Randi Seney¶

SWDV 691¶

Capstone¶

Design Service Layer¶

Overview¶

→ This application uses Angular to execute the service layer.¶¶

Authentication Service¶

Register ¶

Purpose: User must register account in order to access and use application¶. Successful registrations are routed to home page. Users with failed attempts receive a pop up message notifying them of registration failure and to please try again.¶

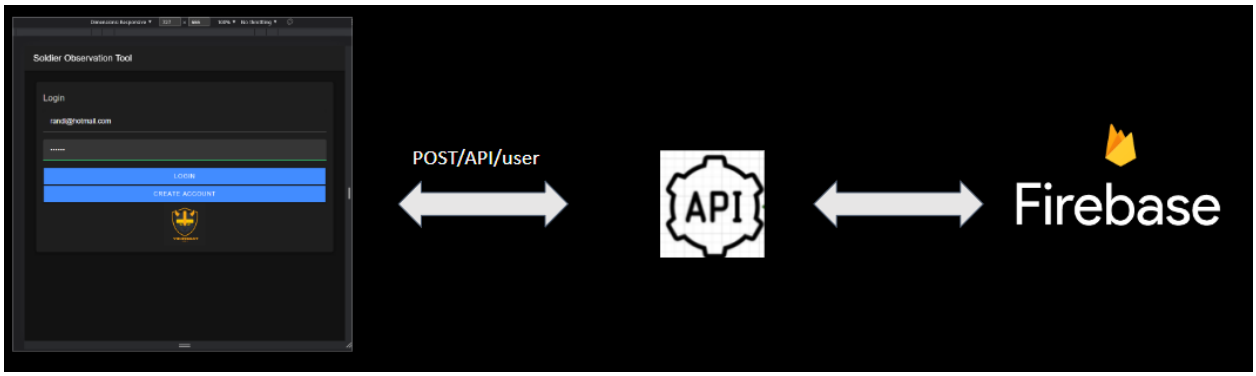
URL: http://localhost:8100/home

Success Response ¶

```
post:
    tags:
      - user
    summary: Create user
    description: This can only be done by the logged in user.
    operationId: createUser
    responses:
      default:
        description: successful operation
    requestBody:
      content:
        application/json:
          schema:
          User:
          type: object
          properties:
            id: "abcdef"
            email_username: "randi@gmail.com"
            password: *****
            profilePic_file: image_file
            size: "1.4"
            updated: "1400 17JUL 2022"
      description: Created user object
      required: true
```

Failed Response

```
responses:
'400':
  description: Login failed, please try again.
```



Login¶

Purpose: User who have previously registered must log in with their credentials. Users who successfully login will be routed to the home page. Failed logins are notified of failed attempt and then directed to try again.¶

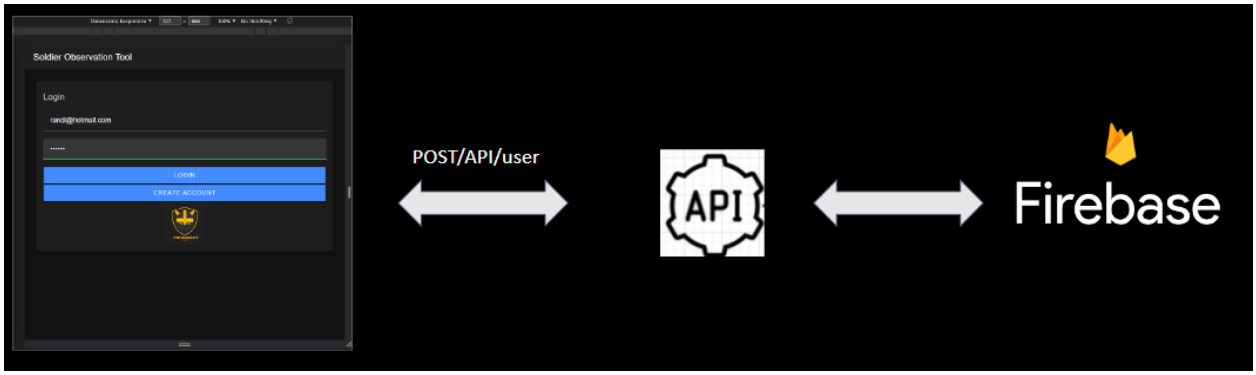
URL: http://localhost:8100/home

```
get:
  tags:
    - user
  summary: Logs user into the system
  operationId: loginUser
  parameters:
    - name: "randi@gmail.com"
      in: query
      description: The user name for login
      required: true
    - name: "*****"
      in: query
      description: The password for login in clear text
      required: true
      schema:
        type: string
```

¶

Failed Response

```
responses:
'400':
  description: Login failed, please try again.
```



Avatar Service¶

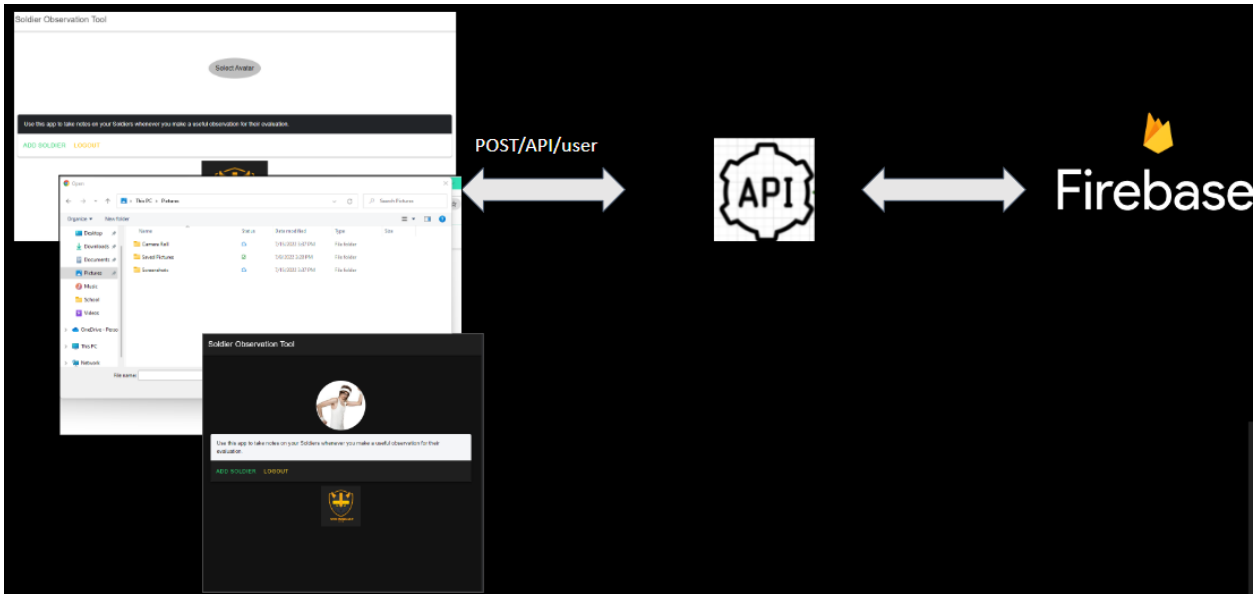
changeImage¶

Purpose: Users can upload an image to serve as their profile picture. If the image uploads successfully then the profile image will update. If the upload fails users receive a message notifying the user that there was a problem uploading the avatar.This method ¶

URL: http://localhost:8100/home

```
post:
tags:
summary: uploads an image
operationId: uploadFile
parameters:
  - name: "randi@gmail.com"
    in: path
    description: ID of user to update
    required: true
```

¶



Data Service¶

addNote¶

Purpose: This is how a user adds a note. After clicking a floating button, the user enters the Soldier's name, rank, email, and observations. Users have the option to cancel or add the note.¶

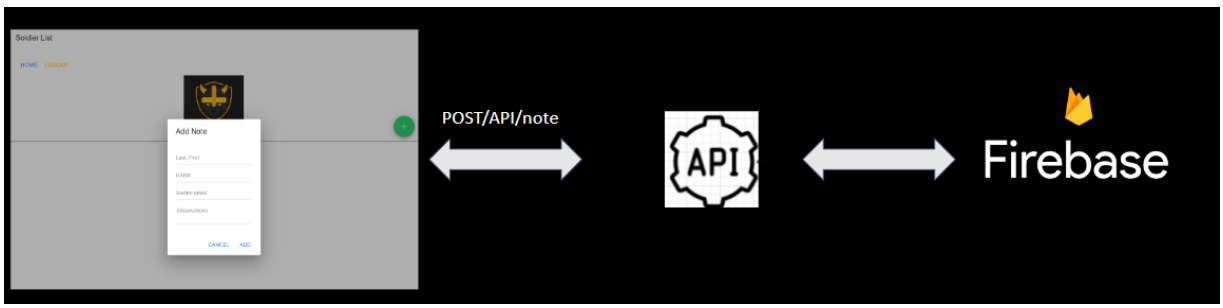
URL: http://localhost:8100/add-note

```
post:

Note:
type: object
required:
  - name
  - observation
properties:
  id:
    type: string

  name: "Smith John"
  soldier_email: "john.smith@gmail.com"
  rank:  "SFC"
  observation: "Soldier physically fit and took charge"
```

¶



updateNote

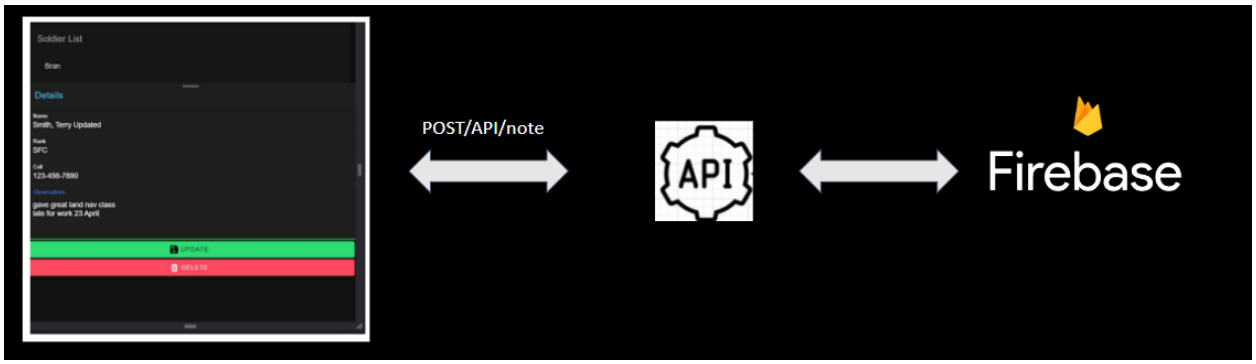
Purpose: Users can update existing notes. Once update is complete Users receive a notice that the note was updated.

URL: http://localhost:8100/add-note

```
async updateNote(){
  await this.dataService.updateNote(this.note);
  const toast = await this.toastCtrl.create({
    message: 'Note updated!',
    duration: 2000
  })
  toast.present();
```

¶

```
    responses:
      '200':
        description: Note updated!
        content:
          application/json:
```



deleteNote

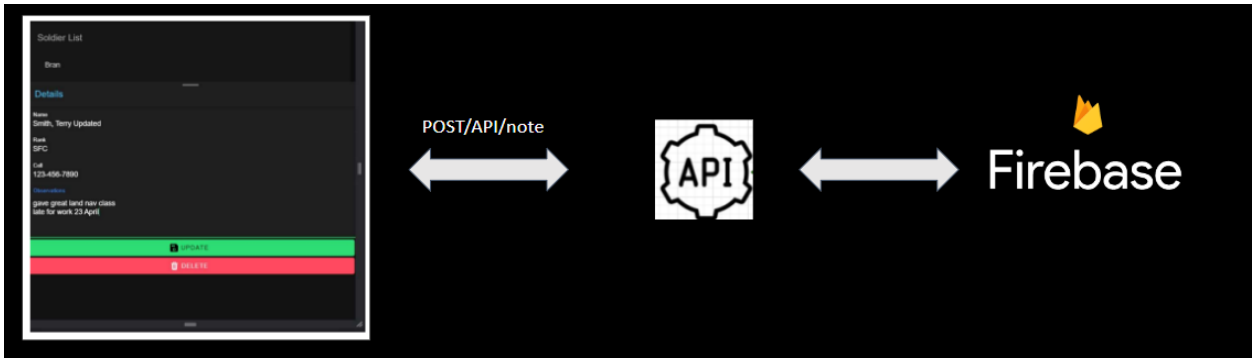
Purpose: Users can delete existing Soldier notes.

URL: http://localhost:8100/add-note

¶

```
async deleteNote(){
  await this.dataService.deleteNote(this.note)
  this.modalCtrl.dismiss();
}
```

¶



sendNote - Stretch

Purpose: Users can send Soldier notes to the Soldier email

URL: http://localhost:8100/add-note

```
GET:

Note:
type: object
required:
  - name
  - observation
properties:
  id:
    type: string

  name: "Smith John"
  soldier_email: "john.smith@gmail.com"
    observation: "Soldier physically fit and took charge"
```