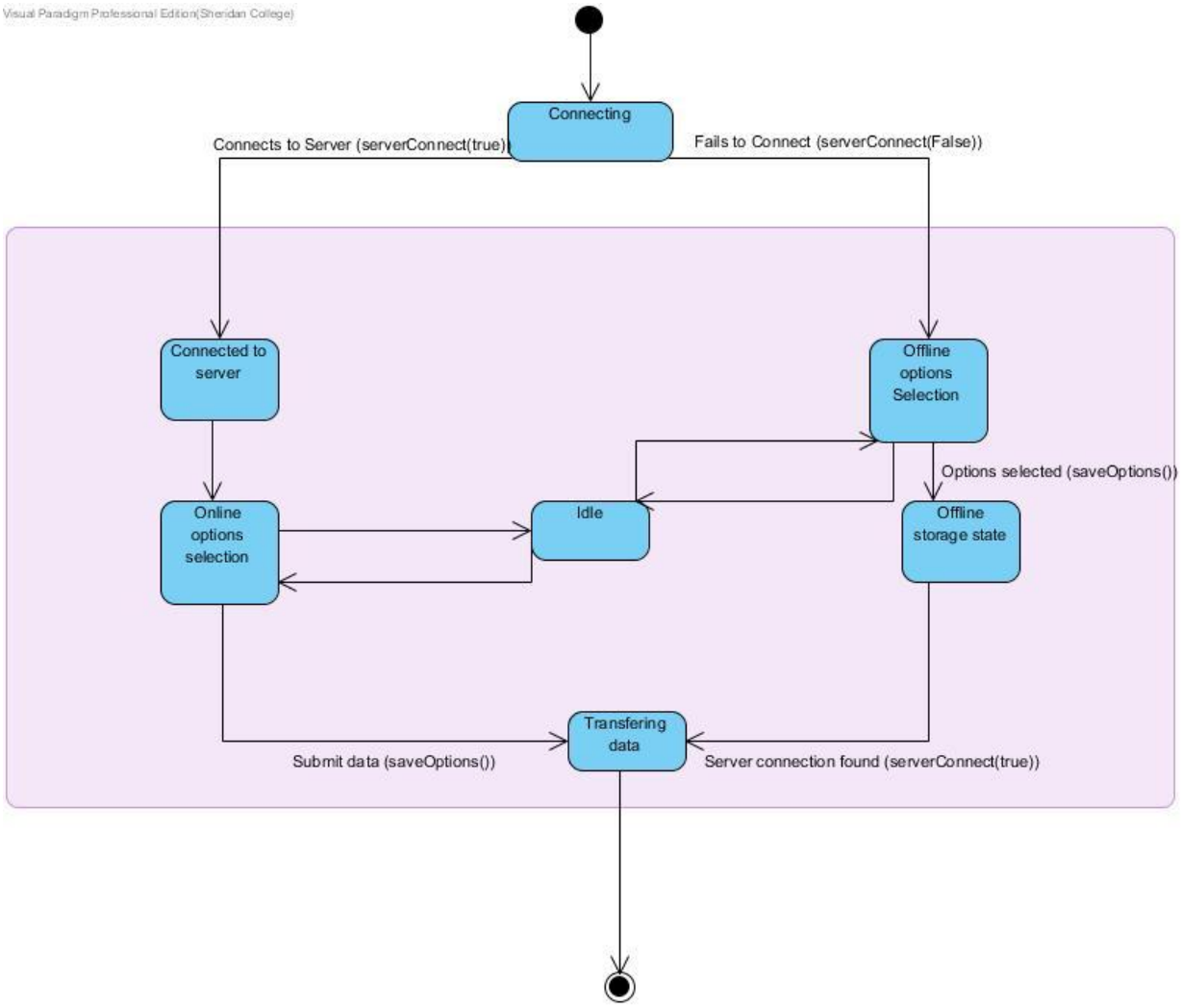
## State 32


Android Phone Wireframe1
12:00
Class Name
Instructors
Class Info
Update Class
Delete Class
Reschedule
Class List

# Domain Model Class Diagram



# State Diagrams




# System Security Requirements

## Physical Security

The project will be using a dedicated host for its database and backend; however, the following are in case the need for administration of a physical server arises.

**Dedicated Space** will be required in case the needs of the application overstep the abilities of the dedicated host; the servers will need a server room in which to operate.

**Door Locks** will be required on the server room door to ensure no one just walks in and tampers with the server.

**Audio/Video Surveillance** is recommended to monitor activity inside and outside the server room to track traffic in and around the room. This will make it easier to identify individuals who have tampered with the servers, and any would-be intruders.

**Rack Mounted Servers** ensure that the servers are secure in their position, making them more difficult to steal.

**Data Breaches** are when secure information is released to an untrusted source, and are amongst the most common issues in physical security, with human error being noted as one of the largest causes. Realistically, there is very little that can be done to eliminate the risk of **Data Breaches**; however, there are a few steps that can be taken, should the need for us to maintain the server ourselves arise. Firstly, all hard drives and company computers will need to be properly encrypted; unencrypted drives may as well be written on paper for how secure they are. Secondly, all hard drives and company devices that have stored and processed data need to be processed, cleaned, and destroyed at the end of their lifecycle. Thirdly, all unnecessary information needs to be automatically removed from the system to remove clutter, and to ensure additional information that no longer needs to be stored is lost.

## Network Security

**The RSA Cryptosystem** is the public-key cryptosystem that will be used in the transference of data between the client and server.

## Application Security

**Server-Side Authentication** and security is something that many people neglect in web development. Many new developers tend to add security protocols to the client side and neglect the server, and thus allow individuals with malicious intent to bypass their security; JavaScript hosted in the browser can be manipulated by the client, and therefore bypassed completely. Though our application will be developed as a Hybrid application, and bundled into an APK, it is possible for users with malicious intent to still decompile the API and make changes to the client. To ensure that this issue doesn't arise, we will ensure that all security protocols are mimicked by the server.

**Cross-Site Scripting** is the most prevalent vulnerability in web security, and is best described as the ability to insert HTML tags into a site through the browser/client. This input is usually submitted through form elements, and can include any HTML tags, including JavaScript script tags. To ensure that cross-site vulnerabilities are minimized, user input will need to be validated at every instance, and sanitized as necessary; no input should go into the database and distributed without first being checked for malicious intent.

**SQL Injection** is another common vulnerability in web security, and is where a malicious user attempts to send SQL commands to the server, generally using form elements. Not ensuring that data sent to the server is sanitized to prevent **SQL Injection** can lead to records, tables, and event entire databases being destroyed. To mitigate the issues caused by **SQL Injection**, all information being sent to the server to be stored in the database needs to be validated and sanitized for SQL commands. In conjunction with validation, the SqlCommand object in the C# language will be used when interacting with the database, which automatically deals with **SQL Injection**.

## File Security

**Remotely Storing Data** is the only way to ensure safety of user information. As we are working in a Hybrid context on Mobile devices, any encryption and security implemented to protect local files is immediately breached the moment someone downloads the client application. To ensure that sensitive user information is kept secure, it will be stored remotely. Only trivial data will be stored locally, through the use of Web SQL Database.

**Triple DES (Data Encryption Standard)** is based on the DES algorithm, and will be used to encrypt data in the database.

## User Security

**Reverse Geocoding Services** will be used to track a user's login locations. As the typical scenario for users is that they will be logging into their application in local areas, the application will send notifications to the user's registered emails, as well as restrict access to many of the application's components, while not in within the user's home country.

**Login Attempts** will be monitored and tracked to mitigate false-logins. Users will be given a total of 5 login attempts before being temporarily locked out of their account, at which point they're free to try and login once again.

**Passwords** will be required to have a minimum length of 8 characters, and may contain both alphanumeric and non-alphanumeric characters. However, the password must include at least 2 upper-case alpha characters, 1 numeric character, and 1 non-alphanumeric character. In addition, the password may not contain the account's username in full.

## Procedural Security

**Form Data** will be processed as outlined in the previous sections. After the user has input the information, the JavaScript security protocols will validate and sanitize the submission as needed and then send it to the server. At the server, the protocols will be run again to ensure that the submitted data is valid and not malicious. The information will then be submitted to the appropriate database.

**Full Database Backups** will be generated daily to ensure that the database is as up-to-date as possible without overloading the network.

**Differential Database Backups** will be generated every 12 hours. While they have the disadvantage of complicating the recovery process, having **Differential Database Backups** further reduce data loss.

**Transaction Log Backups** will be generated every 10 minutes. Being less resource intensive than either **Full Database Backups** or **Differential Database Backups**, **Transaction Log Backups** are the actual log of all activity in the database, and are both able to be generated while the system is working and be used to restore a database to a specific time.

**Backup Storage** will be stored on a remote disk to ensure that even if the hard drive, and indeed the entire RAID 10, goes bad, the backups will still be available to use.

## Operational & executive reports

# Daily Attendee Report

February 16

2016

# Table of Contents

# Executive summary

This report represents the results of the daily activities of the business and its customers. This report is to breakdown the customer's demographical information to which can be used to see who exactly are attending, what and when they are attending classes and what is purchased. Then take that information and display it graphically to see what the current market consists of and where we can set our sights on next.

## Customer Report

| | | |
|---|---|---|
| Number of attendees: | 229 | |
| Average age: | 34 | |
| Number of Females: | 203 | |
| Number of males: | 26 | |
| Peak times: | 4pm – 6pm at 120 attendees | |
| Peak class: | Hot Yoga 90 attendees 3pm-5pm | |
| Total Sales Revenue: $2966.80 | | |
| Top 3 items sold: Class Passes | $2398.80 | 120 sold @ $19.99 |
| Water bottle | $348 | 174 sold @ $2 |
| Yoga mat | $120 | 6 sold @ $20 |

Attendees by geographic location:

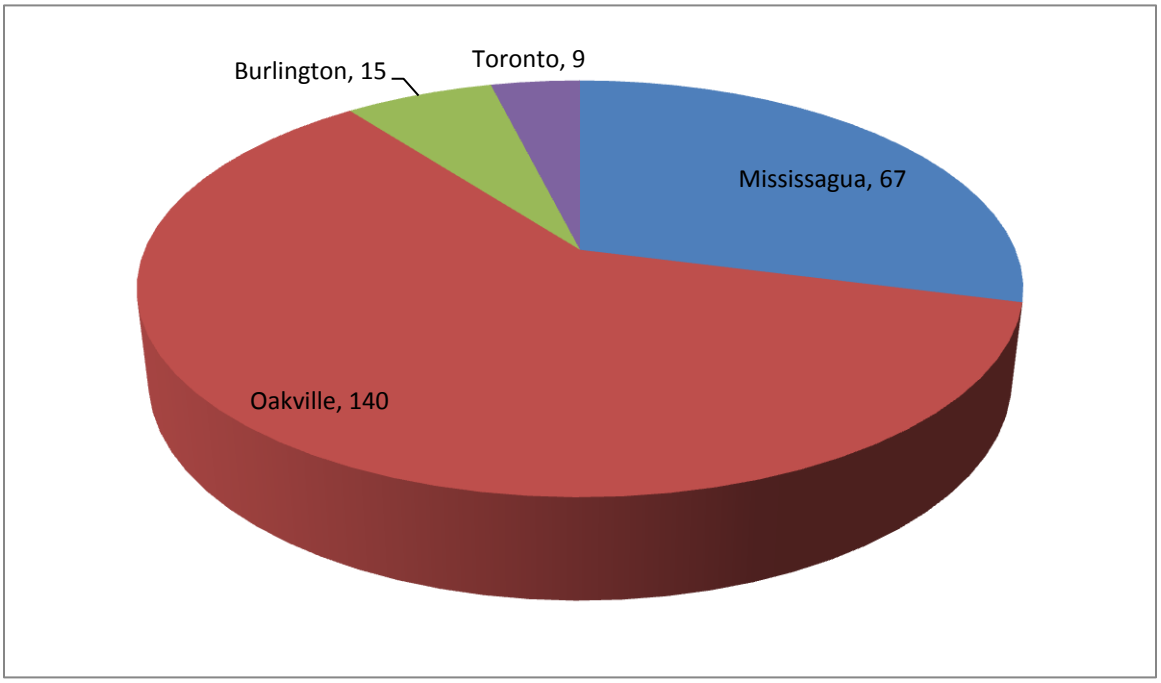



Figure 1 Pie Chart of Geographical Locations of Customers

## Market Opportunity

Based on the demographics it seems most of the clientele are females from the Oakville and Mississauga area. Our clientele seems to purchase passes and water the most often. The peak time our clients come is at 4pm to 6 pm. Potential markets to consider would be the Burlington area with 15% of attendees where from Burlington.

## Recommendations

Based on the information acquired it seems our customer base is females at around the age of 34 from the Oakville and Mississauga area. The recommended options would be to potentially expand to the Burlington region.

## How will the program generate this report?

The report will be generated from the server's database of the daily transactions of the business. Attendees data will be retrieved from the profile data from the ones who have checked in. The sales data will be from the server's sales tables. Peak times and classes will be calculated from the classes with the most attendees at a particular 2 hour period. Potential markets would be looking at the demographical data and selecting ones within a certain distance and under a particular percentage. These reports would be created then be saved daily to the business server.