

# DEVELOPER GUIDE

## Symphony Ltd. Private Institute – Developer Guide

### Table of Contents:

<b>Content</b>	<b>Page</b>
Preface	2
Log In	3
Dashboard	5
Home Page - Section 1 Form	8
Courses	10

# DEVELOPER GUIDE

## Preface

The Symphony Ltd. Online Certification Portal Project is very pleased to welcome you! This project is a collaborative effort between Mesum Bin Shaukat and Syed Ahmed Rooshan to create an online portal for Symphony Ltd., one of the most renowned institutes that offer courses in different IT's and software certifications.

The Symphony Ltd. Online Certification Portal seeks to simplify and digitalize the institute's daily business processes through an online interface for course curriculum information, entrance examinations results (FAQ section, etc.) The project integrates user and administration functionality in order to offer a complete, convenient system.

### **Open Source Availability:**

It is with great satisfaction that we announce the availability of this project as an open source initiative on the GitHub. The public repository can be accessed at <https://github.com/mesumbinshaukat/EProject-Management-Portal>

This decision, to make this project open source complies with our principle of knowledge sharing and collaborative work. Developers, students and enthusiasts are invited to explore the site in all its features as well as submit their contributions. We open the source our code to create an environment of learning, enhancement and progress within the community at large.

If you want to clone, fork or contribute the project on GitHub just do it. With anticipation, we are very keen to see what changes and improvements the community can introduce within Symphony Ltd. Online Certification Portal.

Happy coding!

Mesum Bin Shaukat

# DEVELOPER GUIDE

## Log In Page

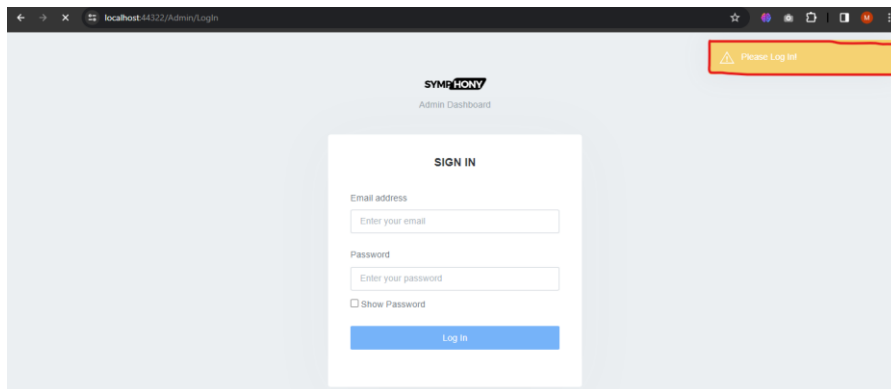


Figure 1

- If the Admin wants to access the dashboard then he/she has to Log In first.
- If Admin will try to access the dashboard or any other pages directly without logging in, then he'll be redirected to **/Admin/Login/** every time.
- Log In page is totally secure from **CSRF**, **XSS** and **SQL Injections**.
- There's no sign up page for the Admin, because according to the scenario there's only one Admin, and admin credentials are stored in the database directly by the developer.
- Once Admin is able to type the correct credentials and then try to log in then he'll be redirected to Dashboard and Session will start, but if session is not destroyed and he/she still tries to access the log in page then they'll be prevented from doing this and will be redirected to **/Admin/Index/**.

```
1 <form method="post" action="@Url.Action("Login")">
2     @Html.AntiForgeryToken()
3
4     <div asp-validation-summary="ModelOnly" class="text-danger"></div>
5     <div class="mb-3">
6         <label for="emailaddress" class="form-label">Email address</label>
7         <input class="form-control" type="text" name="emailaddress" required placeholder="Enter your email">
8     </div>
9
10    <div class="mb-3">
11        <label for="password" class="form-label">Password</label>
12        <input class="form-control" type="password" name="password" required placeholder="Enter your password" id="password">
13
14        <div class="mt-2">
15            <input type="checkbox" id="show"> <label>Show Password</label>
16        </div>
17        <div class="mt-2" style="display:none;">
18            <input type="checkbox" id="hide"> <label>Hide Password</label>
19        </div>
20    </div>
21
22    <div class="mb-3 d-grid text-center">
23        <button class="btn btn-primary" type="submit"> Log In </button>
24    </div>
25 </form>
```

Figure 2

# DEVELOPER GUIDE

```
1 [HttpGet]
2 [AllowAnonymous]
3 public IActionResult LogIn()
4 {
5     if (HttpContext.Session.GetString("s_email") != null)
6     {
7
8         return RedirectToAction("Index", "Admin");
9     }
10    IEnumerable<Admin> objAdmin = _db._Admin;
11
12    return View();
13
14 }
15
16
17
18 [HttpPost]
19 [ValidateAntiForgeryToken]
20 public IActionResult LogIn(Admin data, string emailaddress, string password)
21 {
22     var check_user = _db._Admin.Where(i => i.Email == emailaddress).FirstOrDefault();
23
24     var test_email = check_user ?? null;
25     if (test_email != null)
26     {
27         var check_user_password = _db._Admin.Where(i => i.Password == password).FirstOrDefault();
28
29         var test_password = check_user_password ?? null;
30
31         var verify = check_user.Password == password;
32
33         if (verify && test_password != null)
34         {
35
36             var verify_pass = verify.ToString();
37
38             HttpContext.Session.SetString("s_email", emailaddress); // Set Email In Session
39             HttpContext.Session.SetString("s_pass_verify", verify_pass); // Set Password In Session
40
41
42             TempData["success"] = "Logged In Successfully";
43             return RedirectToAction("Index", "Admin");
44
45         }
46         TempData["failed"] = "Invalid Password";
47         return RedirectToAction("LogIn");
48
49     }
50     TempData["failed"] = "Invalid Email";
51     return RedirectToAction("LogIn");
52
53 }
54
55 }
```

Figure 3

# DEVELOPER GUIDE

## Dashboard

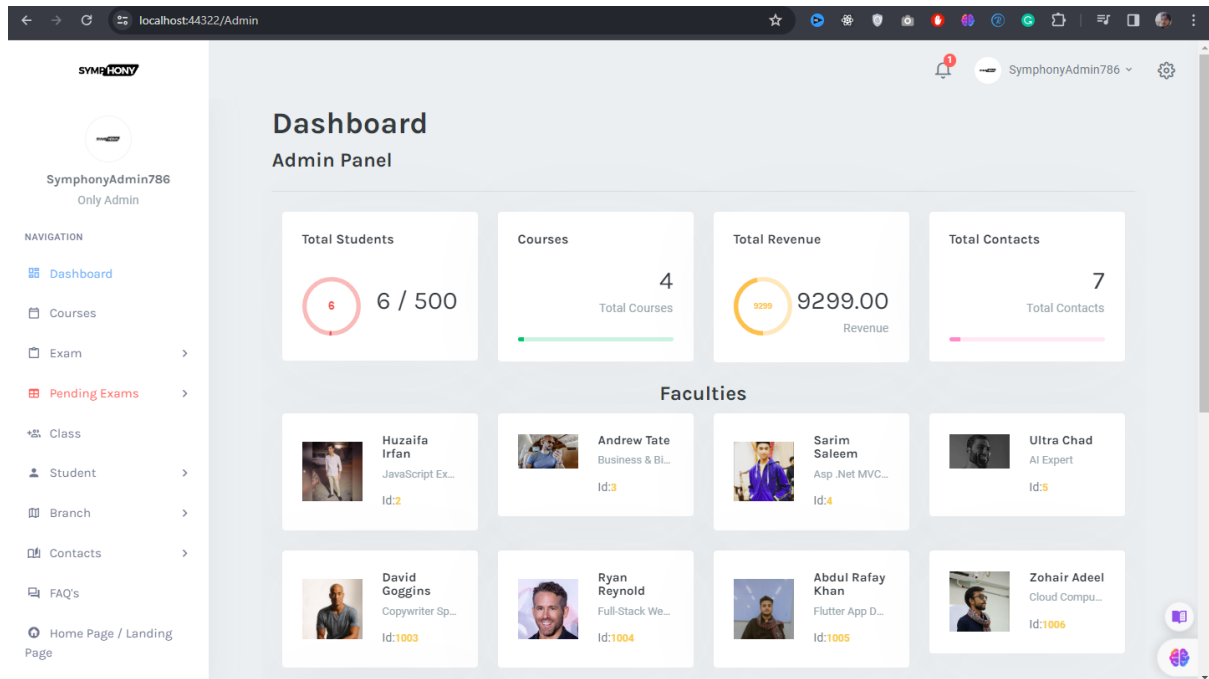


Figure 4

- Section 1: Total Student, Courses Total revenue & Total Contacts are dynamic.
- Section 2: Faculties details are dynamic, images are accurate to corresponding faculties.

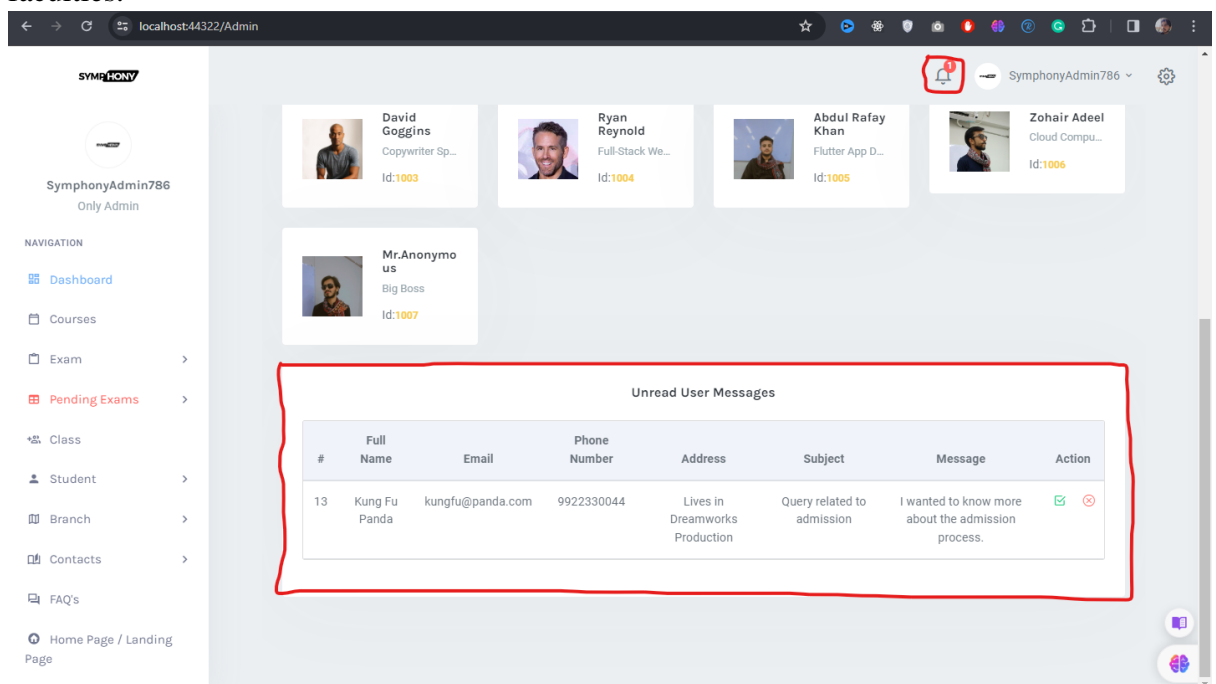


Figure 5

# DEVELOPER GUIDE

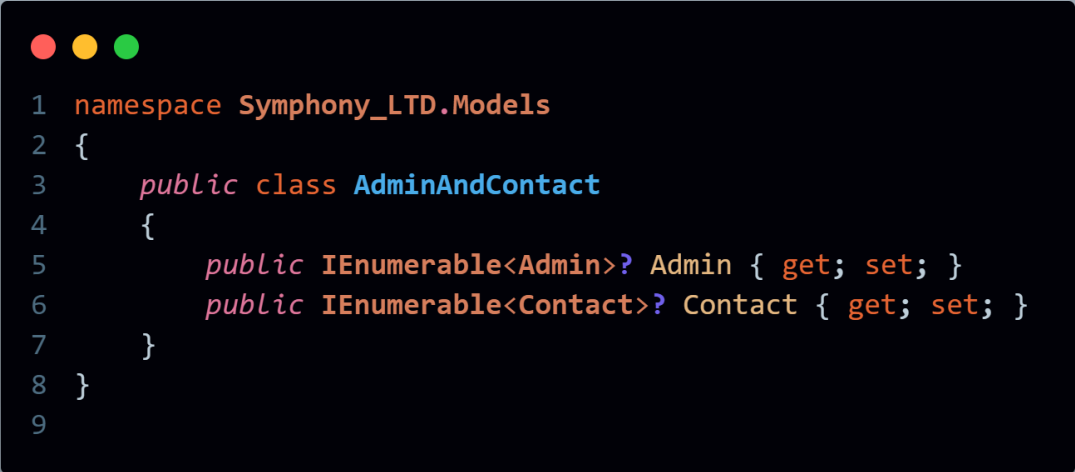
- Section 3: Contact section is also dynamic. Admin can delete the user query, and also can update the user query status to seen, depending on his whims, although the notification icon is also dynamic, it also show the unread user query messages.

```
1 public IActionResult Index()
2 {
3     if (HttpContext.Session.GetString("s_email") != null)
4     {
5         ViewBag.Email = HttpContext.Session.GetString("s_email").ToString();
6         ViewBag.Pass = HttpContext.Session.GetString("s_pass_verify").ToString();
7
8         var adminData = _db._Admin.ToList();
9         var contactData = _db._Contact.ToList();
10
11         var viewModel = new AdminAndContact
12         {
13             Admin = adminData,
14             Contact = contactData
15         };
16
17         var totalCourses = _db.Courses.Count();
18         ViewBag.TotalCourses = totalCourses;
19
20         var studentPropertiesCount = _db.Students.Count();
21         ViewBag.TotalColumns = studentPropertiesCount;
22
23         var totalContacts = _db._Contact.Count();
24         ViewBag.TotalContacts = totalContacts;
25
26         var allFaculties = _db._Faculty.ToList();
27         ViewBag.TotalFaculties = allFaculties;
28
29         var totalRevenue = _db.Courses.ToList();
30         decimal total = 0;
31
32         foreach (var i in totalRevenue)
33         {
34             if (i.CourseFee.HasValue)
35             {
36                 total += i.CourseFee.Value;
37             }
38             else
39             {
40                 ViewBag.TotalRevenue = "0";
41             }
42         }
43
44         ViewBag.TotalRevenue = total;
45
46         var user_details = _db._Admin.FirstOrDefault();
47
48         if (user_details != null)
49         {
50             ViewBag.Username = user_details.Name;
51         }
52
53         return View("Index", viewModel);
54     }
55     TempData["failed"] = "Please Log In!";
56     return RedirectToAction("Login");
57 }
58
59
60 }
```

Figure 6

# DEVELOPER GUIDE

- In Figure 6, line number 11, there are multiple models. We have made another model where we've inserted those two models in one model so we can use it in one View. Down below is the model that is concatenating the Admin and Contact models together in one model:



```
1 namespace Symphony_LTD.Models
2 {
3     public class AdminAndContact
4     {
5         public IEnumerable<Admin>? Admin { get; set; }
6         public IEnumerable<Contact>? Contact { get; set; }
7     }
8 }
9
```

Figure 7

- Down below is the code, how we imported this model in **Index.cshtml** or **Home Page**:



```
1 @model Symphony_LTD.Models.AdminAndContact
```

Figure 8

# DEVELOPER GUIDE

## Landing Page / Home Page

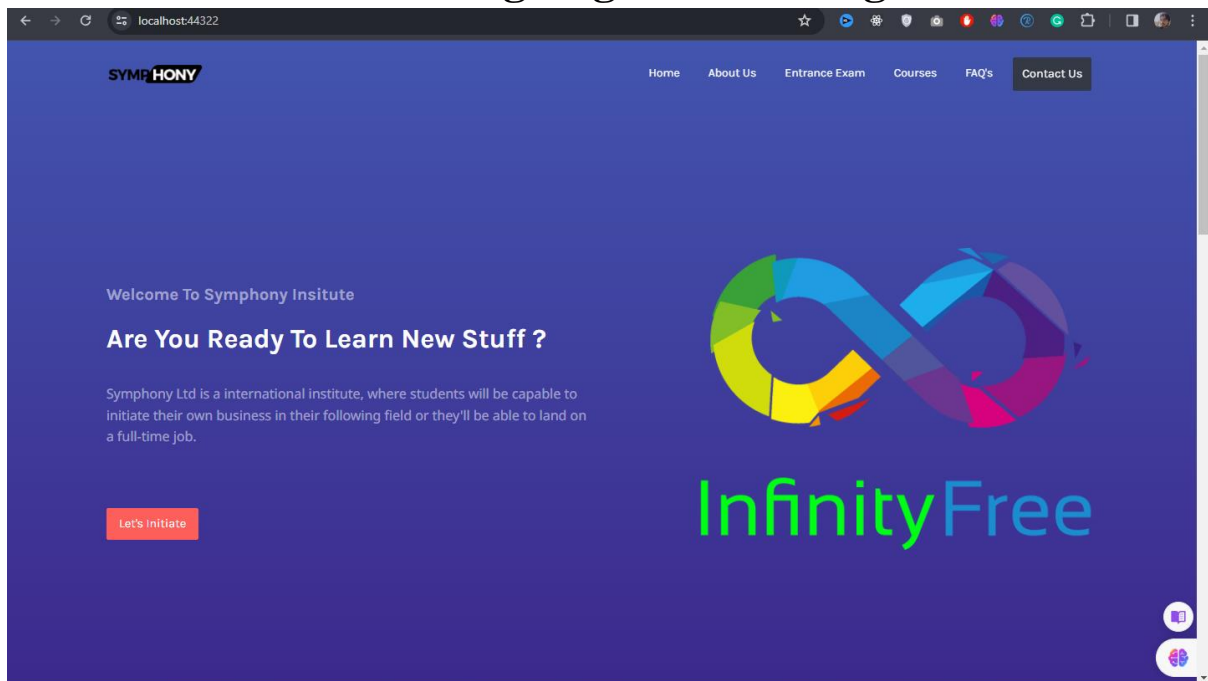


Figure 9

- This is the landing page, which user will be landed on if he hits the base url.
- This home page, section one can be fully customized according to the need or requirement.
- Heading 5, Heading 2, Paragraph, Button Value and HREF, and Image can be replaced or changed according to the Admin's desire.
- Down below is the form where Admin can change the home page's section 1:

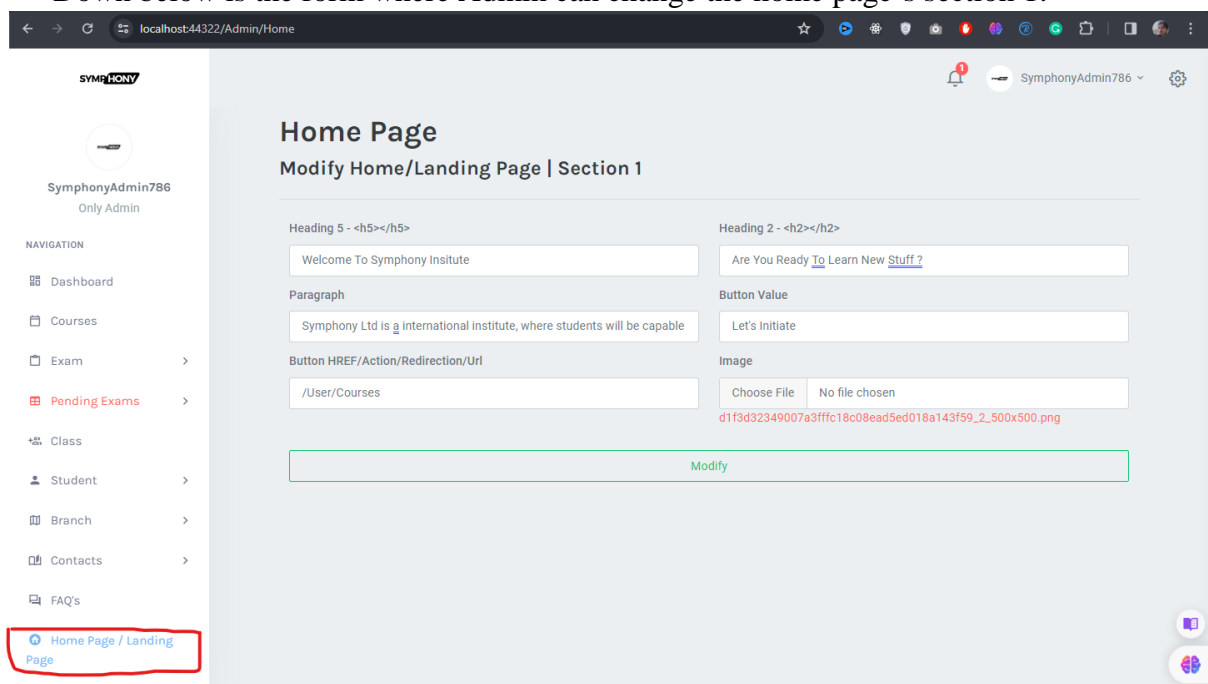


Figure 10



# DEVELOPER GUIDE

```
1 public IActionResult Home ()
2 {
3     if (HttpContext.Session.GetString("s_email") != null)
4     {
5         var existing_data = _db._HomeSectionOne.FirstOrDefault();
6         if (existing_data != null)
7         {
8             ViewBag.H5 = existing_data.H5;
9             ViewBag.H2 = existing_data.H2;
10            ViewBag.Paragraph = existing_data.Paragraph;
11            ViewBag.BtnVal = existing_data.BtnVal;
12            ViewBag.BtnAction = existing_data.BtnAction;
13            ViewBag.Img = existing_data.Img;
14        }
15
16        ViewBag.Email = HttpContext.Session.GetString("s_email").ToString();
17        ViewBag.Pass = HttpContext.Session.GetString("s_pass_verify").ToString();
18        var user_details = _db._Admin.FirstOrDefault();
19
20        if (user_details != null)
21        {
22            ViewBag.Username = user_details.Name;
23        }
24        return View();
25    }
26    TempData["failed"] = "Please Log In!";
27    return RedirectToAction("LogIn");
28 }
29
30 [HttpPost]
31 [ValidateAntiForgeryToken]
32 public IActionResult Home (HomeSectionOne data, IFormFile _img)
33 {
34     var existing_data = _db._HomeSectionOne.FirstOrDefault();
35     if (_img != null)
36     {
37         string path = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/media/home");
38         string filepath = Path.Combine(path, _img.FileName);
39
40         if (!Directory.Exists(path))
41         {
42             Directory.CreateDirectory(path);
43         }
44
45         var stream = new FileStream(filepath, FileMode.Create);
46         _img.CopyTo(stream);
47         string? filename = _img.FileName;
48         data.Img = filename;
49     }
50     else
51     {
52         string defaultFilename = existing_data.Img;
53
54         string extension = Path.GetExtension(defaultFilename)?.ToLowerInvariant();
55
56         string contentType;
57         switch (extension)
58         {
59             case ".svg":
60                 contentType = "image/svg+xml";
61                 break;
62             case ".png":
63                 contentType = "image/png";
64                 break;
65             case ".jpg":
66             case ".jpeg":
67                 contentType = "image/jpeg";
68                 break;
69             default:
70                 contentType = "application/octet-stream";
71                 break;
72         }
73
74         _img = new FormFile(null, 0, 0, "Img", defaultFilename)
75         {
76             Headers = new HeaderDictionary(),
77             ContentType = contentType,
78         };
79     }
80 }
81 }
```

Figure 11

# DEVELOPER GUIDE

## Courses

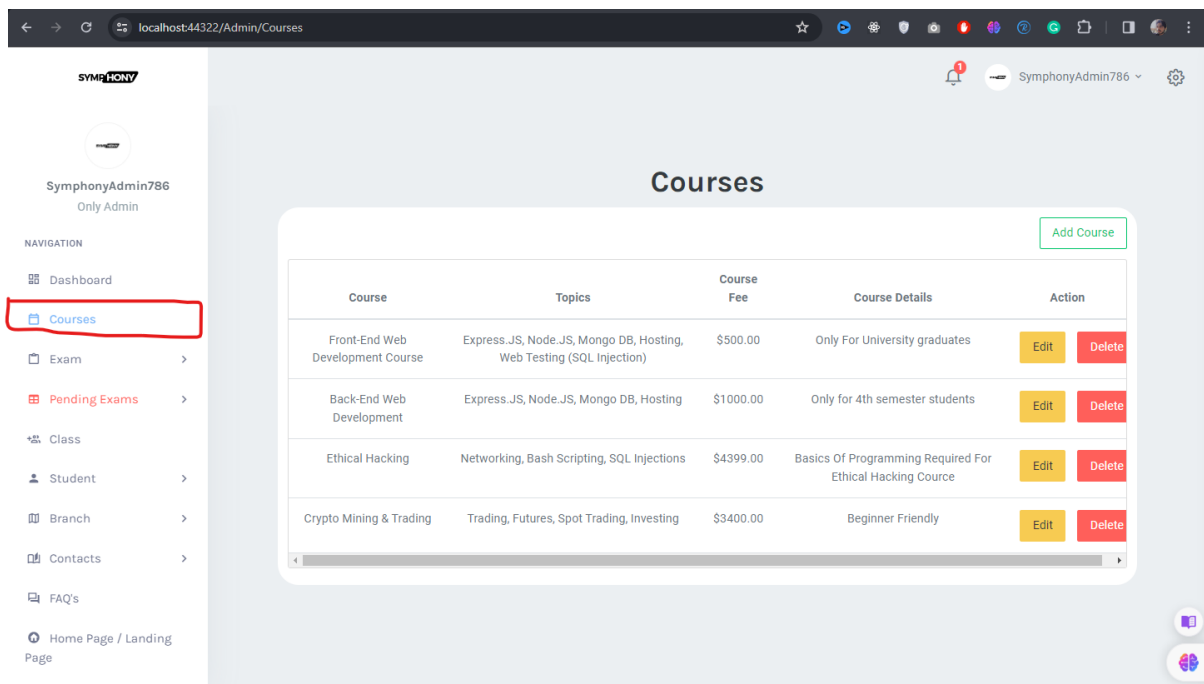


Figure 12

- These are the courses which Admin will be able to add just by clicking on that green button “Add Course”, it will redirect them to </Admin/AddCourse/>
- This **Courses** page has edit and delete option too.
- Down below is action method code for **Courses**:

```
1 public IActionResult Courses()
2 {
3     if (HttpContext.Session.GetString("s_email") != null)
4     {
5         ViewBag.Email = HttpContext.Session.GetString("s_email").ToString();
6         ViewBag.Pass = HttpContext.Session.GetString("s_pass_verify").ToString();
7         IEnumerable<Course> courses = _db.Courses;
8         var user_details = _db._Admin.FirstOrDefault();
9
10        if (user_details != null)
11        {
12            ViewBag.Username = user_details.Name;
13        }
14        return View(courses);
15    }
16    TempData["failed"] = "Please Log In!";
17    return RedirectToAction("LogIn");
18 }
19
20 }
```

Figure 13

# DEVELOPER GUIDE

- Admin can easily add the course depending on him/her.
- Down below is the Add Course form, where Admin can add the course:

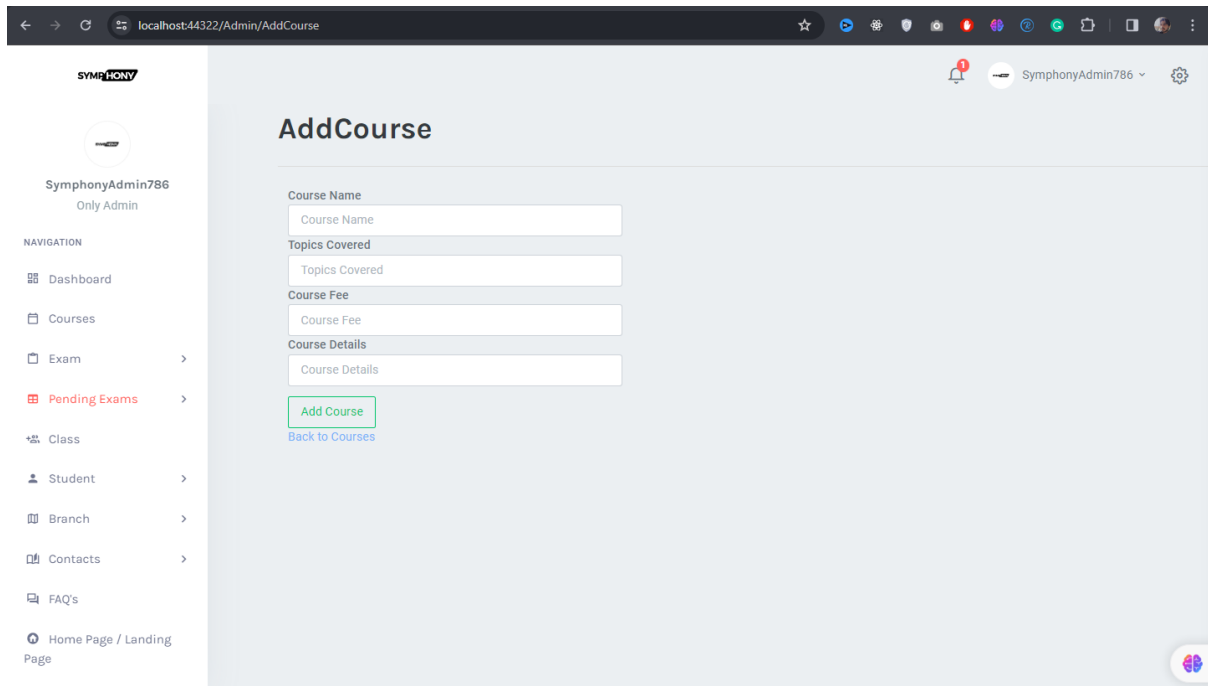


Figure 14

- Down below is Add Course action method code:

```
1 public IActionResult AddCourse()
2 {
3     if (HttpContext.Session.GetString("s_email") != null)
4     {
5         ViewBag.Email = HttpContext.Session.GetString("s_email").ToString();
6         ViewBag.Pass = HttpContext.Session.GetString("s_pass_verify").ToString();
7         var user_details = _db._Admin.FirstOrDefault();
8
9         if (user_details != null)
10        {
11            ViewBag.Username = user_details.Name;
12        }
13    }
14    return View();
15
16    }
17    TempData["failed"] = "Please Log In!";
18    return RedirectToAction("LogIn");
19 }
20
21 [HttpPost]
22 [ValidateAntiForgeryToken]
23 public IActionResult AddCourse(Course obj)
24 {
25     if (ModelState.IsValid)
26     {
27         _db.Courses.Add(obj);
28         _db.SaveChanges();
29         return RedirectToAction("Courses");
30     }
31     else
32     {
33         return View(obj);
34     }
35     //return View("LogIn");
36 }
```

Figure 15

# DEVELOPER GUIDE

- Once courses are added then it will be shown to user on two pages, on Home Page and Courses page.
- Down below is images where courses are being shown.

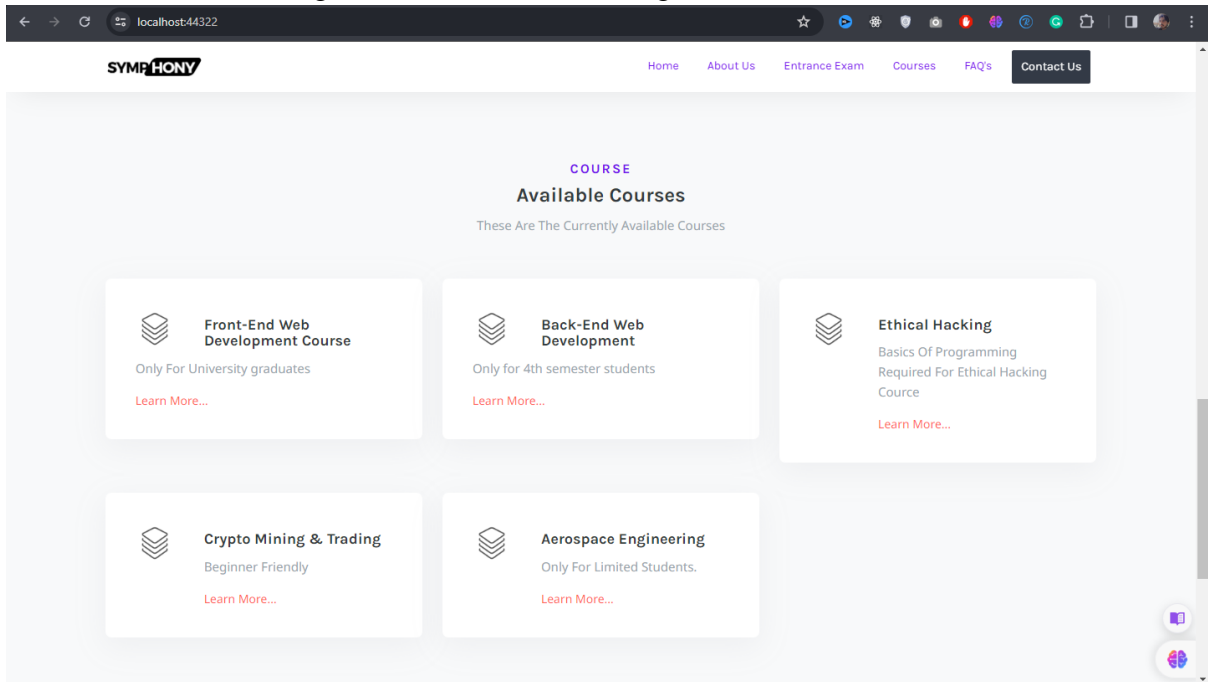


Figure 17 - Home Page

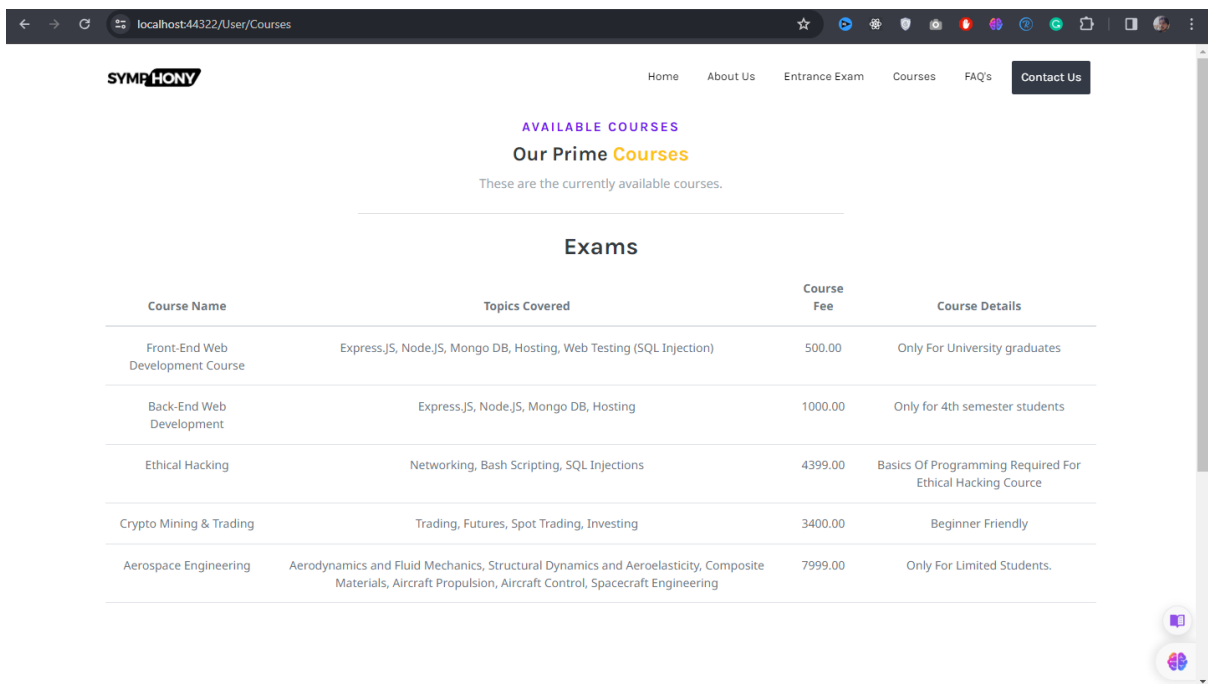


Figure 16 - Courses

# DEVELOPER GUIDE

- Admin can easily edit these courses.

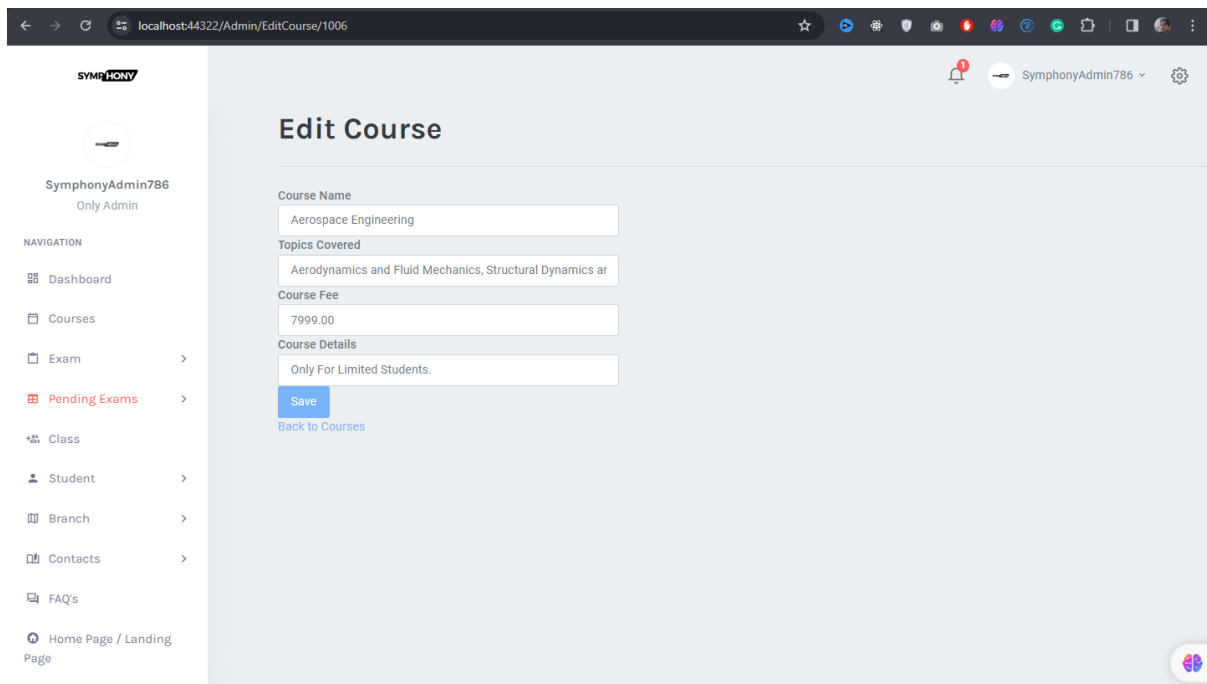


Figure 18

- Edit course action method code:

```
1 public IActionResult EditCourse(int? id)
2 {
3     if (HttpContext.Session.GetString("s_email") != null)
4     {
5         if (id == null || id == 0)
6         {
7             return NotFound();
8         }
9         var studentFromDb = _db.Courses.Find(id);
10
11         if (studentFromDb == null)
12         {
13             return NotFound();
14         }
15         ViewBag.Email = HttpContext.Session.GetString("s_email").ToString();
16         ViewBag.Pass = HttpContext.Session.GetString("s_pass_verify").ToString();
17         ViewBag.Course = _db.Courses.ToList();
18         var user_details = _db._Admin.FirstOrDefault();
19
20         if (user_details != null)
21         {
22             ViewBag.Username = user_details.Name;
23         }
24         return View(studentFromDb);
25     }
26
27     TempData["failed"] = "Please Log In!";
28     return RedirectToAction("Login");
29 }
30
31 [HttpPost]
32 [ValidateAntiForgeryToken]
33 public IActionResult EditCourse(Course obj)
34 {
35     if (ModelState.IsValid)
36     {
37         _db.Courses.Update(obj);
38         _db.SaveChanges();
39         return RedirectToAction("Courses");
40     }
41
42     return View();
43 }
```

Figure 19

# DEVELOPER GUIDE

- Admin can also easily delete the courses.

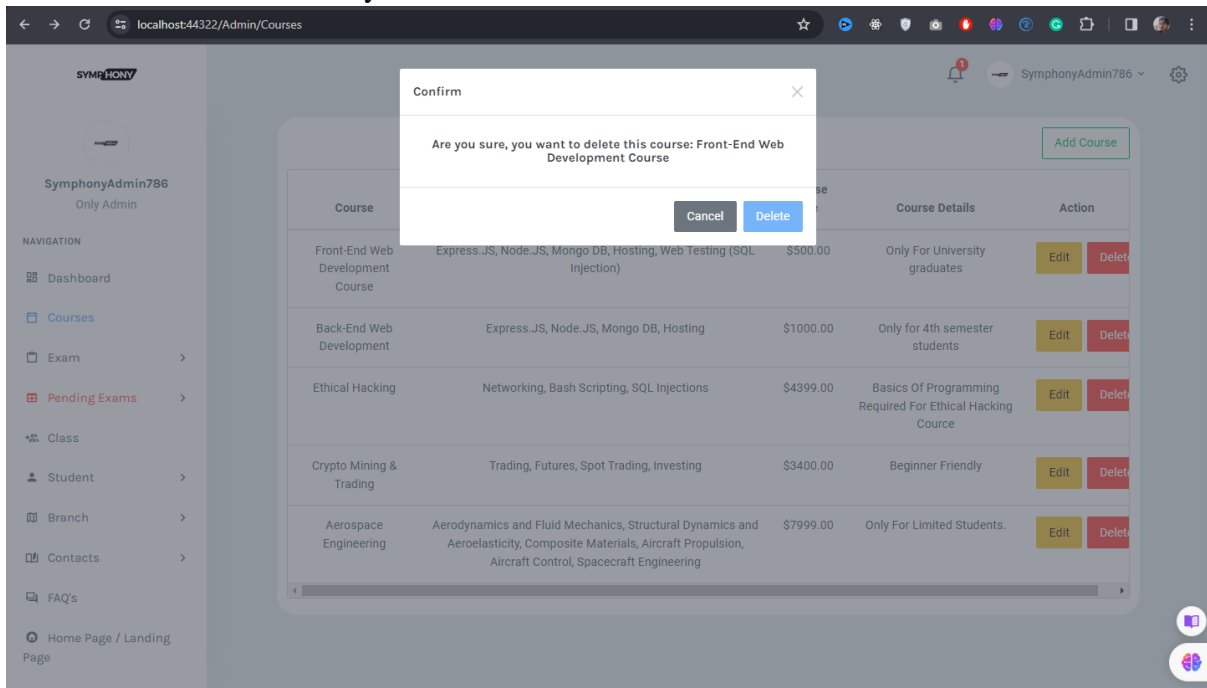


Figure 20

- Delete course action method code:

```
1 public IActionResult DeleteCourse(int? id)
2     {
3         if (HttpContext.Session.GetString("s_email") != null)
4         {
5             var obj = _db.Courses.Find(id);
6             if (id != null)
7             {
8                 _db.Courses.Remove(obj);
9                 _db.SaveChanges();
10                return RedirectToAction("Courses");
11            }
12        }
13
14        return BadRequest();
15    }
```

Figure 21

# DEVELOPER GUIDE

## Conclusion:

This **Developer Guide** will updated, and more content will be added. For latest documentation, consider checking out the github repository.

Repo: <https://github.com/mesumbinshaukat/EProject-Management-Portal>