**Software Design Specification (SDS)**

1. **Introduction** 
   1. **Purpose of this Document**   
        
      To help Meandering Armadillos team members properly ascertain a series of goals and standards related to the completion of the Yuppie City Simulator project. This will standardize the creation of the Yuppie City Simulator Platform and allow a seamless implementation of all required aspects.
   2. **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.
   3. **Definitions, Acronyms, and Abbreviations**

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

**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

* 1. **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>

<https://www.amazon.com/Bagel-Photograph-Picture-Super-Photos-ebook/dp/B07K8ZHT9B/ref=sr_1_3?keywords=everything+bagel+seasoning+book&qid=1552504085&s=gateway&sr=8-3>

<https://docs.google.com/document/d/1RMsAopTiX6urjoYWUacLCH3WspzccYcVgpwQwcavYCQ/edit?usp=sharing>

* 1. **Major software requirements**

The most crucial aspects of our software, and the ones that will allow us to achieve success at the most fundamental level, are the quiz and the city database. With these two elements, a user will be able to take a quiz that is reliant on the data from our database and receive a result based on their answers. Beyond this, a website that houses our quiz and stores past results on a per user basis will allow our software to provide a more customizable user experience.

* 1. **Design constraints, limitations**

In our project’s current state, our main limitation is time. In order to implement many of the features of our project, we will need time to develop a programming strategy and test out the features we are implementing. If we have any hitches in the development of our software, it could result in us being unable to implement major features. Beyond this, our project requires a few elements be present in order for others to be built around them, and until those elements, such as the database, are completed it will be difficult to finish any work on other elements.

Additionally, our group is specialized towards specific areas of development, which means that if one of us has an issue with our particular section of work it may be difficult to assist each other. Our team is not web development focused, which means we will be learning as we go in regards to the creation of our project’s website.

* 1. **Design Goals**

Reliability: The YCS system should be reliable in uptime and general availability to any client [generalization of Non-Functional Requirements 3,8 (NFR-3,8)]

Fault Tolerant: The YCS system should be fault tolerant to the loss of connectivity without corruption of data [generalization of NFR-4]

Response Time: The YCS system must be able to act upon user input, including generation of quiz results, in reasonable amounts of time [generalization of NRF-1,2]

Security: The YCS system must securely store user information to prevent unauthorized access [generalization of NFR-5]

* 1. **Changes to requirements**   
       
     Due to the nature of us not completing any new coding developments since the previous requirements documents, we have not elected to make any changes to our requirements. However, we have considered the importance of bagels, which is now included in our sources.
  2. **Overview of Document**   
       
     The rest of our SDS will follow the format that has been given. If it strays in any way, it is a result of errors from Meandering Armadillos personnel exclusively.

1. **Data Design**

Data Objects and resultant data structures:

User Profile:

User Name: String

Password: String

Email: String

Quiz Results: Quiz Results

City Data:

City Name: String

Attributes: Map <String, Int>

Quiz Results:

Answers: Map <String, Int>

Result Cities : List <String, Int, City Data>

* 1. **File and Database Structures**
     1. **External File Structure**

Our file structure will consist of our website’s homepage and corresponding pages including the quiz and user profile page. A separate python file will house the algorithm that runs our quiz and calculates the user’s results.

* + 1. **Global Data**

Our database is our only “global data” as it is accessed by both our python algorithm and our website. This is housed on a MySQL server hosted by AWS.

* + 1. **File and Data Cross Reference**   
       N/A

1. **System Architecture Description (This is your architecture design.)** 
   1. Overview of Modules / Components:

User Interface: Composed of all pages accessible to the user. Provides user with the means to interact with the Yuppy City Simulator. Broken down into three main sub-components:

Quiz: Contains all pages necessary for the user to execute a quiz workflow or modification to an existing quiz.

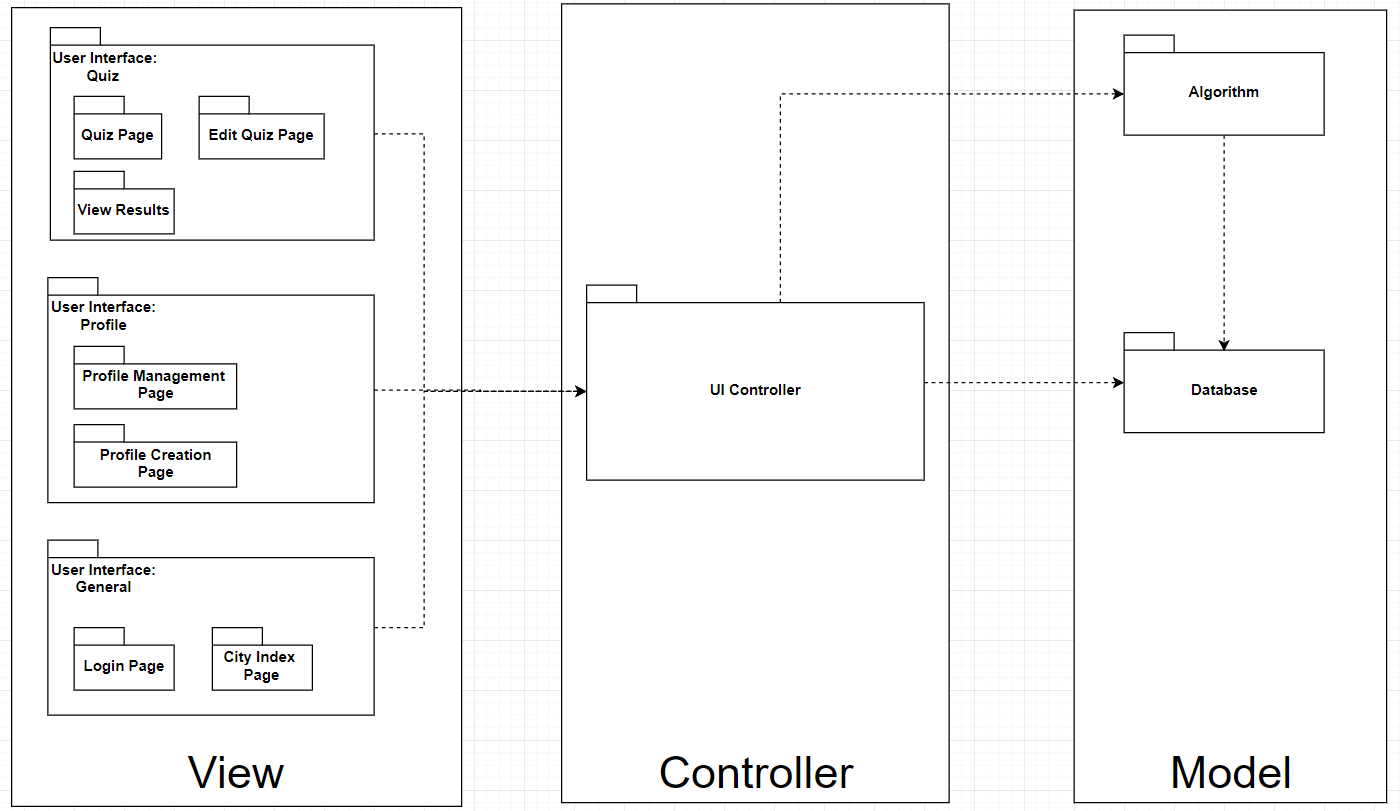
User Profile: Contains all pages necessary to create or modify a user profile.

General: General / miscellaneous pages, including the login, and general city data pages.

UI Controller: Module that handles the processing of user information into the UI, including database querying and algorithm processing.

Algorithm: Module that allows the data gathered from the UI to be processed and a result determined.

Database: The repository for all persistent information in the system, including user profiles, quiz results and city data.

* 1. Structure and relationships:

1. **Detailed description of components (This is your procedural design.): All** 
   1. **Algorithm**

* **ID:** Algorithm
* **Type:** Python Script
* **Purpose:** To create a list of ideal cities for a user to live in based on their quiz results. Should take no longer than 30 seconds as specified by SRS section 3.3. If a user is logged in, it will save the results to their profile for later viewing, otherwise results can still be generated but no persistent results will be created, as specified by SRS section 3.4
* **Function:** The inputs for the Algorithm are:
  1. Quiz Results: as specified in section 2
     + Pulled either from the User Profile, if user is logged in, or directly handed to the algorithm if the user is NOT logged in.
  2. City Data: as specified in section 2

Outputs:). Raw Quiz Results (if user not logged in).

Algorithm will use a weighted technique, where scores are based on user designated weights of certain attributes and matched against a city’s score in that attribute. User input only requires a Quiz Results object and will return a copy of the object with the resulting cities assigned. All city data is pulled straight from the database in an initial query and is not technically an input.

* **Subordinates / Modules used:** Quiz Results, City Data interfaces
* **Dependencies:** The Algorithm will rely on the Database for pulling the city data necessary to compare user results with prefered living conditions.
* **Resources:** CPU execution time, python math library
* **Processing:**  run\_module(){

Query\_database()

Compute\_results()  
 Return current\_quiz

}

Query\_database(){

Query SQL DB by name and table reference, assign results to internal dict.

Return True/False depending on success

}

Compute\_results(Quiz\_results){

for all attributes in Quiz Results{

Get scores for all cities in attributes listed in Quiz\_results.

Go through all scores, compute new total city score by weight \* city attribute score

}

Return top x number of cities by total score

}

* **Data**: Algorithm does not keep any initial data values. Will all be either handed to the algorithm or queried fresh from the database for each run.
  1. **SQL Database**
* **ID:** SQL\_DB
* **Type:** MySQL Database
* **Purpose:** To house all city and user data for utilization by our website and algorithm. The user’s info should be stored securely.
* **Function:** Data is held in individual tables for city data and user data.
* **Subordinates:** N/A
* **Dependencies:** N/A
* **Resources:** AWS Server running MySQL
* **Processing:** Calls will be made retrieving table data on the basis of quiz specific and user specific needs. The entire table will also be retrieved for our City Index page.
* **Data:** Each table will contain rows that define a city or a user, respectively.
  + City\_Index
    - City\_ID: int, not null, auto-increment
    - City\_Name: varchar
    - Walkability, Transit, Population, Population Density, etc: double
  + Users
    - User\_ID: int, not null, auto-increment
    - User\_name: varchar
    - Password: varchar
    - E-mail: varchar
  1. **User Interface: Quiz**
* **ID:** Quiz
* **Type:** Web Form
* **Purpose:** To gather the necessary information from the user in order to calculate unique recommendations.
* **Function:** The user will fill out each field of the form. The quiz page will then ask the user to save their answers and then redirect them to the results page.
* **Subordinates:** Quiz results
* **Dependencies:** The quiz will depend on the algorithm to generate recommendations, as well as the city database so that the city data of each recommendation can be presented to the user.
* **Resources:** Google forms
* **Processing:** When the form is submitted, the quiz will pass the user’s answers to the algorithm in order to generate city recommendations. It will then wait for a response from the algorithm, and upon receiving one, will display the results to the user.
* **Data:** The user’s answers will be stored in JSON format and passed to the algorithm.
  1. **User Interface: User Profile**
* **ID:** User\_Profile
* **Type:** Web Page
* **Purpose:** To provide the user a method of viewing and editing their account information.
* **Function:** The page will initially display the user’s information as read-only. The user can then selectively edit each section and then save their changes.
* **Subordinates:**
* **Dependencies:** The user profile page will rely on the user database in order to retrieve and update the user’s information.
* **Resources:** Password salting and hashing libraries
* **Processing:** Each piece of user information will initially be read-only. The password will appear as bullets or asterisks. Upon clicking the edit button next to each piece of information, the user will be able to edit that information, and then save their changes.
* **Data:** The only data is the username, password, and email address of the user.
  1. **User Interface: General**
* **ID:** UI
* **Type:** Web page
* **Purpose:** To provide an interface for the user to view and interact with the system**.**
* **Function:** The general functionality of the interface will be to process the user input and perform the appropriate action in response.
* **Subordinates:** o
* **Dependencies:** Dependencies will vary from page to page, but most pages will at least depend on the database being functional.
* **Resources:** All pages will make use of JQuery, as well as other page-specific libraries.
* **Processing:** Each page is only utilized when it is the current active page. From every page the user can access the sidebar menu and navigate to the other main pages of the site.

**6. UI Controller:**

* **ID:** Controller
* **Type:** Python Script
* **Purpose:** The purpose of the UI controller is to handle communication and user input between the UI layer and the model (database / algorithm) layer.
* **Function:** Specifically, this layer handles database queries to retrieve user, quiz, or city data information along with any calls necessary to the algorithm module. It does NOT process anything itself but merely facilitates communication, and queries, returning found objects and errors as necessary.
* **Subordinates:** The controller is pretty self contained and is mostly just a collection of functions in a single script, so no subordinates exist.
* **Dependencies:** As stated previously, the User Interface itself (all modules of it**)** rely on this controller to interface with the model layer. Any user call from the UI that requires data pulled or calculations made will trigger Controller behavior and give an appropriate counter-response.
* **Resources:** CPU Execution time.
* **Processing:** Database querying.Algorithm calls
* **Data:** Utilizes the object: User Profile, Quiz Results, City Data

1. **Interface Design:**   
     
   Home Page:  
   This page will serve as the main hub for the user to access the other pages on the site. It will also display basic information about how to use the service.  
     
   Quiz Page:  
   This page will serve as the main method for the user to not only fill out a quiz, but also modify the answers to a previously completed quiz. When modifying an existing quiz, the questions the user has already answered will appear with their answers already filled in.  
     
   Results Page:  
   This page will display the results of a selected completed quiz. The results will consist of the city that scored the highest according to the user’s answers, followed by a list of the 4 other cities receiving high scores.  
     
   City Data Page:  
   This page will serve as the main method for the user to view and interact with the data about each city in the YCS database. The user will be able to sort the data according to a chosen attribute. They will also be able to filter the data based on certain constraints they choose for certain attributes.  
     
   User Profile Page:  
   This page serves as the main method for a user to view and edit their profile info (username, password, and email).  
     
   Sidebar Menu:  
   The sidebar menu will serve as an easy way for the user to navigate to other pages of the site without having to return to the homepage. The menu will normally be collapsed to the side, however when the user click on the hamburger icon in the top corner, the menu will expand and allow them to select a page to navigate to.