FULLSTACK DOTNET TUTORIAL

In this tutorial, we are going to build a full stack dotnet project, which is connected to a database and an HTML frontend

Let Start With The BACKEND

STEP 1

- ➔ Create the project by using the visual studio code application
- ➔ Click on the [project>Manage nugget packages] to install the following packages

1. Microsoft.EntityFrameworkCore
2. Microsoft.EntityFrameworkCore.SqlServer
3. Microsoft.EntityFrameworkCore.Design
4. Microsoft.identity.Client
5. Microsoft.Data.SqlClient
6. Microsoft.SqlServer.Server

- ➔ To add CORS to your application, create a folder in the project root and name it Extensions
- ➔ Inside the Extensions folder, create a file and name it **ServiceExtensions.cs** and paste this code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ProjectManager.Extensions
{
public static class ServiceExtensions
{
public static void ConfigureCors(this IServiceCollection services) =>
services.AddCors(options =>
{
options.AddPolicy("CorsPolicy", builder =>
builder.AllowAnyOrigin()
.AllowAnyMethod()
.AllowAnyHeader());
});


public static void ConfigureIISIntegration(this IServiceCollection services) =>
services.Configure<IISOptions>(options =>
{
});




}
}
```

- ➔ In the Program.cs, add ConfigureCors and ConfigureIISIntegration as a service and also register the CORS in the Program.cs file. In the Program.cs, we need to call the DataContext in it too like this [builder.Services.AddDbContext<DataContext>();]

The entire **Program.cs** should look like this

```
using Microsoft.AspNetCore.HttpOverrides;
using ProjectManager.Data;
using ProjectManager.Extensions;


var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.ConfigureCors();
builder.Services.ConfigureIISIntegration();

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddDbContext<DataContext>();

var app = builder.Build();

if (app.Environment.IsDevelopment())
app.UseDeveloperExceptionPage();
else
app.UseHsts();


// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}

app.UseDefaultFiles();
app.UseStaticFiles();
app.UseForwardedHeaders(new ForwardedHeadersOptions
{
ForwardedHeaders = ForwardedHeaders.All
});
app.UseCors("CorsPolicy");

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();
```

app.Run();

Step 2

In the Root folder, create 2 folders, Models and Data
In the Models folder create this Four Models

Project.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ProjectManager.Models
{
public class Project
{
public int Id {get; set;}
public string name {get;set;}
public string description {get;set;}
public DateTime dueDate {get;set;}
public double cost {get;set;}
public string developer {get;set;}
public string customer {get;set;}
public string status {get;set;}
}
}
```

User.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace FlexiSecure.Models
{
public class User
{
public int Id { get; set; }

public string Name { get; set; } = string.Empty;
public string Email { get; set; } = string.Empty;
public byte[] PasswordHash { get; set; } = new byte[32];
public byte[] PasswordSalt { get; set; } = new byte[32];
public string? VerificationToken{get; set; }
public DateTime? VerifiedAt { get; set; }
```

```
public string? PasswordResetToken { get; set; }
public DateTime? ResetTokenExpires{get; set; }

}
}
```

UserRegisterRequest.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace FlexiSecure.Models
{
public class UserRegisterRequest
{
[Required,MinLength(3, ErrorMessage ="Enter a valid name")]
public string Name { get; set; } = string.Empty;
[Required, EmailAddress]
public string Email { get; set; } = string.Empty;
[Required, MinLength(6,ErrorMessage ="Please enter at least 6 characters ")]
public string Password { get; set; } = string.Empty;
[Required, Compare("Password")]
public string ConfirmPassword{get; set; } = string.Empty;


}
}
```

UserLoginRequest.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace FlexiSecure.Models
{
public class UserLoginRequest
{

[Required, EmailAddress]
public string Email { get; set; } = string.Empty;
[Required]
public string Password { get; set; } = string.Empty;

}
```

```
}
```

In the Data folder, Create a **DataContext.cs** file and paste this content

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using FlexiSecure.Models;
using Microsoft.EntityFrameworkCore;
using ProjectManager.Models;

namespace ProjectManager.Data
{
public class DataContext : DbContext
{
//Empty constructor
public DataContext(): base(){
}


//Database Connection String
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
base.OnConfiguring(optionsBuilder);
optionsBuilder.UseSqlServer("Server=localhost,1433;Database=Projects;User=sa;Passw
ord=HydotTech;TrustServerCertificate=true;");
}


//Data Set, where Project and User are models in the Model folder
public DbSet<Project> Projects => Set<Project>();
public DbSet<User> Users => Set<User>();


}
}
```

Step 3

Open the Controller Folder and Create 2 Controllers,

ProjectController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ProjectManager.Data;
using ProjectManager.Models;

namespace ProjectManager.Controllers
{
[ApiController]
[Route("api/Projects")]
public class ProjectController : ControllerBase
{
private readonly DataContext _context;
public ProjectController(DataContext context){
_context = context;
}
[HttpPost()]
public async Task<IActionResult> AddProject(Project project){
//Call the Project.cs model to create a skeleton
var my_project = new Project{
name = project.name,
description = project.description,
dueDate = project.dueDate,
cost = project.cost,
developer = project.developer,
customer = project.customer,
status = project.status
};
_context.Projects.Add(my_project);
await _context.SaveChangesAsync();
return Ok("Projects Created Successfully");

}
[HttpGet()]
public async Task<ActionResult<IEnumerable<Project>>> GetItems()
{
var items = await _context.Projects.ToListAsync(); // Replace "YourModels" with
your actual DbSet name

return Ok(items);
}

[HttpGet("{Id}")]
public async Task<ActionResult<Project>> GetItem(int Id){
```

```
var item = await _context.Projects.FindAsync(Id);
if (item == null){
return NotFound();
}
return Ok(item);


}
```

```
[HttpPut("{id}")]
public async Task<IActionResult> UpdateItem(int id, Project updatedItem){
if (id != updatedItem.Id)
{
return BadRequest("The IDS DONT MATCH"); // Returns HTTP 400 if the ID in the
route does not match the ID in the updated item
}

_context.Entry(updatedItem).State = EntityState.Modified;

try
{
await _context.SaveChangesAsync();
}
catch(DbUpdateConcurrencyException){
if (!ItemExists(id))
{
return NotFound(); // Returns HTTP 404 if the item with the specified ID is not
found
}
else
{
throw;
}
}
return NoContent();
}
private bool ItemExists(int id)
{
return _context.Projects.Any(e => e.Id == id);
}

[HttpDelete("{id}")]
public async Task<IActionResult> DeleteItem(int id)
{
```

```
var item = await _context.Projects.FindAsync(id);

if (item == null)
{
return NotFound("Item not found");
}

_context.Projects.Remove(item);
await _context.SaveChangesAsync();

return NoContent();
}









}




}
```

UserController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Threading.Tasks;
using FlexiSecure.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ProjectManager.Data;

namespace FlexiSecure.Controllers
{
[ApiController]
[Route("api/Auth")]
public class UserController : ControllerBase
{
private readonly DataContext _context;
public UserController(DataContext context)
{
_context = context;
}

[HttpPost("register")]
public async Task<IActionResult>Register(UserRegisterRequest request)
{

//Check if Email is already registered
if(_context.Users.Any(u=> u.Email == request.Email)){
return BadRequest("The user account is already registered");
};

//Create a Password Harsh
CreatePasswordHash(request.Password, out byte[]passwordHash,out
byte[]passwordSalt);

//Call the user model from User.cs, create a new objects
var user = new User{
Name = request.Name,
Email = request.Email,
PasswordHash=passwordHash,
PasswordSalt = passwordSalt,
VerificationToken = CreateRandomToken(),
};

// The _context is called from the DataContext file and insiidde that file, we
have the Users data set
//The Users dataset is using the User model
```

```
//We will the add the user object we created to the Users dataset
_context.Users.Add(user);
await _context.SaveChangesAsync();

return Ok("Hello "+request.Name+", your account has successfully been created");



}




[HttpPost("login")]
public async Task<IActionResult>Login(UserLoginRequest request)
{

var user = await _context.Users.FirstOrDefaultAsync(u => u.Email ==
request.Email);
if(user==null){
return BadRequest("User not found");
}

if(!VerifyPasswordHash(request.Password, user.PasswordHash, user.PasswordSalt))
{
return BadRequest("Password is incorrect");
}

if(user.VerifiedAt == null){
return BadRequest("User is not verified");
}

return Ok($"Welcome Back, {user.Name}! ");

}



[HttpPost("verify")]
public async Task<IActionResult>Verify(string token)
{

var user = await _context.Users.FirstOrDefaultAsync(u => u.VerificationToken ==
token);
if(user==null){
return BadRequest("Invalid token");
}

user.VerifiedAt = DateTime.Now;
```

```
await _context.SaveChangesAsync();


return Ok("Congratulations! You have been verified");




}
```

```
private static bool VerifyPasswordHash(string password, byte[] passwordHash,
byte[] passwordSalt)
{
using (var hmac = new HMACSHA512(passwordSalt))
{

var computedHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));
return computedHash.SequenceEqual(passwordHash);


}

}
```

```
private static void CreatePasswordHash(string password, out byte[] passwordHash,
out byte[] passwordSalt)
{
using (var hmac = new HMACSHA512())
{
passwordSalt = hmac.Key;
passwordHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));


}
```

```
}

private string CreateRandomToken(){
return Convert.ToHexString(RandomNumberGenerator.GetBytes(64));
}

}
}
```

Bonus Feature

If you want to add forgot password and reset password feature,

1. Create a **ResetPasswordRequest.cs** model and add the following code

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace FlexiSecure.Models
{
public class ResetPasswordRequest
{
[Required]
public string Token { get; set; } = string.Empty;
[Required, MinLength(6,ErrorMessage ="Please enter at least 6 characters ")]
public string Password { get; set; } = string.Empty;
[Required, Compare("Password")]
public string ConfirmPassword{get; set; } = string.Empty;


}
}
```

Inside the **UserController.cs**, add the following code

```
[HttpPost("forgot-password")]
public async Task<IActionResult>ForgotPassword(string email)
{

var user = await _context.Users.FirstOrDefaultAsync(u => u.Email == email);
if(user==null){
return BadRequest("User " + email + " does not exist");
}

user.PasswordResetToken = CreateRandomToken();
user.ResetTokenExpires = DateTime.Now.AddDays(1);
await _context.SaveChangesAsync();

return Ok("You may now reset your password");
```

```
}


[HttpPost("reset-password")]
public async Task<IActionResult>ResetPassword(ResetPasswordRequest request)
{


//The PasswordResetToken which i declared in the User.cs model and it is made
availabe in the
//DataContext.cs through the DbSet shouuld be equal to the Token the user will
declare in the ResetPasswordRequest model
var user = await _context.Users.FirstOrDefaultAsync(u => u.PasswordResetToken ==
request.Token);
if(user==null || user.ResetTokenExpires<DateTime.Now ){
return BadRequest("Invalid Token");
}

CreatePasswordHash(request.Password, out byte[] passwordHash, out byte[]
passwordSalt);
user.PasswordHash = passwordHash;
user.PasswordSalt = passwordSalt;
user.PasswordResetToken = null;
user.ResetTokenExpires = null;

await _context.SaveChangesAsync();

return Ok("Password successfully reset.");

}
```

Step 4
Ensure that the Database is running in Docker or any place. To create a Database in Docker, follow this tutorial

Adding Database

Follow this link to create a database sql server using docker in Mac Os

https://builtin.com/software-engineering-perspectives/sql-server-management-studio-mac

In the terminal, install dotnet-ef tool by running this command [dotnet tool install --global dotnet-ef]

Migrate the DbContext by running the command [dotnet ef migrations add Initial]
This will create a migration folder in the root folder

Also run the command to push the migration to the database [dotnet ef database update]

Install Azure Data Studio and Connect it to the Database you have created

server: *localhost*

Username: *sa*
Password: *YourPassword*

Run The Application By Typing [dotnet watch run]

LET DIVE INTO THE FRONTEND

Step 1
Open the launchSettings.json file from the Properties folder in the root folder and delete the **"launchUrl":"Swagger"** line so that the application will open the default index.html file.
The launchSettings.json file should look like this

```
{
"$schema": "https://json.schemastore.org/launchsettings.json",
"iisSettings": {
"windowsAuthentication": false,
"anonymousAuthentication": true,
"iisExpress": {
"applicationUrl": "http://localhost:29732",
"sslPort": 44348
}
},
"profiles": {
"http": {
"commandName": "Project",
"launchBrowser": true,
"applicationUrl": "http://localhost:5217",
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
},
"dotnetRunMessages": true
```

```
},
"https": {
"commandName": "Project",
"launchBrowser": true,
"applicationUrl": "https://localhost:7273;http://localhost:5217",
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
},
"dotnetRunMessages": true
},
"IIS Express": {
"commandName": "IISExpress",
"launchBrowser": true,
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
}
}
}
}
```

Create a folder called **wwwroot,** inside this folder, create the css, image and js folders
Outside the wwwroot folder, which is the root of the application, create the index.html

inside the index.html file, paste this code

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=p">
<title>Projects Manager</title>

<link rel="stylesheet" href="./css/styles.css">
</head>
<body>
<div class="header">
<div class="header-name">HyProject Manager</div>
</div>

<div id="loginContainer" class="sign-in">
<form action="javascript:void(0);" method="POST" onsubmit="authenticateUser()">
<h1>Sign In</h1>
<div class="empty"></div>
<input type="email" id="username" placeholder="&#9993  Username"/>
```

```
<input type="password" id="password"
placeholder="&#128272  **********"/>
<input type="submit" value="Sign in">
<a onclick="hysignup()"> Not Registered? Sign Up</a>
</form>

</div>
<br/><br/>
<div class="signup" id="signup">
<form action="javascript:void(0);" method="POST" onsubmit="signUP()">
<h1>Sign Up</h1>
<input type="text" name="name" id="name" placeholder="&#128100  Full
Name">
<input type="email" name="email" id="email"
placeholder="&#9993  user@example.com">
<input type="password" name="password" id="password1"
placeholder="&#128274  Password">
<input type="password" name="password" id="password2"
placeholder="&#128272  Confirm Password">
<a onclick="hyverify()"> Already Registered? Verify</a>

<input type="submit" value="Sign Up">
</form>

</div>
<div class="veriToken" id="veriToken">
<form action="javascript:void(0);" method="POST" onsubmit="verifyUser()">
<h1>Verify Account</h1>
<input type="text" name="verificationToken " id="token"
placeholder="&#128477  Enter the verification token">
<input type="submit" value="Verify">
</form>

</div>


<div id="dashboardContainer" class="hidden">

<div class="sidebar" id="sidebar">
<div class="sideproject" id="hyaddnew">

<form action="javascript:void(0);" method="POST" onsubmit="addProject()"
class="addProject">
<fieldset>
<legend>Add Project</legend>
<input type="text" name="customer" id="customer" placeholder="Customer Name ">
<input type="text" id="project-name" placeholder="Name of project">
<textarea name="description" id="description" cols="30" rows="10"
placeholder="Description of project"></textarea>
<input type="date" name="duedate" id="duedate" >
```

```
<input type="number" name="cost" id="cost" placeholder="Project Cost">
<input type="text" name="developer" id="developer" placeholder="Developer Name ">
<input type="text" name="status" id="status" placeholder="Project status">
<input type="submit" value="Add">
<a onclick="hyupdate()"> Want to Update? Update</a>
<a onclick="hydelete()"> Want to Delete? Delete</a>
</fieldset>
</form>

</div>

<div class="sideproject" id="hyupdate">
<form action="javascript:void(0);" onsubmit="updateProject()">
<fieldset>
<legend>Update Project</legend>
<input type="number" id="edit-id" placeholder="Project ID">
<input type="text" name="customer" id="edit-customer" placeholder="Customer Name
">
<input type="text" id="edit-name" placeholder="Name of project">
<textarea name="description" id="edit-description" cols="30" rows="10"
placeholder="Description of project"></textarea>
<input type="date" name="edit-duedate" id="edit-duedate" >
<input type="number" name="edit-cost" id="edit-cost" placeholder="Project Cost">
<input type="text" name="edit-developer" id="edit-developer"
placeholder="Developer Name ">
<input type="text" name="edit-status" id="edit-status" placeholder="Project
status">
<input type="submit" value="Save">
<a onclick="hyaddnew()"> Want to Add New? Add</a>
<a onclick="hydelete()"> Want to Delete? Delete</a>
</fieldset>
</form>
</div>

<div class="sideproject" id="hydelete">

<form action="javascript:void(0);" onsubmit="deleteProject()">
<fieldset>
<legend>Delete Project</legend>
<input type="number" id="delete-id" placeholder="Project ID">
<input type="submit" value="Delete">
<a onclick="hyaddnew()"> Want to Add New? Add</a>
<a onclick="hyupdate()"> Want to Update? Update</a>
</fieldset>
</form>

</div>

<br/><br/><br/>
</div>
```

```
<div class="display_projects" id="display_projects">
<div class="organizer">

<h2 id="welcomeMessage" class="welcome"></h2>

<div class="open" id="opener" onclick="opener()">Click to Add, Edit or Delete
Project</div>

</div>
<div class="container" id="container"></div>


</div>

</div>
```

```
<script src="js/script.js" asp-append-version="true"></script>
<script type="text/javascript">
getProject();
</script>


</body>
</html>
```

Inside the css folder, create the **styles.css** add the following code

```
.header{
height: 10vh;
display: flex;
justify-content: center;
background: rgba(47, 45, 45,1);

border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
```

```
backdrop-filter: blur(19.9px);
-webkit-backdrop-filter: blur(19.9px);
border: 1px solid rgba(255, 255, 255, 0.65);


}

.header-name{
font-size: 2rem;
padding-top: 1rem;
padding-bottom: 1rem;
color: rgba(255, 255, 255, 0.9);
font-family: 'Times New Roman', Times, serif;
}

.hidden {
display: none;
}


.card{
width:25rem;
word-wrap: break-word;
/* From https://css.glass */
background: rgba(47, 45, 45,1);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(13.1px);
-webkit-backdrop-filter: blur(13.1px);
border: 1px solid rgba(255, 255, 255, 0.65);
font-size: 2rem;
font-family: 'Times New Roman', Times, serif;
display: flex;
flex-direction: column;
gap: 1rem;
word-wrap: break-word;

}
.card-1>span{

color:#f06040
}
.card-1{
color: grey;
}


.container{
display: flex;
flex-direction: row;
```

```
gap: 2rem;
flex-wrap: wrap;
}

.card-2{
font-size: 2rem;
font-weight:700;

background: rgba(240, 96, 64, 0.55);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(15.7px);
-webkit-backdrop-filter: blur(15.7px);
border: 1px solid rgba(240, 96, 64, 1);
justify-content: center;
display: flex;
}


.sign-in{
display: flex;
justify-content: center;
margin-top: 10rem;
}

.sign-in>form{
display: flex;
flex-direction: column;
gap: 1rem;
width: 40rem;
height: 30rem;
background: rgba(255, 255, 255, 0.2);
/* From https://css.glass */
background: rgba(255, 255, 255, 0.74);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(16.8px);
-webkit-backdrop-filter: blur(16.8px);
border: 1px solid rgba(255, 255, 255, 0.79);


}

.sign-in>form>input{
border-radius: 2rem;
width: 35rem;
margin-left: 2rem;
margin-right: 2rem;
height: 3rem;
font-size: 2rem;
```

```
border-radius: 16px;
/* From https://css.glass */
background: rgba(255, 255, 255, 0.32);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(3.9px);
-webkit-backdrop-filter: blur(3.9px);
border: 1px solid rgba(255, 255, 255, 0.68);
cursor:pointer;
}
.sign-in>form>a{
border-radius: 2rem;
width: 35rem;
margin-left: 2rem;
margin-right: 2rem;
margin-top: 2rem;
height: 3rem;
font-size: 2rem;
border-radius: 16px;
cursor:pointer;
/* From https://css.glass */
background: rgba(255, 255, 255, 0.32);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(3.9px);
-webkit-backdrop-filter: blur(3.9px);
border: 1px solid rgba(255, 255, 255, 0.68);

}


.sign-in>form>h1{
text-align: center;
font-size: 3rem;
margin-top: 2rem;
margin-bottom: 2rem;
}



body{
background-image: url("../image/2.jpeg");
background-size: cover;
background-repeat: no-repeat;
background-position: center center;
background-attachment: fixed;

}
```

```
.signup{
display: none;
justify-content: center;
margin-top: 10rem;
}

.signup>form{
display: flex;
flex-direction: column;
gap: 1rem;
width: 40rem;
height: 40rem;
background: rgba(255, 255, 255, 0.2);
/* From https://css.glass */
background: rgba(255, 255, 255, 0.74);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(16.8px);
-webkit-backdrop-filter: blur(16.8px);
border: 1px solid rgba(255, 255, 255, 0.79);
}

.signup>form>input{
border-radius: 2rem;
width: 35rem;
margin-left: 2rem;
margin-right: 2rem;
height: 3rem;
font-size: 2rem;
border-radius: 16px;
/* From https://css.glass */
background: rgba(255, 255, 255, 0.32);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(3.9px);
-webkit-backdrop-filter: blur(3.9px);
border: 1px solid rgba(255, 255, 255, 0.68);
cursor:pointer;
}

.signup>form>a{
border-radius: 2rem;
width: 35rem;
margin-left: 2rem;
margin-right: 2rem;
margin-top: 2rem;
height: 3rem;
font-size: 2rem;
border-radius: 16px;
cursor:pointer;
```

```
/* From https://css.glass */
background: rgba(255, 255, 255, 0.32);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(3.9px);
-webkit-backdrop-filter: blur(3.9px);
border: 1px solid rgba(255, 255, 255, 0.68);

}


.signup>form>h1{
text-align: center;
font-size: 3rem;
margin-top: 2rem;
margin-bottom: 2rem;
}

.veriToken{
display: none;
justify-content: center;
margin-top: 10rem;

}


.veriToken>form{
display: flex;
flex-direction: column;
gap: 1rem;
width: 40rem;
height: 20rem;
background: rgba(255, 255, 255, 0.2);
/* From https://css.glass */
background: rgba(255, 255, 255, 0.74);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(16.8px);
-webkit-backdrop-filter: blur(16.8px);
border: 1px solid rgba(255, 255, 255, 0.79);
}

.veriToken>form>input{
border-radius: 2rem;
width: 35rem;
margin-left: 2rem;
margin-right: 2rem;
height: 3rem;
font-size: 2rem;
border-radius: 16px;
```

```
/* From https://css.glass */
background: rgba(255, 255, 255, 0.32);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(3.9px);
-webkit-backdrop-filter: blur(3.9px);
border: 1px solid rgba(255, 255, 255, 0.68);
cursor:pointer;
}

.veriToken>form>h1{
text-align: center;
font-size: 3rem;
margin-top: 2rem;
margin-bottom: 2rem;
}

.sidebar{
display: none;
flex-direction: column;
}



.sideproject>form>fieldset{
display: flex;
flex-direction: column;
gap:2rem;
width: 40rem;
background: rgba(255, 255, 255, 0.74);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(16.8px);
-webkit-backdrop-filter: blur(16.8px);
border: 1px solid rgba(255, 255, 255, 0.79);

}


.sideproject>form>fieldset>input{
border-radius: 2rem;
width: 35rem;
margin-left: 2rem;
margin-right: 2rem;
height: 3rem;
font-size: 2rem;
border-radius: 16px;
/* From https://css.glass */
background: rgba(255, 255, 255, 0.32);
border-radius: 16px;
```

```
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(3.9px);
-webkit-backdrop-filter: blur(3.9px);
border: 1px solid rgba(255, 255, 255, 0.68);
cursor:pointer;
}

.sideproject>form>fieldset>a{
border-radius: 2rem;
width: 35rem;
margin-left: 2rem;
margin-right: 2rem;
height: 3rem;
font-size: 2rem;
border-radius: 16px;
/* From https://css.glass */
background: rgba(255, 255, 255, 0.32);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(3.9px);
-webkit-backdrop-filter: blur(3.9px);
border: 1px solid rgba(255, 255, 255, 0.68);
cursor:pointer;
}

.sideproject>form>fieldset>textarea{
font-size: 2rem;
background: rgba(255, 255, 255, 0.32);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(3.9px);
-webkit-backdrop-filter: blur(3.9px);
border: 1px solid rgba(255, 255, 255, 0.68);
cursor:pointer;
width: 35rem;
margin-left: 2rem;
margin-right: 2rem;
}
.sideproject>form>fieldset>legend{
font-size: 2.5rem;
font-weight: bold;
}



#hyupdate{
display: none;
}

#hydelete{
```

```
display:none;
}


.welcome,.open{
font-size: 2rem;
font-weight:700;
height: 2rem;
background: rgba(240, 96, 64, 0.55);
border-radius: 16px;
box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
backdrop-filter: blur(15.7px);
-webkit-backdrop-filter: blur(15.7px);
border: 1px solid rgba(240, 96, 64, 1);
justify-content: center;
display: flex;
}
.sideproject {
position: absolute;
top: 100%;
left: 50%;

transform: translate(-50%, -50%);
}

.organizer{
display: flex;
flex-direction: row;
justify-content: space-evenly;
flex-wrap: wrap;
margin-bottom: 2rem;
}
```

Inside the js folder, create a **script.js** *and add the following code*

```
window.onload = function(){
Projects()
}
const uri = 'api/Projects';
const auth = "api/Auth";
let projects = [];
function getProject(){
fetch(uri)
.then(response => response.json())
.catch(error=> console.error('Unable to get items.', error))
}
/*
Create a Project
*/
```

```
function addProject(){
const addCustomer = document.getElementById("customer");
const addName = document.getElementById("project-name");
const addDescription = document.getElementById("description");
const addDuedate = document.getElementById("duedate");
const addCost = document.getElementById("cost");
const addDeveloper = document.getElementById("developer");
const addStatus = document.getElementById("status");
const projectItem = {
customer: addCustomer.value.trim(),
name: addName.value.trim(),
description: addDescription.value.trim(),
dueDate: addDuedate.value.trim(),
cost : addCost.value.trim(),
developer : addDeveloper.value.trim(),
status: addStatus.value.trim(),
};
fetch(uri, {
method: 'POST',
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json'
},
body: JSON.stringify(projectItem)
})
.then(response => response.json)
.then(()=>{
getProject();
addCustomer.value = "";
addStatus.value = "";
addDescription.value = "";
addDuedate.value = "";
addCost.value = "";
addDeveloper.value = "";
addName.value = "";
})
.catch(error => console.error('Unable to add Project.', error));
}
/*
Update a Project
*/
function updateProject(){
projectId = document.getElementById('edit-id').value.trim();
const projectItem = {
Id : document.getElementById('edit-id').value.trim(),
customer: document.getElementById('edit-customer').value.trim(),
name: document.getElementById('edit-name').value.trim(),
description: document.getElementById('edit-description').value.trim(),
dueDate: document.getElementById('edit-duedate').value.trim(),
cost: document.getElementById('edit-cost').value.trim(),
```

```
developer: document.getElementById('edit-developer').value.trim(),
status:document.getElementById('edit-status').value.trim(),
};
fetch(`${uri}/${projectId}`,{
method: 'PUT',
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json'
},
body: JSON.stringify(projectItem)
})
.then(response => response.json)
.then(() =>{
getProject();
document.getElementById('edit-id').value = "";
document.getElementById('edit-customer').value = "";
document.getElementById('edit-name').value = "";
document.getElementById('edit-description').value = "";
document.getElementById('edit-duedate').value = "";
document.getElementById('edit-cost').value = "";
document.getElementById('edit-developer').value = "";
document.getElementById('edit-status').value = "";
})
.catch(error => console.error('Unable to update projects.', error));
return false;
}
/*
DELETE PROJECT
*/
function deleteProject() {
projectId = document.getElementById("delete-id").value.trim();
fetch(`${uri}/${projectId}`, {
method: 'DELETE'
})
.then(() => {
getProject();
document.getElementById("delete-id").value = "";
})
.catch(error => console.error('Unable to delete item.', error));
}
async function Projects(){
try{
const response = await fetch(uri);
const items = await response.json();
const container = document.getElementById("container");

items.forEach(function(item){
const Card = document.createElement("div");
Card.className = "card"
const customer = document.createElement("div");
```

```
customer.className = "card-2";
customer.innerHTML = `&#128188;  ${item.customer}`;
Card.appendChild(customer);
const Id = document.createElement("div");
Id.className = "card-1";
Id.innerHTML = `<span >&#128273 ID:</span> ${item.id}`;
Card.appendChild(Id);
const name = document.createElement("div");
name.className = "card-1";
name.innerHTML = `<span >&#128450 Name:</span> ${item.name}`;
Card.appendChild(name);
const description = document.createElement("div");
description.className = "card-1";
description.innerHTML = `<span >&#128452 Description:</span> ${item.description}`;
Card.appendChild(description);
const cost = document.createElement("div");
cost.className = "card-1";
cost.innerHTML = `<span >&#128181 Cost:</span> GHS ${item.cost}`;
Card.appendChild(cost);
const duedate = document.createElement("div");
duedate.className = "card-1";
duedate.innerHTML = `<span >&#128198 Duedate:</span> ${item.dueDate}`;
Card.appendChild(duedate);
const developer = document.createElement("div");
developer.className = "card-1";
developer.innerHTML = `<span >&#128100 Developer:</span> ${item.developer}`;
Card.appendChild(developer);

const status = document.createElement("div");
status.className = "card-1";
status.innerHTML = `<span >&#128640 Status:</span>${item.status}`;
Card.appendChild(status);
container.appendChild(Card)
})
}
catch(error){
console.error("Error fetching Items",error);
}
}
async function verifyUser() {
const token = document.getElementById("token").value.trim();
try {
const response = await fetch(auth+"/verify?token=" + token, {
method: "POST"
});
const data = await response.text();
if (response.ok) {
// Verification successful
alert("Verification successful! " + data);
document.getElementById("signup").style.display = "none";
```

```
document.getElementById("veriToken").style.display = "none";
document.getElementById("loginContainer").style.display="flex"
} else {
// Verification failed
alert("Verification failed: " + data);
}
} catch (error) {
console.error("Error verifying user:", error);
}
}
//Signup
async function signUP(){
const name = document.getElementById("name").value;
const email = document.getElementById("email").value;
const password1 = document.getElementById("password1").value;
const password2 = document.getElementById("password2").value;
const register = {
Name: name.trim(),
Email: email.trim(),
Password: password1.trim(),
ConfirmPassword: password2.trim()
}
try{
const response = await fetch(auth+"/register",{
method: "POST",
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json'
},
body: JSON.stringify(register),
})
const data = await response.json();
if (response.ok){
alert(data);
document.getElementById("signup").style.display = "none";
document.getElementById("veriToken").style.display="flex";
}
else{
alert(data)
}
}
catch(error){
console.error("Error registering user:", error);
}
}
//Login Function
async function authenticateUser() {
const username = document.getElementById("username").value.trim();
const password = document.getElementById("password").value.trim();
const login = {
```

```
Email:username,
Password:password
}
try {
const response = await fetch(auth+'/login', {
method: "POST",
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json'
},
body: JSON.stringify(login)
});
const data = await response.json();
if (response.ok) {
// Authentication successful
showDashboard(data);
document.getElementById("loginContainer").style.display = "none";
} else {
// Authentication failed
alert("Authentication failed: " + data);
}
} catch (error) {
console.error("Error authenticating user:", error);
}
}
function showDashboard(data) {
const welcomeMessage = document.getElementById("welcomeMessage");
document.getElementById("signup").style.display = "none";
document.getElementById("veriToken").style.display = "none";
document.getElementById("loginContainer").style.display = "none";
document.getElementById("dashboardContainer").style.display="flex";
;
// Display the username in the welcome message
welcomeMessage.textContent = data;
}
function hysignup(){
document.getElementById("signup").style.display = "flex";
document.getElementById("loginContainer").style.display = "none";
}

function hyverify(){
document.getElementById("veriToken").style.display = "flex";
document.getElementById("signup").style.display = "none";
}

function opener(){
document.getElementById("sidebar").style.display = "flex";
document.getElementById("display_projects").style.display = "none";
document.getElementById("opener").style.display ='none';
```

```
}
function hyupdate(){
document.getElementById("hyupdate").style.display = "flex"
document.getElementById("hydelete").style.display = "none"
document.getElementById("hyaddnew").style.display = "none"
}

function hydelete(){
document.getElementById("hydelete").style.display = "flex"
document.getElementById("hyaddnew").style.display = "none"
document.getElementById("hyupdate").style.display = "none"
}

function hyaddnew(){
document.getElementById("hyaddnew").style.display = "flex"
document.getElementById("hyupdate").style.display = "none"
document.getElementById("hydelete").style.display = "none"

}
```

SECURITY

The script.js file can be visible to the hacker and the hacker can manipulate the code for illegal means but this is how you can beat the system

1. Make a copy of your code and store it internally now the production code, you can change all the variable to a single letter
2. You can copy other ruubbish code and add it to your code for it to look bulky and meaningless
3. 500 lines of code can be changed to 18000 lines by adding additional ruubbish code
4. Remove all comment in production code and use uglifyjs.net to compress it by removing all spaces in the code
5. Rename all your functions to be a nonsense one
6. We can also minify the html with this website