ScavengeRUs Crash Course Document

ScavengeRUs Application

The ScavengeRUs application is a MVC (Model-View-Controller) application that implements an SQLite database to keep records of hunts, users, locations (Tasks), and the connection table between hunts and locations. Giving the application is functional in several aspects, this document should help with understanding how the application works, finding the code to manipulate the application in the way desired, and overall speed up the process of completing the application!

HOMEPAGE

When first launching the ScavengeRUs application, you will be taken to the homepage. The main focus of this page is logging in. In the upper right-hand corner, there is a login button, but also in the front of the page, is a login through an access code.

Controller:

HomeController.cs

View:

Index.cshtml

LOGIN WITH EMAIL & PASSWORD

Logging in this way requires a username and password that must be created by an admin.

Generic Admin Account Credentials:

Username: waltonca@etsu.edu

Password: YMXH@9J!72kM6Em

Generic User Account Credentials:

Username: thomas.foreman17@gmail.com

Password: Etsupass12!

Controller:

HomeController.cs

View:

Home/LogIn.cshtml

LOGIN WITH ACCESS CODE

Logging in this way requires a specific access code. These codes are sent out via email and text to the users assigned to that specific hunt. This generic access code below will sign you in to a basic account with user privileges and take you to the hunt assigned to this access code.

Generic Access Code:

4239006885/hunt_test

Controller:

HomeController.cs

View:

Home/Index.cshtml

USER IN A HUNT

Once entering an access code, the application will take you to that specific hunt page. This page shows you one of the tasks needed for that hunt. (This does not show all of the unfinished tasks or any completed tasks due to our recent update). There is an arrow button on the far right of the screen that will open a sidebar showing additional stats (Some work, some don't). On this page you can do one of two things, logout, or complete a task.

Controller:

HuntController.cs

View:

Hunt/ViewTasks.cshtml

Services:

HuntRepository

IHuntRepository

ADMIN PORTAL

Once logging in to an admin account, the “Admin Portal” link will appear next to the “Logout”. This link lands you on the Active Users page which shows you every user in the database along with being able to “Edit”, “Delete”, and view “Details” for each of them. This page also has three links that allow you to “Create New User”, “Batch Create Users”, and “Manage Hunts/Tasks”

Controller:

UserController.cs

View:

User/Manage.cshtml

Services:

IUserRepository.cs

UserRepository.cs

Users.csv (For creation of batch users)

CREATE NEW USER

Pretty simple here, just a page to create a new user.

Controller:

UserController.cs

View:

User/Create.cshtml

Services:

IUserRepository.cs

UserRepository.cs

BATCH CREATE USERS

This is not a page, rather a function that creates a list of users that are in the Users.csv file

Controller:

UserController.cs

View:

User/Create.cshtml

Services:

IUserRepository.cs

UserRepository.cs

Users.csv

MANAGE HUNTS/TASKS

After clicking this link, it takes to you the “Displaying all hunts” page. Here we can Edit, View, and Delete each hunt in the database. Also, at the top of the page, we can “Create New Hunt”, “Manage All Tasks”, and go “Back to Admin Portal”.

Controller:

HuntController.cs

Model:

Hunt.cs

HuntLocation.cs (Class that links locations/tasks to a hunt)

View:

Hunt/Index.cshtml

Services:

IHuntRepository.cs

HuntRepository.cs

CREATE NEW HUNT

Page to create a new hunt.

Controller:

HuntController.cs

Model:

Hunt.cs

View:

Hunt/Create.cshtml

Services:

IHuntRepository.cs

HuntRepository.cs

MANAGE ALL TASKS

After clicking this link, it takes to you the “All Tasks” page. Here we can Edit, View, and Delete each Tasks in the database. Also, at the top of the page, we can “Create New Task”, and go “Back to Hunts”. **NOTE: View Hunt also is where you can manage several things within the hunt including managing the tasks and players of the hunt. More info in “View Hunt”.

Controller:

LocationsController.cs

Model:

Location.cs

View:

Locations/Index.cshtml

CREATE NEW TASKS

Page to create a new location/task for a hunt. **Note: When inserting an image, the image must be stored in the wwwroot/images folder.

Controller:

LocationsController.cs

Model:

Location.cs

View:

Locations/Create.cshtml

VIEW HUNT

This page is where you can view the hunt but also manage the tasks and users involved in the hunt. Here we see a list of tasks in the hunt currently. At the top of the page we have "Manage Tasks", "View Players", and go back to "View All Hunts".

Controller:

HuntController.cs

Model:

Hunt.cs

HuntLocation.cs

View:

Hunt/ViewTasks.cshtml

Services:

IHuntRepository.cs

HuntRepository.cs

MANAGE TASK via View Hunt

This page is similar to Manage All Tasks. In this instead of editing the tasks, you assign or remove a tasks from a hunt. This also has the link to “Create a new task”.

Controller:

HuntController.cs

Model:

Hunt.cs

HuntLocation.cs

View:

Hunt/ManageTasks.cshtml

Services:

IHuntRepository.cs

HuntRepository.cs

VIEW PLAYERS via View Hunt

This page is shows you the users that have access to the hunt. Here we can “Revoke Access” along with “Add a user to this Hunt”.

Controller:

HuntController.cs

Model:

Hunt.cs

ApplicationUser.cs

View:

Hunt/ViewPlayers.cshtml

Services:

IHuntRepository.cs

HuntRepository.cs

REVOKE ACCESS

Page that confirms if you want to remove the user from the hunt.

Controller:

HuntController.cs

Model:

Hunt.cs

ApplicationUser.cs

View:

Hunt/RemoveUser.cshtml

Services:

IHuntRepository.cs

HuntRepository.cs

ADD A USER TO THIS HUNT

This page is allows you to add a user to the hunt. If this user doesn't exist, it will create the user and also send the invite.

Controller:

HuntController.cs

Model:

Hunt.cs

ApplicationUser.cs

View:

Hunt/ViewPlayers.cshtml

Services:

IHuntRepository.cs

HuntRepository.cs

ScavnegeRUs Database

AccessCodes

Table that houses the access codes and the Id of the hunt they are assigned to.

Field: **Code**

The access code for users to participate in a hunt.

Field: **HuntId**

The ID of the hunt that the Access Code is matched to.

Hunts

This table contains all the information for the hunts.

Field: **EndDate**

When the hunt is to end.

Field: **HuntName**

The name of the hunt.

Field: **InvitationText**

Header for the invitation.

Field: **StartDate**

When the hunt is to be started.

Field: **Theme**

The theme of the hunt.

Field: **CreationDate**

When the hunt was created.

Field: **InvitationBodyText**

The body of the invitation text.

Location

This table houses all the information of the location/tasks.

Field: **AccessCode**

This field seems to be obsolete as this info is stored in the AccessCode table.

Field: **Answer**

The answer for the task

Field: **ApplicationUserId**

The Application User ID.

Field: **Lat**

The latitude of the location and task.

Field: **Lon**

The longitude of the location and task.

Field: **Place**

The name of the location where the task is.

Field: **QRCode**

This field is currently not in use. Plans are for the ability to scan QR code for the hunt.

Field: **Task**

This is the task itself. A question that is specific to the location its involved with.

HuntLocation

This table connects multiple locations/tasks to each hunt.

Field: **HuntId**

HuntId Field

Field: **LocationId**

LocationId Field

AspNetUsers

This table houses all the information of and about the users in the database.

AspNetUserRoles

This keeps track of which users have the admin role vs the user role

AspNetRoles

This table houses the two roles of the application, Player and Admin.