Yuppie City Simulator

Software Requirements Specification

EECE 4520 - Software Engineering



Meandering Armadillos Members:

Michael Nishida

Andrew Ricci

Nicholas Montaquila

[**INTRODUCTION**](#_a8ieem4tkmoe) **2**

[Purpose of this Document](#_szas4jz5nnnw) 2

[Scope of the Development Project](#_21skppl96olx) 2

[Definitions, Acronyms, and Abbreviations](#_ihaqd0gtf4qb) 2

[References](#_s0puegz8c91m) 3

[Overview of Document](#_irv1ct7odng6) 3

[**GENERAL DESCRIPTION**](#_zdlxprct485w) **3**

[User Characteristics](#_cofj47azzly9) 3

[Product Perspective](#_h1r6ipqs7jzt) 3

[Overview of Functional Requirements](#_9hi2weahjz51) 3

[USER REQUIREMENTS](#_irzn9c7wc7ze) 3

[APPLICATION REQUIREMENTS](#_h5z7yi4crhly) 4

[Overview of Data Requirements](#_83rf8i64y8xf) 4

[General Constraints, Assumptions, Dependencies, Guidelines](#_nke8f44oevpt) 4

[User View of Product Use](#_2basv6e5wwsr) 5

[**SPECIFIC REQUIREMENTS**](#_sqcuiji5a9nu) **6**

[External Interface Requirements](#_kecdlku15qbo) 6

[Detailed Description of Functional Requirements](#_if1ohh6r5em9) 6

[Detailed User Requirements](#_9hwn2y1ex0g7) 6

[Detailed Application Requirements](#_yme9thtilej) 9

[Performance Requirements](#_j892q38d8ay) 11

[Quality Attributes](#_oczfc2mfw1im) 11

[**Other requirements**](#_f7m97bdaroco) **11**

[**Appendix**](#_8w10akrgo2z8) **12**

[Class Diagrams](#_f7gbvtfbaxaw) 12

[Minimal Full Class Diagram](#_da4sakvqujnh) 12

[Full Class Diagram](#_558d67po1qm8) 12

[Page Related Class Diagrams](#_33pkmydqg9xn) 14

[User Profile Related Diagrams](#_kdwz9dqsd9d3) 14

[Database Related Diagrams](#_esvrw5mgdi0l) 14

[Use Case Diagrams](#_6qk962en3gcm) 15

[User Login and Profile Creation/Modification](#_ffb085stdy2c) 15

[View Results, City Index, and Modify Answers](#_d15lh1llw6kb) 16

[Take Quiz](#_800hntm5vf6i) 17

[Full Diagram](#_sqhu759ub7mg) 18

[Sequence Models](#_6xsp01ro03vr) 19

[Create User Profile](#_2lxygnr1e6so) 19

[Modify User Profile](#_pjfszg9talsr) 20

[User Login](#_m7i24tm8dpog) 21

[Take Quiz](#_zeeb0vdqodff) 21

[Modify Quiz Answers](#_swqioq3gg931) 22

[View Results](#_rgab5efnef64) 22

[City Index](#_swqioq3gg931) 23

# **INTRODUCTION**

## Purpose of this Document

In this document, Meandering Armadillos will quantify the scope, requirements, and extraneous details related to the creation of the Yuppie City Simulator project. This will serve to help Meandering Armadillos understand the minutiae of Yuppie City Simulator project details, and serve as a template for all tasks related to the creation of the project. This will also allow Professor Singh and company to understand in thorough detail Meandering Armadillos’ plans for the project, assisting in the assessment of project checkpoints and eventually a finalized work.

## Scope of the Development Project

Yuppie City Simulator will be a software program that provides users with a data based analysis of cities based on user chosen factors. Meandering Armadillos have determined the best format for this to be a website that is complete with user profiles, a quiz, and a city index. The core features offered will be an numerical, algorithmic response to the user specified categories in the quiz, and access to our database of city information. The main limitations of this can be subjectivity in regards to city size, features, and statistics like “walkability” and “standard of living”. Additional limitations could be found in the quality of Meandering Armadillos algorithm, which will be the primary focus of the Yuppie City Simulator platform. However, upon successful implementation, Yuppie City Simulator will offer users a quantitative result of suitable cities.

## Definitions, Acronyms, and Abbreviations

**Meandering Armadillos** - MA: The development team including Nick Montiquila, Michael Nishida and Andrew Ricci

**Yuppie City Simulator** - YCS: The product being developed by the Meandering Armadillos to simplify the search for an appropriate city to live in

**Hypertext Markup Language** - HTML: A commonly used markup language for web development

## References

<https://www.amazon.com/Yuppie-Handbook-State-Manual-Professionals/dp/067147684X/ref=sr_1_1?ie=UTF8&qid=1549401728&sr=8-1&keywords=yuppie+book>

## Overview of Document

This document will offer a detailed analysis of product requirements, both functional and non-functional. The SRS is divided in sections based on technical and non-technical aspects, and will identify goals needed for YCS to achieve basic functionality as well as ideal goals.

# **GENERAL DESCRIPTION**

## User Characteristics

Our users will expect a granular and detailed approach to city selection, as well as more general categories that can give a broad result that still appeals to the users’ choices. This will allow users to get a result they desire in a short time, or a more precise result with all categories entered. Users will also enjoy easy access to the YCS platform, through a straightforward login, simple profile UI and listed results of prior quizzes. The YCS City Index will also need to be easy to access and presented in a fashion that is easy to read and sort.

## Product Perspective

Yuppie City Simulator will be a stand alone website. Though all user interactions will be with data within the YCS site and databases, all data will be gathered from external sources. This means that YCS will be comprised of data from large university studies, compilations of local government statistics, and other aggregates like Wikipedia. Other than the initial collection of data in MA’s databases, all information will be accessed locally to the Yuppie City Simulator platform.

## Overview of Functional Requirements

### USER REQUIREMENTS

* + - 1. User must be able to create a profile.
      2. User must be able to modify an existing profile, given that they provide the correct credentials.
      3. User must be able to log into the system if they provide a correct set of credentials
      4. User must be able to take a quiz outlining their preferences for living conditions
      5. User must be able to modify the results of a previously taken quiz
      6. User must be able to view the results of a quiz, either directly after taking it OR viewing the results of a previously taken quiz.
      7. User must be able to access the raw data on cities as an index (City Index)

### APPLICATION REQUIREMENTS

* + - 1. Application must be able to store and modify user profile data
      2. Application must be able to modify user profile data
      3. Application must be able to store city data
      4. Application must be able to take user quiz input and establish what city they would be most likely to enjoy living in
      5. Application must be able to verify user credentials

## Overview of Data Requirements

There are three main types of data used on the Yuppie City Simulator platform, which interact with each other in various forms.

* User profile data
* User Quiz Data
* City Data

All data related to user profiles that is deemed non-confidential will be stored in a user profile database. This will include all data that is public on a user’s profile, which will include username, number of completed quizzes, top cities, etc. The user quiz data will be stored in a separate table with a corresponding user ID for each quiz. The quiz data will include a score from 1-10 for each category the user has chosen to answer. The city data will be stored in a third table, which will be accessed by the quiz algorithm to determine best fitting cities. The city data will also be accessed when the user opens the city index section of YCS..

## General Constraints, Assumptions, Dependencies, Guidelines

* + 1. The system must be web-based. The user must have a working internet connection. If internet connection is temporarily interaction, system should maintain its state until connection is reestablished or session times out.
    2. The user’s information must be stored securely. Encryption method should include salting and hashing of user’s data. System should be able to survive a crash without loss of user data.
    3. System should be able to display correctly on multiple devices (desktop, laptop, phone/tablet, etc.).
    4. The system should begin responding to user’s input within one second. System should include a method of notification when the system is loading other pages so as to avoid the user’s assumption of a crash.

## User View of Product Use

Our website, yuppiecitysimulator.ru.biz, will be themed around nostalgia for 1990s web use. This means that it will be written primarily in HTML, and minimally use Cascading Style Sheets or other more fancy web development. On login, the user will be greeted with a simple home page that displays a list of links to each service, as follows: Profile Management, Take Quiz, Edit Quizzes, and City Index. Upon selecting Profile Management, the user can edit features of their account like username and password. Every page will contain a link back to the home page. The Take Quiz page will have a single box for each city feature. The user can type from 0-10 the importance they place on each feature, and submit the quiz when completed. Selecting edit quizzes will allow a user to click on a quiz and perform a similar functionality. Our City Index will be a simple table that shows each city as a row, each attribute as a column, and displays a value based on that particular city’s performance. Included below are examples of what data from a user quiz and the City Index might look like.

Sample Quiz Data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| user\_id | quiz\_id | Walkability | Annual Sunshine | Population | Healthcare |
| 61 | 1425 | 10 | 7 | 0 | 8 |

Sample City Index:

|  |  |  |  |
| --- | --- | --- | --- |
| city\_id | Walkability | Annual Sunshine | Healthcare |
| Boston | 9 | 7 | 10 |
| New York City | 10 | 7 | 8 |
| Los Angeles | 4 | 10 | 7 |

# **SPECIFIC REQUIREMENTS**

## External Interface Requirements

The website must be able to display appropriately on various display sizes, resolutions, and aspect ratios. Each page of the website should have a menu that allows the user to navigate to the other main pages of the site. It must also support various devices (monitors, laptops, and smartphones. The website must be able to query the database to retrieve city data according to various attributes.

## Detailed Description of Functional Requirements

Class diagrams outlining the overall structure of our project, as we currently understand it, are available in the appendix section 5.2.1.

Similarly, sequence diagrams for each use case are provided in appendix section

### Detailed User Requirements

* + - 1. User Profile Creation

|  |  |
| --- | --- |
| Purpose | Enable the user to create a profile for the application to store and modify. See Appendix 5.2.1 for diagram |
| Inputs to Component | User designated:   * Username * Password * Email Address |
| Processing | System verification that username and email are NOT already in use. In the event that either are, throw an error the user and have them input a new set. |
| Outputs | A User profile object that will be handed to the application’s backend for storage. (3.2.2.1) |

* + - 1. Modify User Profile

|  |  |
| --- | --- |
| Purpose | Enable the user to modify their existing profile data.  See Appendix 5.2.1 for diagram |
| Inputs to Component | If NOT logged in:   * Current username for login * Current password for login   Updated values for:   * Username (optional) * Password (optional) * Email Address (optional) |
| Processing | IF NOT LOGGED IN:  System follows login procedure outlined in 3.2.1.3  System verification that new username / email are NOT already in use. In the event that either are, throw an error the user and have them input a new set. |
| Outputs | Modified user profile that will be handed to the application’s backend for storage. (3.2.2.2) This new profile will override the existing one in the backend. |

* + - 1. User Login

|  |  |
| --- | --- |
| Purpose | Enable the user to log into the application with an existing set of credentials and begin using the application with the linked user profile  See Appendix 5.2.1 for diagram |
| Inputs to Component | User given:   * Username * Password |
| Processing | System verification that the username / password combination does exist in the system. If the combination does NOT match any existing records, throw an error to the user and have them try again. |
| Outputs | Matching profile is activated and user is brought to the home screen. |

* + - 1. User taking quiz

|  |  |
| --- | --- |
| Purpose | Enable the user to take the quiz that is the cornerstone of the YCS project.  See Appendix 5.2.2 for diagram |
| Inputs to Component | User designated:   * Responses to quiz |
| Processing | System validates the the quiz responses are valid (for free-response type questions. e.g type a value from 1-10). If not, system throws an error asking the user to re-input their response.  System then moves to process outlined in 3.2.2.4 |
| Outputs | User profile is updated with a new quiz-response entry and is handed to the application’s algorithm for use in classification and storage. See outline in 3.2.2.4  Results are then presented in accordance with 3.2.1.6 |

* + - 1. User modifying existing quiz.

|  |  |
| --- | --- |
| Purpose | Enable the user to modify an existing set of quiz responses in order to reflect updated views on preferences.  See Appendix 5.2.2 for diagram |
| Inputs to Component | User designated:   * Quiz Selection * Updated Quiz Responses (Optional)   System pull (automatic):   * User Profile: Quiz result set |
| Processing | Once the user designates the quiz they’d like to modify, the system pulls up that response set to display. User can then update answers as they see fit.  System validates free response questions in the same way as it does for 3.2.1.4 and then moves to process outlined in 3.2.2.4 |
| Outputs | User profile is updated with the updated quiz-response entry and is handed to the application’s algorithm for use in classification and storage. See outline in 3.2.2.3 |

* + - 1. User Viewing Results

|  |  |
| --- | --- |
| Purpose | Enable the user to see the results of their quiz(zes) at any time, provided they already have some existing.  See Appendix 5.2.2 for diagram |
| Inputs to Component | User designated:   * Quiz result selection   System pulls:   * User Profile: Quiz Results. |
| Processing | System verifies that the User profile DOES have at least one set of quiz results. Returns an error to the user if they do NOT.  System then pulls the results of the chosen quiz from the user profile for display.  Alternate: If coming from 3.2.1.4 (and subsequently 3.2.2.3), system will automatically pull results from the just-completed quiz. |
| Outputs | Displays the city that YCS has calculated would best fit the user’s response. |

* + - 1. User Use of City Index

|  |  |
| --- | --- |
| Purpose | Enable the user to view an index of raw data on various aspects of city measurements.  See Appendix 5.2.2 for diagram |
| Inputs to Component | User designated:   * Categories for filtering |
| Processing | System queries application backend for data corresponding to user filter. |
| Outputs | Displays information secured from backend. |

### Detailed Application Requirements

* + - 1. Application Storage of User Profile Data

|  |  |
| --- | --- |
| Purpose | Enable the system to have a persistent database of user profiles for reference in various functions. |
| Inputs to Component | User profile consists of:   * Username * Password * Email * Quiz Responses Set * Quiz Results set |
| Processing | System stores User profile in the back end. |
| Outputs | System returns user profile when requested. Success message. |

* + - 1. Application Modification of User Profile Data

|  |  |
| --- | --- |
| Purpose | Enable the system to update information in existing user profiles. |
| Inputs to Component | System receives updated user profile from 3.2.1.2 or 3.2.2.4 |
| Processing | System overrides original profile stored in backend with new profile data. |
| Outputs | System returns Success message. |

* + - 1. Application Storage of City Data

|  |  |
| --- | --- |
| Purpose | Enable the system to locally store a persistent database of city data. |
| Inputs to Component | City Data (format currently TBD) |
| Processing | System stores City Data in the back end. |
| Outputs | System returns City Data when queried by 3.2.2.3.4 or 3.2.2.1.7 |

* + - 1. Application Algorithm Execution

|  |  |
| --- | --- |
| Purpose | Enable the system to run calculations on user quiz responses in order to determine an ideal city to live in |
| Inputs to Component | User profile handed over from either sections 3.2.1.4 or 3.2.1.5  City Data from query to database, outlined in sec 3.2.2.3 |
| Processing | System runs algorithms to match the user responses to the available city data.  Modifies the User Profile to include a proper result from calculations. |
| Outputs | System returns results to user in accordance with sec 3.2.1.6  System stores updated user profile in accordance with section 3.2.2.2 |

* + - 1. Application Credential Validation

|  |  |
| --- | --- |
| Purpose | Enable the System to verify that the user attempting to login is using correct credentials |
| Inputs to Component | User designated:   * Username * Password |
| Processing | System queries backend for any user profile (in accordance with sec 3.2.2.1) matching the given credentials. If NONE found, throws an error. |
| Outputs | If no credentials are found, error is thrown. Otherwise, success message is given. |

## Performance Requirements

* + - * + System should respond to user’s input in less than 1 second.
        + System should be able to generate recommendations after survey completion in less than 30 seconds.
        + System should be able to support 1,000 simultaneous users.
        + System should be able to survive a crash without loss of user data.

## Quality Attributes

* + - * + The user’s info should be stored securely.
        + User should be able to complete survey and get recommended cities without creating a profile, however their answers will not be saved unless they then complete a profile.
        + Upon the user’s input of invalid credentials, system should prompt user to re-enter credentials without needing to reload the entire page.
        + Website should not be down for more than 30 minutes per week.

# **Other requirements**

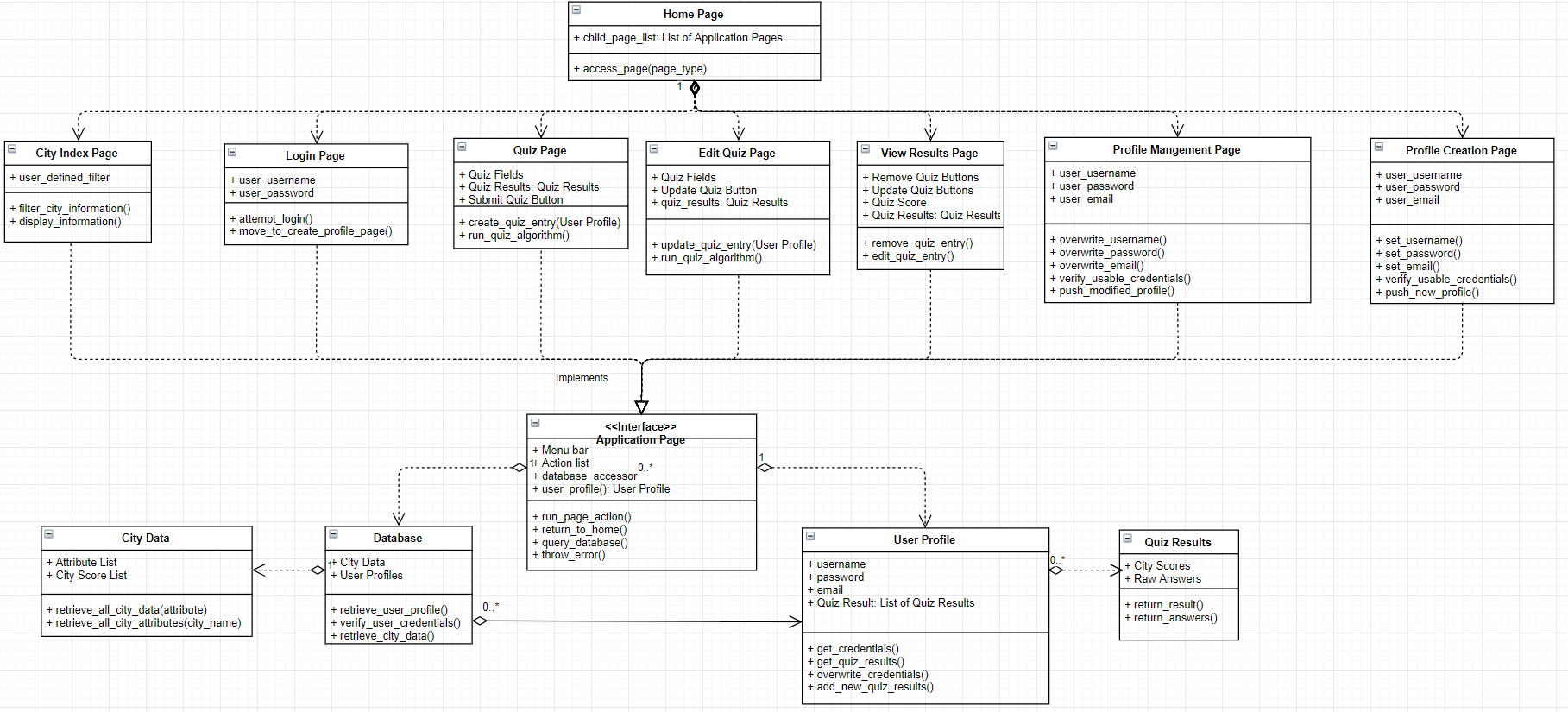
N/A

# **Appendix**

## Class Diagrams

### Minimal Full Class Diagram

### Full Class Diagram



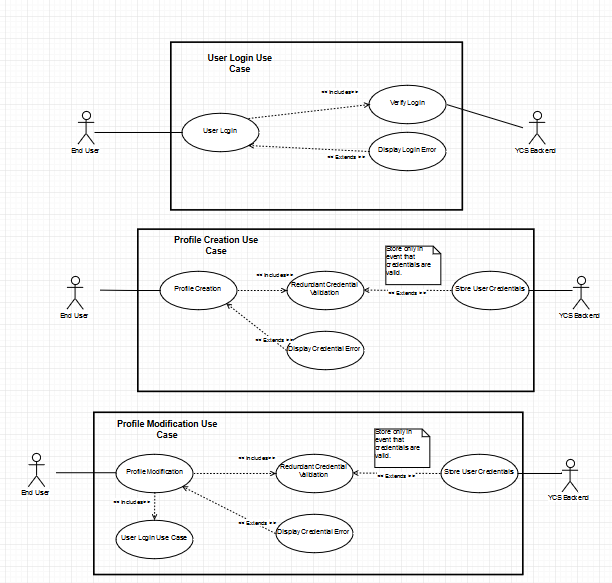
### Page Related Class Diagrams

### User Profile Related Diagrams

### Database Related Diagrams

## Use Case Diagrams

### User Login and Profile Creation/Modification



### 

### 

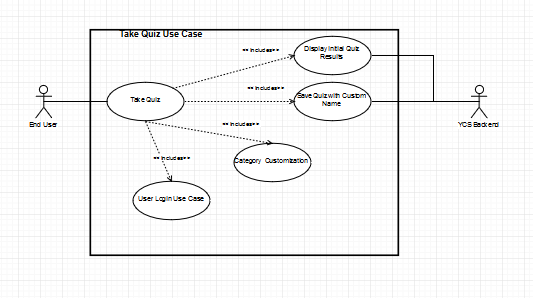
### 

### 

### 

### View Results, City Index, and Modify Answers

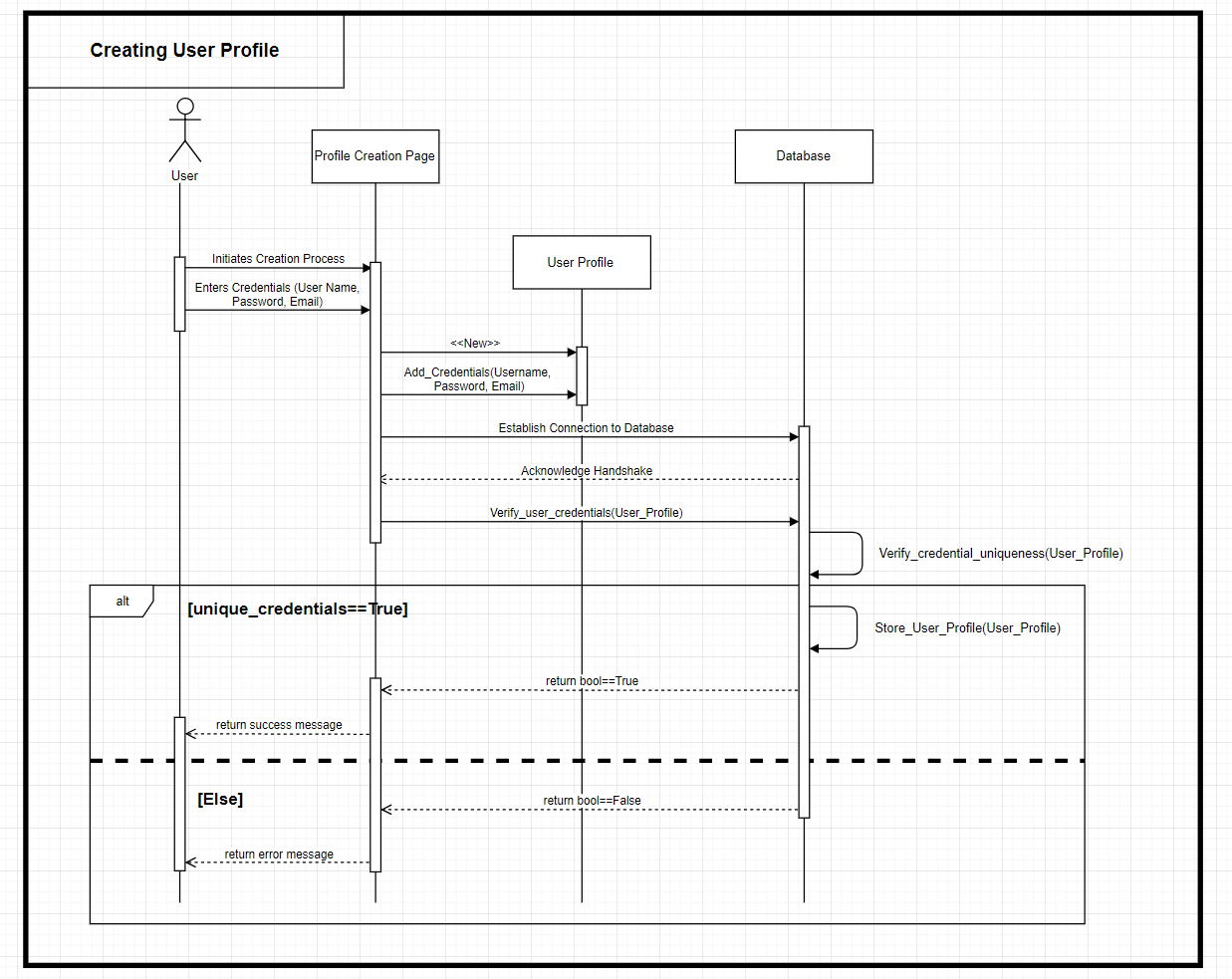
### Take Quiz



### Full Diagramhttps://puu.sh/CJKH7/2ff59bc5f9.png

## Sequence Models

### Create User Profile



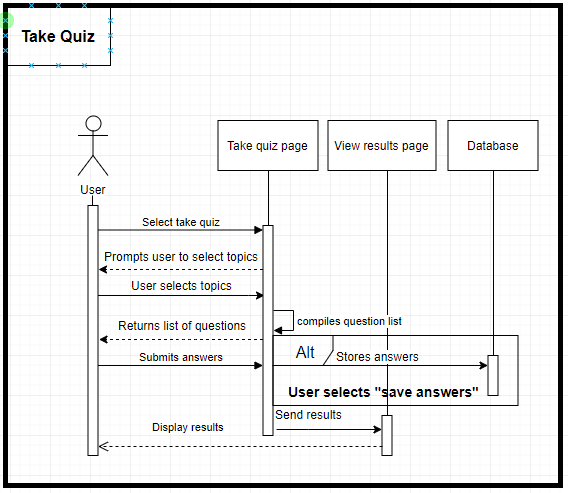
### 

### 

### Modify User Profile

### User Login

### Take Quiz

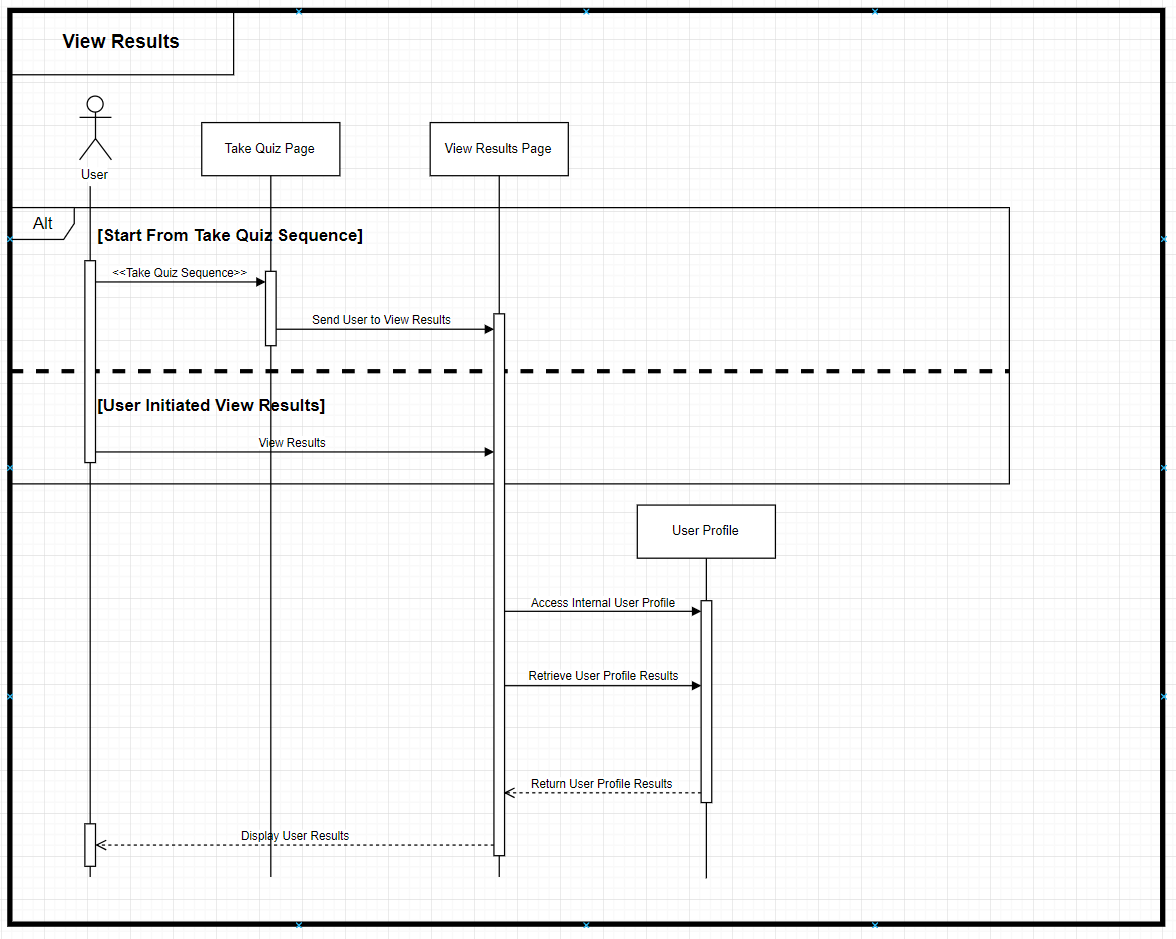


### Modify Quiz Answers

### 

### 

### View Results



### City Index

### 