| **Course: WEB422** | **Shopmart Project**  **1 of 6** | | **Contribution:**  **10%** |
| --- | --- | --- | --- |
| **Professor: Kadeem Best** | **Date Given: January 24th, 2022 Date Due: February 6th, 2022** | | |

**Notes for the Student:** This Assignment is designed to give you practical experience in building Restful APIs using Node.js and Express.js

**Background**: You will need to have access to a code editor. You will also need a thorough understanding of JavaScript, OOP, API design, REST, Node.js & Expres.js

# Assignment Submission Requirements

* Your source code and GitHub link must be uploaded to Blackboard
* **Ensure that your GitHub is private.** Add me as a collaborator (username : **kadeembestteaches**) to your remote repo
* You are required to make consistent sensible commits to your Github, showing constant progress. **Severe** **Mark deductions will be applied if you do not have commits that show you progressively building your API.**
* Ensure that your Github repo has a detailed **Readme.md** file that details your application
* A link to your Heroku instance where your API was deployed

# Assignment Regulations

* This assignment must be done individually.
* A virtual “in-person” demonstration of this project is required. **The date of the presentation would be during the week of February 7th 2022**
* **Failure to demo would result in 0.**
* Failure to answer questions regarding foundational concepts about your project and how the said concepts were used within your code would result in 0.
* **Please review Seneca’s policies on Academic Integrity, specifically:**

*“Each student should be aware of the College's policy regarding Cheating and Plagiarism. Seneca's Academic Policy will be strictly enforced. To support academic honesty at Seneca College, all work submitted by students may be reviewed for authenticity and originality, utilizing software tools and third-party services. Please visit the Academic Honesty site on http://library.senecacollege.ca for further information regarding cheating and plagiarism policies and procedures.  
.”* ***Thus, ensure that your code or any part of it is not duplicated by another student(s). This will result in a percentage of zero (0%) assigned to all parties involved.***

# Technical Requirements

* Your RESTful API must be created using Node.js & Express.js.

# Detailed App Specification

Rest-Inn (this is a fictional company), is a new company that allows visitors to find and/or book vacation rentals, cabins, beach houses and unique homes, around the world, specifically for short-term rental. However, presently, this process is done via phone because they don’t have a web presence.

With the world, slowly but surely, returning to some semblance of normalcy and avid travellers eager to travel the globe once again, Rest-Inn CEO has mandated a travel rental booking web application to be built.

Luckily, you have been “contracted” as a Back-End JavaScript Developer to **ONLY** develop a Back-End, in the form of a RESTful API for Rest-Inn. The API must essentially allow hosts to manage their properties, i.e, add, edit & delete their rental properties. Additionally, the API must also allow an administrator to manage Rest-Inn’s customers.

This Assignment is the first of six (6) and will solely focus on the API functionality for the rental booking web application. The other functionality will be spanned over the remaining 5 Assignments. **Note, YOU DO NOT HAVE TO DEVELOP ANY FRONT-END FOR THIS ASSIGNMENT, I.E, ASSIGNMENT 1**

## Framework

Your RESTful API **MUST** be built using Express.js.

## Database

Your Restful API **MUST** be connected to a MongoDB database.

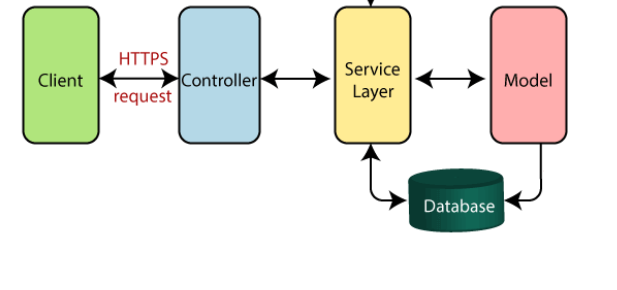
Regarding your database functionality, the following rules must be followed :

1. Setup and configure a MongoDB cloud service using MongoDB Atlas <https://www.mongodb.com/cloud/atlas>.
2. Connect your REST API to the database using an ODM called **Mongoose.**
3. Name your database and collections appropriately. Marks will be deducted if you do not.

.

## Application Architecture

Your RESTful API **MUST** be built in accordance with the MVC (with service layer) design pattern. Thus, you must create Controller files, Service files, and Model files.

* 

## Endpoints

Test all the below endpoints using Postman or any other API Client to ensure that each endpoint is functional and works.

Your API must be designed as follows. Marks will be deducted if your API does not.

1. All communication to and from the API must occur using JSON format.
2. The API expects to receive any request payload data as JSON
3. The API will always respond to a client request with JSON
4. When the API responds to client requests, the API must always return an appropriate status code in its header. You must use the codes per the discussion in class.
5. The name of the endpoints MUST adhere to the conventions discussed in class.

# 

### **None Existing Endpoints**

Requests made to any endpoint that is **not** described in the below documentation should respond with appropriate error code/status message as discussed in class.

### **Customer endpoints**

1. Create an endpoint that will allow a user to register. The below is the data to be added when a customer is created. **Asterisks indicate the fields that are mandatory**
   1. First name (\*),
   2. Last Name(\*),
   3. Email (\*),
   4. password(\*)
   5. Phone numbers - *this should be modelled as an array*

Note, passwords must not be stored in plain text in the database, thus your application must store passwords in an encrypted format. **You can use the Bcrypt Library or any encryption library of your choice.** <https://www.npmjs.com/package/bcryptjs>

This endpoint should also provide validation logic, specifically for requests that do not contain mandatory data.

1. Create an endpoint that retrieves a specific customer by id. This endpoint should also provide validation logic, specifically for requests that do not contain mandatory data.

### **Short Term Property Endpoints**

1. Create an endpoint that will create properties to be added to the database. The below is the data to be added when a property is created. **Asterisks indicate the fields that are mandatory**
   1. Property Title (\*),
   2. Property Rental Price (per night) (\*),
   3. Property Description or Details,
   4. Property Type (\*) (Example : cabins, beach houses, unique homes, condos, apartments,etc)
   5. House Rules (This should be modelled as an array)
   6. Amenities (This should be modelled as an array) (\*),
   7. Location (\*),
   8. A value that will indicate if the property is a bestseller Property (or not) (\*)
   9. Property Photo URL. - For this, you should use the URL for publicly available photos. For example, below is the URL address for a stock photo of an apartment:

<https://unsplash.com/photos/Whgt84a2fSQ>

You are allowed to use any photos from the Internet.

*Before using a photo, you are responsible for ensuring that the website will permit you to use said photo(s) in an academic context. Generally speaking, photos marked as Royalty-Free or Freely-Useable will prevent you from running into intellectual property and copyright issues.*

This endpoint should also provide validation logic, specifically for requests that do not contain mandatory data. Ensure that you return the appropriate ERROR CODE if the requests fail the validation.

1. Create an endpoint that retrieves all the properties in the database.
2. Create an endpoint that retrieves all the properties types in the database.
3. Create an endpoint that retrieves all properties that belong to a specified type.
4. Create an endpoint that retrieves all properties by a particular location.
5. Create an endpoint that will retrieve all properties marked as bestsellers.
6. Create an endpoint that will retrieve a specific property by id. This endpoint should also provide validation logic, specifically for requests that do not contain a valid property id.
7. Create an endpoint that updates an existing property by id. The client should be allowed to update ANY field, except the property id**.** This endpoint should also provide validation logic, specifically for requests that do not contain a valid property id and/or mandatory data. Ensure that you return the appropriate ERROR CODE if the requests fail the validation.
8. Create an endpoint that will delete an existing property by id. This endpoint should also provide validation logic, specifically for requests that do not contain a valid property id.

# Rubric

| **Criteria** | **Not Implemented** | **Fully**  **Implemented** |
| --- | --- | --- |
| **Application Architecture**   * Built-in accordance with MVC | 0 | 4 |
| **Deployment**   * Deployed to Heroku * Used environment variables (where applicable) * Pushed to Github * GitHub contains a well-detailed Readme.md | 0  0  0  0 | 4  2  2  2 |
| **Customer End Points**   * Create A Customer * Encrypting Customer’s Password * Validation for (Creating Customer) * Retrieve A Customer * Retrieve A Customer (Validation) | 0  0  0  0  0  0  0 | 3  2  2  2  3  2 |
| **Property End Point**   * Create A Property * Validation (Create A Property) * Retrieve All Properties * Retrieve All Property Types * Retrieve Properties that Belong To A Specific Type * Retrieve Properties That Are Marked As Bestsellers * Retrieve A Specific Property * Validation * Update A Specific Property Validation * Delete A Specific Property * Validation * Retrieve Properties that Belong To A Location | 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 4  2  2  3  3  3  2  2  2  2  2  2  2 |
| **Non Existing Endpoints** | 0 | 3 |

Total Marks: 62 marks

# THE END