

# Project Deliverables

## Deliverables #1

### Project

Bug Tracker

### Requirements

□ **Determine the DB Schema** – Model the classes in C# and create the database.

1/set project model

2/Identity set project relationship

□ **Login / Register** – New users must be able to register and existing users must be able to successfully log in. News users should be placed on the Submitter role automatically upon registration.

Modify “accountcontroller”

result = UserManager.AddToRole(user.Id, "Submitter");

□ **Authenticated users** must be able to change their name and password.

Identitymodels.cs

ManageviewModels.cs

ChangNameModel

Changname.cshtml

Index.cshtml

□ **Authenticated users** must be able to recover a lost password.

Web.config: modify <appSettings file="private.config">

Add private.config

<appSettings>

<add key="SmtpHost" value="smtp.mailtrap.io"/>

<add key="SmtpPort" value="2525"/>

<add key="SmtpUsername" value="f9c47369b18c9d"/>

<add key="SmtpPassword" value="100954a8e77082"/>

<add key="SmtpFrom" value="vbgirl0527@gmail.com"/>

</appSettings>

EmailService:

namespace bugTracker.Models

{

/// <summary>

/// Represents a e-mail sender

/// </summary>

public class EmailService : IIdentityMessageService

{

//Getting the settings from our private.config setup in web.config

private string SmtpHost = ConfigurationManager.

AppSettings["SmtpHost"];

private int SmtpPort = Convert.ToInt32(ConfigurationManager.AppSettings["SmtpPort"]);

private string SmtpUsername = ConfigurationManager.AppSettings["SmtpUsername"];

private string SmtpPassword = ConfigurationManager.AppSettings["SmtpPassword"];

private string SmtpFrom = ConfigurationManager.AppSettings["SmtpFrom"];

/// <summary>

/// Sends an e-mail

/// </summary>

/// <param name="to">The destination of the e-mail</param>

/// <param name="body">The body of the e-mail</param>

/// <param name="subject">The subject of the e-mail</param>

public void Send(string to,

string body,

string subject)

{

//Creates a MailMessage required to send messages

var message = new MailMessage(SmtpFrom, to);

message.Body = body;

message.Subject = subject;

message.IsBodyHtml = true;

//Creates a SmtpClient required to handle the communication

//between our application and the SMTP Server

var smtpClient = new SmtpClient(SmtpHost, SmtpPort);

smtpClient.Credentials =

new NetworkCredential(SmtpUsername, SmtpPassword);

smtpClient.EnableSsl = true;

//Send the message

smtpClient.Send(message);

}

/// <summary>

/// Required by Microsoft Interface IIdentityMessageService in order to work with Identity Framework

/// </summary>

/// <param name="message">Object containing the necessary information to send e-mails</param>

/// <returns></returns>

public Task SendAsync(IdentityMessage message)

{

//Caling our sync method in a async way.

return Task.Run(() =>

Send(message.Destination, message.Body, message.Subject));

}

}

}

AccountController.cs

if (ModelState.IsValid)

{

var user = await UserManager.FindByNameAsync(model.Email);

if (user == null /\*|| !(await UserManager.IsEmailConfirmedAsync(user.Id))\*/)

{

// Don't reveal that the user does not exist or is not confirmed

return View("ForgotPasswordConfirmation");

}

// For more information on how to enable account confirmation and password reset please visit https://go.microsoft.com/fwlink/?LinkID=320771

// Send an email with this link

string code = await UserManager.GeneratePasswordResetTokenAsync(user.Id);

var callbackUrl = Url.Action("ResetPassword", "Account", new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);

await UserManager.SendEmailAsync(user.Id, "Reset Password", "Please reset your password by clicking <a href=\"" + callbackUrl + "\">here</a>");

return RedirectToAction("ForgotPasswordConfirmation", "Account");

}

□ **Roles**

□ The database should be seeded with **Admin**, **Project Manager**, **Developer**,and **Submitter** roles.

□ The database should be seeded with an **Admin User** with the following credentials:

**Username**: [admin@mybugtracker.com](mailto:admin@mybugtracker.com)

**Password**: Password-1

□ There should be a screen where **Administrators only** must be able to assign and unassign users to and from roles.

ManageRoleviewModel.cs

RoleManagementController

□ **Projects**

□ **Create** Projects – Administrators and Project Managers must be able to create new projects.

□ **Edit** Projects – Administrators and Project Managers must be able to edit existing projects.

□ **Administrators and Project Managers** must be able to assign and unassigned users to and from projects.

□ **List** Projects – Administrators, Project Managers, Developers, and Submitters must be able to view a list of projects they are assigned to. Administrators and Project Managers must be able to view a separate list of all projects. This list should display the name of the project, how many users are assigned, how many tickets the project have, when it was created and when it was last updated.

□ All screens should display the data in a tabular format and should allow the user to search, paginate and sort data. jQuery DataTables or any other plugin of your choice can be used to implement this functionality.

### Sample Site:

[http://zbugtracker.azurewebsites.net](http://zbugtracker.azurewebsites.net/)

### Due Date

Monday, April 8th, 2019 at 8:45