# Links For INF354 Exam:

# Authentication(Register, Login):

<https://www.c-sharpcorner.com/article/authentication-and-authorization-in-asp-net-core-web-api-with-json-web-tokens/> **(Authentication in .net 6 API)**https://www.c-sharpcorner.com/article/jwt-token-authentication-in-angular-14-and-net-core-6-web-api/ **(Authentication in angular)**

# Rasa Chatbot:

<https://dev.to/petr7555/rasa-socket-io-integration-pfo>

<https://www.skypack.dev/view/angular-chat-widget-rasa>

<https://github.com/contribution-jhipster-uga/angular-chat-widget-rasa>

GitJust

API - API2PRACTICE <https://gist.github.com/u21448567/1bef9b51f6556ec8c77cfd729eaea929> API - API1LectureSourceCode <https://gist.github.com/u21448567/a6737549ad3d28f502b10030dc0d8b2d> API-HWA1-MyCode <https://gist.github.com/u21448567/b89c27060c8a78124f8d9dcdf4664ee9> API - HWA1 - TDO <https://gist.github.com/u21448567/b09bc301fc0d80ed52f3c6f6d2dc718e> ANGULAR - TaskManager - Source Code <https://gist.github.com/u21448567/894c9a6bac81979679a63cd15312e9f5> ANGULAR - Angular2 - SourceCode <https://gist.github.com/u21448567/0c9f9179a4ad6ce8e6d37d8225bb9272> ANGULAR - HWA1 - MyCode <https://gist.github.com/u21448567/762faef69ce80982bf33adbfa6d5ee0a> ANGULAR - HWA1 - Memo <https://gist.github.com/u21448567/cc642e15d9d4c809032f188a2468b085> IONIC - HWA2 - MyCode <https://gist.github.com/u21448567/3d466b10d55b55daefe1dfd31169f25e> API - HWA3 - MyCode <https://gist.github.com/u21448567/1667214988fc732b3872544d2312db06> ANGULAR - HWA3 - MyCode <https://gist.github.com/u21448567/ff429bd58729fec543d505b3ee2f0035> API - HWA3 - Memo <https://gist.github.com/u21448567/82ee79ca499419b9ccb2c5d6ad5ee394> ANGULAR - HWA3 - Memo <https://gist.github.com/u21448567/c1118b8a4d42bd411ecad7df0d564077> APPLICATION SECURITY - LECTURE SOURCE CODE <https://gist.github.com/u21448567/b2d421dd79b96d0c162a10a5f2289ac4> RASA - LECTURE SOURCE CODE <https://gist.github.com/u21448567/1e32cc8ebc5d0d3bdf185ff3555e9436> PastExam - 2022 - MyAttempt <https://gist.github.com/u21448567/2083ee85f1d0c8f3ef3a6c63307b31c4> REPORTING - LECTURE SOURCE CODE <https://gist.github.com/u21448567/c0c802633382f09b94b9c62ca167b0d0>

# Example for CRUD

## Add

**For add remember savechangesasync and remember to add to ViewModel**

var example = new example { Name = cvm.Name, Duration = cvm.Duration, Description = cvm.Description };

try

{

\_ExampleRepository.Add(example);

await \_exampleRepository.SaveChangesAsync();

}

catch (Exception)

{

return BadRequest("Invalid transaction");

}

return Ok(Example);

## Read

try

{

var results = await \_exampleRepository.GetAllExampleAsync();

return Ok(results);

}

catch (Exception)

{

return StatusCode(500,"Internal Server Error. Please contact support.");

}

**Update**

**For Update remember ID and Viewmodel**

try

{

var existingExample = await \_exampleRepository.GetExampleAsync(exampleId);

if (existingExample == null) return NotFound($"The example does not exist");

existingExample.Name = exampleModel.Name;

existingExample.Duration = exampleModel.Duration;

existingExample.Description = exampleModel.Description;

if (await \_exampleRepository.SaveChangesAsync())

{

return Ok(existingExample);

}

}

catch (Exception)

{

return StatusCode(500, "Internal Server Error. Please contact support.");

}

return BadRequest("Your request is invalid.");

Delete  
try

{

var existingExample = await \_exampleRepository.GetExampleAsync(exampleId);

if (existingExample == null) return NotFound($"The example does not exist");

\_exampleRepository.Delete(existingExample);

if (await \_exampleRepository.SaveChangesAsync()) return Ok(existingExample);

}

catch (Exception)

{

return StatusCode(500, "Internal Server Error. Please contact support.");

}

return BadRequest("Your request is invalid.");

[Route("api/[controller]")]

[ApiController]

public class ExampleController : ControllerBase

{

private readonly IExampleRepository \_ExampleRepository;

public ExampleController(IExampleRepository ExampleRepository)

{

\_ExampleRepository = ExampleRepository;

}

public async Task<IActionResult> AddExample(ExampleViewModel cvm)

public async Task<IActionResult> GetAllExamples()

public async Task<ActionResult<ExampleViewModel>> UpdateExample(int ExampleId, ExampleViewModel ExampleModel)

public async Task<IActionResult> DeleteExample(int ExampleId)

# Filtering:

[HttpGet]

[Route("GetCourses")]

public async Task<IActionResult> GetCourses(string nameFilter = null)

{

try

{

var results = await \_courseRepository.GetCoursesAsync(nameFilter);

return Ok(results);

}

catch (Exception)

{

return StatusCode(500,"Internal Server Error. Please contact support.");

}

}

# JWT, Register and Login examples:

[Route("api/[controller]")]

[ApiController]

public class AuthenticationController : ControllerBase

{

private readonly UserManager<AppUser> \_userManager;

private readonly IUserClaimsPrincipalFactory<AppUser> \_claimsPrincipalFactory;

private readonly IRepository \_repository;

private readonly IConfiguration \_configuration;

public AuthenticationController(UserManager<AppUser> userManager, IUserClaimsPrincipalFactory<AppUser> claimsPrincipalFactory, IRepository repository, IConfiguration configuration)

{

\_repository = repository;

\_userManager = userManager;

\_claimsPrincipalFactory = claimsPrincipalFactory;

\_configuration = configuration;

}

[HttpPost]

[Route("Register")]

public async Task<IActionResult> Register(UserViewModel uvm)

{

var user = await \_userManager.FindByIdAsync(uvm.emailaddress);

if (user == null)

{

user = new AppUser

{

Id = Guid.NewGuid().ToString(),

UserName = uvm.emailaddress,

Email = uvm.emailaddress

};

var result = await \_userManager.CreateAsync(user, uvm.password);

if (result.Errors.Count() > 0) return StatusCode(StatusCodes.Status500InternalServerError, "Internal Server Error. Please contact support.");

}

else

{

return Forbid("Account already exists.");

}

return Ok();

}

[HttpPost]

[Route("Login")]

public async Task<ActionResult> Login(UserViewModel uvm)

{

var user = await \_userManager.FindByNameAsync(uvm.emailaddress);

if (user != null && await \_userManager.CheckPasswordAsync(user, uvm.password))

{

try

{

var principal = await \_claimsPrincipalFactory.CreateAsync(user);

return GenerateJWTToken(user);

}

catch (Exception)

{

return StatusCode(StatusCodes.Status500InternalServerError, "Internal Server Error. Please contact support.");

}

}

else

{

return NotFound("Does not exist");

}

}

[HttpGet]

private ActionResult GenerateJWTToken(AppUser user)

{

// Create JWT Token

var claims = new[]

{

new Claim(JwtRegisteredClaimNames.Sub, user.Email),

new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),

new Claim(JwtRegisteredClaimNames.UniqueName, user.UserName)

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration["Tokens:Key"]));

var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

var token = new JwtSecurityToken(

\_configuration["Tokens:Issuer"],

\_configuration["Tokens:Audience"],

claims,

signingCredentials: credentials,

expires: DateTime.UtcNow.AddHours(3)

);

return Created("", new

{

token = new JwtSecurityTokenHandler().WriteToken(token),

user = user.UserName

});

}

}

}

# RASA Chatbot Code Example:

import { Component, ViewChild } from '@angular/core';

import { FormGroup, FormControl, Validators } from '@angular/forms';

import { MessageService } from '../service/api.service';

export interface Message {

  type: string;

  message: string;

}

@Component({

  selector: 'app-chat-support',

  templateUrl: './chat-support.component.html',

  styleUrls: ['./chat-support.component.scss'],

})

export class ChatSupportComponent {

  isOpen = false;

  loading = false;

  messages: Message[] = [];

  chatForm = new FormGroup({

    message: new FormControl('', [Validators.required]),

  });

  @ViewChild('scrollMe') private myScrollContainer: any;

  constructor(private messageService: MessageService) {

  }

  openSupportPopup() {

    this.isOpen = !this.isOpen;

  }

  sendMessage() {

    const sentMessage = this.chatForm.value.message!;

    this.loading = true;

    this.messages.push({

      type: 'user',

      message: sentMessage,

    });

    this.chatForm.reset();

    this.scrollToBottom();

    this.messageService.sendMessage(sentMessage).subscribe((response: any) => {

      for (const obj of response) {

        let value

        if (obj.hasOwnProperty('text') ) {

          value = obj['text']

          this.pushMessage(value)

        }

        if (obj.hasOwnProperty('image') ) {

          value = obj['image']

          this.pushMessage(value)

        }

      }

    });

  }

  pushMessage(message:string){

     this.messages.push({

        type: 'client',

        message: message,

      });

      this.scrollToBottom();

  }

  scrollToBottom() {

    setTimeout(() => {

      try {

        this.myScrollContainer.nativeElement.scrollTop =

          this.myScrollContainer.nativeElement.scrollHeight + 500;

      } catch (err) {}

    }, 150);

  }

}

HTML:  
<div id="assistant">

  <button id="assistant-popup-button" (click)="openSupportPopup()">

    Chat Support?

  </button>

  <div id="assistant-popup" [style.display]="isOpen ? 'block' : 'none'">

    <div id="assistant-popup-header">

      Your friendly Assistant

      <button id="assistant-popup-close-button" (click)="openSupportPopup()">

       X

      </button>

    </div>

    <div id="assistant-popup-body">

      <div class="messages" #scrollMe>

        <div \*ngFor="let message of messages" class="message">

          <div [class]="message.type">

            {{ message.message }}

          </div>

        </div>

        <div

          \*ngIf="loading"

          class="message"

          style="width: 100%; display: block"

        >

          <div [class]="'client'">...</div>

        </div>

      </div>

    </div>

    <form id="assistant-popup-footer" [formGroup]="chatForm">

      <input

        formControlName="message"

        type="text"

        id="assistant-popup-input"

        placeholder="Type your message here..."

      />

      <button

        id="assistant-popup-submit-button"

        [disabled]="!chatForm.valid"

        (click)="sendMessage()"

      >

        Submit

      </button>

    </form>

  </div>

</div>