Notes for final project

# To-do:

Due 10am 1/26

1. Add isActive bool to profileModel, itemsModel, authUsersModel
2. Update seeds to reflect isActive = true
3. Add seed for second profile
4. Add seed for second user’s items (John Doe)
5. Add-migration; update-database

Morning 1/26 (through to 1pm)

1. Modify the repositories to reflect:
   1. Generic repository
   2. Interface repository
   3. Correct methods
   4. Add a get profile by user ID to the AccountProfileRepository (& update interface?)
2. Modify the API controllers to reflect:
   1. Repository changes
      1. AccountProfileController
      2. ItemsModelsController
   2. Change AccountprofileController to ProfileController
   3. Authorize on appropriate methods

Afternoon 1/26 (2pm through to 6pm)

1. Modify the methods called in the controllers & services to reflect back end changes (expand on this)
   1. Methods: CRUD for profile
      1. UI Changes:
         1. Reflect the correct fields
         2. Remove auth users
         3. Remove log-in
      2. Read profiles by login
         1. Remember that one login can have multiple profiles
      3. Create profile – will need to associate login on server side
      4. Update profiles by login
      5. Delete profile
         1. Use isActive bool to “soft delete” profile
         2. Also delete related items (did hard delete, but soft delete is possible)
   2. Methods: CRUD for items
      1. Read items by User – DONE
      2. Create item - DONE
      3. Update Item
         1. Ultimately this may also include a method using the AuthUser Id to limit it to just one auth users items (selected user)
      4. Delete Item
         1. Use isActive bool to “soft delete” profile

Evening 1/26 (7pm through to 10pm)

1. Modify the new log-in, register and not found html pages to “prettify” them
2. Modify the Index.cshtml file nav bar to:
   1. Show/hide Items & Manage Profile Page
   2. Add nav bar items for log-in and register pages
3. Check & fix itemModal photo so that photo appears
4. If time: on account page add a href to google maps

1/27 (7.30am – 1pm, 2pm – 6pm)

1. Complete/fix all CRUD
2. Fix all errors

1/27 (7pm to 10pm)

1. Tidy up – remove unused sections that won’t be part of future work
2. Study
3. IF time, consider adding additional minor functionality
   1. If logged in, add current location of first profile in user to google maps
   2. Add ability to filter items by selected profile
   3. Add ability to search items
   4. Add paging for items

# Current issues:

When editing a photo, unable to add/edit photo – item is undefined

After creating an item, client-side validation is triggered on item creation fields

Selected profile isn’t populated into dropdown when editing item

Coder Camps Main Camp Final Project

# Planning document

## Specs

* User should be able to navigate page to page using nav bar pills and appropriate links in text
* Carousel on landing page
* Interactive map with user home address on about us page
  + If not logged in, show default center of US
* Profile page
  + User should be able to create an account and, once logged in, create multiple authorized users on that account
  + User should be able to edit or delete account
  + When logged in, user should be able to see their information
  + If time: user should be able to select authorized users from a drop down & edit/delete users
  + TODO – create log-in form
* Items page
  + If logged in, user should be able to create an item
  + If logged in, user should be able to list items (20 at a time + paging, if time)
  + If logged in, user should be able to click on an item for details
    - Launches ItemModal
    - Lists details + photo
    - Edit or delete item from modal
    - Close button

## User Interface – Views

Index

* Located in:
  + /Views/Home/
* Contains:
  + Site header
  + Navbar
  + Ng-app = “WMS”
  + Ng-view

Main

* Located in:
  + ngApp/Views/
* Controller (routing):
  + MainController
* Contains:
  + Carousel
  + Jumbotron
    - Intro & button to about page
  + Informational blocks
    - Profile blurb & button to profile page
    - Stuff blurb & button to items page
    - About blurb & button to about page

About

* Located in:
  + ngApp/Views/
* Controller (routing):
  + AboutController
* Contains:
  + Jumbotron
    - Info & button to profile page

Profile

* Located in
  + ngApp/Views/
* Controller (routing):
  + ProfileController
  + Needs:
    - public user
    - public authUser
    - login(account)
    - createAccount(account)
    - editAccount() (enables fields only)
    - saveAccount(account)
    - deleteAccount(account)
    - createUser(user)
* Contains:
  + Log-in form
    - Form name = loginform; ng-submit = “wms.login(account)”
    - Input: Username
      * Ng-model = “wms.user.username”
      * Validation = email+required
      * Input name = lUName
    - Input: Password
      * Ng-model = “wms.user.password”
      * Validation – required
      * Input name = lPwd
    - Input: submit
  + Account creation form
    - Form name = pform; ng-submit = “wms.createAccount(account)”
    - Input: Name
    - Input: Email/Username
      * Ng-model = “wms.user.username”
      * Validation – email + required
      * Input name = uName
    - Input: Password
      * Ng-model = “wms.user.password”
      * Validation – required
      * Input name = pwd
    - Input: Phone
      * Ng-model = “wms.user.phone”
    - Input: Street
      * Ng-model = “wms.user.street”
    - Input: City
      * Ng-model = “wms.user.city”
    - Input: State
      * Ng-model = “wms.user.state”
    - Input: Zip
      * Ng-model = “wms.user.zip”
    - Input: Create Account!
      * Type: submit
    - Input: Edit Profile
      * Type: button
      * Ng-click = “wms.editAccount()”
      * Enables fields for editing
    - Input: Save Changes
      * Type: button
      * Ng-click = “wms.saveAccount(account)”
      * Saves changes to account
    - Input: Delete Account
      * Type: button
      * Ng-click = “wms.deleteAccount(account)”
      * Launches confirmation modal and deletes account, all associated users and all associated items
  + Authorized User addition form
    - Form name = authUserForm; ng-submit = “wms.createUser(user)”
    - Input: Name
      * Ng-model = “wms.authUser.auName”
      * Validation – required
      * Input name = aUName
    - Input: Email
      * Ng-model = “wms.authUser.email”
      * Validation – email + required
      * Input name = aUEmail
    - Input: Add User
      * Type: submit

Items

* Located in
  + ngApp/Views/
* Controller (routing):
  + ItemsController
  + Needs:
    - All written – needs testing
    - createItem(item)
      * method to add item stored in “item”
    - displayItems()
      * method to retrieve list of items, add to “items” property
    - showModal(item)
      * opens the modal with the options specified
    - public item (DB field names must match the properties in the view itemName, itemDesc, itemLocation, itemLabel, itemOwner [matches auth user], Photo)
      * object with properties
      * holds item
    - public items
      * array of item objects
      * holds list of items retrieved
* Contains:
  + DONE
  + Item creation form
  + Form name = itemForm; ng-submit = “wms.createItem(item)”
    - Input: Item Name
      * Ng-model = “wms.item.itemName”
      * Validation – required
      * Input name = itemName
    - Input: Item Description
      * Ng-model = “wms.item.itemDesc”
      * Input name = itemDesc
    - Input: Location
      * Ng-model = “wms.item.itemLocation”
      * Validation – required
      * Input name = itemLocation
    - Input: Item Label
      * Ng-model = “wms.item.itemLabel”
      * Input name = itemLabel
    - Input: Item Owner
      * Ng-model = “wms.item.itemOwnder”
      * Validation – required
      * Input name = itemOwner
    - Input: Item Photo
      * Ng-model = “wms.item.Photo”
      * Input name = itemPhoto
    - NEEDS TESTING
    - Input: Create Item button
    - Input: List Items button
      * Ng-click = “displayItems()”
  + Item list
    - NEEDS TESTING
    - Table with ng-repeat tds
    - Each td should be clickable & launch the details modal

ItemModal

* Located in
  + ngApp/Views/
* Controller (routing):
  + ItemModalController
  + Needs:
    - All done NEEDS TESTING
    - editItem()
      * method to save item stored in “item”
    - deleteItem()
      * method to retrieve list of items, add to “items” property
    - closeModal()
      * opens the modal with the options specified
    - public item (DB field names must match the properties in the view itemName, itemDesc, itemLocation, itemLabel, itemOwner [matches auth user], Photo)
      * object with properties
      * holds item
* Contains:
  + DONE
  + Item creation form
  + Form name = detailsForm
    - Input: Item Name
      * Ng-model = “wms.item.itemName”
      * Validation – required
      * Input name = itemName
    - Input: Item Description
      * Ng-model = “wms.item.itemDesc”
      * Input name = itemDesc
    - Input: Location
      * Ng-model = “wms.item.itemLocation”
      * Validation – required
      * Input name = itemLocation
    - Input: Item Label
      * Ng-model = “wms.item.itemLabel”
      * Input name = itemLabel
    - Input: Item Owner
      * Ng-model = “wms.item.itemOwner”
      * Validation – required
      * Input name = itemOwner
    - Input: Item Photo
      * Ng-model = “wms.item.Photo”
      * Input name = itemPhoto
    - Input: Edit Item button
      * Ng-click = “wms.editItem()”
    - Input: Delete Item button
      * Ng-click = “wms.deleteItem()”
    - Input: Close modal button
      * Ng-click = “wms.closeModal()

## User Interface – css

UI-Bootstrap-csp

Bootswatch

Site.css

* Added background image to body tag
* Added pale grey background to major row in views
* Added .ng-invalid/.ng-touched yellow background
* Added .error color of darkred

## Controllers

Main

About

Profile

* public user
* public authUser
* login(account)
* createAccount(account)
* editAccount() (enables fields only)
* saveAccount(account)
* deleteAccount(account)
* createUser(user)

Items

ItemModal

## Application Scripts

App

CRUDServices