UNIT TEST

Table of Unit Test

|  |  |
| --- | --- |
| **Unit Test** | **Test Method** |
| UT-01 | Class AccountController : ActionResult Login(string returnUrl) |
| UT-02 | Class AccountController : ActionResult Login(LoginModel model, string returnUrl) |
| UT-03 | Class AccountController : ActionResult LogOff() |
| UT-04 | Class AccountController : ActionResult Register() |
| UT-05 | Class AccountController : ActionResult Register(RegisterModel model) |
| UT-06 | Class AccountController : ActionResult Disassociate(string provider, string providerUserId) |
| UT-07 | Class AccountController : ActionResult Manage(ManageMessageId? message) |
| UT-08 | Class AccountController : ActionResult Manage(LocalPasswordModel model) |
| UT-09 | Class AccountController : ActionResult ExternalLogin(string provider, string returnUrl) |
| UT-10 | Class AccountController : ActionResult ExternalLoginCallback(string returnUrl) |
| UT-11 | Class AccountController : ActionResult ExternalLoginConfirmation(RegisterExternalLoginModel model, string returnUrl) |
| UT-12 | Class AccountController : ActionResult ExternalLoginFailure() |
| UT-13 | Class AccountController : ActionResult ExternalLoginsList(string returnUrl) |
| UT-14 | Class AccountController : ActionResult RemoveExternalLogins() |
| UT-15 | Class AccountController : ActionResult Redirect(string returnUrl) |
| UT-16 | Class AccountController : enum ManageMessageID |
| UT-17 | Class AccountController : internal ExternalLoginResult : ActionResult  ExternalLoginResult(string provider, string returnUrl),  Provider {get; private set;}, ReturnUrl {get; private set;},  override void ExecuteResult(ControllerContext context) |
| UT-18 | Class AccountController : ErrorCodeToString(MembershipCreateStatus createStatus) |
| UT-19 | Class AdminController : ActionResult Login(string returnUrl) |
| UT-20 | Class AdminController : ActionResult Login(AdminAccount model,string returnUrl) |
| UT-21 | Class AdminController : ActionResult Logout() |
| UT-22 | Class AdminController : ViewResult Index() |

|  |  |
| --- | --- |
| **Unit Test** | **Test Method** |
| UT-23 | Class AdminController : ViewResult IndexManaging() |
| UT-24 | Class AdminController : ViewResult AddWelcomeImage() |
| UT-25 | Class AdminController : ViewResult AddGalleryImage() |
| UT-26 | Class AdminController : ActionResult DeleteWelcomeImage(int ImageID) |
| UT-27 | Class AdminController : ViewResult GalleryManaging() |
| UT-28 | Class AdminController : ViewResult EditBackgroundInfo(int backgroundId) |
| UT-29 | Class AdminController : ViewResult BackgroundManaging() |
| UT-30 | Class AdminController : ViewResult AddActivityClip() |
| UT-31 | Class AdminController : ViewResult AddAnnualEvent() |
| UT-32 | Class AdminController : ViewResult AddOtherEvent() |
| UT-33 | Class AdminController : ViewResult ReplyDhammaQA() |
| UT-34 | Class AdminController : ViewResult ActivityClipManaging() |
| UT-35 | Class AdminController : ViewResult DhammaQAManaging() |
| UT-36 | Class AdminController : ViewResult AnnualEventManaging() |
| UT-37 | Class AdminController : ViewResult OtherEventManaging() |
| UT-38 | Class AdminController : ActionResult DeleteGalleryImage(int ImageID) |
| UT-39 | Class AdminController : ActionResult DeleteActivityClip(int clipID) |
| UT-40 | Class AdminController : ActionResult DeleteDhammaQA(int QAID) |
| UT-41 | Class AdminController : ActionResult DeleteAnnualEvent(int EventID) |
| UT-42 | Class AdminController : ActionResult DeleteOtherEvent(int EventID) |
| UT-43 | Class AdminController : ActionResult IndexManaging(WelcomeImage welcomeImage, HttpPostedFileBase image) |
| UT-44 | Class AdminController : ActionResult Index(GalleryImage galleryImage, HttpPostedFileBase image) |
| UT-45 | Class AdminController : ActionResult EditBackground(Background background) |
| UT-46 | Class AdminController : ActionResult AddAnnualEvent(AnnualEvent annualEvent) |
| UT-47 | Class AdminController : ActionResult AddOtherEvent(OtherEvent otherEvent) |
| UT-48 | Class AdminController : ActionResult AddActivityClip(ActivityClip activityClip) |
| UT-49 | Class AdminController : ActionResult ReplyDhammaQuestion(DhammaQA dhammaQA) |

|  |  |
| --- | --- |
| **Unit Test** | **Test Method** |
| UT-50 | Class HomeController : FileContentResult GetWelcomeImage(int imageId) |
| UT-51 | Class HomeController : FileContentResult GetGalleryImage(int imageId) |
| UT-52 | Class HomeController : ActionResult Index() |
| UT-53 | Class HomeController : ActionResult Background() |
| UT-54 | Class HomeController : ActionResult AnnualEvent() |
| UT-55 | Class HomeController : ActionResult AnnualEventDetail(int aEventID) |
| UT-56 | Class HomeController : ActionResult DhammaContent() |
| UT-57 | Class HomeController : ActionResult DhammaQA() |
| UT-58 | Class HomeController : ViewResult SendDhammaQ() |
| UT-59 | Class HomeController : ActionResult SendDhammaQ(DhammaQA dhammaQA) |
| UT-60 | Class HomeController : ActionResult Gallery() |
| UT-61 | Class HomeController : ActionResult ActivityClip() |
| UT-62 | Class HomeController : ActionResult InternetTV() |
| UT-63 | Class HomeController : ActionResult Map() |
| UT-64 | Class HomeController : ActionResult Contact() |
| UT-65 | Class HomeController : ActionResult Contact(Contact \_objModelMail) |

**UNIT TEST**

**UT-01** Class AccountController : ActionResult Login(string returnUrl)

**Test Date** 16 May 2015

**Description** Test for view the Login page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.ReturnUrl = returnUrl  return View(); | ViewBag.ReturnUrl = returnUrl  return View(); | PASS |

**UT-02** Class AccountController : ActionResult Login(LoginModel model, string returnUrl)

**Test Date** 16 May 2015

**Description** Test for checking login username and password

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid && WebSecurity.Login(model.UserName, model.Password, persistCookie: model.RememberMe))  {  return RedirectToLocal(returnUrl);  }  // If we got this far, something failed, redisplay form  ModelState.AddModelError("", "The user name or password provided is incorrect.");  return View(model); | if (ModelState.IsValid && WebSecurity.Login(model.UserName, model.Password, persistCookie: model.RememberMe))  {  return RedirectToLocal(returnUrl);  }  // If we got this far, something failed, redisplay form  ModelState.AddModelError("", "The user name or password provided is incorrect.");  return View(model); | PASS |

**UT-03** Class AccountController : ActionResult LogOff()

**Test Date** 16 May 2015

**Description** Test for Log off

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | WebSecurity.Logout();  return RedirectToAction("Index", "Home"); | WebSecurity.Logout();  return RedirectToAction("Index", "Home"); | PASS |

**UT-04** Class AccountController : ActionResult Register()

**Test Date** 16 May 2015

**Description** Test for Register view

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(); | return View(); | PASS |

**UT-05** Class AccountController : ActionResult Register(RegisterModel model)

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | string ownerAccount = OAuthWebSecurity.GetUserName(provider, providerUserId);  ManageMessageId? message = null;  if (ownerAccount == User.Identity.Name){  using (var scope = new TransactionScope(TransactionScopeOption.Required, new TransactionOptions { IsolationLevel = IsolationLevel.Serializable })){  bool hasLocalAccount = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));  if (hasLocalAccount || OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name).Count > 1){  OAuthWebSecurity.DeleteAccount(provider, providerUserId);  scope.Complete();  message = ManageMessageId.RemoveLoginSuccess; }}}  return RedirectToAction("Manage", new { Message = message });} | string ownerAccount = OAuthWebSecurity.GetUserName(provider, providerUserId);  ManageMessageId? message = null;  if (ownerAccount == User.Identity.Name){  using (var scope = new TransactionScope(TransactionScopeOption.Required, new TransactionOptions { IsolationLevel = IsolationLevel.Serializable })){  bool hasLocalAccount = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));  if (hasLocalAccount || OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name).Count > 1){  OAuthWebSecurity.DeleteAccount(provider, providerUserId);  scope.Complete();  message = ManageMessageId.RemoveLoginSuccess; }}}  return RedirectToAction("Manage", new { Message = message });} | PASS |

**Test Date** 16 May 2015

**Description** Test for Model Register view

**UT-06** Class AccountController : ActionResult Disassociate(string provider, string providerUserId)

**Test Date** 16 May 2015

**Description** Test for redirect to home page of member

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid) {  try {  WebSecurity.CreateUserAndAccount(model.UserName, model.Password); WebSecurity.Login(model.UserName, model.Password);  return RedirectToAction("Index", “Home"); }  catch (MembershipCreateUserException e) {  ModelState.AddModelError("", ErrorCodeToString(e.StatusCode)); } }  return View(model); | if (ModelState.IsValid) {  try {  WebSecurity.CreateUserAndAccount(model.UserName, model.Password); WebSecurity.Login(model.UserName, model.Password);  return RedirectToAction("Index", “Home"); }  catch (MembershipCreateUserException e) {  ModelState.AddModelError("", ErrorCodeToString(e.StatusCode)); } }  return View(model); | PASS |

**UT-07** Class AccountController : ActionResult Manage(ManageMessageId? message)

**Test Date** 16 May 2015

**Description** Test for showing the massage

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.StatusMessage =  message == ManageMessageId.ChangePasswordSuccess ? "Your password has been changed."  : message == ManageMessageId.SetPasswordSuccess ? "Your password has been set."  : message == ManageMessageId.RemoveLoginSuccess ? "The external login was removed."  : "";  ViewBag.HasLocalPassword = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));  ViewBag.ReturnUrl = Url.Action("Manage");  return View(); | ViewBag.StatusMessage =  message == ManageMessageId.ChangePasswordSuccess ? "Your password has been changed."  : message == ManageMessageId.SetPasswordSuccess ? "Your password has been set."  : message == ManageMessageId.RemoveLoginSuccess ? "The external login was removed."  : "";  ViewBag.HasLocalPassword = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));  ViewBag.ReturnUrl = Url.Action("Manage");  return View(); | PASS |

**UT-08** Class AccountController : ActionResult Manage(LocalPasswordModel model)

**Test Date** 16 May 2015

**Description** Test for verify password

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | bool hasLocalAccount = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));  ViewBag.HasLocalPassword = hasLocalAccount;  ViewBag.ReturnUrl = Url.Action("Manage");  if (hasLocalAccount){  if (ModelState.IsValid){  bool changePasswordSucceeded;  try{ changePasswordSucceeded = WebSecurity.ChangePassword(User.Identity.Name, model.OldPassword, model.NewPassword);  } catch (Exception e) { changePasswordSucceeded = false; }  if (changePasswordSucceeded){  return RedirectToAction("Manage", new { Message = ManageMessageId.ChangePasswordSuccess });} else{  ModelState.AddModelError("", "The current password is incorrect or the new password is invalid.”);}}} else{  ModelState state = ModelState[“OldPassword"];  if (state != null) {state.Errors.Clear(); }  if (ModelState.IsValid) { try { WebSecurity.CreateAccount(User.Identity.Name, model.NewPassword);  return RedirectToAction("Manage", new { Message = ManageMessageId.SetPasswordSuccess });} catch (Exception e){ModelState.AddModelError("", e);  return View(model); | bool hasLocalAccount = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));  ViewBag.HasLocalPassword = hasLocalAccount;  ViewBag.ReturnUrl = Url.Action("Manage");  if (hasLocalAccount){  if (ModelState.IsValid){  bool changePasswordSucceeded;  try{ changePasswordSucceeded = WebSecurity.ChangePassword(User.Identity.Name, model.OldPassword, model.NewPassword);  } catch (Exception e) { changePasswordSucceeded = false; }  if (changePasswordSucceeded){  return RedirectToAction("Manage", new { Message = ManageMessageId.ChangePasswordSuccess });} else{  ModelState.AddModelError("", "The current password is incorrect or the new password is invalid.”);}}} else{  ModelState state = ModelState[“OldPassword"];  if (state != null) {state.Errors.Clear(); }  if (ModelState.IsValid) { try { WebSecurity.CreateAccount(User.Identity.Name, model.NewPassword);  return RedirectToAction("Manage", new { Message = ManageMessageId.SetPasswordSuccess });} catch (Exception e){ModelState.AddModelError("", e);  return View(model); | PASS |

**UT-09** Class AccountController : ActionResult ExternalLogin(string provider, string returnUrl)

**Test Date** 16 May 2015

**Description** Test for external login

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return new ExternalLoginResult(provider, Url.Action("ExternalLoginCallback", new { ReturnUrl = returnUrl })); | return new ExternalLoginResult(provider, Url.Action("ExternalLoginCallback", new { ReturnUrl = returnUrl })); | PASS |

**UT-10** Class AccountController : ActionResult ExternalLoginCallback(string returnUrl)

**Test Date** 16 May 2015

**Description** Test for external login

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | AuthenticationResult result = OAuthWebSecurity.VerifyAuthentication(Url.Action("ExternalLoginCallback", new { ReturnUrl = returnUrl }));  if (!result.IsSuccessful){  return RedirectToAction("ExternalLoginFailure");}if (OAuthWebSecurity.Login(result.Provider, result.ProviderUserId, createPersistentCookie: false)){  return RedirectToLocal(returnUrl);}  if (User.Identity.IsAuthenticated){ OAuthWebSecurity.CreateOrUpdateAccount(result.Provider, result.ProviderUserId, User.Identity.Name);  return RedirectToLocal(returnUrl);} else{  string loginData = OAuthWebSecurity.SerializeProviderUserId(result.Provider, result.ProviderUserId);  ViewBag.ProviderDisplayName = OAuthWebSecurity.GetOAuthClientData(result.Provider).DisplayName;  ViewBag.ReturnUrl = returnUrl;  return View("ExternalLoginConfirmation", new RegisterExternalLoginModel { UserName = result.UserName, ExternalLoginData = loginData }); | AuthenticationResult result = OAuthWebSecurity.VerifyAuthentication(Url.Action("ExternalLoginCallback", new { ReturnUrl = returnUrl }));  if (!result.IsSuccessful){  return RedirectToAction("ExternalLoginFailure");}if (OAuthWebSecurity.Login(result.Provider, result.ProviderUserId, createPersistentCookie: false)){  return RedirectToLocal(returnUrl);}  if (User.Identity.IsAuthenticated){ OAuthWebSecurity.CreateOrUpdateAccount(result.Provider, result.ProviderUserId, User.Identity.Name);  return RedirectToLocal(returnUrl);} else{  string loginData = OAuthWebSecurity.SerializeProviderUserId(result.Provider, result.ProviderUserId);  ViewBag.ProviderDisplayName = OAuthWebSecurity.GetOAuthClientData(result.Provider).DisplayName;  ViewBag.ReturnUrl = returnUrl;  return View("ExternalLoginConfirmation", new RegisterExternalLoginModel { UserName = result.UserName, ExternalLoginData = loginData }); | PASS |

**UT-11** Class AccountController : ActionResult ExternalLoginConfirmation(RegisterExternalLoginModel model, string returnUrl)

**Test Date** 16 May 2015

**Description** Test for confirm register of new member

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | string provider = null; string providerUserId = null;  if (User.Identity.IsAuthenticated || !OAuthWebSecurity.TryDeserializeProviderUserId(model.ExternalLoginData, out provider, out providerUserId)) {  return RedirectToAction("Manage");}  if (ModelState.IsValid) {  using (UsersContext db = new UsersContext()) {  UserProfile user = db.UserProfiles.FirstOrDefault(u => u.UserName.ToLower() == model.UserName.ToLower());  if (user == null) {  db.UserProfiles.Add(new UserProfile { UserName = model.UserName });  db.SaveChanges();  OAuthWebSecurity.CreateOrUpdateAccount(provider, providerUserId, model.UserName); OAuthWebSecurity.Login(provider, providerUserId, createPersistentCookie: false);  return RedirectToLocal(returnUrl); } else{ ModelState.AddModelError("UserName", "User name already exists. Please enter a different user name.”); }}}  ViewBag.ProviderDisplayName = OAuthWebSecurity.GetOAuthClientData(provider).DisplayName;  ViewBag.ReturnUrl = returnUrl;  return View(model); | string provider = null; string providerUserId = null;  if (User.Identity.IsAuthenticated || !OAuthWebSecurity.TryDeserializeProviderUserId(model.ExternalLoginData, out provider, out providerUserId)) {  return RedirectToAction("Manage");}  if (ModelState.IsValid) {  using (UsersContext db = new UsersContext()) {  UserProfile user = db.UserProfiles.FirstOrDefault(u => u.UserName.ToLower() == model.UserName.ToLower());  if (user == null) {  db.UserProfiles.Add(new UserProfile { UserName = model.UserName });  db.SaveChanges();  OAuthWebSecurity.CreateOrUpdateAccount(provider, providerUserId, model.UserName); OAuthWebSecurity.Login(provider, providerUserId, createPersistentCookie: false);  return RedirectToLocal(returnUrl); } else{ ModelState.AddModelError("UserName", "User name already exists. Please enter a different user name.”); }}}  ViewBag.ProviderDisplayName = OAuthWebSecurity.GetOAuthClientData(provider).DisplayName;  ViewBag.ReturnUrl = returnUrl;  return View(model); | PASS |

**UT-12** Class AccountController : ActionResult ExternalLoginFailure()

**Test Date** 16 May 2015

**Description** Test for provide failure page if cannot login

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(); | return View(); | PASS |

**UT-13** Class AccountController : ActionResult ExternalLoginsList(string returnUrl)

**Test Date** 16 May 2015

**Description** Test for show list of external login

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.ReturnUrl = returnUrl;  return PartialView("\_ExternalLoginsListPartial", OAuthWebSecurity.RegisteredClientData); | ViewBag.ReturnUrl = returnUrl;  return PartialView("\_ExternalLoginsListPartial", OAuthWebSecurity.RegisteredClientData); | PASS |

**UT-14** Class AccountController : ActionResult RemoveExternalLogins()

**Test Date** 16 May 2015

**Description** Test for remove external login

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ICollection<OAuthAccount> accounts = OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name);  List<ExternalLogin> externalLogins = new List<ExternalLogin>();  foreach (OAuthAccount account in accounts){  AuthenticationClientData clientData = OAuthWebSecurity.GetOAuthClientData(account.Provider);  externalLogins.Add(new ExternalLogin {  Provider = account.Provider, ProviderDisplayName = clientData.DisplayName, ProviderUserId = account.ProviderUserId,}); }  ViewBag.ShowRemoveButton = externalLogins.Count > 1 || OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));  return PartialView("\_RemoveExternalLoginsPartial", externalLogins); | ICollection<OAuthAccount> accounts = OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name);  List<ExternalLogin> externalLogins = new List<ExternalLogin>();  foreach (OAuthAccount account in accounts){  AuthenticationClientData clientData = OAuthWebSecurity.GetOAuthClientData(account.Provider);  externalLogins.Add(new ExternalLogin {  Provider = account.Provider, ProviderDisplayName = clientData.DisplayName, ProviderUserId = account.ProviderUserId,}); }  ViewBag.ShowRemoveButton = externalLogins.Count > 1 || OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));  return PartialView("\_RemoveExternalLoginsPartial", externalLogins); | PASS |

**UT-15** Class AccountController : ActionResult RedirectToLocal(string returnUrl)

**Test Date** 16 May 2015

**Description** Test for redirect to Index page and home page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (Url.IsLocalUrl(returnUrl)) {  return Redirect(returnUrl);  } else {  return RedirectToAction("Index", “Home"); } | if (Url.IsLocalUrl(returnUrl)) {  return Redirect(returnUrl);  } else {  return RedirectToAction("Index", “Home"); } | PASS |

**UT-16** Class AccountController : enum ManageMessageID

**Test Date** 16 May 2015

**Description** Test for manage message ID

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ChangePasswordSuccess,  SetPasswordSuccess,  RemoveLoginSuccess, | ChangePasswordSuccess,  SetPasswordSuccess,  RemoveLoginSuccess, | PASS |

**UT-17** Class AccountController : internal ExternalLoginResult : ActionResult

ExternalLoginResult(string provider, string returnUrl),

Provider {get; private set;}, ReturnUrl {get; private set;},

override void ExecuteResult(ControllerContext context)

**Test Date** 16 May 2015

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | public ExternalLoginResult(string provider, string returnUrl) {  Provider = provider;  ReturnUrl = returnUrl; }  public string Provider { get; private set; }  public string ReturnUrl { get; private set; }  public override void ExecuteResult(ControllerContext context) {  OAuthWebSecurity.RequestAuthentication(Provider, ReturnUrl);} | public ExternalLoginResult(string provider, string returnUrl) {  Provider = provider;  ReturnUrl = returnUrl; }  public string Provider { get; private set; }  public string ReturnUrl { get; private set; }  public override void ExecuteResult(ControllerContext context) {  OAuthWebSecurity.RequestAuthentication(Provider, ReturnUrl);} | PASS |

**Description** Test for result of external login

**UT-18** Class AccountController : ErrorCodeToString(MembershipCreateStatus createStatus)

**Test Date** 16 May 2015

**Description** Test for status of membership

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | switch (createStatus) {  case MembershipCreateStatus.DuplicateUserName: return "User name already exists. Please enter a different user name.”;  case MembershipCreateStatus.DuplicateEmail: return "A user name for that e-mail address already exists. Please enter a different e-mail address.”;  case MembershipCreateStatus.InvalidPassword: return "The password provided is invalid. Please enter a valid password value.";  case MembershipCreateStatus.InvalidEmail: return "The e-mail address provided is invalid. Please check the value and try again.";  case MembershipCreateStatus.InvalidAnswer: return "The password retrieval answer provided is invalid. Please check the value and try again.";  case MembershipCreateStatus.InvalidQuestion:return "The password retrieval question provided is invalid. Please check the value and try again.";  case MembershipCreateStatus.InvalidUserName: return "The user name provided is invalid. Please check the value and try again.”;  case MembershipCreateStatus.ProviderError: return "The authentication provider returned an error. Please verify your entry and try again. If the problem persists, please contact your system administrator.";case MembershipCreateStatus.UserRejected:return "The user creation request has been canceled. Please verify your entry and try again. If the problem persists, please contact your system administrator.”;  default: return "An unknown error occurred. Please verify your entry and try again. If the problem persists, please contact your system administrator.”; | switch (createStatus) {  case MembershipCreateStatus.DuplicateUserName: return "User name already exists. Please enter a different user name.”;  case MembershipCreateStatus.DuplicateEmail: return "A user name for that e-mail address already exists. Please enter a different e-mail address.”;  case MembershipCreateStatus.InvalidPassword: return "The password provided is invalid. Please enter a valid password value.";  case MembershipCreateStatus.InvalidEmail: return "The e-mail address provided is invalid. Please check the value and try again.";  case MembershipCreateStatus.InvalidAnswer: return "The password retrieval answer provided is invalid. Please check the value and try again.";  case MembershipCreateStatus.InvalidQuestion:return "The password retrieval question provided is invalid. Please check the value and try again.";  case MembershipCreateStatus.InvalidUserName: return "The user name provided is invalid. Please check the value and try again.”;  case MembershipCreateStatus.ProviderError: return "The authentication provider returned an error. Please verify your entry and try again. If the problem persists, please contact your system administrator.";case MembershipCreateStatus.UserRejected:return "The user creation request has been canceled. Please verify your entry and try again. If the problem persists, please contact your system administrator.”;  default: return "An unknown error occurred. Please verify your entry and try again. If the problem persists, please contact your system administrator.”; | PASS |

**UT-19** Class AdminController : ActionResult Login(string returnUrl)

**Test Date** 16 May 2015

**Description** Test for Login page of administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.ReturnUrl = returnUrl;  return View(); | ViewBag.ReturnUrl = returnUrl;  return View(); | PASS |

**UT-20** Class AdminController : ActionResult Login(AdminAccount model,string returnUrl)

**Test Date** 16 May 2015

**Description** Test for Login of administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (model.UserName == "admin" && model.Password == “admin1234") {  Session["LoginStatus"] = “True";  return RedirectToAction(“IndexManaging");  } else {  System.Windows.Forms.MessageBox.Show("ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง กรุณากรอกใหม่อีกครั้ง”);  return View(); } | if (model.UserName == "admin" && model.Password == “admin1234") {  Session["LoginStatus"] = “True";  return RedirectToAction(“IndexManaging");  } else {  System.Windows.Forms.MessageBox.Show("ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง กรุณากรอกใหม่อีกครั้ง”);  return View(); } | PASS |

**UT-21** Class AdminController : ActionResult Logout()

**Test Date** 16 May 2015

**Description** Test for Logout of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | Session.Remove(“LoginStatus");  return RedirectToAction(“Index"); } | Session.Remove(“LoginStatus”);  return RedirectToAction(“Index"); } | PASS |

**UT-22** Class AdminController : ViewResult Index()

**Test Date** 16 May 2015

**Description** Test for show index page of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(); | return View(); | PASS |

**UT-23** Class AdminController : ViewResult IndexManaging()

**Test Date** 16 May 2015

**Description** Test for manage images of home page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(repository.WelcomeImages); | return View(repository.WelcomeImages); | PASS |

**UT-24** Class AdminController : ViewResult AddWelcomeImage()

**Test Date** 16 May 2015

**Description** Test for adding images

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(“AddWelcomeImage”, new WelcomeImage); | return View(“AddWelcomeImage”, new WelcomeImage); | PASS |

**UT-25** Class AdminController : ViewResult AddGalleryImage()

**Test Date** 16 May 2015

**Description** Test for adding images to gallery

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(“AddGalleryImage”, new GalleryImage); | return View(“AddGalleryImage”, new GalleryImage); | PASS |

**UT-26** Class AdminController : ActionResult DeleteWelcomeImage(int ImageID)

**Test Date** 16 May 2015

**Description** Test for delete images of home page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | WelcomeImage deletedWelcomeImage = repository.DeleteWelcomeImage(ImageID)  if (deletedWelcomeImage != null{  TempData["message"] = "Delete success”; }  return RedirectToAction("IndexManaging"); | WelcomeImage deletedWelcomeImage = repository.DeleteWelcomeImage(ImageID)  if (deletedWelcomeImage != null{  TempData["message"] = "Delete success”; }  return RedirectToAction("IndexManaging"); | PASS |

**UT-27** Class AdminController : ViewResult GalleryManaging()

**Test Date** 16 May 2015

**Description** Test for view images into gallery

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(repository2.GalleryImages); | return View(repository2.GalleryImages); | PASS |

**UT-28** Class AdminController : ViewResult EditBackgroundInfo(int backgroundId)

**Test Date** 16 May 2015

**Description** Test for edit background of Wat Pa Thammapirom

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | Background background = repository3.Backgrounds.FirstOrDefault(p => p.BackgroundID == backgroundId);  return View(background); | Background background = repository3.Backgrounds.FirstOrDefault(p => p.BackgroundID == backgroundId);  return View(background); | PASS |

**UT-29** Class AdminController : ViewResult BackgroundManaging()

**Test Date** 16 May 2015

**Description** Test for view background of Wat Pa Thammapirom

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(repository3.Backgrounds); | return View(repository3.Backgrounds); | PASS |

**UT-30** Class AdminController : ViewResult AddActivityClip()

**Test Date** 16 May 2015

**Description** Test for adding activity clips

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View("AddActivityClip", new ActivityClip()); | return View("AddAnnualEvent", new AnnualEvent()); | PASS |

**UT-31** Class AdminController : ViewResult AddAnnualEvent()

**Test Date** 16 May 2015

**Description** Test for adding annual event

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View("AddAnnualEvent", new AnnualEvent()); | return View("AddAnnualEvent", new AnnualEvent()); | PASS |

**UT-32** Class AdminController : ViewResult AddOtherEvent()

**Test Date** 16 May 2015

**Description** Test for adding other event

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View("AddOtherEvent", new OtherEvent()); | return View("AddOtherEvent", new OtherEvent()); | PASS |

**UT-33** Class AdminController : ViewResult ReplyDhammaQA(int qaID)

**Test Date** 16 May 2015

**Description** Test for reply dharma question and answer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | DhammaQA dhammaQA = repository5.DhammaQAs.FirstOrDefault(p => p.QAID == qaID);  return View(dhammaQA); | DhammaQA dhammaQA = repository5.DhammaQAs.FirstOrDefault(p => p.QAID == qaID);  return View(dhammaQA); | PASS |

**UT-34** Class AdminController : ViewResult ActivityClipManaging()

**Test Date** 16 May 2015

**Description** Test for view activity clips

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(repository4.ActivityClips); | return View(repository4.ActivityClips); | PASS |

**UT-35** Class AdminController : ViewResult DhammaQAManaging()

**Test Date** 16 May 2015

**Description** Test for view dharma question and answer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(repository5.DhammaQAs); | return View(repository5.DhammaQAs); | PASS |

**UT-36** Class AdminController : ViewResult AnnualEventManaging()

**Test Date** 16 May 2015

**Description** Test for view annual event

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(repository6.AnnualEvents); | return View(repository6.AnnualEvents); | PASS |

**UT-37** Class AdminController : ViewResult OtherEventManaging()

**Test Date** 16 May 2015

**Description** Test for view other event

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View(repository7.OtherEvents); | return View(repository7.OtherEvents); | PASS |

**UT-38** Class AdminController : ActionResult DeleteGalleryImage(int ImageID)

**Test Date** 16 May 2015

**Description** Test for delete images in gallery

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | GalleryImage deletedGalleryImage = repository2.DeleteGalleryImage(ImageID);  if (deletedGalleryImage != null){ TempData["message"] = "Delete success”; }  return RedirectToAction(“Index"); | GalleryImage deletedGalleryImage = repository2.DeleteGalleryImage(ImageID);  if (deletedGalleryImage != null){ TempData["message"] = "Delete success”; }  return RedirectToAction(“Index"); | PASS |

**UT-39** Class AdminController : ActionResult DeleteActivityClip(int clipID)

**Test Date** 16 May 2015

**Description** Test for delete activity clip

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ActivityClip deletedActivityClip = repository4.DeleteActivityClip(clipID);  if (deletedActivityClip != null) { TempData["message"] = "Delete success”;}  return RedirectToAction("ActivityClipManaging"); | ActivityClip deletedActivityClip = repository4.DeleteActivityClip(clipID);  if (deletedActivityClip != null) { TempData["message"] = "Delete success”;}  return RedirectToAction("ActivityClipManaging"); | PASS |

**UT-40** Class AdminController : ActionResult DeleteDhammaQA()

**Test Date** 16 May 2015

**Description** Test for delete dhamma question and answer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | DhammaQA deletedDhammaQA =repository5.DeleteDhammaQA(QAID);  if (deletedDhammaQA != null) {  TempData["message"] = "Delete success”; }  return RedirectToAction("DhammaQAManaging"); | DhammaQA deletedDhammaQA =repository5.DeleteDhammaQA(QAID);  if (deletedDhammaQA != null) {  TempData["message"] = "Delete success”; }  return RedirectToAction("DhammaQAManaging"); | PASS |

**UT-41** Class AdminController : ActionResult DeleteAnnualEvent(int EventID)

**Test Date** 16 May 2015

**Description** Test for delete annual event

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | AnnualEvent deletedAnnualEvent =repository6.DeleteAnnualEvent(EventID);  if (deletedAnnualEvent != null) {  TempData["message"] = "Delete success”; }  return RedirectToAction(“AnnualEventManaging"); } | AnnualEvent deletedAnnualEvent =repository6.DeleteAnnualEvent(EventID);  if (deletedAnnualEvent != null) {  TempData["message"] = "Delete success”; }  return RedirectToAction(“AnnualEventManaging"); } | PASS |

**UT-42** Class AdminController : ActionResult DeleteOtherEvent(int EventID)

**Test Date** 16 May 2015

**Description** Test for delete other event

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | OtherEvent deletedOtherEvent = repository7.DeleteOtherEvent(EventID);  if (deletedOtherEvent != null) {  TempData["message"] = "Delete success”; }  return RedirectToAction("OtherEventManaging"); | OtherEvent deletedOtherEvent = repository7.DeleteOtherEvent(EventID);  if (deletedOtherEvent != null) {  TempData["message"] = "Delete success”; }  return RedirectToAction("OtherEventManaging"); | PASS |

**UT-43** Class AdminController : ActionResult IndexManaging(WelcomeImage welcomeImage, HttpPostedFileBase image)

**Test Date** 16 May 2015

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid{  if (image != null) {  welcomeImage.ImageMimeType = image.ContentType;  welcomeImage.ImageData = new byte[image.ContentLength];  image.InputStream.Read(welcomeImage.ImageData, 0, image.ContentLength); }  repository.SaveWelcomeImage(welcomeImage);  TempData["message"] = string.Format("{0} has been saved", welcomeImage.ImageID);  return RedirectToAction(“IndexManaging”); } else {  return View(welcomeImage); | if (ModelState.IsValid{  if (image != null) {  welcomeImage.ImageMimeType = image.ContentType;  welcomeImage.ImageData = new byte[image.ContentLength];  image.InputStream.Read(welcomeImage.ImageData, 0, image.ContentLength); }  repository.SaveWelcomeImage(welcomeImage);  TempData["message"] = string.Format("{0} has been saved", welcomeImage.ImageID);  return RedirectToAction(“IndexManaging”); } else {  return View(welcomeImage); | PASS |

**Description** Test for managing of index page of Administer

**UT-44** Class AdminController : ActionResult Index(GalleryImage galleryImage, HttpPostedFileBase image)

**Test Date** 16 May 2015

**Description** Test for saving images by managing of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid){  if (image != null){  galleryImage.ImageMimeType = image.ContentType; galleryImage.ImageData = new byte[image.ContentLength]; image.InputStream.Read(galleryImage.ImageData, 0, image.ContentLength); }  repository2.SaveGalleryImage(galleryImage); TempData["message"] = string.Format("{0} has been saved", galleryImage.ImageID);  return RedirectToAction(“Index"); } else {  return View(galleryImage); | if (ModelState.IsValid){  if (image != null){  galleryImage.ImageMimeType = image.ContentType; galleryImage.ImageData = new byte[image.ContentLength]; image.InputStream.Read(galleryImage.ImageData, 0, image.ContentLength); }  repository2.SaveGalleryImage(galleryImage); TempData["message"] = string.Format("{0} has been saved", galleryImage.ImageID);  return RedirectToAction(“Index"); } else {  return View(galleryImage); | PASS |

**UT-45** Class AdminController : ActionResult EditBackground(Background background)

**Test Date** 16 May 2015

**Description** Test for edit background by managing of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid) {  repository3.SaveBackgroundInfo(background);  return RedirectToAction(“BackgroundManaging"); } else {  return View(background); | if (ModelState.IsValid) {  repository3.SaveBackgroundInfo(background);  return RedirectToAction(“BackgroundManaging"); } else {  return View(background); | PASS |

**UT-46** Class AdminController : ActionResult AddAnnualEvent(AnnualEvent annualEvent)

**Test Date** 16 May 2015

**Description** Test for adding annual event by managing of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid) {  repository6.SaveAnnualEvent(annualEvent);  return RedirectToAction(“AnnualEventManaging"); }  else {  return View(annualEvent); } | if (ModelState.IsValid) {  repository6.SaveAnnualEvent(annualEvent);  return RedirectToAction(“AnnualEventManaging"); }  else {  return View(annualEvent); } | PASS |

**UT-47** Class AdminController : ActionResult AddOtherEvent(OtherEvent otherEvent)

**Test Date** 16 May 2015

**Description** Test for adding other event by managing of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid) {  repository7.SaveOtherEvent(otherEvent);  return RedirectToAction(“AnnualEventManaging”); }  else {  return View(otherEvent); } | if (ModelState.IsValid) {  repository7.SaveOtherEvent(otherEvent);  return RedirectToAction(“AnnualEventManaging”); }  else {  return View(otherEvent); } | PASS |

**UT-48** Class AdminController : ActionResult AddActivityClip(ActivityClip activityClip)

**Test Date** 16 May 2015

**Description** Test for adding activity clips by managing of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid) {  repository4.SaveActivityClip(activityClip);  return RedirectToAction(“ActivityClipManaging"); }  else {  return View(activityClip); } | if (ModelState.IsValid) {  repository4.SaveActivityClip(activityClip);  return RedirectToAction(“ActivityClipManaging"); }  else {  return View(activityClip); } | PASS |

**UT-49** Class AdminController : ActionResult ReplyDhammaQuestion(DhammaQA dhammaQA)

**Test Date** 16 May 2015

**Description** Test for reply dharma question and answer by managing of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid) {  repository5.SaveDhammaQA(dhammaQA);  return RedirectToAction(“DhammaQAManaging");  } else {  return View(dhammaQA); } | if (ModelState.IsValid) {  repository5.SaveDhammaQA(dhammaQA);  return RedirectToAction(“DhammaQAManaging");  } else {  return View(dhammaQA); } | PASS |

**UT-50** Class HomeController : FileContentResult GetWelcomeImage(int imageId)

**Test Date** 16 May 2015

**Description** Test for show slide images of home page by managing of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | WelcomeImage welimg = repository.WelcomeImages.FirstOrDefault(p => p.ImageID == imageId);  if (welimg != null) {  return File(welimg.ImageData, welimg.ImageMimeType);  } else {  return null;}} | WelcomeImage welimg = repository.WelcomeImages.FirstOrDefault(p => p.ImageID == imageId);  if (welimg != null) {  return File(welimg.ImageData, welimg.ImageMimeType);  } else {  return null;}} | Pass |

**UT-51** Class HomeController : FileContentResult GetGalleryImage(int imageId)

**Test Date** 16 May 2015

**Description** Test for get image of gallery by managing of Administer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | GalleryImage galimg = repository2.GalleryImages.FirstOrDefault(p => p.ImageID == imageId);  if (galimg != null) {  return File(galimg.ImageData, galimg.ImageMimeType);  } else {  return null; } } | GalleryImage galimg = repository2.GalleryImages.FirstOrDefault(p => p.ImageID == imageId);  if (galimg != null) {  return File(galimg.ImageData, galimg.ImageMimeType);  } else {  return null; } } | PASS |

**UT-52** Class HomeController : ActionResult Index()

**Test Date** 16 May 2015

**Description** Test for view home page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “หน้าแรก”;  return View(repository.WelcomeImages); | ViewBag.Message = “หน้าแรก”;  return View(repository.WelcomeImages); | PASS |

**UT-53** Class HomeController : ActionResult Background()

**Test Date** 16 May 2015

**Description** Test for view background page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “ประวัติวัด”;  return View(repository3.Backgrounds); | ViewBag.Message = “ประวัติวัด”;  return View(repository3.Backgrounds); | PASS |

**UT-54** Class HomeController : ActionResult AnnualEvent()

**Test Date** 16 May 2015

**Description** Test for view annual event page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “ข่าวสาร-กิจกรรม”;  return View(repository7.AnnualEvents); | ViewBag.Message = “ข่าวสาร-กิจกรรม”;  return View(repository7.AnnualEvents); | PASS |

**UT-55** Class HomeController : ActionResult AnnualEventDetail(int aEventID)

**Test Date** 16 May 2015

**Description** Test for view annual event detail page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | AnnualEvent annualEvent = repository7.AnnualEvents.FirstOrDefault(p => p.EventID == aEventID);  return View(annualEvent); | AnnualEvent annualEvent = repository7.AnnualEvents.FirstOrDefault(p => p.EventID == aEventID);  return View(annualEvent); | PASS |

**UT-56** Class HomeController : ActionResult DhammaContent()

**Test Date** 16 May 2015

**Description** Test for view dharma content page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “สาระธรรม”;  return View(); | ViewBag.Message = “สาระธรรม”;  return View(); | PASS |

**UT-57** Class HomeController : ActionResult DhammaQA()

**Test Date** 16 May 2015

**Description** Test for view dhamma question and answer page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “ถาม-ตอบธรรม”;  return View(repository5.DhammaQAs); | ViewBag.Message = “ถาม-ตอบธรรม”;  return View(repository5.DhammaQAs); | PASS |

**UT-58** Class HomeController : ViewResult SendDhammaQ()

**Test Date** 16 May 2015

**Description** Test for view sending dhamma question and answer page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | return View("SendDhammaQ", new DhammaQA()); | return View("SendDhammaQ", new DhammaQA()); | PASS |

**UT-59** Class HomeController : ActionResult SendDhammaQ(DhammaQA dhammaQA)

**Test Date** 16 May 2015

**Description** Test for send dhamma question and answer

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid) {  repository5.SaveDhammaQA(dhammaQA);  return RedirectToAction(“DhammaQA”); } else {  return View(dhammaQA); } | if (ModelState.IsValid) {  repository5.SaveDhammaQA(dhammaQA);  return RedirectToAction(“DhammaQA”); } else {  return View(dhammaQA); } | PASS |

**UT-60** Class HomeController : ActionResult Gallery()

**Test Date** 16 May 2015

**Description** Test for view gallery page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “แกลอรี่";  return View(repository2.GalleryImages); | ViewBag.Message = “แกลอรี่";  return View(repository2.GalleryImages); | PASS |

**UT-61** Class HomeController : ActionResult ActivityClip()

**Test Date** 16 May 2015

**Description** Test for view activity clip page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “วิดีโอกิจกรรม";  return View(repository4.ActivityClips); | ViewBag.Message = “วิดีโอกิจกรรม";  return View(repository4.ActivityClips); | PASS |

**UT-62** Class HomeController : ActionResult InternetTV()

**Test Date** 16 May 2015

**Description** Test for view internet TV page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “อินเตอร์เน็ตทีวี";  return View(); | ViewBag.Message = “อินเตอร์เน็ตทีวี";  return View(); | PASS |

**UT-63** Class HomeController : ActionResult Map()

**Test Date** 16 May 2015

**Description** Test for map page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “แผนที่วัด”;  return View(); | ViewBag.Message = “แผนที่วัด”;  return View(); | PASS |

**UT-64** Class HomeController : ActionResult Contact()

**Test Date** 16 May 2015

**Description** Test for view contact page

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | ViewBag.Message = “ติดต่อวัด”;  return View(“Contact”, new Contact()); | ViewBag.Message = “ติดต่อวัด”;  return View(“Contact”, new Contact()); | PASS |

**UT-65** Class HomeController : ActionResult Contact(Contact -objModelMail)

**Test Date** 16 May 2015

**Description** Test for sending email for contact

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | if (ModelState.IsValid) {  repository6.SaveContact(\_objModelMail);  MailMessage mail = new MailMessage(); mail.To.Add(“se552115005@vr.camt.info"); mail.From = new MailAddress(\_objModelMail.SenderEmail);  mail.Subject = \_objModelMail.ContactTitle;  string Body = "โปรดติตต่อกลับ: "+\_objModelMail.Sender+"\n อีเมลล์: "+\_objModelMail.SenderEmail+"\n เรื่อง: “+\_objModelMail.ContactMessage;  mail.Body = Body;  mail.IsBodyHtml = true;  SmtpClient smtp = new SmtpClient();  smtp.Host = "smtp.gmail.com";  smtp.Port = 587;  smtp.UseDefaultCredentials = true;  smtp.Credentials = new System.Net.NetworkCredential("se552115005@vr.camt.info","552115005");// Enter seders User name and password  smtp.EnableSsl = true;  smtp.Send(mail);  return View("Contact", \_objModelMail);} else {  return View(); } | if (ModelState.IsValid) {  repository6.SaveContact(\_objModelMail);  MailMessage mail = new MailMessage(); mail.To.Add(“se552115005@vr.camt.info"); mail.From = new MailAddress(\_objModelMail.SenderEmail);  mail.Subject = \_objModelMail.ContactTitle;  string Body = "โปรดติตต่อกลับ: "+\_objModelMail.Sender+"\n อีเมลล์: "+\_objModelMail.SenderEmail+"\n เรื่อง: “+\_objModelMail.ContactMessage;  mail.Body = Body;  mail.IsBodyHtml = true;  SmtpClient smtp = new SmtpClient();  smtp.Host = "smtp.gmail.com";  smtp.Port = 587;  smtp.UseDefaultCredentials = true;  smtp.Credentials = new System.Net.NetworkCredential("se552115005@vr.camt.info","552115005");// Enter seders User name and password  smtp.EnableSsl = true;  smtp.Send(mail);  return View("Contact", \_objModelMail);} else {  return View(); } | PASS |