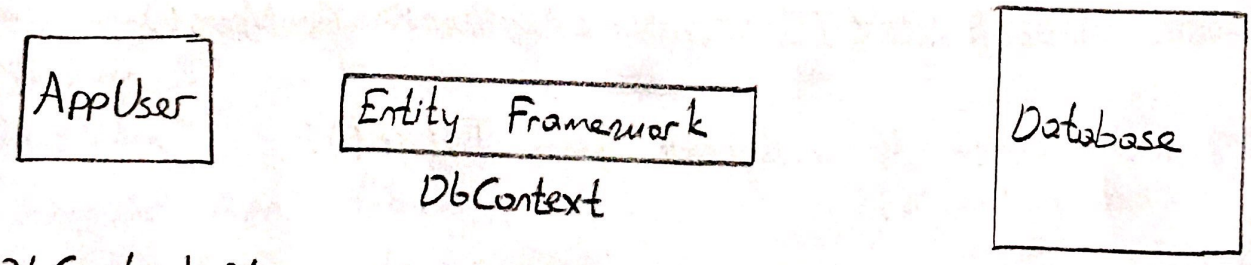
- ASPNET Core and Angular -

[MS 30.07.2021]

Section 2: Building Walking Skeleton Part One - API

- ls , lists folders
- mkdir DatingApp
- cd DatingApp
- dotnet --info , to check .net installation
- dotnet new sln
- dotnet new webapi -o API
- dotnet sln add API
- cd API , dotnet run
- Entity
- Entity Framework ⇒ Database



DbContext.Users.Add(new User {Id = 4, Name = John})

⟹ INSERT INTO Users (Id, Name)
VALUES (4, John)

- API: dotnet dev-certs https --trust
- WeatherForecastController
  ↳ localhost:5001/weatherForecast

API

- ↓ dotnet ef database update, to create database

- API > Controllers > UsersController.cs

```
using System.Collections.Generic;
 "      "    .Linq;
 "      API.Data;
 "      "    .Entities;
 "     Microsoft.AspNetCore.Mvc;
namespace API.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class UsersController : ControllerBase
    {
        private readonly DataContext _context;
        public UsersController(DataContext context)
        {
            _context = context;
        }

        [HttpGet]
        public ActionResult<IEnumerable<AppUser>> GetUsers()
        {
            var users = _context.Users.ToList();
            return users;
        }
        [HttpGet("{id}")]
        public ActionResult<AppUser> GetUser(int id)
        {
            var user = _context.Users.Find(id);
            return user
        }
    }
}
```

- Users Controller:

[MS 01.08.2021]

```
public async Task<ActionResult<IEnumerable<AppUser>>> GetUsers()
{
    return await _context.Users.ToListAsync();
}

public async Task<ActionResult<AppUser>> GetUser(int id)
{
    return await _context.Users.FindAsync(id);
}
```

- Source Code:

Dating App: git status ⇒ "fatal: not a git repository"

Dating App: git init

Dating App: dotnet new gitignore

→ Add "appsettings.json" to .gitignore

→ git remote add origin [Link]

→ git push -u origin master

Section 3:
(Part two)

- Angular Application:

DatingApp: node --version
npm --version
npm install -g @angular/cli
ng new client
⇒ Angular routing: yes
⇒ CSS

strict Mode: "NO"

- Starting angular development:

client: ng serve

- app.component.html ⇒ <h1>Dating app</h1>

↳ app.component.ts ⇒ export class AppComponent
{
title = 'The Dating app';
}

app.component.html ⇒ <h1>{{title}}</h1>

- app.component.ts ⇒ export class AppComponent implements OnInit
{
users: any;
constructor(private http: HttpClient) {}
:
ngOnInit(): void {
this.getUsers();
} ⊛

app.module.ts ⇒ }

```
import {HttpClientModule} from '@angular/common/http';
:
imports:[
....,
HttpClientModule
],
```

⊛ ⇒ getUsers()
{
this.http.get('https://localhost:5001/api/users').subscribe(
response ⇒ { this.users = response; }, error ⇒ {console.log(error);
})
}

[MS 02.08.2021]

- Startup.cs ⇒

```
public void ConfigureServices (....)
{
    :
    Service.AddCors();
}

public void Configure (...)
{
    app.UseRating();
```

↳ `app.UseCors( x => x.AllowAnyHeader().AllowAnyMethod().WithOrigins(`

↳ `"http://localhost:4200"`

```
    app.UseAuthorization();
```

- app.component.html ⇒

```
<ul>
    <li *ngFor = 'let user of users' > {{{user.id}}} - {user.userName}}
</ul>
```


↳ 1 - Bob
2 - Tom
3 - Jane

- client: ng add ngx-bootstrap
npm install font-awesome

- After certificate, Startup.cs ⇒ http ⇒ https

Section 4 Authentication Basics :

- Password hash & password salt

→ AppUser.cs ⇒

```
public class AppUser
{
    :
    public byte[] PasswordHash { get; set; }
    public byte[] PasswordSalt { get; set; }
}
```

→ API: dotnet ef migrations add UserPasswordAdded
dotnet ef database update

- New class in API/Controllers : BaseApiController.cs

BaseApiController.cs ⇒

```
using Microsoft.AspNetCore.Mvc;

namespace API.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class BaseApiController : ControllerBase
    {
    }
}
```

UsersController.cs ⇒ → Delete until namespace

public class UsersController : BaseApiController

- New class in API/Controller: AccountControllers.cs

AccountControllers.cs ⇒

```
...
Using API.Data;

namespace API.Controllers
{
    public class AccountController : BaseApiController
    {
        private readonly DataContext _context;

        public AccountController (DataContext context)
        {
            _context = context;
```

(**)

```
if (await UserExists(registerDto.
Username)) return
    BadRequest ("Username is
    taken");
```

```
ActionResult<AppUser>> Register (RegisterDto
                                 registerDto)
```

```
Using var hmac = new HMACSHA512();
(**)
var user = new AppUser
{
```

```
    ...
    ...isterDto.Username,
    ...mac.ComputeHash(Encoding.UTF8.GetBytes
    ...ssword)),
    ...mac.Key
```

(*) private async Task<bool>
UserExists(string username)
{
    return await _context.
    Users.AnyAsync(x => x.UserName
    == username.ToLower());
}

```
    d(user);
    SaveChangesAsync();
```

- New folder in API : DTOs

New class in API/DTOs : RegisterDto.cs

RegisterDto.cs ⇒

```
namespace API.DTOs
{
    public class RegisterDto
    {
        public string Username { get; set; }
        public string Password { get; set; }
    }
}
```

[Required]

- New class in API/DTOs : LoginDto.cs [MS 03.08.2021]

```
namespace API.DTOs
{
    public class LoginDto
    {
        public string Username { get; set;}

        public string Password { get; set; }
    }
}
```

- AccountController.cs ⇒

```
[HttpPost("login")]
public async Task<ActionResult<AppUser>> Login(LoginDto loginDto)
{
    var user = await _context.Users
        .SingleOrDefaultAsync(x => x.UserName == loginDto.Username);

    if(user == null) return Unauthorized("Invalid username");

    using var hmac = new HMACSHA512(user.PasswordSalt);

    var computedHash = hmac.ComputedHash(Encoding.UTF8.GetBytes(loginDto.
      Password));

    for(int i=0; i< computedHash.Length; i++)
    {
        if(computedHash[i] != user.PasswordHash[i])
          return Unauthorized("Invalid password");
    }
    return user;
}
```

- dotnet ef database drop ⇒ delete the database
- dotnet ef database update ⇒ create the database

- Adding Token Service;

→ New folder in API: Interfaces

→ New interface in API/Interfaces: ITokenService.cs ⇒

```
namespace API.Interfaces
{
    public interface ITokenService
    {
        string CreateToken(AppUser user);
    }
}
```

→ New folder in API: Services

→ New class in API/Services: TokenService.cs ⇒

```
namespace API.Services
{
    public class TokenService : ITokenService
    {
        public string CreateToken(AppUser user)
        {
            return "";
        }
    }
}
```

→ Startup.cs ⇒

```
public void ConfigureServices(...)
{
    Services.AddScoped<ITokenService, TokenService>();
    :
    :
}
```

- Adding the create token logic

→ Nuget: System.Identity.Model.Tokens.Jwt

→ TokenService.cs ⇒

```
public class Token....
{
    private readonly SymmetricSecurityKey _key;

    public TokenService(IConfiguration config)
    {
        _key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(
        config["TokenKey"]));
    }
    public string CreateToken(AppUser user)
    {
        var claims = new List<Claim>
        {
            new claim(JwtRegisteredClaimNames.NameId, user.UserName)
        };
        var creds = new SigningCredentials(_key, SecurityAlgorithms.
        HmacSha512Signature);
```

...Cont...

```
var tokenDescriptor = new SecurityTokenDescriptor
{
        Subject = new ClaimsIdentity(claims),
        Expires = DateTime.Now.AddDays(7),
        SigningCredentials = creds
};

var tokenHandler = new JwtSecurityTokenHandler();
var token = tokenHandler.CreateToken(tokenDescriptor);
return tokenHandler.WriteToken(token);

}
```

- Creating a User DTO and returning the token

→ AccountController.cs ⇒

```
public class Account ...
{
    :
    Private readonly ITokenService _tokenService;
    Public AccountController(DataContext context, ITokenService tokenService)
    {
        _tokenService = tokenService;
        :
    }

    registerMethod ⇒ Task<ActionResult<UserDto>>
    {
        :
        return new UserDto
        {
            Username = user.UserName,
            Token = _tokenService.CreateToken(user)
        };
    }
```

... cont ...

```
loginMethod => Task<ActionResult<UserDto>>
{
    :
    :
    return new UserDto
    {
        Username = user.UserName,
        Token = _tokenService.CreateToken(user)
    };
}
```

→ New class in API/DTOs: UserDto.cs =>

```
public class UserDto
{
    public string Username { get; set; }
    public string Token { get; set; }
}
```

→ appsettings.Development.json =>

```
"TokenKey": "super secret unguessable key",
"Logging" ....
    !
```

Student No:	Full Name:	Section:	Department:

Question No.:_____

- Adding the authentication middleware

→ UsersController.cs ⇒

```
[HttpGet]
[AllowAnonymous]
public async ... GetUsers()
    ⋮
[Authorize]
[HttpGet("{id}")]
```

→ Nuget: Microsoft.AspNetCore.Authentication.JwtBearer

→ Startup.cs ⇒

```
public void ConfigureServices...
{
    ⋮
    services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
        .AddJwtBearer(options =>
        {
            options.TokenValidationParameters = new TokenValidationParameters
            {
                ValidateIssuerSigningKey = true,
                IssuerSigningKey = new SymmetricSecurityKey(
                    Encoding.UTF8.GetBytes(config["TokenKey"])),
                ValidateIssuer = false,
                ValidateAudience = false,
            };
        });
```

... cont ...

ConfigureMethod

app.UseCors...

app.UseAuthentication();

app.UseAuthorization();

- Adding extension methods:

→ New Folder in API: Extensions
→ New class in API/Extensions: ApplicationServiceExtensions.cs
→ " " " " : IdentityServiceExtensions.cs
→ ApplicationServiceExtensions.cs ⇒

```
public static class ApplicationServiceExtensions
{
    public static IServiceCollection AddApplicationServices(
        this IServiceCollection services, IConfiguration config)
    {
        ⇒ Cut Copy from ConfigureServices: AddScoped, AddDbContext, UseSqlite
          change UseSqlite (config ...)

        return services;
    }
}
```

→ Startup.cs ⇒

```
ConfigureServices(...)
{
    services.AddApplicationServices(_config);
    :
    services.AddIdentityServices(_config);
```

Student No:	Full Name:	Section:	Department:

Question No.:_____

→ IdentityServiceExtensions.cs ⇒

```
public static class IdentityServiceExtensions
{
    public static IServiceCollection AddIdentityServices(this
     IServiceCollection services, IConfiguration config)
    {
        cut
        copy from ConfigureServices: AddAuthentication
        change _config to config
        return services;
    }
}
```

Section 5 - Client Login and Register:

- Creating a nav bar:

→ client: ng g -h
↑ generate

client: cd src/app

app: ng g c nav --skip-tests
↑ component

→ Inspect examples/carousel in getbootstrap.com
<nav ..... >-</nav> ⇒ copy to nav.component.html

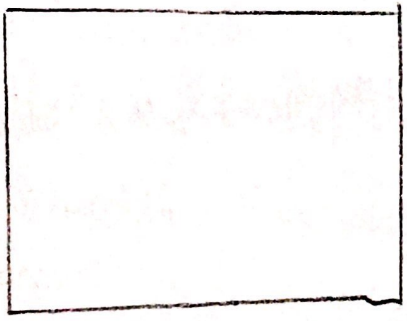
→ nav.component.ts ⇒ see the selector 'app-nav'

→ app.component.html ⇒

Line 1: <app-nav></app-nav>

→ nav.component.html:

<nav class="navbar ...



⇒ take out any

⇒ delete <button>...</button>
⇒ Change the name to "Dating App"
⇒ take out <div class="collapse navbar ...
⇒ " " active and span for "Home"
⇒ " " disabled and aria and tabindex for "Disabled"

Student No:	Full Name:	Section:	Department:

Question No.:_____

⇒ Make first link "Matches"
" second " "Lists"
" third " "Messages"

⇒ Delete aria-label from "form"
Change ↓"Search" to "Login"
Change ↑"Search" to "Username"
↳ copy the line and change "Username" to "Password"
↓ type="text" ↳ make type="password"

→ div.container , tab

→ app.component.html ⇒

<app-nav></app-nav>

<UL>
copy {{user.id}} - {{user.userName}} line here
</UL>

[MS 04.08.2021]

- Introduction to Angular Template Forms:

→ app.module.ts ⇒

```
imports: [
    :
  FormsModule
],
```

```
import {FormsModule} from '@angular/forms';
```

→ nav.component.html ⇒

```
<form #loginForm="ngForm" class "..." (ngSubmit)="login()"
 autocomplete="off">

<input
   name="username"
   [(ngModel)]="model.username"        ← a-ngm, tab
   class ...
   type ...
   placeholder= ...
<input
   name="password"
   [(ngModel)]="model.password"
   class...
    :
```

→ nav.component.ts ⇒

```
export class ...
{
   model: any = {}
      :
   login()
   {
     console.log(this.model);
   }
}
```

PHYS 101-102 Midterm Exam 1 Spring Semester 2020-21

Student No:	Full Name:	Section:	Department:

Question No.:_____

- Introduction to Angular services:

→ New folder in client/app: _services

→ cd _services

-services: ng g s account --skip-tests

→ account.service.ts ⇒

```
export class AccountService
{
    baseUrl = 'https://localhost:5001/api/';

    constructor (private http: HttpClient) {}

    login(model: any)
    {
        return this.http.post(this.baseUrl + 'account/login', model);
    }
```

- Injecting services into components:

→ nav.component.ts ⇒

```
export class Nav...
{
    :→ loggedIn: boolean;
    :
    constructor (private accountService: AccountService) {}
    :
    :
    login()
    {
        this.accountService.login(this.model).subscribe(response =>
        {
            console.log(response);
            this.loggedIn = true;
        }, error => { console.log(error); })
```

- Using conditionals to show and remove content:

→ nav.component.ts ⇒

```
login() ...
    :
}

logout() {
    this.loggedIn = false;
}
```

→ nav.component.html ⇒

```
<ul class="navbar-nav mr-auto" *ngIf="loggedIn">
    :
    <li class="nav-item">
        <a class="nav-link" (click)="logout()" href="#">
        Logout </a>
    </li>
</ul>

<div class="dropdown" *ngIf="loggedIn">
    <a class="dropdown-toggle text-light">Welcome user</a>
    <div class="dropdown-menu">
        <a class="dropdown-item">Edit Profile</a>
        <a class="dropdown-item" (click)="logout()">Logout
        </a>
    </div>
</div>

<form *ngIf="!loggedIn" #loginForm="ngForm" class="form-inline
mt-2 mt-md-0" (ngSubmit)="login()" autocomplete="off">
```

PHYS 101-102 Midterm Exam 1 Spring Semester 2020-21

Student No:	Full Name:	Section:	Department:

Question No.:_____

- Using the angular bootstrap components - dropdown :

→ app.module.ts ⇒

```
import { BsDropdownModule } from 'ngx-bootstrap/dropdown';
  :
imports: [
    :
    BsDropdownModule.forRoot()
```

→ nav.component.html ⇒

```
<div class "dropdown" *ngIf = "loggedIn" dropdown >
  <a class = "dropdown-toggle text-light" dropdownToggle >
    Welcome user</a>
  <div class = "dropdown-menu mt-3" *dropdownMenu>
    <a class = "dropdown-item"> Edit Profile</a>
    <div class = "dropdown-divider"></div>
    <a class = "dropdown-item" (click) = "logout()">Logout
      </a>
  </div>
</div>
```

→ nav.component.css ⇒

```
.dropdown-toggle, .dropdown-item
{
    cursor : pointer;
}
```

- Introduction to observables:
- Persisting the login:

→ account.service.ts ⇒

```
import {map} from 'rxjs/operators';
  :
  private currentUserSource = new ReplaySubject<User>(1);
  currentUser$ = this.currentUserSource.asObservable();
  constructor ...
  login (model: any)
```

```
setCurrentUser(user: User)
{
  this.currentUserSource.next
    (user);
}
```

```
st(this.baseUrl + 'account/login', model).pipe(
User) => {
esponse;

.setItem ('user', JSON.stringify(user));
UserSource.next(user);
  })
  )
}
```

```
logout()
{
  localStorage.removeItem('user');
  this.currentUserSource.next(null);
}
```

→ New folder in client/app/ : _models

New file in _models: user.ts

user.ts ⇒

```
export interface User
{
    username: string;
    token: string;
}
```

PHYS 101-102 Midterm Exam 1 Spring Semester 2020-21

Student No:	Full Name:	Section:	Department:

Question No.:_____

→ app.component.ts ⇒

```
constructor(private http: ... , private accountService: AccountService) {}

ngOnInit()
{
    this.getUsers();
    this.setCurrentUser();
}

setCurrentUser()
{
   const user: User = JSON.parse(localStorage.getItem('user'));
   this.accountService.setCurrentUser(user);
}
```

→ nav.component.ts ⇒

```
logout()
{
    this.accountService.logout();
     :
}

getCurrentUser()
{
    this.accountService.currentUser$.subcribe(user => {
      this.loggedIn = !!user;
    }, error => {
      console.log(error);
    })
}
```

```
ngOnInit(): void {
  this.getCurrentUser();
}
```

Ms 06.08.2021

- Using the async pipe:

→ nav.component.ts ⇒

```
export class ...
{
    ...                                  constructor(public account ...)
    currentUser$: Observable<User>;
    :
    ngOnInit(): void
    {
        this.currentUser$ = this.accountService.currentUser$;
    }

    login()
    {
        :
        this.loggedIn = true;  X
        :
    }
    logout()
    {
        :
        this.loggedIn = false;  X
    }

    setCurrentUser()  X
```

→ nav.component.html ⇒

for every loggedIn replace with accountService.currentUser$ | async

!loggedIn " " (currentUser$ | async) === null

remove "logout" item

Student No:	Full Name:	Section:	Department:

Question No.:_____

- Adding a home page:

→ app: ng g c home --skip-tests

⇒ home.component.html ⇒ ⊛ ⇒ *ngIF="!registerMode"

```
<div class="container mt-5">
    <div *style='text-align: center;'>
        <h1> Find your match </h1>
        <p class="lead"> Come on in to view your matches... all you need to do is sign up! </p>

        <div class="text-center">
            <button ***class="btn btn-primary btn-lg mr-2"> Register
            </button>

            <button class="btn btn-info btn-lg"> Learn more </button>

        </div>

    </div>
```

→ *ngIF="registerMode"

```
<div ***class="container">
    <div class="row justify-content-center">
        <div class="col-4"> </div>
    </div>
</div>
```

→ Register Form goes here

*** ⇒ (click)="registerToggle()"

→ app.component.html

```
<div class="conta...
    <app-home></app-home>
</div>
```

→ home.component.ts →

```
export class ...
{
    registerMode = false;
      :

    registerToggle(){
        this.registerMode = !this.registerMode;
    }
}
```

- Adding a register form:

→ app: ng g c register --skip-tests

→ register.component.ts →

```
export class ...
{
    model: any = {};
    :
    register()
    {
        console.log(this.model);
    }

    cancel(){
      console.log('cancelled');
    }
}
```

Student No:	Full Name:	Section:	Department:

Question No.:_____

→ register.component.html ⇒

```
<form #registerForm="ngForm" (ngSubmit)="register()" autocomplete="off">
    <h2 class="text-center text-primary">Sign up</h2>
    <hr>

    <div class="form-group">
        <input type="text" class="form-control" name="username"
        [(ngModel)]="model.username" placeholder="Username">
    </div>

    <div class="form-group">
        <input type="password" class="form-control" name="password"
        [(ngModel)]="model.password" placeholder="Password">
    </div>

    <div class="form-group text-center">
        <button class="btn btn-success mr-2" type="submit">Register
        </button>
        <button class="btn btn-default mr-2" (click)="cancel()"
        type="button">Cancel</button>
    </div>

</form>
```

→ ~~this~~ home.component.html =>

Register form goes here <app-register></app-register>

PHYS 101 / 102 Midterm Exam 1 Spring Semester 2020/21

Student No:	Full Name:	Section:	Department:

Question No.:_____

MS 07.08.2021

- Parent to child communication:

→ app.component.ts ⇒

Remove getUsers() {...}, check ngOnInit

constructor (private accountService ...) { }

→ home.components.ts ⇒

```
export class HomeCom...
{
    :
    users: any;
    constructor (private http: HttpClient) { }
    ⟶ ngOnInit(): void { this.getUsers() };
    getUsers()
    {
        this.http.get('https://localhost:5001/api/users').subscribe (
            users => this.users = users);
    }
}
```

→ register.component.ts ⇒

```
export class ...
{
    @Input() usersFromHomeComponent: any;
    :
}
```

→ home.component.html ⇒

```
<app-register [usersFromHomeComponent] = "users"></app-register>
```

Student No:	Full Name:	Section:	Department:

Question No.:_____

- Child to parent communication:

→ register.component.ts ⇒

```
export class ...
{
  :
  @Output()  cancelRegister = new EventEmitter();
    .
    .
    .
  cancel()
  {
      this.cancelRegister.emit(false);
  }
}
```

→ home.component.html ⇒

```
<app-register [user...]="users" (cancelRegister)="cancelRegisterMode
    ($event)"></app-register>
```

→ home.component.ts ⇒

```
getUsers()
{
  ...
}

cancelRegisterMode(event: boolean)
{
    this.registerMode = event;
}
```

- Hooking up the register method to the service:

→ account.service.ts ⇒

```
login()
{
...
}
register(model: any)
{
    return this.http.post(this.baseUrl + 'account/register', model).pipe
        ( map((user: User) => {
            if(user)
            {
                localStorage.setItem('user', JSON.stringify(user));
                this.currentUserSource.next(user);
            }
        })
    )
}
```

→ register.component.ts ⇒

delete @Input() ...

→ home.component.ts ⇒

```
ngOnInit(): void { }

remove getUsers(){...}
constructor()
remove users: any;
```

Student No:	Full Name:	Section:	Department:

Question No.:____

→ home.component.html ⇒

remove [usersFromHomeComponent] = "users"

→ register.component.ts ⇒

```
constructor (private accountService: AccountService) {}

register()
{
    this.accountService.register(this.model).subscribe (response =>
        {
            console.log(response);
            this.cancel();
        }, error =>
          {
              console.log(error);
          })
}
```

— End of section 5 —

• Start of section 6: Routing In Angular

- Creating some more components:

→ app: mkdir members

members: ng g c member-list --skip-tests

members: ng g c member-detail --skip-tests

app: ng g c lists --skip-tests

app: ng g c messages --skip-tests

→ app-routing.module.ts ⇒

```
const routes:Routes = [
  { path: '', component: HomeComponent },
  { path: 'members', "" : MemberListComponent },
  { path: 'members/:id', "" : MemberDetailComponent },
  { path: 'lists', "" : ListsComponent },
  { path: 'messages', "" : MessagesComponent },
  { path: '**', "" : HomeComponent, pathMatch: 'full' },
];
```

→ app.component.html ⇒

```
<app-home></app-home>
<router-outlet></router-outlet>
```

Student No:	Full Name:	Section:	Department:

Question No.:_____

- Adding the nav links:

→ nav.component.html ⇒

```
<a class="navbar-brand" routerLink='/'>...
    <a class="nav-link" routerLink='/members'> Matches ...
    <"  "  "  "  ='/lists'> Lists ...
    <"  "  "  "  ='/messages'> Messages ...
```

routerLinkActive='active'

- Routing in code:

→ nav.component.ts ⇒

```
constructor( ...., private router: Router){ }

login()
{
    :
    console.log(response)  this.router.navigateByUrl('/members');
    :
}

logout()
{
    :
    this.router.navigateByUrl('/')
}
```

- Adding a toast service for notifications:

→ client: npm install ngx-toastr

→ angular.json ⇒

```
"styles": [
  :
  "./node_modules/font-awesome/css/font-awesome.css",
  "./node_modules/ngx-toastr/toastr.css",
  :
],
```

→ app.module.ts ⇒

```
imports: [
  :
  :
  :
                    ⟹ import {ToastrModule} from 'ngx-toastr';
  ToastrModule.forRoot({
    positionClass: 'toast-bottom-right'
  })
```

→ nav.component.ts ⇒

```
constructor(..., ..., private toastr: ToastrService) {}
```

→ register.component.ts ⇒

```
register()  ⟺ login()
{
  ;
  :), error => {
    console.log(error);
    this.toastr.error(error.error);
  })
}
```

Student No:	Full Name:	Section:	Department:

Question No.:_____

- Adding an Angular route guard;

→ Menu folder in app: _guards

-guards: ng g guard auth --skip-tests

"can activate"

→ auth.guards.ts ⇒

constructor (private accountService: AccountService, private toastr: ToastrService)

canActivate(...): Observable<boolean> {

Remove

return this.accountService.currentUser$.pipe(

map(user => {

if(user) return true;

this.toastr.error('You shall not pass!')

})

)

}

- Adding a dummy route:

→ app-routing.module.ts ⇒

```
const routes: Routes = [

  { path: '', component: HomeComponent },
  {
    path: '',
    runGuardsAndResolvers: 'always',
    canActivate: [AuthGuard],
    children: [
      Cut paste the routes {path: ...}, except '**'
    ]
  },...
```

→ nav.component.html ⇒

```
<ul class="navbar-nav mr-auto">
  <ng-container *ngIf="accountService.currentUser$ | async">
    <li ...>
    </li>
    <li
      :
    </li>
  </ng-container>
</ul>
```

PHYS 101-102 Midterm Exam 1 Spring Semester 2020-21

Student No:	Full Name:	Section:	Department:

Question No.:_____

- Adding a new theme:

→ Bootswatch

→ client: npm install bootswatch

→ angular.json ⇒

```
"styles": [
    :
    :
  "./node_modules/bootswatch/dist/united/bootstrap.css",
    :
    :
],
```

→ nav.component.html ⇒

```
<nav class="navbar ... bg-primary">
  :
  :
  <div class="dropdown" *ngIf="(...) as user" dropdown>
    <a class ..... dropdownToggle>Welcome {{user.username}}</a>
      :
      :
    <button class="btn btn-success my-2 ..." ...>Login
    </button>
  </form>
</div>
</nav>
```

... | async

| titlecase

- Tidying up the app module by using a shared module:

→ app: mkdir -modules

-modules: ng g m shared --flat

→ shared.module.ts ⇒

```
imports: [
    CommonModule,
    BsDropdownModule.forRoot(),
    ToastrModule.forRoot({
      positionClass: 'toast-bottom-right'
    })
],

exports: [
    BsDropdownModule,
    ToastrModule
]
})
```

Replace with SharedModule in app.module.ts

- End of Section 6 -