# Final Project – Part #1

1. Create a new database in SQLServer Management Studio called BugTracker.
2. Set up the following tables with the indicated column names and data types. Based on this information, determine the relationships between the tables and make sure to set up the appropriate Primary Key and Identity columns.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Column Name** | **Data Type** |
| Users | Id | int |
| UserName | nvarchar(50) |
| FirstName | nvarchar(50) |
| LastName | nvarchar(50) |
| Projects | Id | int |
| Name | nvarchar(50) |
| ProjectUsers | Id | int |
| ProjectId | int |
| UserName | nvarchar(50) |
| Tickets | Id | int |
| Title | nvarchar(50) |
| Description | nvarchar(MAX) |
| Created | datetimeoffset |
| Updated | datetimeoffset |
| ProjectId | int |
| TicketTypeId | int |
| TicketPriorityId | int |
| TicketStatusId | int |
| OwnerUserName | nvarchar(50) |
| AssignedToUserName | nvarchar(50) |
| TicketTypes | Id | int |
| Name | nvarchar(50) |
| TicketPriorities | Id | int |
| Name | nvarchar(50) |
| TicketStatuses | Id | int |
| Name | nvarchar(50) |
| TicketHistories | Id | int |
| TicketId | int |
| Property | nvarchar(50) |
| OldValue | nvarchar(MAX) |
| NewValue | nvarchar(MAX) |
| Changed | datetimeoffset |
| UserName | nvarchar(50) |
| TicketNotifications | Id | int |
| TicketId | int |
| UserName | nvarchar(50) |
| TicketComments | Id | int |
| TicketId | Int |
| Comment | nvarchar(MAX) |
| Created | datetimeoffset |
| UserName | nvarchar(50) |
| TicketAttachments | Id | int |
| TicketId | int |
| FilePath | nvarchar(MAX) |
| Description | nvarchar(MAX) |
| Created | datetimeoffset |
| UserName | nvarchar(50) |

1. Create a NEW MVC project in Visual Studio. Name this project BugTracker.
2. Like you did in your previous projects, set this one up to accept third-party logins (i.e. Facebook, Google, Linked-In).
3. Again, as in your previous projects, modify the Login and Register pages to accept a login with EITHER a username or email and to make your usernames and emails distinct from each other.
4. Set up a new data source for your project. Use your newly created BugTracker database.
5. Import the data model from your database into your project (ADO.NET Entity Data Model).
6. Modify the RegisterViewModel, the Register view, and the Register controller as you did in the OrderTracker project, so that each time a user registers with your website you create a new Users data entry in your database (just like Customers in OrderTracker). Remember to set the Users.UserName attribute equal to the ApplicationUser.UserName attribute from the MVC Entity Framework.
7. Test your program. Make sure you can register new users, that a BTUsers entry is made, and that you can log out and log back in.
8. Now you’re ready for Part 2.

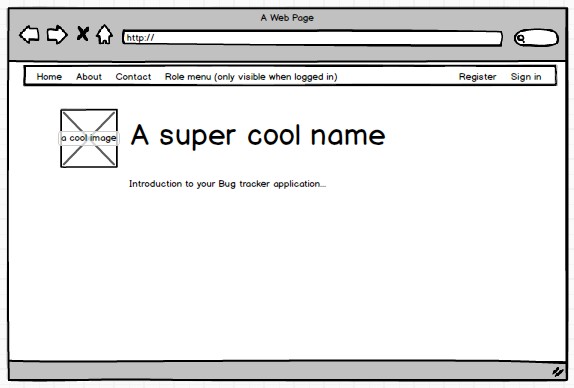
# Part 2

## Task 1: View Models

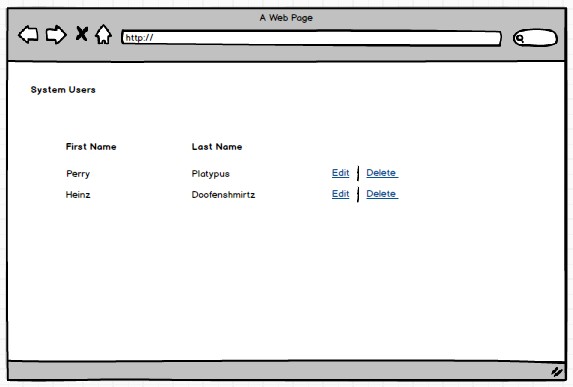
You are going to need to build a few view models (RolesViewModel, UserRolesViewModel, BTUserViewModel). Try to determine what to include in these view models by looking at the related tables in your database and the mockups below.

### Mockups for Views (Users and Roles)

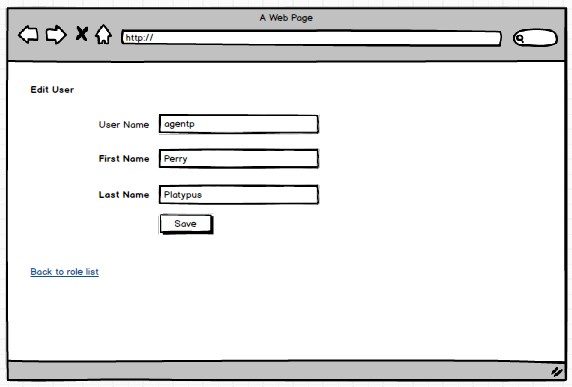
Landing page. This is your home page, found under Views > Home > Index.cshtml. Work on making it cool, but only after you’ve got working code in the critical areas.



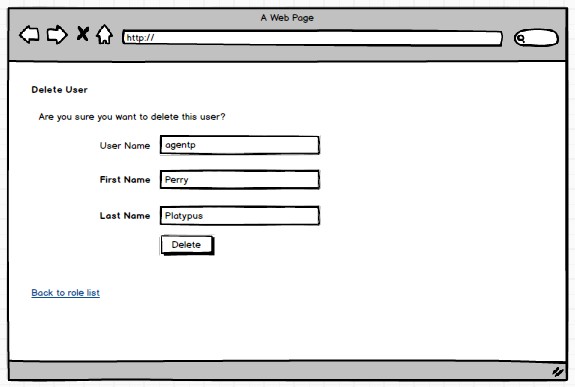
User List



Edit User



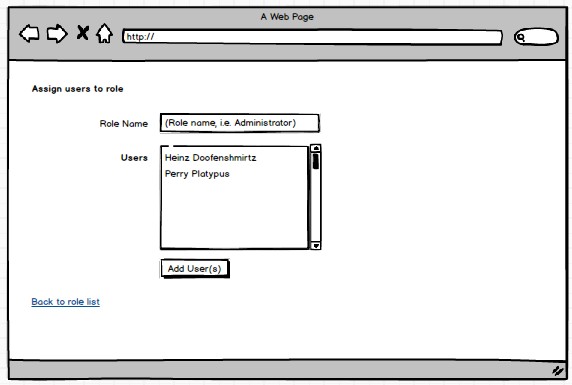
Delete User



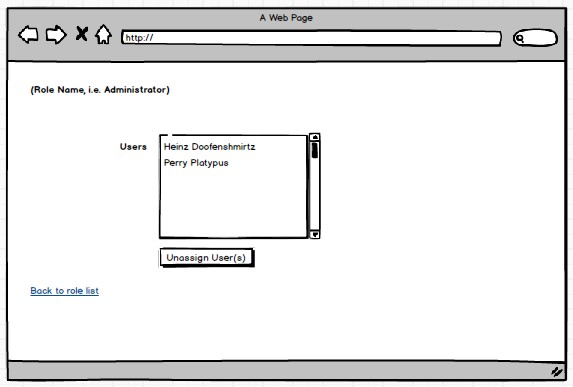
Role List



Assign user(s) to role



Unassign user(s) from role



## Task 2

Build the views represented by the mockups. Generate default views by selecting the appropriate model (you created them in task 1) and an appropriate template (create, edit, delete, list), then modify those views to suit your needs. Put the views in an appropriately named folder under Views (i.e. Users, Roles).

## Task 3

Now that you have built these views, to make the process of creating, assigning, and unassigning roles MUCH simpler, you need to add three methods to the ApplicationUser class, found in the IdentityModels.cs file.

First, management of user roles is handled through an object called the UserManager. Add the following statement at the beginning of the ApplicationUser class (found in IdentityModels.cs):

private UserManager<ApplicationUser> usermanager = new UserManager<ApplicationUser>(  
 new UserStore<ApplicationUser>(new ApplicationDbContext()));

With the UserManager in place, you need to create three methods to make managing individual user roles very simple:

1. AddUserToRole
2. RemoveUserFromRole
3. IsUserInRole

All three will receive a string variable (the role name) as a parameter. There are corresponding methods within the UserManager that you will call. We are just embedding these calls inside the ApplicationUser class to make the job just a little bit simpler. So, for example, the AddUserToRole method would look something like this:

public bool AddUserToRole(string roleName)

{

var idResult = usermanager.AddToRole(this.Id, roleName);

return idResult.Succeeded;

}

Now write the other two on your own.