## Credentials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Email Address** | **Username/UserId** | **Password (Encrypt)** | **Purpose** | **Is Active** |
| dribbly.dev@google.com |  | dribbly | For development purposes   * Google Maps Api | Y |
| dribbly.dev@google.com | devdribbly | smarterasp | Test Web Client (<https://member3-1.smarterasp.net/account/loginform?go=1>) | N |
| dribbly.test1@gmail.com | test1dribbly | dribbly |  | Y |

## Keys

|  |  |
| --- | --- |
| Google Map API Key | AIzaSyCQwPkj7HcSjORBr6z8ZGf56e4uXNPHUuY |

## Terms and Definitions

|  |  |  |
| --- | --- | --- |
| Term | Definition | References |
| OWIN | An interface that breaks the tight coupling between web servers (such as IIS) and .Net web applications. It allows .Net web application to be hosted on any OWIN-compliant server (not just IIS) |  |
| Asp.Net Identity | Authentication system for ASP.NET applications. It implements authentication not just using Username and Password, but also Facebook and Google authentication and some more advance features | <https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity> |
| OAuth | An authorization pattern which uses tokens instead of Username/Password combination | <https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth> |

## Setup

### Tools

* Node JS v12.x.x

## Source Code

### Projects

#### Dribbly.Authentication

##### Dependencies:

* Entity Framework 6.4.0

#### Dribbly.Service

* Entity Framework 6.4.0

##### Repositories

* Should be able to access only 1 entity of the DbContext

##### Services

* Should contain the main logic for each Web API action method
* Multiple Repositories can be injected
* No other service should be injected except for the common service

#### DribblyWebApi

Project Type: Asp.Net Core Web API

Reference for adding Identity

## Testing Environments

|  |  |  |
| --- | --- | --- |
| Project | Web Client | Web API |
| URL | <http://devdribbly-001-site1.gtempurl.com/> | http://dribbly001.somee.com/ |
| CPanel URL | <https://member3-1.smarterasp.net/account/account_screen> | <https://somee.com/DOKA/DOU/DOUserDefault.aspx> |
| Email Used | [dribbly.dev@google.com](mailto:dribbly.dev@google.com) | rjpablo16@yahoo.com |
| Username | Devdribbly-001 | rjpablo112 |
| Password | Encrypt(devdribbly) | Encrypt(somee) |
| ConnStr |  | workstation id=dribblyAuth.mssql.somee.com;packet size=4096;user id=rjpablo112\_SQLLogin\_1;pwd=nse64gv37m;data source=dribblyAuth.mssql.somee.com;persist security info=False;initial catalog=dribblyAuth |

Web Client

|  |  |
| --- | --- |
| Hosting Account ID： | test1dribbly-001 |
| First Website Root Folder Name： | dribblyweb |
| Hosting Plan Name： | W60-US |
|  |  |
|  | *Please Record The Following FTP Infomation:* |
| FTP Address ： | FTP.SITE4NOW.NET |
| FTP Username ： | test1dribbly-001 |
| FTP Password ： | (same as your control panel password) |
| FTP Directory | \ |

## Creating the project

### Tools used:

* Visual Studio 2017

### Steps:

1. Open Visual Studio
2. Under **File** menu, select **New** > **Project**
3. Copy the options below, then hit OK.



1. On the next screen select the options below, then hit OK.

## Requirements for Phase 1

* Landing page
* Global Search – Search court or user
* Main Navigator
* Authentication
  + ~~Registration~~
  + ~~Login~~
  + ~~Reset~~ Password
  + ~~Change Password~~
* Account Maintenance
  + View Account Details
    - ~~Functionality~~
      * ~~Username~~
      * ~~Member Since~~
      * ~~Sex~~
      * ~~Location~~
    - Design (WEB-0089)
  + Account Settings
    - ~~Update password~~
    - Update Email (WEB-0077)
    - Preferences
      * Profile Settings
        + Show birth date
        + Public/Private
      * Notification Settings
        + Send to email
        + Filter notifications
      * Messaging Settings
        + Only friends can send
    - Deactivate/Re-activate Account (WEB-0090)
    - Remove Account (WEB-0091)
  + Photos (WEB-0092)
  + List User Activities
  + View Owned Courts (WEB-0093)
  + List of Games
  + Court Bookings
    - View list of bookings (WEB-0097)
    - View booking details
      * Display booking details
        + Schedule
        + Court
        + Booked by
        + Status (Approved/Pending Approval/)
      * Update booking status (done by court owner)
      * Cancel booking
* Access/Permission (WEB-0094)
* Error logging (WEB-0095)
* Courts
  + ~~Registration~~
  + Photos
    - ~~Uploading~~
    - Viewing
    - Photo options
      * ~~Delete~~
      * Use as primary Photo (WEB-0096)
      * Make Private
  + Details Viewing
    - View location on map
  + ~~Updating of details~~
  + ~~Searching~~
  + Games
    - Booking
      * ~~functionality~~
      * design
    - ~~Viewing~~
  + Follow
* Notifications
* Messaging
* Player profiles
  + Creation
  + Search
* Team Management
* Games management
  + Add Game
  + Update Game Details
  + Search Games

## App Structure

## Types of Setting

### Account Settings

* Settings that are specific to Accounts
* Set by the owner of the account

Fields

* Account ID (from account table, not aspNetUsers)
* Setting\_Id (Id column from Settings table)

Items

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Description** | **Allowed Values** | **Default Value** |
| show\_birthdate | Show birth date on user profile | Y/N | N |
| keep\_profile\_private | Keep the user’s profile private. If set to Y, only limited information is visible on the user’s profile | Y/N | N |
|  |  |  |  |

### System Settings

* Settings for controlling the website
* Set by Admin

Fields

* Id
* Key
* DescriptionKey
* IsActive
* Value

### User Preferences

* User-specific settings
* Set by users

### Settings

Fields

* Id
* Key
* Description
* Default Value

## Types of Pages

### Search Page

Page used for searching players, courts, games, etc

### Browser Page

Page used for viewing player details, court details, game details

## Rules

### DateTime Objects

* Always expect dates from Web API in **UTC** timezone and in **ISO** format (ie. 2020-03-01T17:59:23Z). If value from WEB API does not have ‘Z’ suffix, add right upon receiving it.

## Conventions

### Dribbly

* Prefix components, directives, filters with ‘drbbly’
* Add a newline character at the end of every files if allowed

### Git

Branch name prefix

|  |  |
| --- | --- |
| **Prefix** | **Purpose** |
| feat | For new components/functionalities |
| improv | For improvement of existing components/functionalities |
| fix | For fixing error in existing components/functionalities |
| setup | For tasks that are not directly related to the app (e.g. compiler setup, publish profile changes, git setup updates) |
| clean | For removing unused code/files |
| docu | For adding/updating comments |
| design | For prototype design changes (usually photoshop files) |
| refactor | For changes in code that do not affect any functionalities (e.g. renaming variables, making code more organized) |

## Guides

### Using modals

* Modals must always invoke context.dismiss when closing. Each modal instance subscribes to navigation events. Invoking context.dismiss assures that we unsubscribe from these events before disposing the modal.
* Make sure to do validation (e.g. check unsaved changes), if needed, before letting the modal close. After making sure that it’s safe to close the modal, set context.okToClose to true before invoking context.dismiss
* Subscribe to modal.closing event to handle closing initiated by navigation, cliking the overlay or the Esc button, and check for the value of context.okToClose. A value of true indicates that it is ok to close the modal without doing validations. context.okToClose is set to true when a navigation occurs with the custom option `force` set to true. Validations should be done when context.okToClose is false.

N

Y

modal.closing event occurs

okToClose?

Y

N

Check for unsaved changes

User still wants to close?

set okToClose to Y

invoke context.dismiss

Modal closes

Keep modal

User clicks Close/Cancel

Navigation occurs / User presses Esc / backdrp is clicked

### Adding new NPM dependencies

* Install npm module
* Update the **noteLibs** array in gulpfile.js (**Note:** reference required files only)
* Add reference the files in \_SiteLayout.cshtml
* Inject the module into siteModule

### Adding new component files

* Add reference the files in \_SiteLayout.cshtml

## Pages

### Login

### Signup

### Forgot Passwordsky

### Courts List

Features:

* Viewing of list of courts
* Filtering list of courts
* Registering a court
* Searching for courts

### Account

#### Account Details

#### Account Settings

### Court

#### Details

#### Photos

#### Games

#### Schedule

### Game

#### Details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Admin | Game Creator | Participant | Guest |  |
| Change Status | Y | Y |  |  |  |
| Edit Details | Y | Y |  |  |  |
| Join/Leave | Y | Y | Y | Y |  |
| Follow/Unfollow | Y |  | Y | Y |  |

## Modules

### SiteModule

* The “foundation” module
* Should only contain components that will be used both by **authModule** and the **mainModules**
* All 3rd party modules except those that are only for a specific mainModule should be injected here

### AuthModule

* Should only contain services and components used for authenticating user

### AppModule

* Should only contain components and services that are shared among the **mainModules**

### MainModule

* Should only contain components and services for itself
* **Dribbly** is currently the only mainModule. Dribbly is focused in Basketball. We may add other mainModule for other sports and/or a **store** later

## Modules Hierarchy

Site Module

Auth Module

App Module

Store Module

Main Module

## Functionality

### Games

#### Starting a Game

Rules:

* A game can only be started if its current status is Waiting To Start
* Only the user who created the game can start it
* A user activity log should be created when a user starts a game
* Game status should change to “Started”
* The “End Game” option should become available

## Publishing

### Local IIS Server

#### Publishing

Pre-requisites:

* Make sure .NET Core Hosting Bundle is installed on the machine (This is required for .NET Core applications to run on local IIS Server)
* Open IIS Manager



* Add Website



* Click OK to create the website
* Run Visual Studio as admin, then open the solution
* In Visual Studio, right-click on the project then select Publish
* Click on New Profile…
* Select “IIS, FTP, etc”
* click on “Create Profile”



* Click Next
* For now, set Configuration to Debug, until the website is configured for release (minify files, etc.)



* Click Save.

Helpful article: <https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/index?tabs=aspnetcore2x&view=aspnetcore-2.2>

#### Accessing local IIS Instance from another machine via IP address

1. Add Website a binding

* Open IIS Manager, select the website, and then click on “Bindings”



* Click “Add”
* Set type to “http”, select the IP from the dropdown, then enter the port. Leave Host Name empty. (Note: IPs will only show in the dropdown if the network is already connected to the network. Otherwise, type in the IP address. We can also enter \* to allow it to be accessed whatever the current IP address) Click OK



1. Add Firewall rule

* Open Firewall advanced settings window
* Select “Inbound Rules” then click on “New Rule…”



* Select “Port”, then click Next



* Select TCP, then enter the same port number used in the previous step. Click Next



* Select “Allow Connection”. Click Next



* Check all options. Click next



* Enter a name for the rule, then click Finish



* Any device on the same network should now be able to access the website using the IP address and port (example: http//:192.168.43.179:9010)

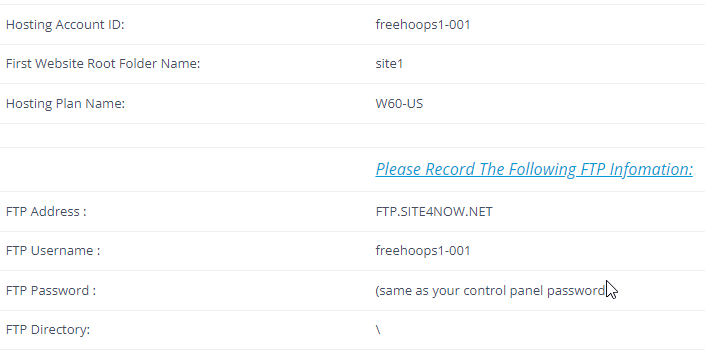
### Test Server (somee.com)

|  |  |
| --- | --- |
| Email address used | dev.[dribbly@gmail.com](mailto:dribbly@gmail.com) |
| Credentials | freehoops1/encrypt(password) |
| Site name | <http://rjpablo112-001-site1.dtempurl.com/> |
| FTP Addresses/path | [FTP.SITE4NOW.NET/drbbly-web](ftp://FTP.SITE4NOW.NET/drbbly-web) |
| FTP User Credential | rjpablo112-001/encrypt(myasp) |



### MyWindowsHosting

|  |  |
| --- | --- |
| Username | freehoops1 |
| Email | dev.[dribbly@gmail.com](mailto:dribbly@gmail.com) |
| Password | enypt(password) |
| File Management page | <https://member3.mywindowshosting.com/cp/filemanager?pop=1&d=h%3A%5Croot%5Chome%5Cdribbly%2D001%5Cwww%5Cdribbly%2Dtest> |
| Hosting Account ID | freehoops1-001 |



## Random Notes

Max width for mobile design is **575px** (bootstrap’s breakpoint for sm (small) screens)

## App Flow

Under Courts show limited number of Featured Courts, New Courts, etc. A “more…” link at the bottom of each category would load more items for that category or show all the items in a modal with pagination

## To Test

|  |  |  |
| --- | --- | --- |
| Scenario | Expected Result | Actual Result |
| Join an active team, of which you are not currently a member | Show be able to join successfully |  |
| Join an inactive team | Show appropriate error message |  |
| Join a team that has been deleted | Show appropriate error |  |
| Join an active team, of which you are currently a member | Show appropriate error |  |

## References

* Password Reset <https://code-maze.com/password-reset-aspnet-core-identity/>

## Ideas

* Each player should have the option to vote for one mvp. The player who get the most mvp votes becomes the Game MVP. All MVP votes received by a player will reflect on his profile, in addition to how many times he became the Game MVP. Players who selects an MVP will receive points as reward
* Upon publish, register public basketball courts as initial Featured content
* In Store, add a separate category for uniform makers