



Team members: (Top Left to right) Cailín Smith, Amogelang P Moloko, Christopher J Oakes

(Bottom Left to Right) Achal Seechoonparsad, Jacquiline L Lawler

Team Information

15022014	Jacquiline L Lawler	u15022014@tuks.co.za	0797581409
15213626	Christopher J Oakes	u15213626@tuks.co.za	0834074027
14284783	Amogelang P Moloko	u14284783@tuks.co.za	0741020360
15035892	Cailín Smith [Project Leader]	u15035892@tuks.co.za	0769363737
15278043	Achal Seechoonparsad	u15278043@tuks.co.za	0718894149

TRWLA System: Technical Specification

The following document contains the technical specification of the TRWLA System, including all technical use case documentation, physical process modelling, physical ERD modelling and estimations, input design, output design, test specification, hardware & software requirements and specifications and validations across all aspects of the proposed system.

Contents

Figures.....	v
1. Introduction	17
2. Use Case	19
2.1 Introduction	19
2.2 Use Case Diagrams.....	20
2.3 Design Use Case Documentation.....	29
2.3.1 User Narratives	29
2.3.2 Volunteer Narratives.....	90
2.3.3 Student Narratives	156
2.3.4 Venues Narratives.....	227
2.3.5 Events Narratives	316
2.3.6 Guest Speaker Narratives	404
2.3.7 Content Narratives.....	440
2.3.8 Marketing Narratives	465
2.3.9 Reports Narratives	479
2.4 Conclusion.....	519
3. Process-Oriented Design.....	521
3.1 Introduction	521
3.2 Context Diagram	522
3.3 Decomposition Diagram	531
3.4 Physical Data Flow Diagrams	540
3.4.1 High Level Data Flow Diagrams.....	540
3.4.2 Mid-Level Data Flow Diagrams	549
3.4.3 Primitive Level Data Flow Diagrams	579

3.5 Physical Description of Each Process (Pseudo Code)	652
3.5.1 User Subsystem Pseudo Code	652
3.5.2 Volunteer Subsystem Pseudo Code.....	673
3.5.3 Student Subsystem Pseudo Code	695
3.5.4 Venues Subsystem Pseudo Code	717
3.5.5 Events Subsystem Pseudo Code	742
3.5.6 Guest Speaker Subsystem Pseudo Code.....	765
3.5.7 Content Subsystem Pseudo Code.....	772
3.5.8 Marketing Subsystem Pseudo Code	781
3.5.9 Report Subsystem Pseudo Code	785
3.6 Conclusion.....	793
4. Entity Relationship Model.....	795
4.1 Introduction	795
4.2 Entity Relationship Diagram	795
4.3 Conclusion	797
5. Size Estimation of Proposed Database	799
5.1 Introduction	799
5.2 Size Estimation	799
5.3 Five Year Projection	801
5.4 Conclusion.....	801
6. Input Design	803
6.1 Introduction	803
6.2 Prototype Screens.....	804
6.3 Conclusion.....	952
7. Output Design	953

7.1 Introduction	953
7.2 Reports	954
7.3 Outputs	970
7.4 Conclusion.....	979
8. Test Specification	981
8.1 Introduction	981
8.2 Test plan.....	981
8.2.1 Usability Testing.....	981
8.2.2 Unit Testing	981
8.2.3 Testing of Database	981
8.2.4 Final release Testing.....	981
8.3 Test Scenarios and Procedures.....	982
8.4 Test Data	997
8.5 Conclusion.....	1007
9. Hardware and Software Requirements	1009
9.1 Introduction	1009
9.2 Requirements.....	1009
9.2.1 Hardware	1009
9.2.2 Software.....	1011
9.3 Conclusion.....	1012
10. Network / Web Layout Specifications	1013
10.1 Introduction	1013
10.2 Diagram.....	1013
10.3 Conclusion.....	1013
11. Validation	1015

11.1 Introduction	1015
11.2 Validation	1015
11.2.1 Subsystem 1	1015
11.2.2 Subsystem 2	1020
11.2.3 Subsystem 3	1026
11.2.4 Subsystem 4	1031
11.2.5 Subsystem 5	1037
11.2.6 Subsystem 6	1043
11.2.7 Subsystem 7	1045
11.2.8 Subsystem 8	1047
11.2.9 Subsystem 9	1049
11.3 Conclusion.....	1052
12. Complexity	1053
13. Sign-off by Team	1061
14. Sign-off by Client	1063

Figures

Figure 1: 1. User Subsystem Use Case Diagram	20
Figure 2: 2. Volunteer Subsystem Use Case Diagram	21
Figure 3: 3. Student Subsystem Use Case Diagram	22
Figure 4: 4. Venue Subsystem Use Case Diagram	23
Figure 5: Event Subsystem Use Case Diagram	24
Figure 6: 6. Guest Speaker Subsystem Use Case Diagram	25
Figure 7: 7 Content Subsystem Use Case Diagram	26
Figure 8: 8. Marketing Subsystem Use Case Diagram	27
Figure 9: 9. Report Subsystem Use Case Diagram	28
Figure 10: Context Diagram Subsystem 1	522
Figure 11: Context Diagram Subsystem 2	523
Figure 12: Context Diagram Subsystem 3	524
Figure 13: Context Diagram Subsystem 4	525
Figure 14: Context Diagram Subsystem 5	526
Figure 15: Context Diagram Subsystem 6	527
Figure 16: Context Diagram Subsystem 7	528
Figure 17: Context Diagram Subsystem 8	529
Figure 18: Context Diagram Subsystem 9	530
Figure 19: Function 1 Decomposition Diagram	531
Figure 20: Function 2 Decomposition Diagram	532
Figure 21: Function 3 Decomposition Diagram	533
Figure 22: Function 4 Decomposition Diagram	534
Figure 23: Function 5 Decomposition Diagram	535
Figure 24: Function 6 Decomposition Diagram	536
Figure 25: Function 7 Decomposition Diagram	537
Figure 26: Function 8 Decomposition Diagram	538
Figure 27: Function 9 Decomposition Diagram	539
Figure 28: 1 High Level Data Flow Diagram	540
Figure 29: 2 High Level Data Flow Diagram	541
Figure 30: 3 High Level Data Flow Diagram	542

Figure 31: 4 High Level Data Flow Diagram	543
Figure 32: 5 High Level Data Flow Diagram	544
Figure 33: 6 High Level Data Flow Diagrams	545
Figure 34: 7 High Level Data Flow Diagram	546
Figure 35: 8 High Level Data Flow Diagram	547
Figure 36: 9 High Level Data Flow Diagrams	548
Figure 37: AUC 1 Mid-Level Data Flow Diagram	549
Figure 38: 1.1 Mid-Level Data Flow Diagram	549
Figure 39: 1.2 Mid-Level Data Flow Diagram	549
Figure 40: 1.3 Mid-Level Data Flow Diagram	550
Figure 41: 1.4 Mid-Level Data Flow Diagram	550
Figure 42: 1.5 Mid-Level Data Flow Diagram	551
Figure 43: 1.6 Mid-Level Data Flow Diagram	551
Figure 44: 1.7 Mid-Level Data Flow Diagram	551
Figure 45: 1.8 Mid-Level Data Flow Diagram	552
Figure 46: 1.9 Mid-Level Data Flow Diagram	552
Figure 47: 1.10 Mid-Level Data Flow Diagram	553
Figure 48: 2.1 Mid-Level Data Flow Diagram	554
Figure 49: 2.2 Mid-Level Data Flow Diagram	554
Figure 50: 2.3 Mid-Level Data Flow Diagram	555
Figure 51: 2.4 Mid-Level Data Flow Diagram	555
Figure 52: 2.5 Mid-Level Data Flow Diagram	555
Figure 53: 2.6 Mid-Level Data Flow Diagram	556
Figure 54: 2.7 Mid-Level Data Flow Diagram	556
Figure 55: 2.8 Mid-Level Data Flow Diagram	557
Figure 56: 2.9 Mid-Level Data Flow Diagram	557
Figure 57: 2.10 Mid-Level Data Flow Diagram	558
Figure 58: 2.11 Mid-Level Data Flow Diagram	558
Figure 59: 3.1 Mid-Level Data Flow Diagram	558
Figure 60: 3.2 Mid-Level Data Flow Diagram	559
Figure 61: 3.3 Mid-Level Data Flow Diagram	559

Figure 62: 3.4 Mid-Level Data Flow Diagram	560
Figure 63: 3.5 Mid-Level Data Flow Diagram	560
Figure 64: 3.6 Mid-Level Data Flow Diagram	560
Figure 65: 3.7 Mid-Level Data Flow Diagram	561
Figure 66: 3.8 Mid-Level Data Flow Diagram	561
Figure 67: 3.9 Mid-Level Data Flow Diagram	561
Figure 68: 4.1 Mid-Level Data Flow Diagram	562
Figure 69: 4.2 Mid-Level Data Flow Diagram	562
Figure 70: 4.3 Mid-Level Data Flow Diagram	563
Figure 71: 4.4 Mid-Level Data Flow Diagram	563
Figure 72: 4.5 Mid-Level Data Flow Diagram	564
Figure 73: 4.6 Mid-Level Data Flow Diagram	564
Figure 74: 4.7 Mid-Level Data Flow Diagram	565
Figure 75: 4.8 Mid-Level Data Flow Diagram	565
Figure 76: 4.9 Mid-Level Data Flow Diagram	566
Figure 77: 4.10 Mid-Level Data Flow Diagram	566
Figure 78: 4.11 Mid-Level Data Flow Diagram	566
Figure 79: 4.12 Mid-Level Data Flow Diagram	567
Figure 80: AUC 2 Mid-Level Data Flow Diagram	567
Figure 81: 5.1 Mid-Level Data Flow Diagram	568
Figure 82: 5.2 Mid-Level Data Flow Diagram	568
Figure 83: 5.3 Mid-Level Data Flow Diagram	569
Figure 84: 5.4 Mid-Level Data Flow Diagram	569
Figure 85: 5.5 Mid-Level Data Flow Diagram	570
Figure 86: 5.6 Mid-Level Data Flow Diagram	570
Figure 87: 5.7 Mid-Level Data Flow Diagram	570
Figure 88: 5.8 Mid-Level Data Flow Diagram	571
Figure 89: 5.9 Mid-Level Data Flow Diagram	571
Figure 90: 6.1 Mid-Level Data Flow Diagram	572
Figure 91: 6.2 Mid-Level Data Flow Diagram	572
Figure 92: 6.3 Mid-Level Data Flow Diagram	572

Figure 93: 6.4 Mid-Level Data Flow Diagram	572
Figure 94: 7.1 Mid-Level Data Flow Diagram	573
Figure 95: 7.2 Mid-Level Data Flow Diagram	573
Figure 96: 7.3 Mid-Level Data Flow Diagram	573
Figure 97: 7.4 Mid-Level Data Flow Diagram	574
Figure 98: 7.5 Mid-Level Data Flow Diagram	574
Figure 99: 8.1 Mid-Level Data Flow Diagram	574
Figure 100: 8.2 Mid-Level Data Flow Diagram	575
Figure 101: 8.3 Mid-Level Data Flow Diagram	575
Figure 102: 9.1 Mid-Level Data Flow Diagram	576
Figure 103: 9.2 Mid-Level Data Flow Diagram	576
Figure 104: 9.3 Mid-Level Data Flow Diagram	576
Figure 105: 9.4 Mid-Level Data Flow Diagram	577
Figure 106: 9.5 Mid-Level Data Flow Diagram	577
Figure 107: 9.6 Mid-Level Data Flow Diagram	577
Figure 108: 9.7 Mid-Level Data Flow Diagram	578
Figure 109: 9.8 Mid-Level Data Flow Diagram	578
Figure 110: AUC 1 Primitive Level Data Flow Diagram	579
Figure 111: 1.1 Primitive Level Data Flow Diagram	580
Figure 112: 1.2 Primitive Level Data Flow Diagram	581
Figure 113: 1.3 Primitive Level Data Flow Diagram	582
Figure 114: 1.4 Primitive Level Data Flow Diagram	583
Figure 115: 1.5 Primitive Level Data Flow Diagram	584
Figure 116: 1.6 Primitive Level Data Flow Diagram	585
Figure 117: 1.7 Primitive Level Data Flow Diagram	586
Figure 118: 1.8 Primitive Level Data Flow Diagram	587
Figure 119: 1.9 Primitive Level Data Flow Diagram	588
Figure 120: 1.10 Primitive Level Data Flow Diagram	589
Figure 121: 2.1 Primitive Level Data Flow Diagram	590
Figure 122: 2.2 Primitive Level Data Flow Diagram	591
Figure 123: 2.3 Primitive Level Data Flow Diagram	592

Figure 124: 2.4 Primitive Level Data Flow Diagram	593
Figure 125: 2.5 Primitive Level Data Flow Diagram	594
Figure 126: 2.6 Primitive Level Data Flow Diagram	595
Figure 127: 2.7 Primitive Level Data Flow Diagram	596
Figure 128 2.8 Primitive Level Data Flow Diagram	597
Figure 129: 2.9 Primitive Level Data Flow Diagram	598
Figure 130: 2.10 Primitive Level Data Flow Diagram	599
Figure 131: 2.11 Primitive Level Data Flow Diagram	600
Figure 132: 3.1 Primitive Level Data Flow Diagram	601
Figure 133: 3.2 Primitive Level Data Flow Diagram	602
Figure 134: 3.3 Primitive Level Data Flow Diagram	603
Figure 135: 3.4 Primitive Level Data Flow Diagram	604
Figure 136: 3.5 Primitive Level Data Flow Diagram	605
Figure 137: 3.6 Primitive Level Data Flow Diagram	606
Figure 138: 3.7 Primitive Level Data Flow Diagram	607
Figure 139: 3.8 Primitive Level Data Flow Diagram	608
Figure 140: 3.9 Primitive Level Data Flow Diagram	609
Figure 141: 4.1 Primitive Level Data Flow Diagram	610
Figure 142: 4.2 Primitive Level Data Flow Diagram	611
Figure 143: 4.3 Primitive Level Data Flow Diagram	612
Figure 144: 4.4 Primitive Level Data Flow Diagram	613
Figure 145: 4.5 Primitive Level Data Flow Diagram	614
Figure 146: 4.6 Primitive Level Data Flow Diagram	615
Figure 147: 4.7 Primitive Level Data Flow Diagram	616
Figure 148: 4.8 Primitive Level Data Flow Diagram	617
Figure 149: 4.9: Primitive Level Data Flow Diagram	618
Figure 150: 4.10 Primitive Level Data Flow Diagram	619
Figure 151: 4.11 Primitive Level Data Flow Diagram	620
Figure 152: 4.12 Primitive Level Data Flow Diagram	621
Figure 153: AUC 2 Primitive Level Data Flow Diagram	622
Figure 154: 5.1 Primitive Level Data Flow Diagram	623

Figure 155: 5.2 Primitive Level Data Flow Diagram	624
Figure 156- 5.3 Primitive Level Data Flow Diagram	625
Figure 157: 5.4 Primitive Level Data Flow Diagram	626
Figure 158: 5.5 Primitive Level Data Flow Diagram	627
Figure 159: 5.6 Primitive Level Data Flow Diagram	628
Figure 160: 5.7 Primitive Level Data Flow Diagram	629
Figure 161: 5.8 Primitive Level Data Flow Diagram	630
Figure 162: 5.9 Primitive Level Data Flow Diagram	631
Figure 163: 6.1 Primitive Level Data Flow Diagram	632
Figure 164: 6.2 Primitive Level Data Flow Diagram	633
Figure 165: 6.3 Primitive Level Data Flow Diagram	634
Figure 166: 6.4 Primitive Level Data Flow Diagram	635
Figure 167: 7.1 Primitive Level Data Flow Diagram	636
Figure 168: 7.2 Primitive Level Data Flow Diagram	637
Figure 169: 7.3 Primitive Level Data Flow Diagram	637
Figure 170: 7.4 Primitive Level Data Flow Diagram	638
Figure 171- 7.5 Primitive Level Data Flow Diagram	639
Figure 172: 8.1 Primitive Level Data Flow Diagram	640
Figure 173: 8.2 Primitive Level Data Flow Diagram	641
Figure 174: 8.3 Primitive Level Data Flow Diagram	642
Figure 175: 9.1 Primitive Level Data Flow Diagram	643
Figure 176: 9.2 Primitive Level Data Flow Diagram	644
Figure 177: 9.3 Primitive Level Data Flow Diagram	645
Figure 178: 9.4 Primitive Level Data Flow Diagram	646
Figure 179: 9.5 Primitive Level Data Flow Diagram	647
Figure 180: 9.6 Primitive Level Data Flow Diagram	648
Figure 181: 9.7 Primitive Level Data Flow Diagram	649
Figure 182: 9.8 Primitive Level Data Flow Diagram	650
Figure 183: Physical ERD	796
Figure 184: Navigation Bar for Students	804
Figure 185: Navigation Bar for Volunteers: Members Dropdown List	804

Figure 186: Navigation Bar for Volunteers: Events Dropdown List	805
Figure 187: Navigation Bar for Volunteers: User Dropdown List	805
Figure 188: General Profile page	806
Figure 189: 1.1.1 Check Forgotten Password	808
Figure 190: 1.1.2 Check Forgotten Password	809
Figure 191: 1.2.2 Change Password	810
Figure 192: 1.3.1 Login	811
Figure 193: 1.4.1 Logout	812
Figure 194: 1.5.2 Deactivate Account	813
Figure 195: 1.6.1 Create User type	814
Figure 196: 1.6.2 Create User Type	815
Figure 197: 1.7.1 Update User Type	816
Figure 198: 1.8.1 Delete User Type	817
Figure 199: 1.9.1 Search Alumni	818
Figure 200: Home page	819
Figure 201: About Page	820
Figure 202: Contact Page	820
Figure 203: Gallery Page	821
Figure 204: 2.1 Register	822
Figure 205: 2.1.1 Register Volunteer	823
Figure 206: 2.1.2 Register Volunteer	824
Figure 207: 2.1.3 Register Volunteer	825
Figure 208: 2.2.1 Search Volunteer	827
Figure 209: 2.3.1 Update Volunteer	829
Figure 210: 2.4.1 Delete Volunteer	831
Figure 211: 2.5.2 Create Volunteer Type	833
Figure 212: 2.6.1 Search Volunteer Type	834
Figure 213: 2.7.1 Update Volunteer Type	836
Figure 214: 2.8.1 Delete Volunteer Type	837
Figure 215: 2.9 Generate Unique Code	838
Figure 216: 2.10 and 2.11 Register and Deregister Admin	839

Figure 217: 3.1.1 Register Student	840
Figure 218: 3.1.2 Register Student	842
Figure 219: 3.1.3 Register Student	843
Figure 220: 3.2.1 Search Student	845
Figure 221: 3.3.1 Update Student	846
Figure 222: 3.4.1 Delete Student	848
Figure 223: 3.5 Generate Graduate List	850
Figure 224: 3.6.2 Add Student Type	851
Figure 225: 3.7 Search Student Type	852
Figure 226: 3.8.1 Update Student Type	853
Figure 227: 3.9.1 Delete Student Type	854
Figure 228: 4.3.1 Update venue	857
Figure 229: 4.4.1 Delete Venue	859
Figure 230: 4.7.1 Update Venue Type	863
Figure 231: 4.8.1 Delete Venue Type	864
Figure 232: 4.9.2 Add Residence	865
Figure 233: 4.11.1 Update Residence	867
Figure 234: 4.12.1 Delete a Residence	868
Figure 235: 5.0.1 Events Home VA	869
Figure 236: 5.0.2 Events Home VA	871
Figure 237: 5.1.1 Create Function VA	872
Figure 238: 5.1.2 Create Function VA	875
Figure 239: 5.1.3 Create Function VA	876
Figure 240: 5.1.4 Create Function VA	877
Figure 241: 5.2.1 Create Community Outreach VA	878
Figure 242: 5.2.2 Create Community Outreach VA	880
Figure 243: 5.2.3 Create Community Outreach VA	881
Figure 244: 5.2.4 Create Community Outreach VA	882
Figure 245: 5.3.1 Create Lecture VA	883
Figure 246: 5.3.2 Create Lecture VA	886
Figure 247: 5.3.3 Create Lecture VA	887

Figure 248: 5.3.4 Create Lecture VA	888
Figure 249: 5.4.1 Search Event VA	889
Figure 250: 5.4.2 Search Event VA ComEng	891
Figure 251: 5.4.2 Search Event VA Function	893
Figure 252: 5.4.2 Search Event VA Lecture	895
Figure 253: 5.5.1 RSVP to an Event VA ComEng	897
Figure 254: 5.5.1 RSVP to an Event VA Lecture	898
Figure 255: 5.5.2 RSVP to an Event VA Function	899
Figure 256: 5.6.1 Update Event Information VA ComEng	900
Figure 257: 5.6.1 Update Event Information VA Lecture	901
Figure 258: 5.6.2 Update Event Information VA Function	902
Figure 259: 5.7.1 Cancel Event VA ComEng	903
Figure 260: 5.7.1 Cancel Event VA Lecture	904
Figure 261: 5.7.2 Cancel Event Function VA	905
Figure 262: 5.7.2 Cancel Event VA From home page	906
Figure 263: 5.8.1 Log Event Attendance VA	907
Figure 264: 5.8.3 Log Event Attendance VA	908
Figure 265: 5.8.4 Log Event Attendance VA	909
Figure 266: 5.9.1 Send Notification VA	910
Figure 267: 5.9.3 Send Notification VA	912
Figure 268: 5.9.4 Send Notification VA	913
Figure 269: 5.9.5 Send Notification VA	914
Figure 270: 6.0.1 Function VA	915
Figure 271: 6.1.1 Register Guest Speaker VA	916
Figure 272: 6.1.2 Register Guest Speaker VA	918
Figure 273: 6.2.2 Search Guest Speaker VA	919
Figure 274: 6.3.1 Update Guest Speaker VA	921
Figure 275: 6.4.1 Delete Guest Speaker VA	922
Figure 276: 6.4.2 Delete Guest Speaker VA	923
Figure 277: 7.0.1 Content VA	924
Figure 278: 7.1.1 Upload Content VA	926

Figure 279: 7.1.2 Upload Content VA	928
Figure 280: 7.2.1 Search Content VA Locked	929
Figure 281: 7.3.2 Update Content VA	931
Figure 282: 7.4.1 Delete Content VA	932
Figure 283: 7.5.1 Review Lecture Content S	933
Figure 284: 7.5.2 Review Lecture Content S	935
Figure 285: 8.1.1Upload Photo Admin	936
Figure 286: 8.1.1 Upload Photo S	938
Figure 287: 8.1.2 Upload Photo S	939
Figure 288: 8.1.4 Upload Photo S	940
Figure 289: 8.2.1 Post Photo	941
Figure 290: 8.2.2 Post Photo	942
Figure 291: 8.3.1 Delete Photo	943
Figure 292: 8.3.2 Delete Photo	944
Figure 293: 9.1 Generate Class Attendance	945
Figure 294: 9.2 Generate Function Attendance	946
Figure 295: 9.3 Generate Com Engagement	947
Figure 296:9.4 Generate Demographics	948
Figure 297:9.5 Generate Event Popularity	949
Figure 298: 9.6 Generate Feedback	950
Figure 299: 9.7 Generate User Statistics	951
Figure 300: 9.8: Generate Student Progress	952
Figure 301: Suggested Confirmation Message Example	970
Figure 302: Suggested Successful Update Message Example	972
Figure 303: Suggested Deletion Warning Message Example	973
Figure 304: Suggested Update Warning Message Example	974
Figure 305: Suggested Error Message Example	976
Figure 306: Suggested No Search Results Message Example	977
Figure 307: Example of email	978
Figure 308: Suggested Graduate List Attachment in Email Example	978
Figure 309: Network Layout Structure	1013

1. Introduction

1.1 This document entails the technical specification for the TuksRes Women in Leadership Academy system. In the first deliverable, the business problem was addressed whereby the group identified that the organization was in dire need of automating most of their business processes. The second deliverable involved the functional specification whereby the group proposed a solution to the business problem. This document is a continuation of the proposed solution but involves the more technical aspects needed to develop the overall solution for the organisation. This document addresses all the technical requirements and specifications that need to be met in order to create the TRWLA system. The first section describes the to-be system which is illustrated using use case diagrams. Each use case from the use case diagram is then discussed and explained in the technical use case narratives. The process models are then illustrated and explained to describe the proposed system. Process models included are the context diagram, a complete functional decomposition diagram and a full set of high-level, mid-level and primitive-level data flow diagrams. All diagrams depict the technical process of how data will be obtained for each process in the process models. For each process there is also pseudo code which is a structured English method that explains the coding or logic behind every process in the process model so that the system is easier to code in the next deliverable. The data modelling is then illustrated using an entity-relationship diagram with the necessary referential integrity. After the ERD, the size estimation of the proposed database is depicted whereby a five year projection is illustrated for the organisation to determine their future growth. The full input prototype designs are illustrated and describe all of the possible actions a user may be presented with when interacting with the system. After the input designs, the output designs are depicted by the use of reports and other various outputs that the system will produce. The test specifications are then depicted and test certain use cases against specific scenarios. The hardware and software requirements are also identified in this deliverable for the organisation to be aware of what is needed for the final system. After the hardware and software has been identified, a network layout is illustrated. Lastly, a validation table is then presented whereby all the details of the functional specification are validated against the requirements. The document is then concluded by the sign-off by each of the members, as well as the client.

2. Use Case

2.1 Introduction

2.1.1 This section of the technical specification illustrates the technical requirements by means of a use case diagram. In the technical narratives, the actions are divided into three actors namely, Actor (PBA), Manual (PSA) and Automated (System). Each screen is also explained in technical detail to match the high-fidelity prototypes. The use case diagram explains who interacts directly with the system and who is an external actor. The use case diagram includes the user, volunteer, student, venues, event, guest speaker, content, marketing and reports subsystem. These diagrams are illustrated in figures 1 to 9. Each use case in the use case diagrams are then discussed in depth in the use case narratives to determine the steps involved in completing each use case.

2.2 Use Case Diagrams

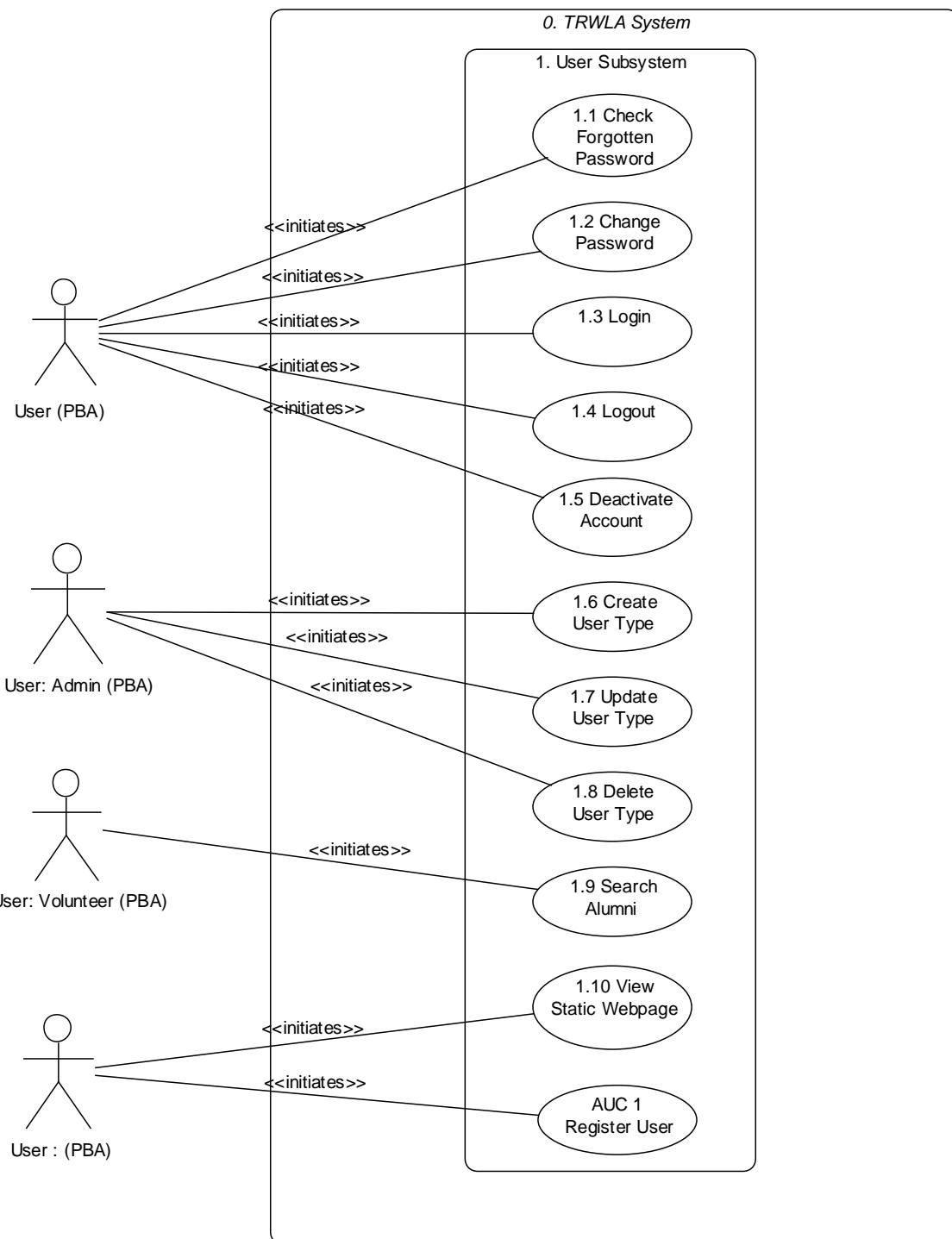


Figure 1: 1. User Subsystem Use Case Diagram

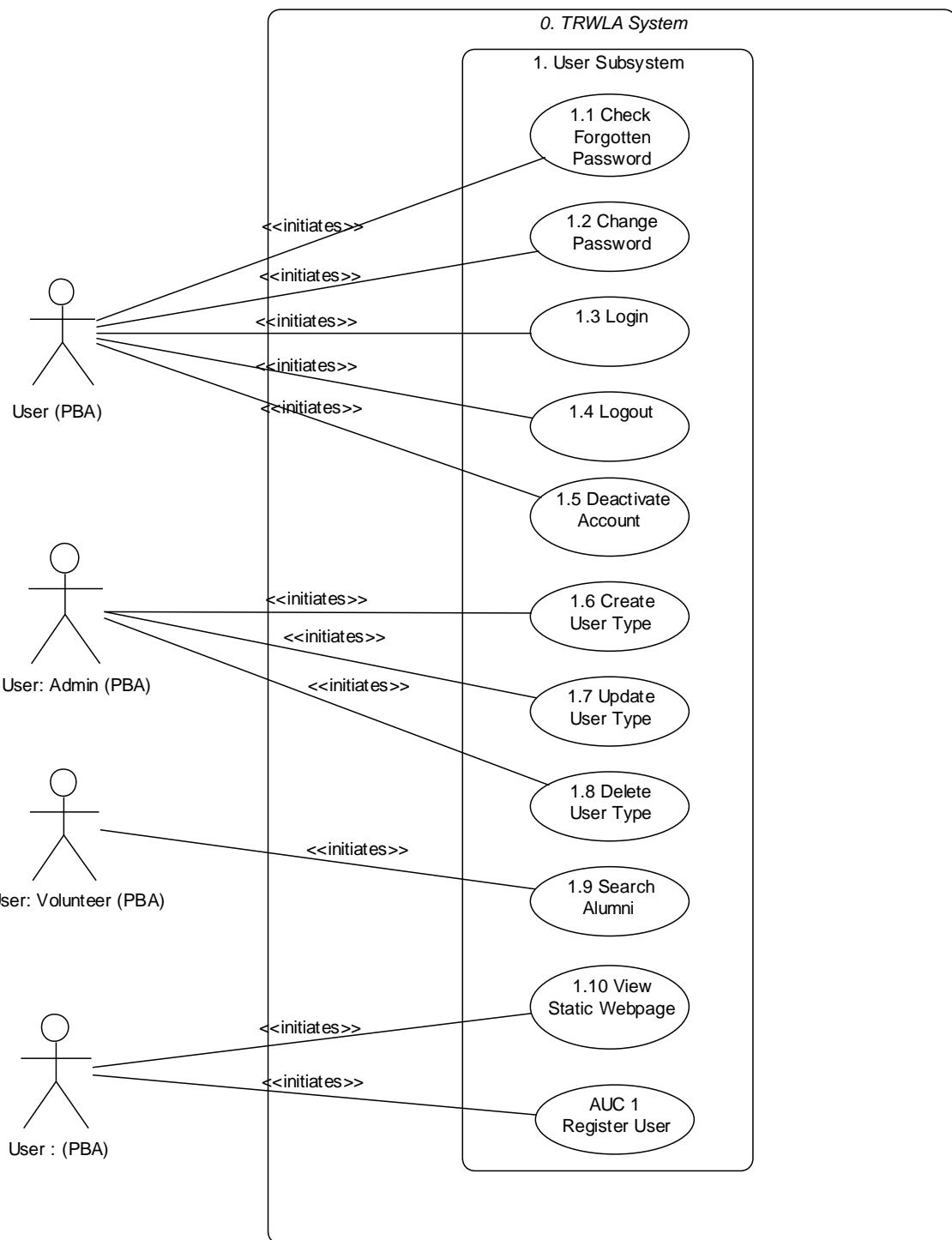


Figure 2: 2. Volunteer Subsystem Use Case Diagram

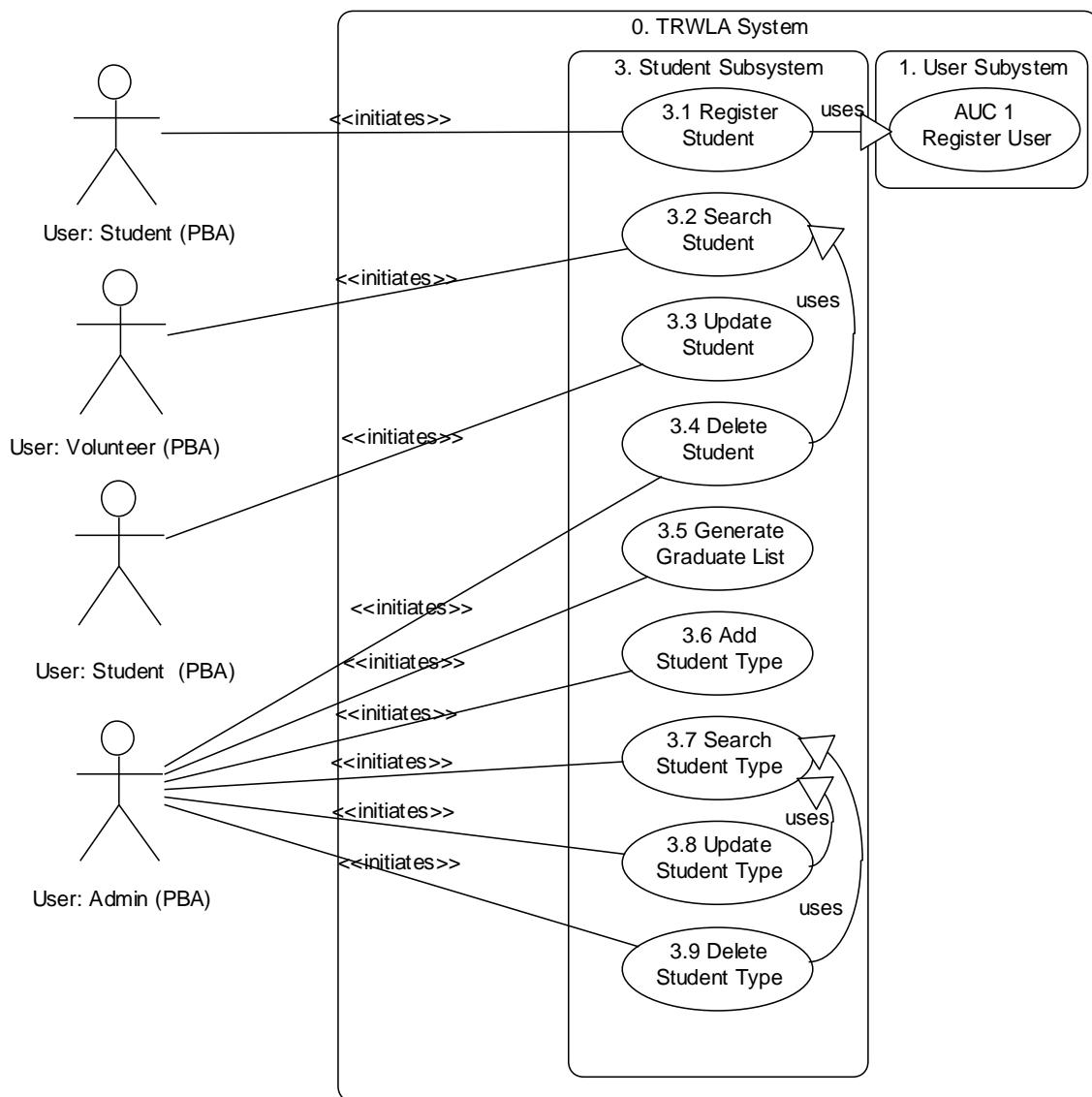


Figure 3: 3. Student Subsystem Use Case Diagram

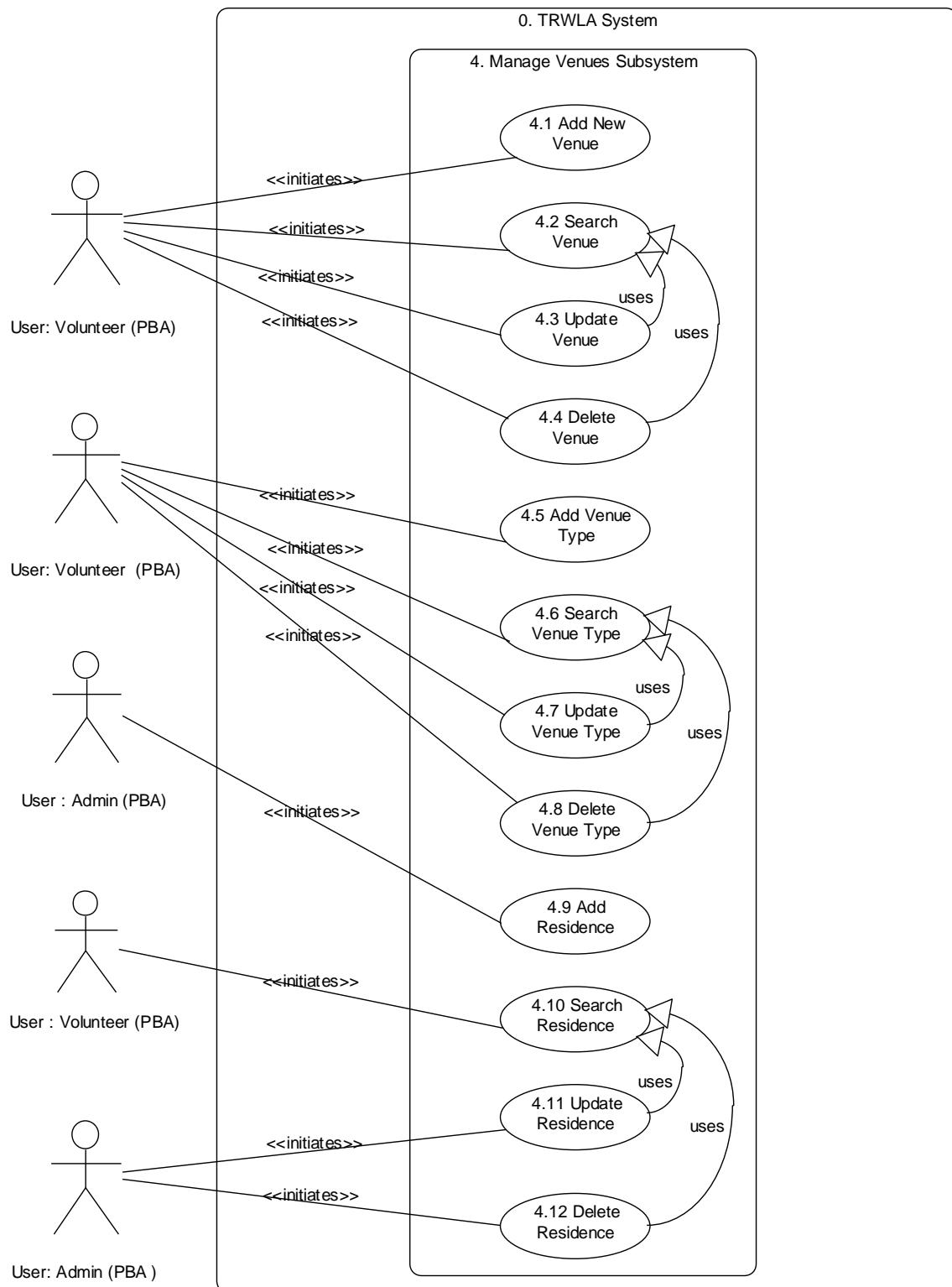


Figure 4: 4. Venue Subsystem Use Case Diagram

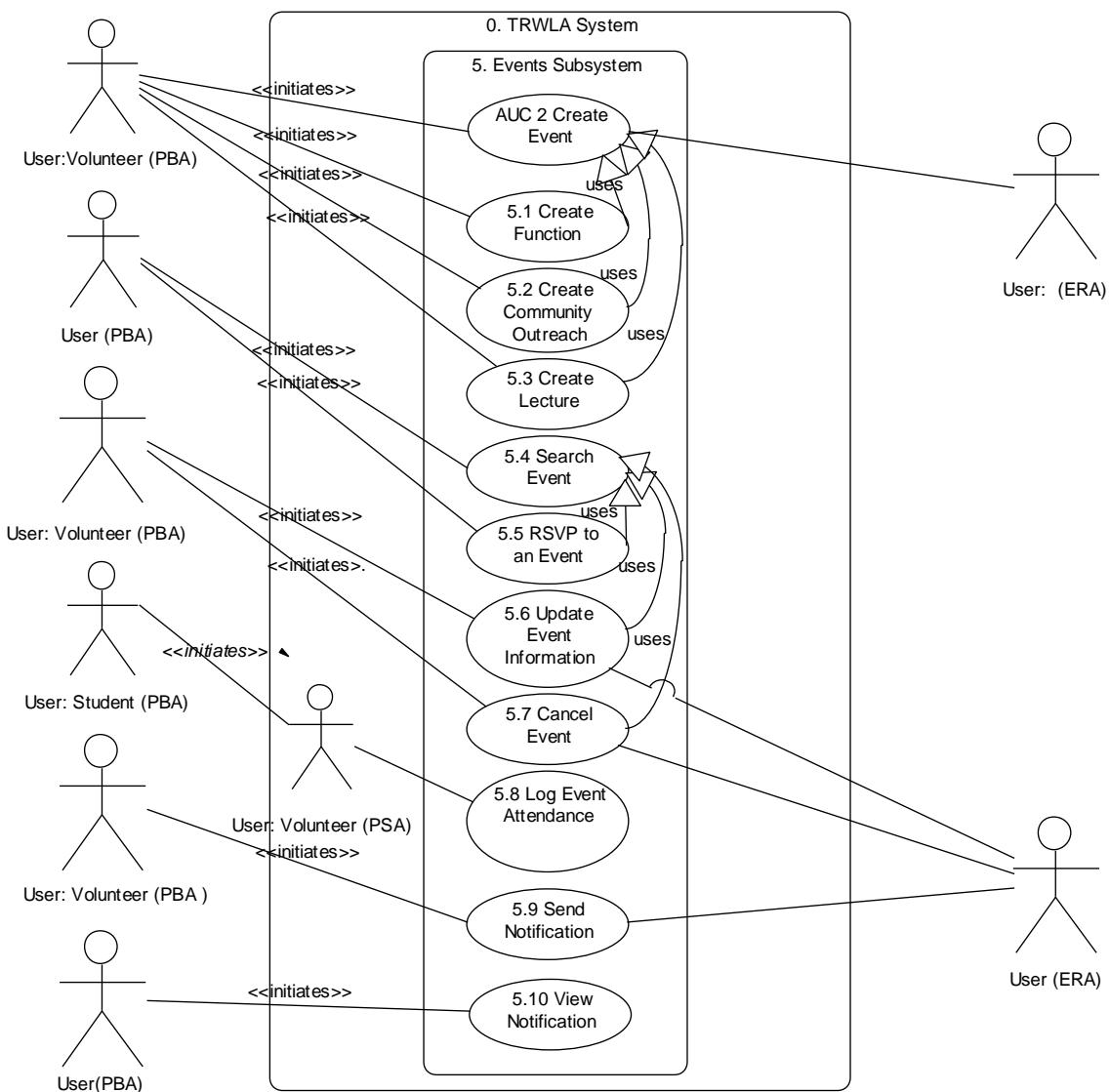


Figure 5: Event Subsystem Use Case Diagram

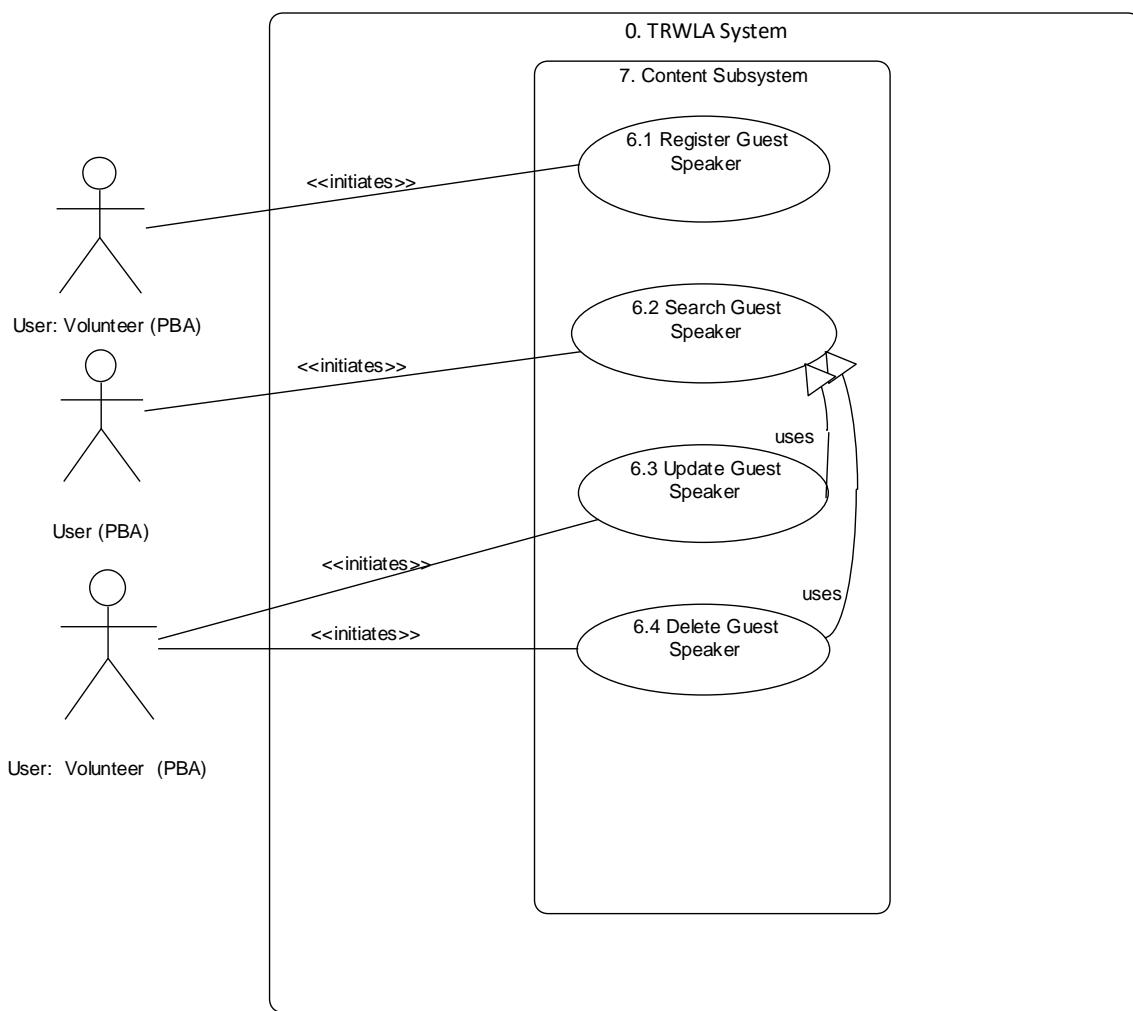


Figure 6: 6. Guest Speaker Subsystem Use Case Diagram

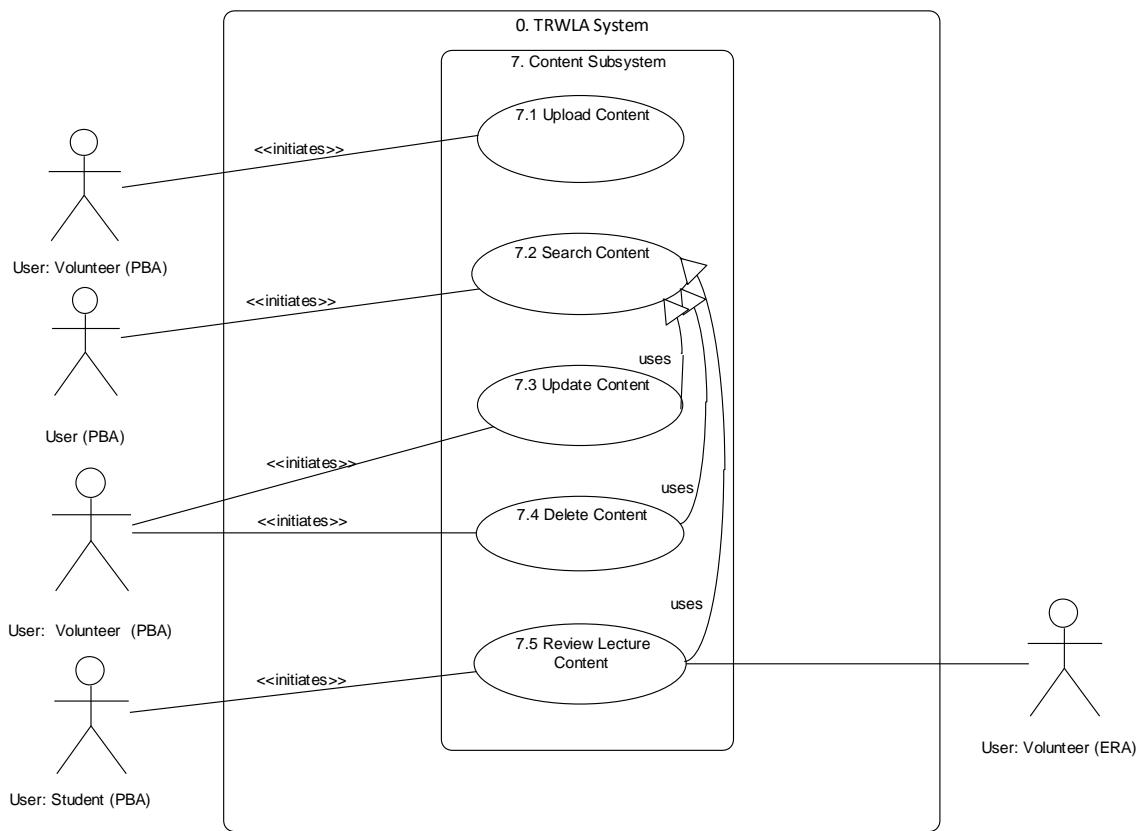


Figure 7: 7 Content Subsystem Use Case Diagram

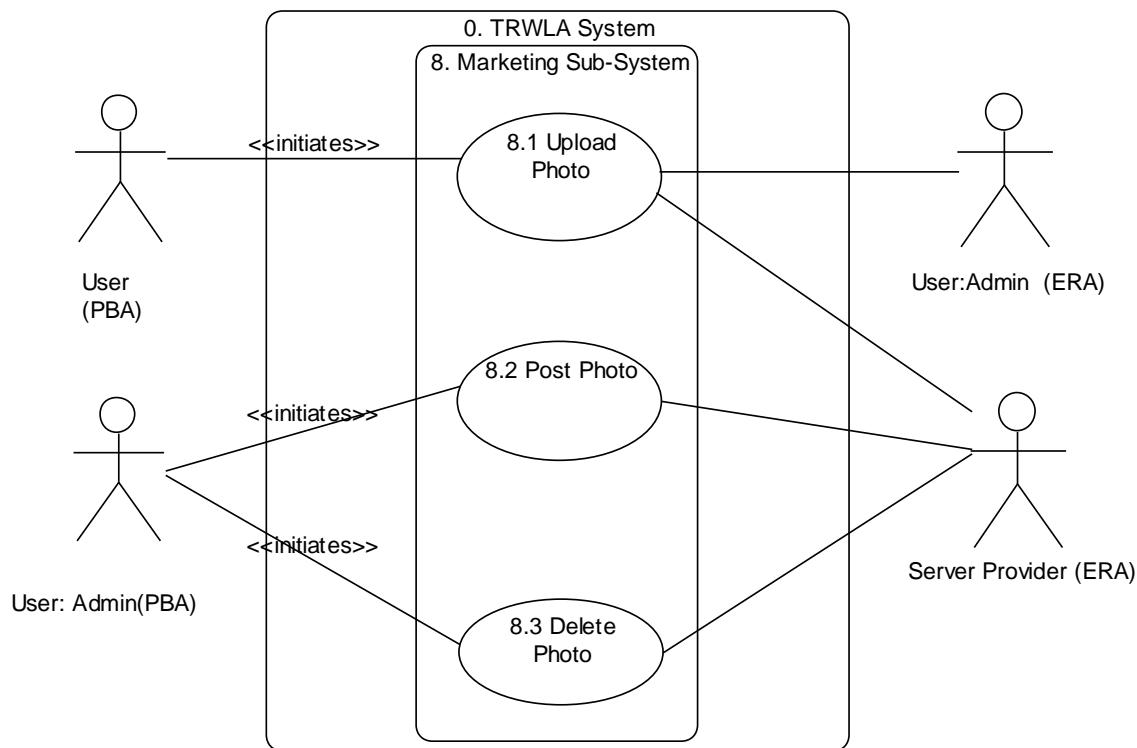


Figure 8: 8. Marketing Subsystem Use Case Diagram

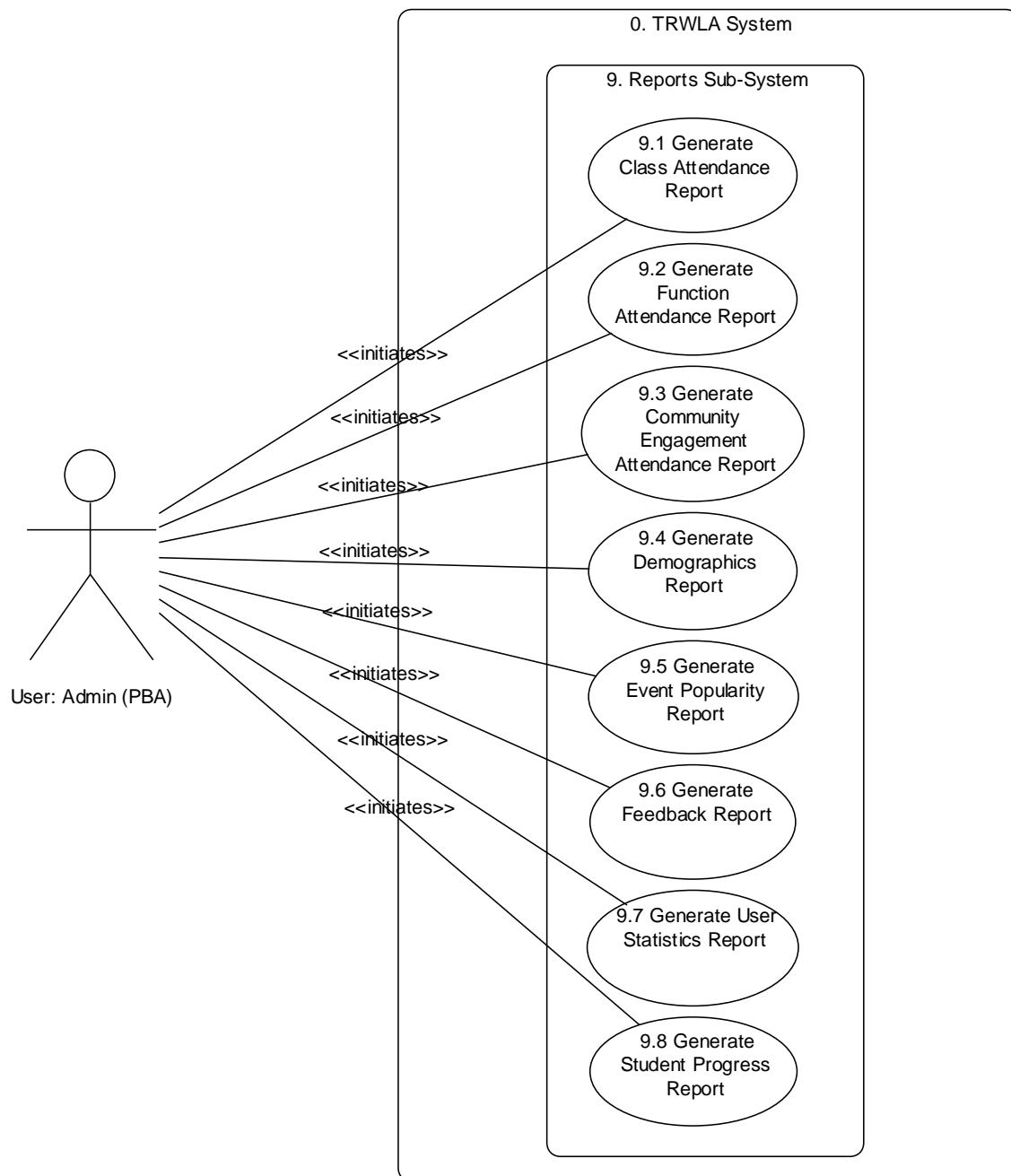


Figure 9: 9. Report Subsystem Use Case Diagram

2.3 Design Use Case Documentation

2.3.1 User Narratives

USE CASE NAME:	Change Forgotten Password		USE CASE TYPE
USE CASE ID:	1.1		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a user changing their password after forgetting their current password. The use case begins when the user has forgotten their password and wants to change it. The user will go to the TRWLA website, go to the login page of the website and click “Forgot Password?” The System will redirect the user to the forgotten password page, where the user will be required to enter their email address and answer a security question. Once the security question has been answered correctly the user will be taken to the change password page. The use case concludes when the change password webpage has been displayed.</p>		
PRE-CONDITION:	None		
TRIGGER:	The user has forgotten their password and wants to change it.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1: The user wants to change their forgotten password and goes to the TRWLA website.</p>		<p>Step 2: The system displays the webpage: ‘Static Website Home’ using HTML, CSS and Javascript, which contains the following:</p> <p>Header</p> <p>The header contains a navigation bar with the</p>

			<p>following items from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Home➤ About➤ Contact➤ Gallery➤ Login <p>Body</p> <p>The body contains an left aligned h1 header labelled “TuksRes Women in Leadership Academy” as well as 3 left aligned h2 headers labelled “Background”, “3 Core Pillars” and “Follow Us On:” Beneath the “Background” header is a paragraph giving a brief background of the organisation, beneath the “3 Core Pillars” header is a list of the core principles of the organisation and beneath the “Follow us On:” header are 5 round icons next to each other from left to right which represent Facebook, Instagram, Twitter, Blog and Make a Donation. To the right of the header and paragraphs is a large TRWLA Logo icon.</p> <p>Footer</p> <p>The footer contains a label “ © 2017 – TRWLA System Web Application” On Webpage: Home1</p>
--	--	--	--

	<p>Step 3: The user clicks the Login navigation bar item.</p>	<p>Step 4: The system displays the webpage: Login using HTML, CSS and Javascript, which contains the following:</p> <p>Header</p> <p>The header contains a navigation bar with the following items from left to right:</p> <ul style="list-style-type: none">➤ TRWLA➤ Home➤ About➤ Contact➤ Gallery➤ Login <p>Body</p> <p>The body contains a left aligned h1 header labelled “Login”. Below this is another header labelled “Use a local account to login”. Below the header is an h2 label “Email Address”, to the right of which is a textbox. Below that is another h2 header labelled “Password” followed by another textbox to the right of it. Beneath the “Password” header is a checkbox with the label “Remember Me”. Below this is a button entitled “Login”. Below this are two hyperlinks labelled “Register as a New User” and “Forgot Password”.</p> <p>Footer</p>
--	--	--

			<p>The footer contains a label “ © 2017 – TRWLA System Web Application” On Webpage: 1.3.1 Login</p>
	<p>Step 5: The user clicks the “Forgot Password” hyperlink.</p>		<p>Step 6: The system displays the webpage: 1.1.1 Check Forgotten Password using C#, HTML, CSS and Javascript, which contains the following:</p> <p>Header</p> <p>The header contains a navigation bar with the following items from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Home ➤ About ➤ Contact ➤ Gallery ➤ Login <p>Body</p> <p>The body contains a left aligned h1 header labelled “Check Forgotten Password”. Below the header is an h2 labelled “Enter your email” as well as an h4 header labelled “Email” with a corresponding textbox next to it. Beneath the textbox is a button labelled “Email Link”.</p> <p>Footer</p> <p>The footer contains a label “© 2017 – TRWLA System Web Application”</p>

			On Webpage: 1.1.1 Check forgotten Password
	Step 7: The user types their email address into the textbox.		Step 8: The system emails a hyperlink to the email address typed into the “Email Address” textbox. This hyperlink will redirect the user to answer a security question.
	Step 9: The user checks their email and clicks on the hyperlink sent by the system.		Step 10: Using JavaScript the system displays the following on the webpage 1.1.2 Check forgotten Password: Beneath the email address textbox a new h2 header is displayed labelled “Please answer the following security question:” Below this header is a label populated with text according to the <ul style="list-style-type: none"> • Question Attribute as retrieved from the SecurityAnswer table. The system uses c# and LINQ to verify whether the “Email Address” entered into the textbox exists as an <ul style="list-style-type: none"> • EmailAddress in the Person table, and if it does it displays the security question related to that person. The SecurityAnswer table and the Person table are connected via the PersonID key. Below this

			label is a textbox and below the textbox is a button entitled “Change Password”. On Webpage: 1.1.2 Check Forgotten Password
	Step 11: The user types in the answer to the security question in the textbox and clicks the “Change Password” button.		Step 12: The system verifies that the security answer entered by the user matches the <ul style="list-style-type: none"> Answer Attribute in the SecurityAnswer table using C# and LINQ.
			Step 12: The system displays the Change Password webpage (Invoke Use Case 1.2 Change Password)
ALTERNATE COURSES:	Alt Step 10: There is no email address in the Person table that matches the email address typed in by the user. The system displays a modal containing an error message. → Go to Step 7		
	Alt Step 11: The answer entered by the user does not match the answer in the SecurityAnswer table. The system displays a modal with an error message. → Go to Step 10		
CONCLUSION:	The user has answered the security question correctly and the system directs the user to the change password webpage, which invokes Use Case 1.2 Change Password.		
POST-CONDITION:	Use Case 1.2 Change Password has been invoked.		
BUSINESS RULES	<ul style="list-style-type: none"> Only users that can answer the security question correctly may change their forgotten password. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> The user has a current password which they have forgotten. 		
OPEN ISSUES:	None		

USE CASE NAME:	Change Password	USE CASE TYPE	
USE CASE ID:	1.2	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	Student (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a student changing their password. The use case begins when the student wants to change their password by clicking the 'Change Password' button in their profile. The system will direct the user to the 'Change Password' webpage. The user will have to provide their current login details as well as the new password they want and a confirmation of the new password. The system verifies that the information is complete and valid and updates the password. The use case concludes when a student's password has been successfully updated.</p>		
PRE-CONDITION:	<p>The student must be logged in OR must have answered the security question correctly to change their password.</p>		
TRIGGER:	<p>The user has forgotten their password and wants to change it.</p>		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1: The student wants to change their password and clicks on the 'My Profile' icon.</p>		<p>Step 2: The system, using C#, JavaScript, CSS and HTML, displays the Student Profile webpage with the following layout:</p> <p>Header</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Navigation bar with the following items from left to right:

			<ul style="list-style-type: none"> ➤ Volunteers ➤ Home ➤ Reading Content ➤ Gallery ➤ Message icon that displays notifications. ➤ Profile picture icon ➤ User drop down list with the following list items; <ul style="list-style-type: none"> ➤ “My Profile” ➤ “Logout” <p>Body</p> <ul style="list-style-type: none"> ➤ “My Profile” header label ➤ “Name” textbox ➤ “Surname” textbox ➤ “Phone Number” textbox ➤ “Email Address” textbox ➤ “Change Password” hyperlink ➤ “Date Of Birth” date picker ➤ “Student Type” drop down list populated from the StudentType table using LINQ queries. ➤ “Residence” drop down list populated from the Residence table using LINQ queries. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Save” button ➤ “Return” button <p>All fields are populated from the Student table using LINQ queries.</p>
--	--	--	---

	<p>Step 3: The student clicks the “Change Password” hyperlink.</p>	<p>Step 4: Using C#, HTML, CSS and JavaScript the system displays the 1.2.2 Change Password webpage. The webpage contains the following:</p> <p>Header</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Navigation bar with the following links:<ul style="list-style-type: none">➤ Volunteers➤ Home➤ Reading Content➤ Gallery➤ Message icon that displays notifications.➤ Profile picturebox➤ User drop down list with the following list items;<ul style="list-style-type: none">➤ “My Profile”➤ “Logout” <p>Body</p> <p>Aligned to the left of the body are the following:</p> <ul style="list-style-type: none">➤ H1 Header labelled “Change Password”➤ H2 header labelled “Email Address”➤ Email Address textbox➤ H2 header labelled “Current Password”➤ Password textbox➤ H2 Header labelled “New Password”➤ New Password Textbox➤ H2 header labelled “Confirm Password”
--	---	--

			<ul style="list-style-type: none"> ➤ Confirm Password textbox ➤ “Change” button <p>Footer The footer contains a label “© 2017 – TRWLA System Web Application” On Webpage: 1.2.2 Change Password</p>
	<p>Step 5: The student types in the required information including their email address, password, new password and confirmed password and clicks the “Change” button.</p>		<p>Step 6: The system verifies whether the information typed in by the student is complete and valid such that the email address and password matches the students</p> <ul style="list-style-type: none"> • Email Address • Password <p>in the Person table, and that the new password and the confirmed password are identical and are a minimum of 8 characters, a maximum of 35 characters and contains at least one numeric value.</p>
			<p>Step 7: Using C# and LINQ the system updates the</p> <ul style="list-style-type: none"> • Password <p>in the Person table.</p>
			<p>Step 8: Using JavaScript the system displays a modal with the message “Success Your password has been successfully updated!”</p>
ALTERNATE COURSES:	<p>Alt Step 1: The student has forgotten their password and has already completed use case 1.1, using C#, HTML, CSS and JavaScript the system</p>		

	<p>displays the “Change Password” webpage without the “Current Password” field.</p>
	<p>Alt Step 4: The user is not a student but a volunteer. The system displays the “Volunteer Profile” webpage which contains the following:</p> <p>Header</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Navigation bar with the following links: <ul style="list-style-type: none"> ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ Message icon that displays notifications. ➤ Profile picturebox ➤ User drop down list with the following list items; ➤ “My Profile” ➤ “Logout” <p>Body</p> <ul style="list-style-type: none"> ➤ “My Profile” header label ➤ “Name” textbox ➤ “Surname” textbox ➤ “Phone Number” textbox ➤ “Email Address” textbox ➤ “Change Password” hyperlink ➤ “Date Of Birth” date picker ➤ Volunteer Type drop down list populated from the VolunteerType table using entity framework and LINQ queries. ➤ “Residence” drop down list populated from the Residence table using Entity Framework and LINQ queries. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button ➤ “Save” button <p>All fields are populated from the Volunteer table using Entity Framework and LINQ queries</p>
	<p>Alt Step 7: The information typed in by the user is invalid and the system prompts the user with error messages using JavaScript.</p> <p>→ Go to Step 5</p>
CONCLUSION:	The student has successfully updated their password and the system has updated the password in the Person table in the database.
POST-CONDITION:	The student’s password has changed.
BUSINESS RULES	<ul style="list-style-type: none"> • Only users that can provide their valid login information or answer the security question can change their password.

IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> None
ASSUMPTIONS:	<ul style="list-style-type: none"> The user has a current password which they have forgotten.
OPEN ISSUES:	None

USE CASE NAME:	Login		USE CASE TYPE
USE CASE ID:	1.3		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Student		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> None 		
DESCRIPTION:	<p>This use case begins with a student wanting to login to the system. The student will navigate to the application login page. The student will then enter their registered email and password, and click the login button. The student's login details are verified by the system using the <u>Person</u> table to check if they are a valid user. The use case will conclude by displaying the main menu according to their user type.</p>		
PRE-CONDITION:	The student must be registered on the system.		
TRIGGER:	A student wants to login to the system.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: A student wants to login to the TRWLA management system and proceeds to the login webpage. Screen: 1.3.1 Login</p>		<p>Step 2: The system displays the Login webpage. This webpage contains a header, body and footer. The header contains the following navigation bar items displayed from left to right respectively:</p>

			<ul style="list-style-type: none">• TRWLA Management System• Home• About• Contact• Gallery• Log in <p>The body of the webpage contains the following displayed consecutively:</p> <ul style="list-style-type: none">➤ A label on the top left corner entitled, "Log in"➤ Required fields to login to the TRWLA Management Systemly, Email Address (textbox) and Password (masked textbox)➤ Checkbox labelled "Remember Me?"➤ Button labelled "Login"➤ Two hyperlinks entitled, "Register as a new user" and "Forgot Password". <p>The footer contains the following information:</p> <ul style="list-style-type: none">➤ A label on the left entitled "© 2017 - TRWLA
--	--	--	--

			<p>System Web Application” Using CSS, JavaScript, C# and HTML.</p>
	<p>Step 3: The student enters the required information, namely:</p> <ul style="list-style-type: none"> • Email address • Password <p>And clicks the “Log in” button. Screen: 1.3.1 Login</p>		<p>Step 4: The system verifies the required information entered by the user exists as the following attributes in the Person table:</p> <ul style="list-style-type: none"> • Email • Password <p>Using C# and SQL.</p>
			<p>Step 5: The system determines the access right of the user, according to the UserType table which is linked to the Person table via the UserID key. Using SQL and C#.</p>
			<p>Step 6: The system displays the Main Menu webpage. This webpage contains a header, body and footer. The header contains a navigation bar with the following menu items following elements from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Home ➤ Volunteers ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of

			<p>notifications, which contains the link to the user's notifications.</p> <p>➤ A profile picture which contains a link to the User dropdown list items.</p> <p>The body contains the following elements displayed vertically:</p> <ul style="list-style-type: none">➤ A container at the top of the body with the following message "Welcome (Name retrieved from <u>Person</u> table)"➤ A textbox with a button under entitled "Search upcoming events by name".➤ To the right of the textbox is a label entitled "Sort with the following" with two buttons below the label "Event" and "Date"➤ A container containing a list of upcoming events populated by
--	--	--	---

			<p>the Event table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the Event table:</p> <ul style="list-style-type: none">• Name• Date• Time• Venue• Summary <p>Each list item contains two buttons under the item entitled “RSVP” and “Details.”</p> <p>➤ To the right of the body is another container with a list of upcoming events that the user has rsvp’d to, which is determined by the RSVP_Event table, and is populated by the Event table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the Event table:</p> <ul style="list-style-type: none">• Name
--	--	--	--

			<ul style="list-style-type: none"> • Date • Time • Venue <p>Using CSS, JavaScript, C#, SQL and HTML.</p>
			<p>Step 7: The system updates the <u>AuditLog</u> table with the following information:</p> <ul style="list-style-type: none"> • PersonID (Determined according to the current logged in user) • LoginTime (Current timestamp) • LoginDate (Current date) • LoginDuration (Time between login and logout). <p>Using SQL, and LINQ.</p>
ALTERNATE COURSES:			<p>Alt-Step 4: The user entered an invalid email address or password, the system displays viewbag error messages to notify the user that their details are invalid.</p> <p>→ Go to Step 3</p>
			<p>Alt-Step 6: The Primary Business Actor (PBA) is a volunteer or admin. The system displays the Main Menu webpage. This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items following elements from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items.

	<p>The body contains the following elements displayed vertically:</p> <ul style="list-style-type: none"> ➤ A container at the top of the body with the following message “Welcome (Name retrieved from Person table)” ➤ A textbox with a button under entitled “Search upcoming events by name”. ➤ To the right of the textbox is a label entitled “Sort with the following” with two buttons below the label “Event” and “Date” ➤ A container containing a list of upcoming events populated by the Event table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the Event table: <ul style="list-style-type: none"> • Name • Date • Time • Venue • Summary <p>Each list item contains two buttons under the item entitled “RSVP” and “Details”. Below the container are four buttons labelled “Create” “Update” and “Delete”.</p> <ul style="list-style-type: none"> ➤ To the right of the body is another container with a list of upcoming events that the user has rsvp'd to, which is determined by the RSVP_Event table, and is populated by the Event table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the Event table: <ul style="list-style-type: none"> • Name • Date • Time • Venue <p>Using CSS, JavaScript, C#, SQL and HTML.</p>
CONCLUSION:	The user has entered a registered and valid email address and password, and has successfully logged into the system.
POST-CONDITION:	The system displays an appropriate main menu according to the type of user logging in.
BUSINESS RULES	<ul style="list-style-type: none"> • Only registered users may login to the TRWLA system.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • The user must use a valid email address as their login ‘username’
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Logout	USE CASE TYPE
USE CASE ID:	1.4	Business Requirements: <input type="checkbox"/>
PRIORITY:	Low	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User (Student or Volunteer)	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 	
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 	
DESCRIPTION:	<p>This use case describes the event of a user wanting to logout of the system. The use case begins by a user being logged into the system. The user clicks the dropdown list linked to their profile and clicks the “Logout” list item. The use case concludes by the user being logged off the system and the Login webpage is displayed.</p>	
PRE-CONDITION:	The user is logged into the system.	
TRIGGER:	The user wants to logout of the system.	
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: The user clicks their profile picture, picture box, which contains a link to their User dropdown list on the navigation bar in the header of the webpage.	System Response Manual Action Automated Action
	Step 3: The user clicks the Logout list item.	Step 4: The system displays Screen 1.4.1 Logout. The webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:

			<ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Home ➤ Volunteers ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains the following elements beneath one another:</p> <ul style="list-style-type: none"> ➤ A profile picture ➤ "Check Forgotten Password" button ➤ "Deactivate Account" button ➤ "Logout" button <p>Using CSS, C#, HTML, JavaScript.</p>
	<p>Step 5: The user clicks the "Logout" Button. Screen 1.4.1 Logout</p>		<p>Step 6: The system updates the following information in the <u>AuditLog</u> table:</p> <ul style="list-style-type: none"> • LoginDuration per the PersonID <p>Using SQL and LINQ.</p>
			<p>Step 7: The system displays Screen: 1.3.1 Login. This webpage</p>

			<p>contains a header, body and footer.</p> <p>The header contains the following navigation bar items displayed from left to right respectively:</p> <ul style="list-style-type: none">• TRWLA Management System• Home• About• Contact• Gallery• Log in <p>The body of the webpage contains the following displayed consecutively:</p> <ul style="list-style-type: none">➤ A label on the top left corner entitled, "Log in"➤ Required fields to login to the TRWLA Management Systemly, Email Address (textbox) and Password (masked textbox)➤ Checkbox labelled "Remember Me?"➤ Button labelled "Login"➤ Two hyperlinks entitled, "Register as a new user" and "Forgot Password".
--	--	--	---

			<p>The footer contains the following information:</p> <ul style="list-style-type: none"> ➤ A label on the left entitled “© 2017 - TRWLA System Web Application” <p>Using CSS, JavaScript, C# and HTML.</p>
ALTERNATE COURSES:	Alt Step 2a: The Primary Business Actor (PBA) is admin. The system displays the dropdown list items, namely: Reports, Register Admin, User Type and Logout. Using JavaScript, CSS and C#.		
CONCLUSION:	The user is logged out of the system.		
POST-CONDITION:	The Login webpage is displayed and the <u>Audit Log</u> table is updated		
BUSINESS RULES	<ul style="list-style-type: none"> • None 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Deactivate Account	USE CASE TYPE			
USE CASE ID:	1.5	Business Requirements: <input type="checkbox"/>			
PRIORITY:	Low	System Analysis: <input type="checkbox"/>			
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>			
PRIMARY BUSINESS ACTOR	User				
PRIMARY SYSTEM ACTOR	None				
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 				
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 				
DESCRIPTION:	<p>This use case describes the event of a user deactivating their account. The user navigates to the Logout Webpage and clicks the Deactivate Account Button. The system will confirm the deactivation, log the user out of the system and change the status of the user in the <u>Person</u> table.</p>				
PRE-CONDITION:	The user must be logged into the system.				
TRIGGER:	The user wants to deactivate their account.				
TYPICAL COURSE OF EVENTS:	Actor Action	System Response			
		Manual Action	Automated Action		
	<p>Step 1: The user wants to deactivate their account and clicks their profile picture in the header of the webpage.</p>		<p>Step 2: The system displays the dropdown list items, namely: User, Logout. Using JavaScript, CSS and C#.</p>		
	<p>Step 3: The user clicks the “Logout” list item.</p>		<p>Step 4: The system displays Screen 1.4.1 Logout. The webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p>		

			<ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Home ➤ Volunteers ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains the following elements beneath one another:</p> <ul style="list-style-type: none"> ➤ A profile picture ➤ "Check Forgotten Password" button ➤ "Deactivate Account" button ➤ "Logout" button <p>Using CSS, C#, HTML, JavaScript.</p>
	<p>Step 5: The user clicks the "Deactivate Account" button. Screen 1.4.1 Logout</p>		<p>Step 6: The system displays Screen 1.5.2 Deactivate Account. The webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p>

			<ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none"> ➤ A Header entitled "Deactivate Account" in the top centre. <p>Below, to the left the Header are blank fields listed vertically:</p> <ul style="list-style-type: none"> ➤ Enter Password (textbox), ➤ Re-enter Password (textbox), ➤ "Confirm" button <p>Using C#, CSS, HTML, JavaScript and SQL.</p>
	<p>Step 7: The user enters the required information and clicks the "Confirm" button.</p>		<p>Step 8: The system validates the information with the</p>

	Screen: 1.5.2 Deactivate Account	<p>following attributes from the Person table:</p> <ul style="list-style-type: none"> • Password <p>and displays a modal with the following elements:</p> <ul style="list-style-type: none"> ➤ Header entitled “Are you Sure” ➤ Paragraph text “You are about to deactivate your account. You will not be able to access the system with the email address and password you registered with” ➤ “Deactivate” button in the bottom right corner <p>Using CSS, C#, JavaScript, HTML.</p>
	<p>Step 9: The user clicks the “Confirm” button.</p>	<p>Step 10: The system displays the Login webpage. This webpage contains a header, body and footer.</p> <p>The header contains the following navigation bar items displayed from left to right respectively:</p> <ul style="list-style-type: none"> • TRWLA Management System • Home • About • Contact • Gallery • Log in

			<p>The body of the webpage contains the following displayed consecutively:</p> <ul style="list-style-type: none"> ➤ A label on the top left corner entitled, “Log in” ➤ Required fields to login to the TRWLA Management Systemly, Email Address (textbox) and Password (masked textbox) ➤ Checkbox labelled “Remember Me?” ➤ Button labelled “Login” ➤ Two hyperlinks entitled, “Register as a new user” and “Forgot Password”. <p>The footer contains the following information:</p> <ul style="list-style-type: none"> ➤ A label on the left entitled “© 2017 - TRWLA System Web Application” <p>Using CSS, JavaScript, C# and HTML.</p>
			<p>Step 9: The system updates the following attributes in the <u>Person</u> table:</p> <ul style="list-style-type: none"> • Status

		per the User's PersonID. Using SQL and C#.
ALTERNATE COURSES:	Alt Step 8: The information entered is invalid. The system displays viewbag error under the required fields. → Go to Step 6	
	Alt Step 9: The user clicks the “Cancel” button → Go to Step 6	
CONCLUSION:	The user has deactivated their account.	
POST-CONDITION:	The user's status has been updated in the Person table per the PersonID.	
BUSINESS RULES	<ul style="list-style-type: none"> • All users may deactivate their account's. 	
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 	
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 	
OPEN ISSUES:	None	

USE CASE NAME:	Create User Type	USE CASE TYPE
USE CASE ID:	1.6	Business Requirements: <input type="checkbox"/>
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	• None	
OTHER INTERESTED STAKEHOLDERS:	Students Volunteers	
DESCRIPTION:	This use case describes the event of an admin member creating a new user type on the system. The use case begins when the admin member clicks on the “User Type” list item under the “User” drop down list. Once the user has clicked on the list item, the system will display the User Type Webpage that displays the existing user types. The admin member will then click on the “Create a New User Type” button. The system will then display a list of empty fields that the admin member must complete in order to create a new user type. The use case concludes when the admin member has successfully created a new user type.	
PRE-CONDITION:	The admin member is logged into the system.	
TRIGGER:	The admin member wants to create a new user type.	
TYPICAL COURSE OF EVENTS:	Actor Action	System Response
	Manual Action	Automated Action
	Step 1: The admin member wants to create a new user type and clicks on the “User” drop down list.	Step 2: The system, using JavaScript, displays the following list items: ➤ Reports ➤ Register Admin ➤ User Type ➤ Log Out
	Step 3: The admin member clicks the “User Type” button.	Step 4: The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout:

			<p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Create User Type” h1 header➤ “Create a New User Type” button➤ “Description” h3 header➤ “Access Right” h3 header➤ “Update” button➤ “Details” button➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <ul style="list-style-type: none">➤ “Return” button
--	--	--	---

	Step 5: The admin member clicks on the “Create a New User Type” button.		Step 6: The system, using JavaScript, CSS and HTML, displays the Create User Type webpage with the following layout: Header Navigation bar from left to right: <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item Body <ul style="list-style-type: none">➤ “Create a New User Type” h1 header➤ “UserType” h2 header➤ “Description” textbox➤ “AccessRight” textbox➤ “Create” button➤ “Back to list” button
	Step 7: The admin member enters the relevant details		Step 8: The system, using C# and JavaScript,

	<p>in the textboxes and clicks the “Create” button.</p>		<p>validate that the information entered is correct, such as:</p> <ul style="list-style-type: none"> • Description (Max 20 characters) • AccessRight (Max 10 characters)
			<p>Step 9: The system, using C# and SQL, then adds the new user type to the UserType table using Entity Framework and LINQ queries.</p>
			<p>Step 10: The system, using JavaScript, displays a Modal with the following message: “The user type has been successfully created” with a “Confirm” button.</p>
	<p>Step 11: The admin member clicks the “Confirm” button.</p>		<p>Step 12 The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout:</p> <p>Header Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item

			<ul style="list-style-type: none"> ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Create User Type” h1 header ➤ “Create a New User Type” button ➤ “Description” h3 header ➤ “Access Right” h3 header ➤ “Update” button ➤ “Details” button ➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p>
ALTERNATE COURSES:			<p>Alt Step 5a: The admin member clicks the “Return” button. The system, using JavaScript, CSS and HTML, displays the main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button

	<ul style="list-style-type: none"> ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button ● “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button <p>On webpage: Admin Main Menu Webpage</p>
	<p>Alt Step 7: The admin member clicks the “Back to list” button. The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list ➤ “Events” drop down list ➤ “Reading Content” drop down list ➤ “Gallery” hyperlink button ➤ Message icon to display notifications ➤ Profile Picturebox ➤ “User” drop down list <p>Body</p> <ul style="list-style-type: none"> ➤ “Create User Type” h1 header ➤ “Create a New User Type” button ➤ “Description” h3 header ➤ “Access Right” h3 header ➤ “Update” button ➤ “Details” button ➤ “Delete” button

	<p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p> <p>→ Go to step 4</p> <p>On webpage: User Type</p>
	<p>At Step 8: The information entered is not correct and according to the system standards. The system displays a viewbag error message beneath the relevant textboxes that the correct information must be entered, i.e., maximum 20 characters are allowed.</p>
CONCLUSION:	<p>The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Create User Type” h1 header ➤ “Create a New User Type” button ➤ “Description” h3 header ➤ “Access Right” h3 header ➤ “Update” button ➤ “Details” button ➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p>
POST-CONDITION:	A new user type has been successfully created.
BUSINESS RULES	<ul style="list-style-type: none"> • Only admin members can create a new user type.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Update User Type	USE CASE TYPE	
USE CASE ID:	1.7	Business Requirements: <input type="checkbox"/>	
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	• None		
OTHER INTERESTED STAKEHOLDERS:	Students Volunteers		
DESCRIPTION:	<p>This use case describes the event of an admin member updating a user type on the system. The use case begins when the admin member clicks on the “User Type” list item under the “User” drop down list. Once the user has clicked on the list item, the system will display the User Type Webpage that displays the existing user types. The admin member will then click on the “Update” button next to the specific user type. The system will then display the user type details in editable textboxes. The user will then click on the “Update” button once they have updated the relevant details. The use case concludes when the user type has been successfully updated.</p>		
PRE-CONDITION:	The admin member is logged into the system.		
TRIGGER:	The admin member wants to update a user type.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	Step 1: The admin member wants to update a user type and clicks on the “User” drop down list.		Step 2: The system, using JavaScript, displays the following list items: <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Log Out
	Step 3: The admin member clicks the “User Type” button.		Step 4: The system, using JavaScript, CSS and HTML, displays the User

		<p>Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Create User Type” h1 header➤ “Create a New User Type” button➤ “Description” h3 header➤ “Access Right” h3 header➤ “Update” button➤ “Details” button➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <ul style="list-style-type: none">➤ “Return” button
--	--	--

	<p>Step 5: The admin member clicks on the “Update” button next to the specific user type.</p>	<p>Step 6: The system, using JavaScript, CSS and HTML, displays the Update User Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Update User Type” h1 header➤ “Description” textbox➤ “AccessRight” textbox➤ “Save” button➤ “Back to list” button <p>The details in the textboxes are populated from the <u>UserType</u> table using Entity Framework and LINQ queries.</p>
--	--	---

	Step 7: The admin member updates the relevant details and clicks the “Save” button.		Step 8: The system, using C# and JavaScript, validate that the information entered is correct, such as: <ul style="list-style-type: none"> • Description (Max 20 characters) • AccessRight (Max 10 characters)
			Step 9: The system, using C# and SQL, then updates the user type in the UserType table using Entity Framework and LINQ queries.
			Step 10: The system, using JavaScript, displays a Modal with the following message: “The user type has been successfully updated.” with a “Confirm” button.
	Step 11: The admin member clicks the “Confirm” button.		Step 12: The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout: Header Navigation bar from left to right: <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item

			<ul style="list-style-type: none"> ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Create User Type” h1 header ➤ “Create a New User Type” button ➤ “Description” h3 header ➤ “Access Right” h3 header ➤ “Update” button ➤ “Details” button ➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p>
ALTERNATE COURSES:	Alt Step 5a: The admin member clicks the “Return” button. The system, using JavaScript, CSS and HTML, displays the main menu with the following layout:		<p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox

	<ul style="list-style-type: none"> ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button ● “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button <p>On webpage: Admin Main Menu Webpage</p>
	<p>Alt Step 7: The admin member clicks the “Back to list” button. The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Create User Type” h1 header ➤ “Create a New User Type” button ➤ “Description” h3 header ➤ “Access Right” h3 header ➤ “Update” button ➤ “Details” button

	<ul style="list-style-type: none"> ➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p> <p>→ Go to step 4</p> <p>On webpage: User Type</p>
	<p>At Step 8: The information entered is not correct and according to the system standards. The system displays a viewbag error message beneath the relevant textboxes that the correct information must be entered, i.e., maximum 20 characters are allowed.</p>
CONCLUSION:	<p>The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Create User Type” h1 header ➤ “Create a New User Type” button ➤ “Description” h3 header ➤ “Access Right” h3 header ➤ “Update” button ➤ “Details” button ➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p>
POST-CONDITION:	A user type has been successfully updated.
BUSINESS RULES	<ul style="list-style-type: none"> • Only admin members can update a user type.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None

OPEN ISSUES:	None
---------------------	------

USE CASE NAME:	Delete User Type		USE CASE TYPE
USE CASE ID:	1.8		Business Requirements: <input type="checkbox"/>
PRIORITY:	Low		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	Students Volunteers		
DESCRIPTION:	<p>This use case describes the event of an admin member deleting a user type on the system. The use case begins when the admin member clicks on the “User Type” list item under the “User” drop down list. Once the user has clicked on the list item, the system will display the User Type Webpage that displays the existing user types. The admin member will then click on the “Delete” button next to the specific user type. The system will then display the user type details with buttons to delete or go back to the list. The admin member will then click on the “Delete” button. The use case concludes when the user type has been successfully deleted.</p>		
PRE-CONDITION:	The admin member is logged into the system.		
TRIGGER:	The admin member wants to delete a user type.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The admin member wants to delete a user type and clicks on the “User” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the following list items:</p> <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Log Out
	<p>Step 3: The admin member clicks the “User Type” button.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the User</p>

		<p>Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Create User Type” h1 header➤ “Create a New User Type” button➤ “Description” h3 header➤ “Access Right” h3 header➤ “Update” button➤ “Details” button➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <ul style="list-style-type: none">➤ “Return” button
--	--	--

	<p>Step 5: The admin member clicks on the “Delete” button next to the specific user type.</p>		<p>Step 6: The system, using JavaScript, CSS and HTML, displays the Delete User Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Delete” h1 header➤ “Are you sure you want to delete this?” h2 header➤ “UserType” h3 header➤ “Description” textbox➤ “AccessRight” textbox➤ “Delete” button➤ “Back to list” button
--	--	--	--

			The details in the textboxes are populated from the UserType table using Entity Framework and LINQ queries.
	Step 7: The admin member clicks the “Delete” button.		Step 8: The system, using JavaScript, CSS and HTML, displays a modal with the following message: “This user type will be deleted forever and is unrecoverable.” with a “Confirm” button.
	Step 9: The admin member clicks the “Confirm” button.		Step 10: The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout: Header Navigation bar from left to right: <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item Body <ul style="list-style-type: none"> ➤ “Create User Type” h1 header

			<ul style="list-style-type: none"> ➤ “Create a New User Type” button ➤ “Description” h3 header ➤ “Access Right” h3 header ➤ “Update” button ➤ “Details” button ➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p>
ALTERNATE COURSES:			<p>Alt Step 5a: The admin member clicks the “Return” button. The system, using JavaScript, CSS and HTML, displays the main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> • “Event Name” h3 header • “Summary” h4 header • “Date of the event” h4 header

	<ul style="list-style-type: none"> • “Time of event” h4 header • “Venue name”h4 header • “RSVP” button • “Details” button • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➢ “Create” button ➢ “Update” button ➢ “Delete” button ➢ “Return” button ➢ “Log Attendance” button <p>On webpage: Admin Main Menu Webpage</p>
	<p>Alt Step 7: The admin member clicks the “Back to list” button. The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➢ TRWLA Management System h1 header ➢ “Members” drop down list navigation bar item ➢ “Events” drop down list navigation bar item ➢ “Reading Content” drop down list navigation bar item ➢ “Gallery” navigation bar item ➢ Message icon which displays notifications. ➢ Profile picturebox ➢ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➢ “Create User Type” h1 header ➢ “Create a New User Type” button ➢ “Description” h3 header ➢ “Access Right” h3 header ➢ “Update” button ➢ “Details” button ➢ “Delete” button <p>The user types displayed are populated from the <u>User Type</u> table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p> <p>→ Go to step 4</p> <p>On webpage: User Type</p>
CONCLUSION:	<p>The system, using JavaScript, CSS and HTML, displays the User Type webpage with the following layout:</p> <p>Header</p>

	<p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Create User Type” h1 header ➤ “Create a New User Type” button ➤ “Description” h3 header ➤ “Access Right” h3 header ➤ “Update” button ➤ “Details” button ➤ “Delete” button <p>The user types displayed are populated from the UserType table using Entity Framework and LINQ queries.</p> <p>Footer</p> <p>“Return” button</p>
POST-CONDITION:	A user type has been successfully deleted.
BUSINESS RULES	<ul style="list-style-type: none"> • Only admin members can delete a user type.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Search Alumni	USE CASE TYPE	
USE CASE ID:	1.9	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design:	<input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a user searching for an alumni on the TRWLA System. The use case begins when the user wants to search for an alumni and goes to the alumni webpage. The user can either scroll through the list of alumni's or enter specific search parameters into the search textbox. Once the user has found the alumni they are looking for, they will be able to view the details of that particular alumni. The use case concludes when the user is satisfied with their search.</p>		
PRE-CONDITION:	The volunteer has logged into the system.		
TRIGGER:	The volunteer wants to search for an alumni.		
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: The user wants to search for an alumni and clicks on the Members navigation bar drop down list item. Step 3: The user clicks the "Alumni" dropdown list item.	System Response	
		Manual Action	Automated Action
	Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as: ➤ Volunteers ➤ Students ➤ Alumni		
		Step 4: The system, using JavaScript, CSS and HTML, displays the 1.9.1 Search Alumni webpage with the following layout:	

		Header <ul style="list-style-type: none">➤ TRWLA Management System➤ Navigation bar with the following drop down lists:<ul style="list-style-type: none">➤ Members➤ Events➤ Reading Content➤ Gallery➤ Message icon which displays notifications➤ User with profile picture icon Body <ul style="list-style-type: none">➤ “Alumni” header label➤ “You can search for students who have graduated from the TuksRes Women in Leadership Academy” label below the header➤ “Find by name” textbox below label➤ “Search” button to the right of search textbox➤ A container with a list of alumni, with the following headings above the container:<ul style="list-style-type: none">• First Name• Surname• Phone Number• Email AddressThe alumni in the alumni container has been retrieved from the	
--	--	---	--

			Alumni table using LINQ Queries.
	Step 5: The user scrolls through the list of alumni until they find the information of the relevant alumni.		
ALTERNATE COURSES:	Alt Step 5: The volunteer has some search criteria to enter into the textbox and enters it in the search by name textbox. The system uses LINQ queries to read a list of alumni that meets the search criteria and displays this in the alumni container. → Display webpage 1.9.2 Search Alumni		
CONCLUSION:	The volunteer has successfully found the alumni that they are searching for. The system has displayed the relevant alumni's information to the volunteer.		
POST-CONDITION:	The volunteer has searched for an alumni.		
BUSINESS RULES	<ul style="list-style-type: none"> • None 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	View Static Webpage		USE CASE TYPE
USE CASE ID:	1.10		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	Donors Potential students Potential volunteers		
DESCRIPTION:	<p>This use case describes the event of a user viewing the static webpage of the organization. The use case begins when the user enters the webpage domain name and the website home page is displayed to the user. The user will have access to a navigation bar with various list items to choose from such as, Home, About, Contact, Gallery, Register and Login. The use case concludes when the user exits the website or logs into the system.</p>		
PRE-CONDITION:	The user has navigated to TRWLA's website.		
TRIGGER:	The user wants to view the static webpage.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	Step 1: The user wants to view the static webpage and navigates to the website.		Step 2: The system, using JavaScript, CSS and HTML, displays the static Home webpage with the following layout: Header Navigation bar from left to right: <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ “Home” navigation bar item

			<ul style="list-style-type: none"> ➤ “About” navigation bar item ➤ “Contact” navigation bar item ➤ “Gallery” navigation bar item ➤ “Login” navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “TuksRes Women In Leadership Academy” h1 header ➤ “Background” h2 header ➤ Background information paragraph container ➤ “3 Core Pillars” h2 header ➤ “Service” h4 header with paragraph container ➤ “Skills” h4 header with paragraph container ➤ “Networking” h4 header with paragraph container ➤ Picturebox <p>Footer</p> <ul style="list-style-type: none"> ➤ Facebook clickable icon ➤ Twitter clickable icon
--	--	--	--

			<ul style="list-style-type: none"> ➤ Instagram clickable icon ➤ Blog clickable icon ➤ Donation clickable icon
	<p>Step 3: The user clicks the “About” hyperlink button.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the About Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ “Home” navigation bar item ➤ “About” navigation bar item ➤ “Contact” navigation bar item ➤ “Gallery” navigation bar item ➤ “Login” navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “About” h1 header ➤ “Service” h2 header with paragraph container ➤ “Skills” h2 header with paragraph container

			➤ “Networking” h2 header with paragraph container.
ALTERNATE COURSES:	<p>Alt Step 3a: The user clicks the “Contact” button. The system, using JavaScript, CSS and HTML, displays the Contact Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ “Home” navigation bar item ➤ “About” navigation bar item ➤ “Contact” navigation bar item ➤ “Gallery” navigation bar item ➤ “Login” navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Contact” h1 header ➤ “Duxbury Palace Elandspoort 357-Jr, Pretoria, 0002” h4 header ➤ “Email: office@trwla.co.za” h4 header <p>On webpage: Contact Webpage</p>		
	<p>Alt Step 3b: The user clicks the “Gallery” button. The system, using JavaScript, CSS and HTML, displays the Gallery webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ “Home” navigation bar item ➤ “About” navigation bar item ➤ “Contact” navigation bar item ➤ “Gallery” navigation bar item ➤ “Login” navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Gallery” h1 header ➤ Thumbnails of photos <p>On webpage: Gallery Webpage</p>		
	Alt Step 3c: The user clicks on the Facebook icon. The user will be redirected to TRWLA’s Facebook page.		
	Alt Step 3d: The user clicks on the Twitter icon. The user will be redirected to TRWLA’s Twitter page.		
	Alt Step 3e: The user clicks on the Instagram icon. The user will be redirected to TRWLA’s Instagram page.		

	Alt Step 3f: The user clicks on the Blog icon. The user will be redirected to TRWLA's Blog page.
	Alt Step 3g: The user clicks on the Donation icon. The user will be redirected to TRWLA's Donation page.
CONCLUSION:	<p>The system, using JavaScript, CSS and HTML, displays the About Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ "Home" navigation bar item ➤ "About" navigation bar item ➤ "Contact" navigation bar item ➤ "Gallery" navigation bar item ➤ "Login" navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ "About" h1 header ➤ "Service" h2 header with paragraph container ➤ "Skills" h2 header with paragraph container ➤ "Networking" h2 header with paragraph container.
POST-CONDITION:	The user has successfully viewed the static webpage.
BUSINESS RULES	<ul style="list-style-type: none"> • Anyone in the world can view the static webpage.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

2.3.2 Volunteer Narratives

USE CASE NAME:	Register Volunteer	USE CASE TYPE	
USE CASE ID:	2.1	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • User: Volunteer 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a volunteer registering themselves on the system. The use case begins when the volunteer is ‘hired’ by TRWLA and wants to register on the system. The volunteer will go to the static webpage and click on the “Login” link in the header section of the webpage. The system will then display the Login webpage which uses UC 1.3 Login. The volunteer will then click on the “New User?” hyperlink. The system will then display the Register User webpage where the volunteer will enter the necessary required information. The system will have to verify that the volunteer is registering valid details. The use case concludes when the volunteer has successfully registered on the system.</p>		
PRE-CONDITION:	The volunteer has been ‘hired’ by TRWLA and wants to register as a volunteer on the system		
TRIGGER:	The student wants to register on the TRWLA system.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	Step 1: The volunteer wants to register on the TRWLA system and opens the website.		Step 2: The system prompts the volunteer to register their user details (Invoke Abstract Use Case 1 Register User)
	Step 3: The volunteer wants to register as a volunteer and clicks on the “Volunteer” radio button.		Step 4: Using JavaScript, the system displays a modal with the following layout:

	See Webpage 2.1.1 Register Volunteer		<ul style="list-style-type: none"> ➤ H1 header labelled "Enter Unique Code" ➤ Unique Code textbox ➤ "Continue" button <p>On Webpage 2.1.2 Register Volunteer</p>
	Step 5: The volunteer enters their unique code and clicks the "Continue" button.		Step 6: Using C# and LINQ the system verifies whether the unique code is valid such that it is a 5 digit numeric value that exists in the UniqueCode table and is marked as 'unused'
			Step 7: Using C# and LINQ the system marks the UniqueCode as 'used'
			<p>Step 8: The system, using JavaScript, CSS and HTML, displays the 2.1.3 Register Volunteer webpage with the following layout:</p> <p>Header</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Navigation bar with the following links: <ul style="list-style-type: none"> ➤ Home ➤ About ➤ Contact ➤ Gallery ➤ Login <p>Body</p> <ul style="list-style-type: none"> • "Register as a Volunteer" Header <h1> Label • "Register" g2 label • "Name" textbox • "Surname" textbox

			<ul style="list-style-type: none"> “Phone Number” textbox “Email Address” textbox populated with email address entered during AUC 1. “Date Of Birth” date picker “Security Question” Dropdown List (Hardcoded population) “Security Answer” textbox “Volunteer type” dropdown list (Populated by the <u>VolunteerType</u> table using Entity Framework and LINQ queries.) “Residence to Facilitate” drop down list (Populated from the <u>Residence</u> table using Entity Framework and LINQ queries.) Not Required* <p>To the bottom right of the body are two buttons namely</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Return” button <p>Footer</p> <p>The footer contains a label “© 2017 – TRWLA System Web Application” On Webpage 2.1.3 Register Volunteer</p>
--	--	--	---

	<p>Step 9: The volunteer enters all of the relevant information in the textboxes and clicks the “Create” button.</p>	<p>Step 10: The system, using C#, JavaScript and CSS, verifies that all required fields have been filled in and that the fields contain valid details:</p> <ul style="list-style-type: none">• Name (String value of maximum 35 characters)• Surname (String value of maximum 35 characters)• Phone Number (Max 15 characters (No spaces))• Email Address (Minimum 7 characters, Maximum 255 characters. Must be a full validated email address.)• Date of Birth (Maximum 10 characters in the format CCYY-MM-DD)• Security Question (Selected from the available dropdown list options)• Security Answer (Maximum 200 characters)• Volunteer Type (Selected from predetermined options as retrieved from the VolunteerType table. The Volunteer and VolunteerType tables
--	---	--

			<ul style="list-style-type: none"> • Residence (Selected from predetermined options as retrieved from the Residence table. The Student and Residence tables are connected via the ResID)
			<p>Step 11: The system, using C# and LINQ, adds the new volunteer details to the Person, Volunteer and SecurityAnswer tables using LINQ queries.</p>
			<p>Step 12: The system, using C#, JavaScript and CSS, displays a modal with a message that says “Congratulations! Your registration was successful!” with a “Confirm” button. On Webpage: 2.1.4 Register Volunteer</p>
ALTERNATE COURSES:	<p>Alt Step 5: The volunteer has not yet received a unique code from management and cannot register on the TRWLA system yet.</p>		
	<p>Alt Step 7: The unique code entered by the volunteer is invalid. Using JavaScript, the system displays an error message on the modal “Please enter a valid unique code” → Go to Step 5</p>		
	<p>Alt Step 11: The personal information entered by the volunteer is invalid, using JavaScript the system displays error messages beneath the invalid fields and prompts the user to enter valid information. → Go to Step 9</p>		
CONCLUSION:	<p>The volunteer has successfully registered their details on the system. The system has verified that the information registered by the user is complete and valid and has inserted it into the Volunteer table.</p>		
POST-CONDITION:	<p>The volunteer has been successfully registered on the system.</p>		

BUSINESS RULES	<ul style="list-style-type: none">• None
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none">• None
ASSUMPTIONS:	<ul style="list-style-type: none">• None
OPEN ISSUES:	None

USE CASE NAME:	Search Volunteer	USE CASE TYPE	
USE CASE ID:	2.2	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a user searching for a volunteer on the TRWLA System. The use case begins when the user wants to search for a volunteer and goes to the volunteer webpage. The user can either scroll through the list of volunteers or enter specific search parameters into the search textbox. Once the user has found the volunteer they are looking for, they will be able to view the details of that particular volunteer. The use case concludes when the user is satisfied with their search.</p>		
PRE-CONDITION:	The user has logged into the system.		
TRIGGER:	The user wants to search for a student.		
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: The user wants to search for a volunteer and clicks on the Members navigation bar drop down list item.	System Response Manual Action	Automated Action Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as: ➤ Volunteers ➤ Students ➤ Alumni
	Step 3: The user clicks the “Volunteers” dropdown list item.		Step 4: The system, using JavaScript, CSS and HTML, displays the TRWLA Volunteers webpage with the following layout: Header ➤ TRWLA Management System

			<ul style="list-style-type: none"> ➤ Navigation bar with the following drop down lists: ➤ Members ➤ Events ➤ Reading Content ➤ Message icon which displays notifications ➤ Profile picturebox ➤ User drop down list with “My Profile” and “Logout” list items. <p>Body</p> <ul style="list-style-type: none"> ➤ “Volunteers” header label ➤ “Find by name” textbox ➤ “Search” button to the right of textbox ➤ A container with a list of volunteers, with the following headings above the container: <ul style="list-style-type: none"> ○ Name ○ Surname ○ Phone Number ○ Email Address ○ Date of Birth ○ Volunteer Type ○ Residence ➤ Three buttons beneath the container to the right of the body labelled “Update”, “Delete” and “Return” <p>Footer</p>
--	--	--	--

			<p>➤ Nothing</p> <p>The volunteers in the volunteer container have been retrieved from the Person and Volunteer tables LINQ Queries.</p> <p>On Webpage: 2.2.1 Search Volunteer</p>
	<p>Step 5: The user types search criteria in the search by name textbox and clicks on the search button</p> <p>See Webpage 2.2.2 Search Volunteer</p>		<p>Step 6: Using C# and LINQ the system retrieves all the volunteers from the Person and Volunteer tables that matches the search criteria entered by the user.</p>
			<p>Step 7: Using JavaScript the system replaces the volunteers in the volunteer container with only volunteer that match the search criteria.</p> <p>On Webpage: 2.2.3 Search Volunteer</p>
ALTERNATE COURSES:	<p>Alt Step 1: The user is a student and the student clicks the “Volunteer” navigation bar item directly.</p> <p>→ Go to Step 4</p>		
	<p>Alt Step 5: The user scrolls through the volunteers in the container and finds the relevant volunteer they are looking for.</p>		
CONCLUSION:	<p>The user has successfully found the volunteer that they are searching for. The system has displayed the relevant volunteers' information to the user.</p>		
POST-CONDITION:	<p>The user has searched for a volunteer.</p>		
BUSINESS RULES	<ul style="list-style-type: none"> • None 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	<p>None</p>		

USE CASE NAME:	Update Volunteer	USE CASE TYPE
USE CASE ID:	2.3	Business Requirements: <input type="checkbox"/>
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 	
OTHER INTERESTED STAKEHOLDERS:	None	
DESCRIPTION:	<p>This use case describes the event of a volunteer updating their details on the TRWLA system. The use case begins when the volunteer has logged onto the system and wants to update their profile. The system then displays the volunteer Main Menu webpage. The volunteer will then click on the drop down list arrow next to the profile picture icon and click on "My Profile". The system will then display the volunteer's details and the volunteer can then update her profile accordingly. They will then update and save their relevant details, and the use case will conclude when they have successfully updated their details.</p>	
PRE-CONDITION:	The volunteer has logged into the system.	
TRIGGER:	The volunteer wants to update their profile details.	
TYPICAL COURSE OF EVENTS:	Actor Action	System Response
	<p>Step 1: The volunteer wants to update their profile details and clicks on the User drop down list on the navigation bar.</p>	<p>Manual Action</p> <p>Step 2: Using JavaScript, the system displays the following list items:</p> <ul style="list-style-type: none"> ➤ My Profile ➤ Register Admin (if the volunteer is admin) ➤ User Types (if the volunteer is admin) ➤ Logout
	<p>Step 3: The volunteer clicks on the "My Profile" list item.</p>	<p>Automated Action</p> <p>Step 4: The system, using JavaScript, CSS and HTML, displays the 2.3.1 Update Volunteer webpage with the following layout:</p>

			<p>Header</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Navigation bar with the following dropdown lists:<ul style="list-style-type: none">○ Members○ Events○ Reading Content○ Profile picture icon○ User dropdown list <p>Body</p> <ul style="list-style-type: none">➤ “My Profile” header label➤ “Name” textbox➤ “Surname” textbox➤ “Phone Number” textbox➤ “Email Address” textbox➤ “Date Of Birth” date picker➤ “Volunteer type” dropdown list (Populated by the VolunteerType table using Entity Framework and LINQ queries.)➤ “Residence to Facilitate” drop down list (Populated from the Residence table using Entity Framework and LINQ queries.) Not Required* <p>Each of the fields is in an editable format and is</p>
--	--	--	---

			<p>populated with the volunteers information as retrieved from the Volunteer and Person tables in the database.</p> <p>Footer</p> <ul style="list-style-type: none"> ➤ “Save” button ➤ “Return” button ➤ “Delete” button <p>On Webpage 2.3.1 Update Volunteer</p>
	<p>Step 5: The volunteer updates their details in the fields and clicks on the “Save” button.</p>		<p>Step 6: Using C# and SQL the system verifies that all required fields have been filled in and that the fields contain valid details:</p> <ul style="list-style-type: none"> • Name (Maximum 35 characters) • Surname (Maximum 35 characters) • Phone Number (Max 15 characters (No spaces)) • Email Address (Minimum 7 characters, Maximum 255 characters. Must be a full validated email address.) • Date of Birth (Maximum 10 characters in the format CCYY-MM-DD) • Volunteer Type (Selected from predetermined options as retrieved from the VolunteerType table. The Volunteer and

			<p>VolunteerType tables are connected via the VolunteerTypeID)</p> <ul style="list-style-type: none"> • Residence (Selected from predetermined options as retrieved from the Residence table. The Student and Residence tables are connected via the ResID)
			<p>Step 7: The system, using C# and JavaScript, displays a modal with the following message “Are you sure you would like to update your account?”, with the following two buttons:</p> <ul style="list-style-type: none"> ➤ “Confirm” ➤ “Cancel” <p>On Webpage: 2.3.2 Update Volunteer</p>
	<p>Step 8: The volunteer clicks the “Confirm” button.</p>		<p>Step 9: The system, using C# and LINQ, updates the volunteer’s details in the Volunteer and Person tables.</p>
			<p>Step 10: The system, using C# and JavaScript, displays a modal with the following message “Success! You have successfully updated your profile” and a “Confirm” button.</p>
ALTERNATE COURSES:	<p>Alt Step 6: The information entered is incorrect. Using JavaScript, the system displays error messages underneath the incorrect fields.</p> <p>→ Return to Step 5</p>		

CONCLUSION:	The volunteer has updated her details and the system has verified that the updated details are complete and valid and has updated them in the Volunteer and Person tables in the database using LINQ.
POST-CONDITION:	The volunteer has updated their details on the system.
BUSINESS RULES	<ul style="list-style-type: none">• Only a volunteer can update their details on their profile.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none">• None
ASSUMPTIONS:	<ul style="list-style-type: none">• None
OPEN ISSUES:	None

USE CASE NAME:	Delete Volunteer	USE CASE TYPE	
USE CASE ID:	2.4	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Low	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer (Admin)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	Student		
DESCRIPTION:	<p>This use case describes the event of admin deleting a volunteer from the system. The use case begins when admin has logged onto the system via the static webpage and clicks on the “Members” drop down list on the navigation bar. The admin will then click on the “Volunteer” dropdown list item and the Volunteers webpage will be displayed. The admin will search for the relevant volunteer by using the list box containers or the search textbox functionality. Once the admin has selected a volunteer and clicked on the trashcan icon, the system will display a message confirming deletion of the volunteer. The use case concludes when the admin has successfully deleted a volunteer from the system.</p>		
PRE-CONDITION:	The admin has logged into the system.		
TRIGGER:	The admin wants to delete a volunteer from the system.		
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: The admin wants to delete a volunteer and clicks the “Members” drop down list.	System Response Manual Action	Automated Action Step 2: The system prompts admin to search for a volunteer (Invoke Use Case 2.2 Search Volunteer)
	Step 3: The admin clicks the “Delete” button. See Webpage 2.4.1 Delete Volunteer		Step 4: Using JavaScript, the system displays a modal with the message “Are you sure? This volunteer will be deleted forever from the system?” with a “Cancel” and “Confirm” button beneath the message.

			On webpage 2.4.2 Delete Volunteer
	Step 5: The admin clicks on the “Confirm” button.		Step 6: Using C# and SQL the system deletes the relevant volunteer from the Volunteer and Person tables in the database. The Volunteer and Person tables are connected via the PersonID key.
			Step 7: Using JavaScript, the system displays a modal with the message “You have successfully deleted the volunteer” followed by a “Confirm” button.
ALTERNATE COURSES:	None		
CONCLUSION:	Admin has searched for and deleted a volunteer. The system has displayed a warning message and the admin has confirmed that they wish to delete the volunteer. The use case concludes when the system confirms that the volunteer has been successfully deleted.		
POST-CONDITION:	The admin has deleted a volunteer/s on the system.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only an admin member can delete a volunteer from the system. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • After a volunteer is deleted, the volunteer should not be able to log into the system. 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Create Volunteer Type	USE CASE TYPE
USE CASE ID:	2.5	Business Requirements: <input type="checkbox"/>
PRIORITY:	High	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 	
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 	
DESCRIPTION:	<p>This use case describes the event of Admin wanting to create a volunteer type. Admin clicks the members menu item, which displays a dropdown list with a “Volunteers” item. Admin clicks the “Volunteers” item and the system displays the Volunteer webpage with two tabs, “Volunteers” and “Volunteer Type”. Admin clicks the “Volunteer Type” tab then clicks create volunteer type. Admin enters the required information and the system validates and stores the information. This use case concludes by a volunteer type being added to the system.</p>	
PRE-CONDITION:	Admin is logged into the System.	
TRIGGER:	Admin wants to add a new volunteer type to the system.	
TYPICAL COURSE	Actor Action	System Response
OF EVENTS:	Manual Action	Automated Action
	<p>Step 1: Admin clicks the “Volunteers” dropdown list item under the “Members” navigation bar menu item. Members > Volunteers</p>	<p>Step 2: The system displays Screen: 2.2.1 Search Volunteer. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System

			<ul style="list-style-type: none">➤ Members➤ Events➤ Reading content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Volunteers" and "Volunteer Type", the "Volunteer" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the volunteers' details populated by the Volunteer table. The list contains the following information about the volunteers as retrieved from the Person table:<ul style="list-style-type: none">• Name• Surname• Email Address
--	--	--	--

		<ul style="list-style-type: none"> • Phone Number • DateOfBirth <p>The following information from the <u>Volunteer</u> table:</p> <ul style="list-style-type: none"> • VolunteerTypeID • ResID <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are three buttons entitled "Create a new volunteer type", "Update", "Delete" and "Return". Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the "Volunteer Type" tab. Screen:2.2.1 Search Volunteer</p>	<p>Step 4: The system displays Screen: 2.6.1 Search Volunteer Type. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains

			<p>the link to the user's notifications.</p> <ul style="list-style-type: none"> ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Volunteers" and "Volunteer Type", the "Volunteer Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled "Volunteer Type". ➤ A container below the header containing a list of all the volunteer types details populated by the <u>Volunteer Type</u> table. The list contains the following information about the volunteer type as retrieved from the <u>Volunteer Type</u> table: • Description <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button</p>
--	--	--	--

			<p>labelled “Search”. Below the container are four buttons entitled “Create a new volunteer type”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin clicks the “Create a New Volunteer Type” button. Screen: 2.6.1 Search Volunteer Type</p>		<p>Step 6: The system displays Screen: 2.5.2 Create Volunteer Type. This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains a header entitled “Create</p>

			<p>a new Volunteer" with the following elements in a container below:</p> <ul style="list-style-type: none"> ➤ "Description" (textbox) ➤ "Create" button <p>Below to the right of the container are two buttons entitled "Create" and "Return"</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks</p>
	<p>Step 7: Admin enters the required information into the textbox and clicks the "Create" button.</p> <p>Screen: 2.5.2 Create Volunteer Type</p>		<p>Step 8: The system validates that the "Volunteer Type Description" textbox contains no more than 25 characters.</p>
			<p>Step 9: The system adds the following information to the Volunteer Type table:</p> <ul style="list-style-type: none"> • Description • VolunteerTypeID (automatically generated per addition)
			<p>Step 10: The system displays a modal with the following message, "This volunteer type has been successfully created" with a button below the message entitled "Confirm".</p>
	<p>Step 11: Admin clicks the "Confirm" button.</p>		<p>Step 12: The system displays Screen: 2.5.2 Create Volunteer Type. This webpage contains a header, body and footer.</p>

			<p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains a header entitled "Create a new Volunteer" with the following elements in a container below:</p> <ul style="list-style-type: none"> ➤ "Description" (textbox) ➤ "Create" button <p>Below to the right of the container are two buttons entitled "Create" and "Return"</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks</p>
ALTERNATE COURSES:	Alt Step 7: Admin chooses not to create a volunteer type and clicks the "Return" button.		

	<p>→ Go to Step 4</p> <p>Alt Step 8: The “Volunteer Type Description” textbox contains more than 25 characters. The system displays a viewbag error message below the textbox indicating the maximum characters allowed is 25.</p> <p>→ Go to Step 6</p>
CONCLUSION:	Admin has created a volunteer type.
POST-CONDITION:	A volunteer type is added to the Volunteer Type table.
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can create a volunteer type. • None
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Search Volunteer Type	USE CASE TYPE
USE CASE ID:	2.6	Business Requirements: <input type="checkbox"/>
PRIORITY:	High	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 	
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 	
DESCRIPTION:	<p>This use case describes the event of admin wanting to search for a specific volunteer type. Admin navigates to the Volunteer webpage and clicks on the Volunteer Type tab. The system displays a container containing a list of volunteer types, admin uses the search bar functionality to search for a specific volunteer type. The system displays the search results. The use case concludes by admin viewing the specific volunteer type information.</p>	
PRE-CONDITION:	Admin is logged into the System.	
TRIGGER:	Admin wants to search for a volunteer type.	
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: Admin clicks the “Volunteers” dropdown list item under the “Members” navigation bar menu item. Members > Volunteers	System Response Manual Action Automated Action Step 2: The system displays Screen: 2.2.1 Search Volunteer. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively: ➤ TRWLA Management System ➤ Members ➤ Events

			<ul style="list-style-type: none">➤ Reading content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Volunteers" and "Volunteer Type", the "Volunteer" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the volunteers' details populated by the Volunteer table. The list contains the following information about the volunteers as retrieved from the Person table:• Name• Surname• Email Address• Phone Number• DateOfBirth
--	--	--	---

			<p>The following information from the <u>Volunteer</u> table:</p> <ul style="list-style-type: none"> • VolunteerTypeID • ResID <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are three buttons entitled "Create a new volunteer type", "Update", "Delete" and "Return". Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the "Volunteer Type" tab. Screen: 2.2.1 Search Volunteer</p>		<p>Step 4: The system displays Screen: 2.6.1 Search Volunteer Type. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the

			<p>user's notifications.</p> <ul style="list-style-type: none"> ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Volunteers" and "Volunteer Type", the "Volunteer Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled "Volunteer Type". ➤ A container below the header containing a list of all the volunteer types details populated by the <u>Volunteer Type</u> table. The list contains the following information about the volunteer type as retrieved from the <u>Volunteer Type</u> table: • Description <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below</p>
--	--	--	---

			<p>the container are four buttons entitled “Create a new volunteer type”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the textbox and clicks the “Search” button. Screen: 2.6.1 Search Volunteer Type</p>		<p>Step 6: The system performs a search and displays Screen: 2.6.1 Search Volunteer Type with all the appropriate results. This webpage contains a header and body.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items.

			<p>The body contains two tabs, “Volunteers” and “Volunteer Type”, the “Volunteer Type” tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Volunteer Type”.➤ A container below the header containing a list of all the relevant volunteer types details according to the search populated by the <u>Volunteer Type</u> table. The list contains the following information about the volunteer type as retrieved from the <u>Volunteer Type</u> table:<ul style="list-style-type: none">● DescriptionAbove the container is a textbox to the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create a New Volunteer Type”, “Update”, “Delete” and “Return”.
--	--	--	--

			Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
	<p>Step 7: Admin views the search results and clicks the “Return” button. Screen: 2.6.1 Search Volunteer Type</p>		<p>Step 8: The system displays the Main Menu webpage. This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items following elements from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of upcoming events populated by the Event table using entity

			<p>framework and LINQ queries. The list contains the following information about events as retrieved from the Event table:</p> <ul style="list-style-type: none">• Name• Date• Time• Venue• Summary <p>Each list item contains two buttons to the right of the item entitled “RSVP” and “View More Details”. Above the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. To the right of the body is another container with a list of upcoming events that the user has rsvp’d to, which is determined by the RSVP_Event table, and is populated by the Event table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the Event table:</p> <ul style="list-style-type: none">• Name• Date• Time• Venue
--	--	--	---

		Using CSS, JavaScript, C# and HTML.
ALTERNATE COURSES:	Alt Step 5: Admin scrolls through the container containing a list of volunteers types and finds the volunteer type information that they are looking for and clicks the “Return” button. On webpage: Volunteer Type → Go to Step 8	
	Alt Step 6: The system has no matches. The following message will be displayed in the container, “Your search has no results”. On webpage: Search Volunteer Type	
CONCLUSION:	The system displays the search results to admin.	
POST-CONDITION:	Admin has searched for a volunteer type.	
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can search a volunteer type. 	
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 	
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 	
OPEN ISSUES:	None	

USE CASE NAME:	Update Volunteer Type	USE CASE TYPE	
USE CASE ID:	2.7	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Medium	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy.		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of admin wanting to update a volunteer type. The use case begins by admin navigating to the Volunteers webpage and clicking on the Volunteer Type tab. The system displays a container with a list of all the volunteer types, which admin will search and click the “Update” button for the relevant list item. The system will display the list item in an editable format, which admin can update and save. This use case concludes by a volunteer type being updated.</p>		
PRE-CONDITION:	Admin is logged into the System.		
TRIGGER:	Admin wants to update a volunteer type.		
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: Admin clicks the “Volunteers” dropdown list item under the “Members” navigation bar menu item. Members > Volunteers	System Response	
		Manual Action	Automated Action
			<p>Step 2: The system displays Screen:2.2.1 Search Volunteer. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members

			<ul style="list-style-type: none"> ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Volunteers" and "Volunteer Type", the "Volunteer" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the volunteers' details populated by the Volunteer table. The list contains the following information about the volunteers as retrieved from the Person table: • Name • Surname • Email Address • Phone Number
--	--	--	--

			<ul style="list-style-type: none"> • DateOfBirth <p>The following information from the <u>Volunteer</u> table:</p> <ul style="list-style-type: none"> • VolunteerTypeID • ResID <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are three buttons entitled "Update", "Delete" and "Return". Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the "Volunteer Type" tab. Screen:2.2.1 Search Volunteer</p>		<p>Step 4: The system displays Screen: 2.6.1 Search Volunteer Type. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the

			<p>user's notifications.</p> <ul style="list-style-type: none"> ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Volunteers" and "Volunteer Type", the "Volunteer Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled "Volunteer Type". ➤ A container below the header containing a list of all the volunteer types details populated by the <u>Volunteer Type</u> table. The list contains the following information about the volunteer type as retrieved from the <u>Volunteer Type</u> table: • Description <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below</p>
--	--	--	---

			<p>the container are four buttons entitled “Create a new volunteer type”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the textbox and clicks the “Search” button. 2.6.1 Search Volunteer Type</p>		<p>Step 6: The system performs a search and displays Screen: 2.6.1 Search Volunteer Type. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, “Volunteers” and “Volunteer Type”, the</p>

			<p>“Volunteer Type” tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Volunteer Type”.➤ A container below the header containing a list of all the volunteer types details populated by the <u>Volunteer Type</u> table according to the search. The list contains the following information about the volunteer type as retrieved from the <u>Volunteer Type</u> table:<ul style="list-style-type: none">● DescriptionAbove the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create a new volunteer type”, “Update”, “Delete” and “Return”.Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
--	--	--	--

	<p>Step 7: Admin clicks “Update” On webpage: Search Volunteer Type</p>	<p>Step 8: The system displays Screen: 2.7.1 Update Volunteer Type. This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains a header entitled “Create a new Volunteer” with the following elements in a container below:</p> <ul style="list-style-type: none"> ➤ “Description” (textbox) ➤ “Save” button <p>Below to the right of the container are two</p>
--	---	---

			buttons entitled “Create” and “Return” Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks
	Step 9: Admin updates the volunteer type description textbox and clicks “Save” Screen: 2.7.1 Update Volunteer Type		Step 10: The system validates the information updated by admin is no more than 25 characters.
			Step 11: The system updates the following attributes in the <u>Volunteer Type</u> table: <ul style="list-style-type: none"> • Description per the VolunteerTypeID
			Step 12: The system displays a modal with the following message, “Volunteer type successfully updated” with a button below the message entitled “Confirm”.
	Step 13: Admin clicks the “Confirm” button.		Step 14: The system displays the Volunteer Type Webpage. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively: <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery

			<ul style="list-style-type: none">➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Volunteers" and "Volunteer Type", the "Volunteer Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Volunteer Type".➤ A container below the header containing a list of all the volunteer types details populated by the <u>Volunteer Type</u> table. The list contains the following information about the volunteer type as retrieved from the <u>Volunteer Type</u> table:
--	--	--	--

			<ul style="list-style-type: none"> • Description <p>Each list item contains two buttons to the right of the item entitled “Update” and “Delete”. Above the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are two buttons entitled “Create” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
ALTERNATE COURSES:	Alt Step 5a: Admin chooses not to update a volunteer type and clicks the “Return” button. → Go to Step 2		
	Alt Step 5b: Admin scrolls through the container containing a list of volunteer types and finds the volunteer type information that they are looking for and clicks the “Update” button. On webpage: Volunteer Type → Go to Step 8		
	Alt Step 12: The “Volunteer Type Description” textbox contains more than 25 characters. The system displays a ViewBag error message below the textbox indicating the maximum characters allowed is 25. → Go to Step 8		
CONCLUSION:	Admin has updated a volunteer type.		
POST-CONDITION:	The system updates the Volunteer Type table		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can update a volunteer type. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Delete Volunteer Type		USE CASE TYPE
USE CASE ID:	2.8	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Low	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of admin deleting a volunteer type. The use case begins by admin navigating to the Volunteers webpage and clicking on the Volunteer Type tab. The system displays a container with a list of all the volunteer types, which admin will search and click the “Delete” button for the relevant list item. The system confirms if admin wants to delete that volunteer type. This use case concludes by a volunteer type being deleted.</p>		
PRE-CONDITION:	Admin is logged into the System.		
TRIGGER:	Admin wants to delete a volunteer type.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: Admin clicks the “Volunteers” dropdown list item under the “Members” navigation bar menu item. Members > Volunteers</p>		<p>Step 2: The system displays Screen:2.2.1 Search Volunteer. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events

			<ul style="list-style-type: none">➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Volunteers" and "Volunteer Type", the "Volunteer" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the volunteers' details populated by the Volunteer table. The list contains the following information about the volunteers as retrieved from the Person table:• Name• Surname• Email Address• Phone Number• DateOfBirth
--	--	--	---

			<p>The following information from the <u>Volunteer</u> table:</p> <ul style="list-style-type: none"> • VolunteerTypeID • ResID <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are three buttons entitled "Update", "Delete" and "Return". Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the "Volunteer Type" tab. Screen:2.2.1 Search Volunteer</p>		<p>Step 4: The system displays Screen: 2.6.1 Search Volunteer Type. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.

			<p>➤ A profile picture which contains a link to the User dropdown list items.</p> <p>The body contains two tabs, “Volunteers” and “Volunteer Type”, the “Volunteer Type” tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Volunteer Type”.➤ A container below the header containing a list of all the volunteer types details populated by the Volunteer Type table. The list contains the following information about the volunteer type as retrieved from the Volunteer Type table:<ul style="list-style-type: none">• DescriptionAbove the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create”
--	--	--	---

			<p>a new volunteer type”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the textbox and clicks the “Search” button. 2.6.1 Search Volunteer Type</p>		<p>Step 6: The system performs a search and displays Screen: 2.6.1 Search Volunteer Type. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, “Volunteers” and “Volunteer Type”, the “Volunteer Type” tab is</p>

			<p>displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Volunteer Type”.➤ A container below the header containing a list of all the volunteer types details populated by the <u>Volunteer Type</u> table according to the search. The list contains the following information about the volunteer type as retrieved from the <u>Volunteer Type</u> table:<ul style="list-style-type: none">• DescriptionAbove the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create a new volunteer type”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
--	--	--	---

	<p>Step 7: Admin clicks the “Delete” button on the specific list item in the container.</p> <p>On webpage: Search Volunteer Type</p>		<p>Step 8: The system displays a modal with the following message, Header, “Are you sure?” “This Volunteer type will be deleted from the system forever and is not recoverable” with a button below entitled “Confirm”.</p>
	<p>Step 9: Admin clicks the “Delete” button on the modal.</p>		<p>Step 10: The system deleted the following attributes in the Volunteer Type table:</p> <ul style="list-style-type: none"> • Description • VolunteerTypeID <p>Per the VolunteerTypeID</p>
			<p>Step 4: The system displays Screen: 2.6.1 Search Volunteer Type. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications.

			<ul style="list-style-type: none">➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, “Volunteers” and “Volunteer Type”, the “Volunteer Type” tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Volunteer Type”.➤ A container below the header containing a list of all the volunteer types details populated by the Volunteer Type table. The list contains the following information about the volunteer type as retrieved from the Volunteer Type table:<ul style="list-style-type: none">• DescriptionAbove the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create”
--	--	--	--

			a new volunteer type”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
ALTERNATE COURSES:	Alt Step 9: Admin clicks the “Cancel” button. → Go to Step 6		
CONCLUSION:	Admin deletes a volunteer type.		
POST-CONDITION:	The Volunteer Type table has been updated.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can delete a volunteer type. • None 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS			
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:			

USE CASE NAME:	Generate Unique Code	USE CASE TYPE
USE CASE ID:	2.9	Business Requirements: <input type="checkbox"/>
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin)	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • User: Volunteer (ERA) 	
OTHER INTERESTED STAKEHOLDERS:	None	
DESCRIPTION:	<p>This use case describes the event of generating a unique code for a volunteer to enter when registering on the TRWLA System. The use case is initiated when the user has been officially recognized as a volunteer in the academy. An admin member will login to the system, click on the “Members” drop down list on the navigation bar on their main menu and click the “Volunteer” list item. Once the admin member has clicked on the “Volunteer” list item, a list of existing volunteers will be displayed on the TRWLA Volunteers webpage. The admin member will then click on the “Generate Unique Code” button where a modal will display the unique code generated from the system and a textbox where the email address of the potential volunteer needs to be entered. Once the email address of the volunteer has been entered, the admin member will click on the “Email” button and the system will send the email to the intended volunteer in order to register on the system.</p>	
PRE-CONDITION:	Admin has logged onto the system.	
TRIGGER:	The admin member wishes to generate a unique code for a specific volunteer.	
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: The admin member wishes to generate a unique code for a specific volunteer. Step 2: The admin member clicks on the “Members” drop down list.	System Response Manual Action Automated Action

			<p>of available items on the drop down list such as:</p> <ul style="list-style-type: none"> ➤ Volunteers ➤ Students ➤ Alumni
	<p>Step 4: The admin member clicks on the “Volunteers” list item.</p>		<p>Step 5: The system, using JavaScript, HTML and CSS, displays the TRWLA Volunteers webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Generate unique code” button. ➤ “Volunteers” h1 header ➤ “Find by name:” textbox ➤ “Search” button

			<ul style="list-style-type: none"> ➤ Containers with the following h4 headers above them: <ul style="list-style-type: none"> ➤ Access Right ➤ First Name ➤ ResName ➤ Description <p>All populated from the Volunteer table using Entity Framework and LINQ queries.</p> <p>Footer</p> <ul style="list-style-type: none"> • “Return” button.
	<p>Step 6: The admin member clicks on the “Generate Unique Code” button.</p>		<p>Step 7: The system, using JavaScript, CSS and HTML, displays the Generate Unique Code Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p>

			<ul style="list-style-type: none"> ➤ “Generate Unique Code” h1 header ➤ “Unique codes can only be used once” h4 header ➤ “Generate Unique Code” button ➤ Code container with the following headers: ➤ “Code” ➤ “Status” <p>These codes are populated and stored in the <u>UniqueCode</u> table using Entity Framework and LINQ queries.</p>
	<p>Step 8: The admin member clicks the “Generate Unique Code” button.</p>		<p>Step 9: The system, using C#, generates a random unique code which is a 5 digit numeric value and stores it in the <u>UniqueCode</u> table, using SQL Entity Framework and LINQ queries.</p>
			<p>Step 10: The system, using JavaScript, adds the unique code to the Generate Unique Code Webpage and marks the status as “Active”.</p>
ALTERNATE COURSES:			
CONCLUSION:			
<p>The system, using JavaScript, adds the unique code to the Generate Unique Code Webpage and marks the status as “Inactive”.</p>			
POST-CONDITION:			
<p>A unique code has been generated.</p>			
BUSINESS RULES			
<ul style="list-style-type: none"> • Only an admin member can generate a unique code for a potential volunteer. 			

	<ul style="list-style-type: none"> Once a unique code has been generated, it cannot be re-used and it must be stored in the <u>UniqueCode</u> table. A unique code must be given to the volunteer in order to register. Once the volunteer registers, the code's status will be changed to "Inactive". An active status will be changed to inactive after a period of 7 days if it is unused.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> None
ASSUMPTIONS:	<ul style="list-style-type: none"> None
OPEN ISSUES:	None

USE CASE NAME:	Register Admin	USE CASE TYPE	
USE CASE ID:	2.10	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • User: Volunteer 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of an admin member registering a volunteer as admin on the system. The use case begins when the admin member clicks on the “Register Admin” list item on the “User” drop down list. The system will then display a list of volunteers registered on the TRWLA system. The admin member will then click on the toggle button to register a volunteer as admin. Once the admin member has registered the volunteer, their admin status will change to green active. The use case concludes when a volunteer has been registered as an admin member on the system.</p>		
PRE-CONDITION:	The volunteer has already been registered on the system.		
TRIGGER:	The admin member wants to register a volunteer as an admin member on the system.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The admin member wants to register a volunteer as an admin member on the system and clicks on the “User” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the following list items:</p> <ul style="list-style-type: none"> ➢ Reports ➢ Register Admin ➢ User Type ➢ Log Out
	<p>Step 3: The admin member clicks the “Register Admin” list item.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Register Admin webpage</p>

			<p>with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Register & Deregister Admin” h1 header➤ “Number” h3 header➤ “Name” h3 header➤ “Surname” h3 header➤ “Admin Status” h3 header➤ “Register/Deregister” h3 button● List of volunteer names populated from the <u>Volunteer</u> and <u>UserType</u> tables using Entity Framework and LINQ queries. <p>Footer</p>
--	--	--	---

			➤ “Return” button
	Step 5: The admin member locates the volunteer and clicks on the “on” toggle button under the “Register/Deregister” h3 header.		Step 6: The system, using JavaScript, displays the toggle button by the selected volunteer as “On” and changes the “Admin Status” to a green “Active” symbol.
			Step 7: The system, using C# and SQL, then grants the selected volunteer the admin rights to change features on the system and updates the UserType table using Entity Framework and LINQ queries.
ALTERNATE COURSES:			<p>Alt Step 5a: The admin member clicks the “Return” button. The system, using JavaScript, CSS and HTML, displays the main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> • “Event Name” h3 header • “Summary” h4 header

	<ul style="list-style-type: none"> • “Date of the event” h4 header • “Time of event” h4 header • “Venue name”h4 header • “RSVP” button • “Details” button • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➢ “Create” button ➢ “Update” button ➢ “Delete” button ➢ “Return” button ➢ “Log Attendance” button <p>On webpage: Admin Main Menu Webpage</p>
	<p>Alt Step 5b: The admin member clicks on the “off” toggle button. The system, using JavaScript, displays the toggle button by the selected volunteer as “Off” and changes the “Admin Status” to a red “Inactive” symbol.</p> <p>→ Invoke UC 2.11 Deregister Admin</p> <p>On webpage: Deregister Admin</p>
CONCLUSION:	The system, using C# and SQL, then grants the selected volunteer the admin rights to change features on the system and updates the UserType table using Entity Framework and LINQ queries.
POST-CONDITION:	The volunteer has been successfully registered as an admin member.
BUSINESS RULES	<ul style="list-style-type: none"> • Only existing admin members can register other volunteers as admin members.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • Once a volunteer becomes admin, a new main menu and functionality of the system should be displayed to them when logged in.
ASSUMPTIONS:	<ul style="list-style-type: none"> • Board members have decided to allow the specific volunteer to be an admin member.
OPEN ISSUES:	None

USE CASE NAME:	Deregister Admin	USE CASE TYPE	
USE CASE ID:	2.11	Business Requirements: <input type="checkbox"/>	
PRIORITY:	Low	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • User: Volunteer 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of an admin member deregistering a volunteer as admin on the system. The use case begins when the admin member clicks on the “Register Admin” list item on the “User” drop down list. The system will then display a list of volunteers registered on the TRWLA system. The admin member will then click on the toggle button to deregister a volunteer as admin. Once the admin member has deregistered the volunteer, their admin status will change to red inactive. The use case concludes when a volunteer has been deregistered as an admin member on the system.</p>		
PRE-CONDITION:	<p>The volunteer has already been registered on the system and is an existing admin member.</p>		
TRIGGER:	<p>The admin member wants to deregister a volunteer as an admin member on the system.</p>		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The admin member wants to deregister a volunteer as an admin member on the system and clicks on the “User” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the following list items:</p> <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Log Out
	<p>Step 3: The admin member clicks the “Register Admin” button.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the</p>

			<p>Register Admin webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Register & Deregister Admin” h1 header ➤ “Number” h3 header ➤ “Name” h3 header ➤ “Surname” h3 header ➤ “Admin Status” h3 header ➤ “Register/Deregister” h3 button • List of volunteer names populated from the <u>Volunteer</u> and <u>UserType</u> tables using Entity Framework and LINQ queries.
--	--	--	--

			Footer ➤ “Return” button
	Step 5: The admin member locates the volunteer and clicks on the “off” toggle button under the “Register/Deregister” h3 header.		Step 6: The system, using JavaScript, displays the toggle button by the selected volunteer as “Off” and changes the “Admin Status” to a red “Inactive” symbol.
			Step 7: The system, using C# and SQL, then denies the selected volunteer the admin rights to change features on the system and updates the UserType table using Entity Framework and LINQ queries.
ALTERNATE COURSES:	<p>Alt Step 5a: The admin member clicks the “Return” button. The system, using JavaScript, CSS and HTML, displays the main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> • “Event Name” h3 header 		

	<ul style="list-style-type: none"> • “Summary” h4 header • “Date of the event” h4 header • “Time of event” h4 header • “Venue name” h4 header • “RSVP” button • “Details” button • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➢ “Create” button ➢ “Update” button ➢ “Delete” button ➢ “Return” button ➢ “Log Attendance” button <p>On webpage: Admin Main Menu Webpage</p>
	<p>Alt Step 5b: The admin member clicks on the “On” toggle button. The system, using JavaScript, displays the toggle button by the selected volunteer as “On” and changes the “Admin Status” to a green “Active” symbol.</p> <p>→ Invoke UC 2.10 Register Admin</p> <p>On webpage: Register Admin Webpage</p>
CONCLUSION:	The system, using C# and SQL, then denies the selected volunteer the admin rights to change features on the system and updates the UserType table using Entity Framework and LINQ queries.
POST-CONDITION:	The volunteer has been successfully deregistered as an admin member.
BUSINESS RULES	<ul style="list-style-type: none"> • Only existing admin members can deregister other volunteers as admin members.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • Once a volunteer has been deregistered, only the volunteer main menu will be displayed to them.
ASSUMPTIONS:	<ul style="list-style-type: none"> • Board members have decided to allow the specific volunteer to be removed as an admin member.
OPEN ISSUES:	None

2.3.3 Student Narratives

USE CASE NAME:	Register Student	USE CASE TYPE	
USE CASE ID:	3.1	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Student (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • User: Volunteer 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a student registering themselves on the system. The use case begins when the student joins TRWLA and wants to register on the system. The student will go to the static webpage and click on the “Login” link in the header section of the webpage. The system will then display the Login webpage. The student will then click on the “Register as a new user” button. The system will then display the Register Webpage where the student must enter required details, this uses the AUC 1 Register User. The student will enter the required details. The system will validate that the entered details is correct and display a modal with two radio button, namely “Student” and “Volunteer”. The student will then click on the “Student” radio button and the Register Student webpage will be displayed with empty fields that the student needs to complete. Once the student has entered the required details, the system will validate if the details entered is correct. The use case concludes when the student has successfully registered themselves on the system.</p>		
PRE-CONDITION:	The student has joined TRWLA and wants to register as a student on the system		
TRIGGER:	The student wants to register on the TRWLA system.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1: The student wants to register on the system and clicks on the “Student” radio button.</p>		<p>Step 2: The system, using JavaScript, CSS and HTML, displays the Register Student</p>

			<p>webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Home” navigation bar item➤ “About” navigation bar item➤ “Contact” navigation bar item➤ “Gallery” navigation bar item➤ “Login” navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Register As a New Student” h1 header➤ “Register” h2 header➤ “Name” textbox➤ “Surname” textbox➤ “Phone Number” textbox➤ “Email Address” textbox➤ “Date Of Birth” date picker➤ “Security Question” drop down list
--	--	--	--

			<ul style="list-style-type: none"> ➤ “Security Question Answer” textbox ➤ “Student type” drop down list populated from the <u>StudentType</u> table using Entity Framework and LINQ queries. ➤ “Residence” drop down list populated from the <u>Residence</u> table using Entity Framework and LINQ queries. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Register” button ➤ “Return” button <p>Invoke AUC 1 Register User</p>
	<p>Step 3: The student enters all of the relevant information in the textboxes and clicks the “Register” button.</p>		<p>Step 4: The system, using C#, JavaScript and CSS, verifies that all required fields have been filled in and that the fields contain valid details:</p> <ul style="list-style-type: none"> • Name (Maximum 35 characters) • Surname (Maximum 35 characters) • Phone Number (Max 10 characters (No spaces)) • Email Address (Minimum 7 characters, Maximum 255 characters. Must be a full validated email address.)

			<ul style="list-style-type: none"> • Date of Birth (Maximum 10 characters in the format CCYY-MM-DD) • Security Question (Selected from predetermined options) • Security Question Answer (Maximum 25 characters) • Residence (Selected from predetermined options as retrieved from the Residence table. The Student and Residence tables are connected via the ResID) • Student Type (Selected from predetermined options as retrieved from the StudentType table, using Entity Framework and LINQ queries.
			<p>Step 5: The system, using C# and SQL, adds the new student details to the Student table as well as the security answer to the SecurityAnswer table using Entity Framework and LINQ queries.</p>
			<p>Step 6: The system, using C# and JavaScript, displays a modal with a message that says “Registration was successful” with a “Confirm” button.</p>

	<p>Step 7: The student clicks on the “Confirm” button.</p>	<p>Step 8: The system, using JavaScript, CSS and HTML, displays the login webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ “Home” navigation bar item ➤ “About” navigation bar item ➤ “Contact” navigation bar item ➤ “Gallery” navigation bar item ➤ “Login” navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Login” h1 header ➤ “Use a local account to log in” h2 header ➤ “Email Address” textbox ➤ “Password” textbox ➤ “Remember me?” checkbox ➤ “Login” button ➤ “Register as a new user” hyperlink button ➤ “Forgot Password” hyperlink button <p>Invoke UC 1.3 Login</p>
ALTERNATE COURSES:		<p>Alt Step 4: The information entered is incorrect and does not meet the system standards as specified. The system will display a ViewBag error message below the required textbox stating what the minimum and maximum characters are as well as a Modal with the following message: “The required fields have not been completed or the information</p>

	<p>entered is invalid" with a "Try Again" button. The student clicks the "Try Again" button.</p> <p>→ Go back to Step 3</p> <p>On webpage: Register Student Webpage</p>
CONCLUSION:	<p>The system, using JavaScript, CSS and HTML, displays the login webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ "Home" navigation bar item ➤ "About" navigation bar item ➤ "Contact" navigation bar item ➤ "Gallery" navigation bar item ➤ "Login" navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ "Login" h1 header ➤ "Use a local account to log in" h2 header ➤ "Email Address" textbox ➤ "Password" textbox ➤ "Remember me?" checkbox ➤ "Login" button ➤ "Register as a new user" hyperlink button ➤ "Forgot Password" hyperlink button <p>Invoke UC 1.3 Login</p>
POST-CONDITION:	The student has been successfully registered on the system.
BUSINESS RULES	<ul style="list-style-type: none"> • None
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Search Student	USE CASE TYPE	
USE CASE ID:	3.2	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	• None		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a volunteer searching for a student. The use case begins when the volunteer has logged onto the system via the static webpage and clicks on the “Members” drop down list on the navigation bar. The volunteer will then click on the “Students” link item and the Students webpage will be displayed. The volunteer will search for the relevant student by using the list box containers or the search textbox functionality. Once the volunteer has selected a student, the system will display the selected student’s details on the Search student webpage. The use case concludes when the user is satisfied with the search details and returns to their main menu.</p>		
PRE-CONDITION:	The volunteer has logged into the system.		
TRIGGER:	The volunteer wants to search for a student.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	Step 1: The volunteer wants to search for a student and clicks the “Members” drop down list.		Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as: ➤ Volunteers ➤ Students ➤ Alumni
	Step 3: The volunteer clicks the “Students” list item from the drop down list.		Step 4: The system, using JavaScript, CSS and HTML, displays the Search Students

			<p>webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Students” h1 header➤ “Find by name” textbox➤ “Search” button.➤ A container with a list of students with the following headings above the container:<ul style="list-style-type: none">➤ Name➤ Surname➤ Phone number➤ Email Address➤ Date of Birth➤ Student Type➤ Residence <p>Footer</p>
--	--	--	--

			<ul style="list-style-type: none"> ➤ “Delete” button ➤ “Return” button <p>The students in the student container has been retrieved from the <u>Student</u> table using Entity Framework and LINQ Queries.</p>
	<p>Step 5: The volunteer enters a student name in the textbox and clicks the “Search” button.</p>		<p>Step 6: The system, using C# and JavaScript, processes the search and displays an updated Search Student webpage based on the search result with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Students” h1 header ➤ “Find by name” textbox

			<ul style="list-style-type: none"> ➤ “Search” button. ➤ A container with a list of students with the following headings above the container: <ul style="list-style-type: none"> ➤ Name ➤ Surname ➤ Phone number ➤ Email Address ➤ Date of Birth ➤ Student Type ➤ Residence <p>Footer</p> <ul style="list-style-type: none"> ➤ “Delete” button ➤ “Return” button <p>The students in the student container has been retrieved from the Student table using Entity Framework and LINQ Queries.</p>
ALTERNATE COURSES:	<p>Alt Step 5a: The volunteer searches for the relevant student and clicks on the “Delete” button to delete the student.</p> <p>→ Invoke UC 3.4 Delete Student</p> <p>Alt Step 5d: The volunteer clicks on the “Return” button. The system displays the Volunteer Main Menu webpage.</p>		
	<p>Alt Step 6: The search returned no results and the system displays a Modal with the following message: “No students match your search criteria” with a “Try Again” button. The volunteer clicks the “Try Again” button.</p> <p>→ Return to Step 5</p> <p>On Webpage: Search Student</p>		
CONCLUSION:	<p>The system, using C# and JavaScript, processes the search and displays an updated Search Student webpage based on the search result with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item 		

	<ul style="list-style-type: none"> ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Students” h1 header ➤ “Find by name” textbox ➤ “Search” button. ➤ A container with a list of students with the following headings above the container: ➤ Name ➤ Surname ➤ Phone number ➤ Email Address ➤ Date of Birth ➤ Student Type ➤ Residence <p>Footer</p> <ul style="list-style-type: none"> ➤ “Delete” button ➤ “Return” button <p>The students in the student container has been retrieved from the Student table using Entity Framework and LINQ Queries.</p>
POST-CONDITION:	The volunteer has searched for a student.
BUSINESS RULES	<ul style="list-style-type: none"> • None
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • The student’s username and password must be kept private.
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Update Student	USE CASE TYPE	
USE CASE ID:	3.3	Business Requirements: <input type="checkbox"/>	
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Student (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a student updating their details on the TRWLA system. The use case begins when the student has logged onto the system via the Login webpage. The system then displays the Student Main Menu webpage. The student will then click on the drop down list arrow next to the profile picture icon and click on “My Profile”. The system will then display the student’s details and the student can then update her profile accordingly. They will then update and save their relevant details, and the use case will conclude when they have successfully updated their details.</p>		
PRE-CONDITION:	The student has logged into the system.		
TRIGGER:	The student wants to update their profile details.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	Step 1: The student wants to update their profile details and clicks on the “User drop down list”.		Step 2: The system, using JavaScript, displays the following list items: <ul style="list-style-type: none"> ➤ My Profile ➤ Logout
	Step 3: The student clicks on the “My Profile” list item.		Step 4: The system, using C#, JavaScript, CSS and HTML, displays the My Profile webpage with the following layout: Header Navigation bar from left to right:

			<ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Volunteers” drop down list navigation bar item➤ “Home” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “My Profile” header label➤ “Name” textbox➤ “Surname” textbox➤ “Student Number” textbox➤ “Phone Number” textbox➤ “Email Address” textbox➤ “Date Of Birth” date picker➤ “Race” textbox➤ “Language” textbox➤ “Year of Study” textbox➤ “Degree” drop down list➤ “Residence” drop down list populated from the <u>Residence</u> table using Entity
--	--	--	--

			<p>Framework and LINQ queries.</p> <ul style="list-style-type: none"> ➤ “Student type” drop down list populated from the <u>StudentType</u> table using Entity Framework and LINQ queries. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button ➤ “Save” button <p>All fields are populated from the <u>Student</u> table using Entity Framework and LINQ queries.</p>
	<p>Step 5: The student updates their details and clicks on the “Save” button.</p>		<p>Step 6: The system, using C#, JavaScript and CSS, verifies that all required fields have been filled in and that the fields contain valid details:</p> <ul style="list-style-type: none"> • Name (Maximum 35 characters) • Surname (Maximum 35 characters) • Phone Number (Max 15 characters (No spaces)) • Race (Maximum 10 characters) • Language (Maximum 10 characters) • Email Address (Minimum 7 characters, Maximum 255 characters. Must be a full validated email address.) • Date of Birth (Maximum 10

			<p>characters in the format CCYY-MM-DD)</p> <ul style="list-style-type: none"> • Year of Study (Maximum characters 2, numeric amount between 1-10) • Degree (Selected from pre-determined options) • Student Number (8 numeric characters) • Residence (Selected from predetermined options as retrieved from the Residence table. The Student and Residence tables are connected via the ResID) • Student Type (Selected from predetermined options as retrieved from the StudentType table.
			<p>Step 7: The system, using C# and JavaScript, displays a modal with the following message “Are you sure you would like to update your account?”, with the following two buttons:</p> <ul style="list-style-type: none"> ➤ “Confirm” ➤ “Cancel”
	<p>Step 8: The student clicks the “Confirm” button.</p>		<p>Step 9: The system, using C#, updates the student’s details in the Student table using Entity Framework and LINQ queries.</p>
			<p>Step 10: The system, using JavaScript, CSS and</p>

			<p>HTML, redirects the student to the main menu webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Volunteers” drop down list navigation bar item➤ “Home” navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon that is used to display notifications➤ Profile Picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Welcome User Name” h1 header➤ “Search upcoming events by Name” textbox
--	--	--	---

			<ul style="list-style-type: none">➤ “Here is your progress!” button➤ “Sort with the following:” h4 header➤ “Event name” button➤ “Date” button➤ “Upcoming TRWLA Events” h2 header➤ “My Upcoming Events” h2 header➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries:<ul style="list-style-type: none">• “Event Name” h3 header• “Summary” h4 header• “Date of the event” h4 header• “Time of event” h4 header• “Venue name” h4 header• “RSVP” button• “Details” button• “Not going” button in the “My upcoming events” container.
			<p>Footer</p> <ul style="list-style-type: none">➤ “Return” button

ALTERNATE COURSES:	<p>Alt Step 6: The information entered is incorrect and is not in the format of the specified system standards. The system displays a Modal with the following message: “The required fields have not been completed or the information entered is invalid” with a “Try Again” button. The student clicks the “Try Again” button.</p> <p>→ Return to step 5</p> <p>On webpage: My Profile Webpage</p>
	<p>Alt Step 8: The student clicks the “Cancel” button on the modal.</p> <p>→ Go to step 12</p> <p>On webpage: My Profile Webpage</p>
CONCLUSION:	<p>The system, using JavaScript, CSS and HTML, redirects the student to the main menu webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Here is your progress” button ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button ● “Not going” button in the “My upcoming events” container.

	Footer ➤ “Return” button
POST-CONDITION:	The student has updated their details on the system.
BUSINESS RULES	<ul style="list-style-type: none"> • Only a student can update their details on their profile.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Delete Student		USE CASE TYPE
USE CASE ID:	3.4		Business Requirements: <input type="checkbox"/>
PRIORITY:	Low		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	Student		
DESCRIPTION:	<p>This use case describes the event of a volunteer deleting a student from the system. The use case begins when the volunteer has logged onto the system via the static webpage and clicks on the “Members” drop down list on the navigation bar. The volunteer will then click on the “Students” link item and the Students webpage will be displayed. The volunteer will search for the relevant student by using the list box containers or the search textbox functionality. Once the volunteer has selected a student and clicked on the “Delete” button, the system will display a message confirming deletion of the student. The use case concludes when the volunteer has successfully deleted a student from the system.</p>		
PRE-CONDITION:	The volunteer has logged into the system.		
TRIGGER:	The volunteer wants to delete a student from the system.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to delete a student and clicks the “Members” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as:</p> <ul style="list-style-type: none"> ➤ Volunteers ➤ Students ➤ Alumni
	<p>Step 3: The volunteer clicks the “Students” list item from the drop down list.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the TRWLA Students</p>

			<p>webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Generate Graduate List” button➤ “Students” h1 header➤ “Find by degree” textbox➤ “Search” button.➤ A container with a list of students with the following headings above the container:<ul style="list-style-type: none">➤ Student Number➤ FirstName➤ Graduate➤ Degree➤ Year of Study➤ ResName
--	--	--	---

			<ul style="list-style-type: none"> ➤ “Update” button next to each individual student as per MVC layout ➤ “Details” button next to each individual student as per MVC layout ➤ “Delete” button next to each individual student as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The students in the student container has been retrieved from the Student table using Entity Framework and LINQ Queries.</p>
	<p>Step 5: The volunteer selects the relevant student and clicks the “Delete” button.</p>		<p>Step 6: The system, using C#, JavaScript, displays a modal with the following message, “This student profile will be deleted forever from the system?” with a “Confirm” button.</p>
	<p>Step 7: The volunteer clicks the “Confirm” button.</p>		<p>Step 8: The system, using c# and SQL, deletes the student from the Student table using Entity Framework and LINQ queries.</p>
	<p>Step 9: The volunteer clicks the “Return” button.</p>		<p>Step 10: The system, using JavaScript, CSS and HTML, displays the TRWLA Students webpage with the following layout: Header</p>

			<p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Generate Graduate List” button ➤ “Students” h1 header ➤ “Find by degree” textbox ➤ “Search” button. ➤ A container with a list of students with the following headings above the container: ➤ Student Number ➤ FirstName ➤ Graduate ➤ Degree ➤ Year of Study ➤ ResName ➤ “Update” button next to each individual student as per MVC layout
--	--	--	---

			<ul style="list-style-type: none"> ➤ “Details” button next to each individual student as per MVC layout ➤ “Delete” button next to each individual student as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The students in the student container has been retrieved from the Student table using Entity Framework and LINQ Queries.</p>
ALTERNATE COURSES:	Alt Step 6: The volunteer clicks the “Cancel” button. → Go to step 11 On webpage: TRWLA Students		
CONCLUSION:	The system, using JavaScript, CSS and HTML, displays the TRWLA Students webpage with the following layout: Header Navigation bar from left to right: <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item Body <ul style="list-style-type: none"> ➤ “Generate Graduate List” button ➤ “Students” h1 header ➤ “Find by degree” textbox ➤ “Search” button. ➤ A container with a list of students with the following headings above the container:<ul style="list-style-type: none"> ➤ Student Number ➤ FirstName ➤ Graduate ➤ Degree 		

	<ul style="list-style-type: none"> ➤ Year of Study ➤ ResName ➤ “Update” button next to each individual student as per MVC layout ➤ “Details” button next to each individual student as per MVC layout ➤ “Delete” button next to each individual student as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The students in the student container has been retrieved from the Student table using Entity Framework and LINQ Queries.</p>
POST-CONDITION:	The volunteer has deleted a student/s on the system.
BUSINESS RULES	<ul style="list-style-type: none"> • Only a volunteer or admin member can delete a student from the system.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • After a student is deleted, the student should not be able to log into the system.
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Generate Graduate List		USE CASE TYPE
USE CASE ID:	3.5		Business Requirements: <input type="checkbox"/>
PRIORITY:	Medium		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	Students		
DESCRIPTION:	<p>This use case describes the event of a volunteer generating a graduate list to view all the students who have progressed enough to graduate from the academy. The use case begins when the volunteer has logged onto the system via the static webpage and clicks on the “Members” drop down list on the navigation bar. The volunteer will then click on the “Students” link item and the Students webpage will be displayed. The volunteer will click on the “Generate Graduate List” hyperlink button. The system will then display a webpage with a list of the current year’s graduate students. The use case concludes when the volunteer has successfully generated a graduate list for the current year.</p>		
PRE-CONDITION:	The volunteer has logged into the system.		
TRIGGER:	The volunteer wants to generate a graduate list for the current year.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to generate a graduate list for the current year and clicks the “Members” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as:</p> <ul style="list-style-type: none"> ➤ Volunteers ➤ Students ➤ Alumni
	<p>Step 3: The volunteer clicks the “Students” list item from the drop down list.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the TRWLA Students</p>

			<p>webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Generate Graduate List” button➤ “Students” h1 header➤ “Find by degree” textbox➤ “Search” button.➤ A container with a list of students with the following headings above the container:<ul style="list-style-type: none">➤ Student Number➤ FirstName➤ Graduate➤ Degree➤ Year of Study➤ ResName
--	--	--	---

			<ul style="list-style-type: none"> ➤ “Update” button next to each individual student as per MVC layout ➤ “Details” button next to each individual student as per MVC layout ➤ “Delete” button next to each individual student as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The students in the student container has been retrieved from the Student table using Entity Framework and LINQ Queries.</p>
	<p>Step 5: The volunteer clicks the “Generate Graduate List” hyperlink.</p>		<p>Step 6: The system, using C#, JavaScript, CSS and HTML, displays the Graduate List Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item

			<ul style="list-style-type: none"> ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Graduate List” h1 header ➤ “You can generate a list of graduates in the academy that will qualify based on the requirements you have set” h3 header ➤ Student Container with the following label headings above the container: <ul style="list-style-type: none"> ➤ Name ➤ Student Number ➤ Email Address ➤ Phone Number <p>Footer</p> <ul style="list-style-type: none"> ➤ “Email List” button ➤ “Return” button <p>The students in the student graduate container is populated from the <u>Graduate</u> and <u>Student</u> tables where StudentID connects the two tables using Entity Framework and LINQ queries.</p>
	<p>Step 7: The volunteer clicks the “Email List” button.</p>		<p>Step 8: The system, using C# and JavaScript, displays a Modal with</p>

			the following message: “Please enter an email address:” with an “Email” textbox, “Email” button and “Cancel” button.
	Step 9: The volunteer enters the email address of the intended person and clicks the “Email” button.		Step 10: The system, using C# and JavaScript, validates that the email address entered is in the correct format as follows: <ul style="list-style-type: none"> • Email Address (Minimum 7 characters, Maximum 255 characters. Must be a full validated email address.)
			Step 11: The system, using C#, emails the respected individual the graduate list.
			Step 12: The system, using C# and JavaScript, displays a modal with the following message: “Graduate list has been successfully emailed!” and a “Confirm” button.
	Step 13: The volunteer clicks on the “Confirm” button.		Step 14: The system, using JavaScript, CSS and HTML, displays the TRWLA Students webpage with the following layout: Header Navigation bar from left to right: <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header

		<ul style="list-style-type: none">➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Generate Graduate List” button➤ “Students” h1 header➤ “Find by degree” textbox➤ “Search” button.➤ A container with a list of students with the following headings above the container:➤ Student Number➤ FirstName➤ Graduate➤ Degree➤ Year of Study➤ ResName➤ “Update” button next to each individual student as per MVC layout➤ “Details” button next to each individual student as per MVC layout
--	--	--

			<ul style="list-style-type: none"> ➤ “Delete” button next to each individual student as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The students in the student container has been retrieved from the Student table using Entity Framework and LINQ Queries.</p>
ALTERNATE COURSES:	<p>Alt Step 7: The volunteer clicks the “Return” button. The system, using JavaScript, CSS and HTML, displays the TRWLA Students webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Generate Graduate List” button ➤ “Students” h1 header ➤ “Find by degree” textbox ➤ “Search” button. ➤ A container with a list of students with the following headings above the container: <ul style="list-style-type: none"> ➤ Student Number ➤ FirstName ➤ Graduate ➤ Degree ➤ Year of Study ➤ ResName ➤ “Update” button next to each individual student as per MVC layout ➤ “Details” button next to each individual student as per MVC layout ➤ “Delete” button next to each individual student as per MVC layout <p>Footer</p>		

	<ul style="list-style-type: none"> ➤ “Return” button <p>The students in the student container has been retrieved from the <u>Student</u> table using Entity Framework and LINQ Queries.</p> <p>→ Go to step 14</p> <p>On webpage: TRWLA Students</p>
	<p>Alt Step 9: The volunteer clicks the “Cancel” button. The system displays the Generate Graduate List Webpage.</p> <p>→ Go to step 6</p>
	<p>Alt Step 10: The email address is in the incorrect format according to the system’s standards. The system, using C# and JavaScript, displays viewbag error messages on the modal that indicate what the required format standard is.</p> <p>→ Go to step 9</p>
CONCLUSION:	<p>The system, using JavaScript, CSS and HTML, displays the TRWLA Students webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Generate Graduate List” button ➤ “Students” h1 header ➤ “Find by degree” textbox ➤ “Search” button. ➤ A container with a list of students with the following headings above the container: <ul style="list-style-type: none"> ➤ Student Number ➤ FirstName ➤ Graduate ➤ Degree ➤ Year of Study ➤ ResName ➤ “Update” button next to each individual student as per MVC layout ➤ “Details” button next to each individual student as per MVC layout ➤ “Delete” button next to each individual student as per MVC layout <p>Footer</p>

	<p>➤ “Return” button</p> <p>The students in the student container has been retrieved from the Student table using Entity Framework and LINQ Queries.</p>
POST-CONDITION:	The volunteer has generated a graduate list for the current year.
BUSINESS RULES	<ul style="list-style-type: none"> • Only a volunteer or admin member can generate a graduate list. • Only students that have a progress of 100%, i.e. they have completed all of the milestones expected of them, can graduate from the academy.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • Ensure that all students that have 100% progress based on logged attendance can graduate from the academy. • Students with progress less than 100% should not be able to graduate.
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Add Student Type	USE CASE TYPE	
USE CASE ID:	3.6	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy.		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of Admin wanting to add a student type. Admin clicks the members menu item, which displays a dropdown list with a “Students” item. Admin clicks the “Students” item and the system displays the Student webpage with two tabs, “Students” and “Student Type”. Admin clicks the “Student Type” tab then clicks create student type. Admin enters the required information and the system validates and stores the information. This use case concludes by a student type being added to the system.</p>		
PRE-CONDITION:	Admin is logged into the System.		
TRIGGER:	Admin wants to add a student type.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	Step 1: Admin clicks the “Students” dropdown list item under the “Members” navigation bar menu item. Members > Students		Step 2: The system displays Screen: 3.2.1 Search Student. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively: <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members

			<ul style="list-style-type: none">➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Students" and "Student Type", the "Student" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the students' details populated by the Student table. The list contains the following information about the students as retrieved from the Person table:• Name• Surname• Email Address• Phone Number• DateOfBirth
--	--	--	--

			<p>The following information from the <u>Student</u> table:</p> <ul style="list-style-type: none"> • StudentTypeID • ResID <p>Above the container is a textbox labelled “Find by Name:”. To the right of the textbox is a button labelled “Search”. Below the container are two buttons entitled “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the “Student Type” tab. Screen: 3.2.1 Search Student</p>		<p>Step 4: The system displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications.

			<p>➤ A profile picture which contains a link to the User dropdown list items.</p> <p>The body contains two tabs, “Students” and “Student Type”, the “Student Type” tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Student Type”.➤ A container below the header containing a list of all the student types details populated by the Student Type table. The list contains the following information about the student type as retrieved from the Student Type table:<ul style="list-style-type: none">● DescriptionAbove the container is a textbox labelled “Find by Name”. To the right of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create a new student type”,
--	--	--	--

			<p>“Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin clicks the add button. Screen: 3.7 Search Student Type</p>		<p>Step 6: The system displays Screen: 3.6.2 Add Student Type. This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, “Students” and “Student Type”, the “Student Type” tab is displayed with the following elements:</p>

			<ul style="list-style-type: none"> ➤ A header entitled "Student Type". ➤ A container below the header containing a "Student Type Description" (textbox) and "Create" button ➤ Below the container are two buttons entitled "Cancel" and "Return" <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks</p>
	<p>Step 7: Admin enters the required information into the textbox and clicks the "Create" button.</p> <p>Screen: 3.6.2 Add Student Type</p>		<p>Step 8: The system validates that the "Student Type Description" textbox contains no more than 25 characters.</p>
			<p>Step 9: The system adds the following information to the <u>Student Type</u> table:</p> <ul style="list-style-type: none"> • Description • StudentTypeID (automatically generated per addition)
			<p>Step 10: The system displays a modal with the following message, "You have successfully added a new student type" with a button</p>

			below the message entitled "Confirm".
	<p>Step 11: Admin clicks the "Confirm" button.</p>		<p>Step 12: The system displays Screen: 3.6.2 Add Student Type. This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Students" and "Student Type", the "Student Type" tab is displayed with the following elements:</p>

			<ul style="list-style-type: none"> ➤ A header entitled "Student Type". ➤ A container below the header containing a "Student Type Description" (textbox) and "Create" button ➤ Below the container are two buttons entitled "Cancel" and "Return" <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks</p>
ALTERNATE COURSES:	Alt Step 7: Admin chooses not to create a student type and clicks the "Return" button. → Go to Step 4		
	Alt Step 8: The "Student Type Description" textbox contains more than 25 characters. The system displays a viewbag error message below the textbox indicating the maximum characters allowed is 25. → Go to Step 6		
CONCLUSION:	Admin has created a student type.		
POST-CONDITION:	A student type is added to the <u>Student Type</u> table.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can add a student type. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Search Student Type	USE CASE TYPE				
USE CASE ID:	3.7	Business Requirements: <input type="checkbox"/>				
PRIORITY:	High	System Analysis: <input type="checkbox"/>				
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>				
PRIMARY BUSINESS ACTOR	Admin					
PRIMARY SYSTEM ACTOR	None					
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 					
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 					
DESCRIPTION:	<p>This use case describes the event of admin wanting to search for a specific student type. Admin navigates to the Student webpage and clicks on the Student Type tab. The system displays a container containing a list of student types, admin uses the search bar functionality to search for a specific student type. The system displays the search results. The use case concludes by admin viewing the specific student type information.</p>					
PRE-CONDITION:	Admin is logged into the System.					
TRIGGER:	Admin wants to search for a student type.					
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: Admin clicks the "Students" dropdown list item under the "Members" navigation bar menu item. Members > Students	System Response <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Manual Action</th> <th>Automated Action</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> </tr> </tbody> </table>	Manual Action	Automated Action		
Manual Action	Automated Action					

			<ul style="list-style-type: none">➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Students" and "Student Type", the "Student" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the students' details populated by the Student table. The list contains the following information about the students as retrieved from the Person table:• Name• Surname• Email Address• Phone Number• DateOfBirth
--	--	--	---

			<p>The following information from the <u>Student</u> table:</p> <ul style="list-style-type: none"> • StudentTypeID • ResID <p>Above the container is a textbox labelled “Find by Name:”. To the right of the textbox is a button labelled “Search”. Below the container are two buttons entitled “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the “Student Type” tab. Screen: 3.2.1 Search Student</p>		<p>Step 4: The system displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications.

			<p>➤ A profile picture which contains a link to the User dropdown list items.</p> <p>The body contains two tabs, "Students" and "Student Type", the "Student Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Student Type".➤ A container below the header containing a list of all the student types details populated by the <u>Student Type</u> table. The list contains the following information about the student type as retrieved from the <u>Student Type</u> table:<ul style="list-style-type: none">● DescriptionAbove the container is a textbox labelled "Find by Name". To the right of the textbox is a button labelled "Search". Below the container are four buttons entitled "Create a new student type",
--	--	--	--

			<p>“Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the textbox above the container and clicks the “search” button. Screen: 3.7 Search Student Type</p>		<p>Step 4: The system performs a search and displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, “Students” and “Student Type”, the “Student Type” tab is</p>

			<p>displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Student Type”. ➤ A container below the header containing a list of all the student types details populated by the <u>Student Type</u> table. The list contains the following information about the student type as retrieved from the <u>Student Type</u> table: ● Description <p>Above the container is a textbox labelled “Find by Name”. To the right of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create a new student type”, “Update”, “Delete” and “Return”.</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 7: Admin views the search results and clicks the “Return” button.</p>		<p>Step 8: The system displays the Main Menu webpage. This webpage</p>

	Screen: 3.7 Search Student Type	<p>contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items following elements from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of upcoming events populated by the Event table using entity framework and LINQ queries.➤ The list contains the following information about events as
--	---------------------------------	--

			<p>retrieved from the <u>Event</u> table:</p> <ul style="list-style-type: none"> • Name • Date • Time • Venue • Summary <p>Each list item contains two buttons to the right of the item entitled “RSVP” and “View More Details”. Above the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. To the right of the body is another container with a list of upcoming events that the user has rsvp’d to, which is determined by the RSVP_Event table, and is populated by the <u>Event</u> table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the <u>Event</u> table:</p> <ul style="list-style-type: none"> • Name • Date • Time • Venue <p>Using CSS, JavaScript, C# and HTML.</p>
ALTERNATE COURSES:			<p>Alt Step 5: Admin scrolls through the container containing a list of student types and finds the student type information that they are searching for and clicks the “Return” button.</p> <p>On webpage: Student Type</p> <p>→ Go to Step 8</p>

	Alt Step 6: The system has no matches. The following message will be displayed in the container, "Your search has no results". On webpage: Search Student Type
CONCLUSION:	The system displays the search results to admin.
POST-CONDITION:	Admin has searched for a student type.
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can search a student type.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Update Student Type		USE CASE TYPE
USE CASE ID:	3.8	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Medium	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of admin updating a student type. The use case begins by admin navigating to the Students webpage and clicking on the Student Type tab. The system displays a container with a list of all the student types, which admin will search and click the “Update” button for the relevant list item. The system will display the list item in an editable format, which admin can update and save. This use case concludes by a student type being updated.</p>		
PRE-CONDITION:	Admin is logged into the System.		
TRIGGER:	Admin wants to update a student type.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: Admin clicks the “Students” dropdown list item under the “Members” navigation bar menu item. Members > Students On webpage: Main Menu Admin</p>		<p>Step 2: The system displays Screen: 3.2.1 Search Student. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events

			<ul style="list-style-type: none">➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Students" and "Student Type", the "Student" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the students' details populated by the Student table. The list contains the following information about the students as retrieved from the Person table:• Name• Surname• Email Address• Phone Number• DateOfBirth
--	--	--	---

			<p>The following information from the <u>Student</u> table:</p> <ul style="list-style-type: none"> • StudentTypeID • ResID <p>Above the container is a textbox labelled “Find by Name:”. To the right of the textbox is a button labelled “Search”. Below the container are two buttons entitled “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the “Student Type” tab. Screen: 3.2.1 Search Student</p>		<p>Step 4: The system displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications.

			<p>➤ A profile picture which contains a link to the User dropdown list items.</p> <p>The body contains two tabs, "Students" and "Student Type", the "Student Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Student Type".➤ A container below the header containing a list of all the student types details populated by the <u>Student Type</u> table. The list contains the following information about the student type as retrieved from the <u>Student Type</u> table:<ul style="list-style-type: none">● DescriptionAbove the container is a textbox labelled "Find by Name". To the right of the textbox is a button labelled "Search". Below the container are four buttons entitled "Create a new student type",
--	--	--	--

			<p>“Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the textbox and clicks the “search” button. Screen: 3.7 Search Student Type</p>		<p>Step 6: The system performs a search and displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, “Students” and “Student Type”, the “Student Type” tab is</p>

			<p>displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Student Type”. ➤ A container below the header containing a list of all the student types details populated by the <u>Student Type</u> table. The list contains the following information about the student type as retrieved from the <u>Student Type</u> table: ● Description <p>Above the container is a textbox labelled “Find by Name”. To the right of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create a new student type”, “Update”, “Delete” and “Return”.</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 7: Admin clicks the “Update” button for the specific list item in the container.</p>		<p>Step 8: The system displays Screen: 3.8.1 Update Student Type.</p>

	Screen: 3.7 Search Student Type	<p>This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Students" and "Student Type", the "Student Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Update a Student Type".➤ A container below the header containing a "Student Type"
--	---------------------------------	--

			<p>Description” (textbox) and “Save” button</p> <ul style="list-style-type: none"> ➤ Below the container are two buttons entitled “Cancel” and “Return” <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks</p>
	<p>Step 9: Admin enters the information and clicks the “Update” button</p> <p>Screen: 3.8.1 Update Student Type</p>		<p>Step 10: The system validates the information updated by admin is no more than 25 characters.</p>
			<p>Step 11: The system updates the following attributes in the <u>Student Type</u> table:</p> <ul style="list-style-type: none"> • Description per the StudentTypeID
			<p>Step 12: The system displays a modal with the following message, “You have successfully updated the student type” with a button below the message entitled “Confirm”.</p>
	<p>Step 13: Admin clicks the “Confirm” button.</p>		<p>Step 14: The system displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p>

			<ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Students" and "Student Type", the "Student Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Student Type".➤ A container below the header containing a list of all the student types details populated by the <u>Student Type</u> table. The list contains the following information
--	--	--	---

			<p>about the student type as retrieved from the <u>Student Type</u> table:</p> <ul style="list-style-type: none"> • Description <p>Above the container is a textbox labelled “Find by Name”. To the right of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create a new student type”, “Update”, “Delete” and “Return”.</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
ALTERNATE COURSES:	Alt Step 5a: Admin chooses not to update a student type and clicks the “Return” button. → Go to Step 2		
	Alt Step 5b: Admin scrolls through the container containing a list of student types and finds the student type information that they are looking for and clicks the “Update” button. On webpage: Student Type → Go to Step 8		
	Alt Step 10: The “Student Type Description” textbox contains more than 25 characters. The system displays a ViewBag error message below the textbox indicating the maximum characters allowed is 25. → Go to Step 8		
CONCLUSION:	Admin has updated a student type.		
POST-CONDITION:	The system updates the <u>Student Type</u> table		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can update a student type. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Delete Student Type	USE CASE TYPE	
USE CASE ID:	3.9	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Low	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of admin deleting a student type. The use case begins by admin navigating to the Students webpage and clicking on the Student Type tab. The system displays a container with a list of all the student types, which admin will search and click the “Delete” button for the relevant list item. The system confirms if admin wants to delete that student type. This use case concludes by a student type being deleted.</p>		
PRE-CONDITION:	Admin is logged into the System.		
TRIGGER:	Admin wants to delete a student type.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: Admin clicks the “Students” dropdown list item under the “Members” navigation bar menu item. Members > Students</p>		<p>Step 2: The system displays Screen: 3.2.1 Search Student. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events

			<ul style="list-style-type: none">➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Students" and "Student Type", the "Student" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the students' details populated by the Student table. The list contains the following information about the students as retrieved from the Person table:• Name• Surname• Email Address• Phone Number• DateOfBirth
--	--	--	---

			<p>The following information from the <u>Student</u> table:</p> <ul style="list-style-type: none"> • StudentTypeID • ResID <p>Above the container is a textbox labelled “Find by Name:”. To the right of the textbox is a button labelled “Search”. Below the container are two buttons entitled “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the “Student Type” tab. Screen: 3.2.1 Search Student</p>		<p>Step 4: The system displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications.

			<p>➤ A profile picture which contains a link to the User dropdown list items.</p> <p>The body contains two tabs, "Students" and "Student Type", the "Student Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Student Type".➤ A container below the header containing a list of all the student types details populated by the <u>Student Type</u> table. The list contains the following information about the student type as retrieved from the <u>Student Type</u> table:<ul style="list-style-type: none">● DescriptionAbove the container is a textbox labelled "Find by Name". To the right of the textbox is a button labelled "Search". Below the container are four buttons entitled "Create a new student type",
--	--	--	--

			<p>“Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the textbox and clicks the “search” button. Screen: 3.7 Search Student Type</p>		<p>Step 6: The system performs a search and displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, “Students” and “Student Type”, the “Student Type” tab is displayed with the following elements:</p>

			<ul style="list-style-type: none"> ➤ A header entitled “Student Type”. ➤ A container below the header containing a list of all the student types details populated by the <u>Student Type</u> table. The list contains the following information about the student type as retrieved from the <u>Student Type</u> table: <ul style="list-style-type: none"> ● Description <p>Above the container is a textbox labelled “Find by Name”. To the right of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Create a new student type”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 7: Admin clicks the “Delete” button on the specific list item in the container.</p> <p>On webpage: Search Student Type</p>		<p>Step 8: The system displays a modal with the following message, “Are you sure?” “Once the student type is deleted it is unrecoverable” with a</p>

			button below the message entitled "Confirm".
	<p>Step 9: Admin clicks the "Delete" button on the modal.</p> <p>On webpage: Search Student Type</p>		<p>Step 10: The system deleted the following attributes in the Student Type table:</p> <ul style="list-style-type: none"> • Description • StudentTypeID <p>Per the StudentTypeID</p>
			<p>Step 11: The system displays a modal with the following message, "You have successfully deleted the following student type: (Description from Student Type table)" with a button below the message entitled "Confirm".</p>
	<p>Step 12: Admin clicks the "Confirm" button.</p>		<p>Step 13: The system displays Screen: 3.7 Search Student Type. This webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications,

			<p>which contains the link to the user's notifications.</p> <p>➤ A profile picture which contains a link to the User dropdown list items.</p> <p>The body contains two tabs, "Students" and "Student Type", the "Student Type" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Student Type".➤ A container below the header containing a list of all the student types details populated by the <u>Student Type</u> table. The list contains the following information about the student type as retrieved from the <u>Student Type</u> table:<ul style="list-style-type: none">• DescriptionAbove the container is a textbox labelled "Find by Name". To the right of the textbox is a button labelled "Search". Below
--	--	--	---

			the container are four buttons entitled “Create a new student type”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
ALTERNATE COURSES:	Alt Step 9: Admin clicks the “Cancel” button. → Go to Step 6		
CONCLUSION:	Admin deletes a student type.		
POST-CONDITION:	The Student Type table has been updated.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can delete a student type. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

2.3.4 Venues Narratives

USE CASE NAME:	Add Venue		USE CASE TYPE
USE CASE ID:	4.1		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:			
PRE-CONDITION:	The volunteer must be logged into the system.		
TRIGGER:	The volunteer wants to add a venue.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to add a new venue and clicks the “Events” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as:</p> <ul style="list-style-type: none"> ➢ Create Event ➢ Manage Events ➢ Manage Venues ➢ Log Attendance ➢ Manage Guest Speakers <p>On webpage: Menu-Events</p>
	<p>Step 3: The volunteer clicks the “Manage Venues” list item from the drop down list.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Venues webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p>

		<ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members Drop Down List ➤ Events Drop Down List ➤ Reading Content Drop Down List ➤ Message icon which displays notifications ➤ Profile picturebox ➤ User drop down list with “My Profile” and “Logout” list items. <p>Body</p> <ul style="list-style-type: none"> ➤ Venues <h1> header ➤ “Find by name:” textbox ➤ “Search” button ➤ “Venues” table with a list of current venues with the following headers above the table: <ul style="list-style-type: none"> • Name • Capacity • Accessibility • Street Number • Suburb • Province • Venue Type <p>Four buttons beneath the venues table to the right of the body labelled “Add a new venue”, “Update”, “Delete”, “Return”</p> <p>The venues in the venue table have been retrieved from the Venue table using LINQ Queries.</p>
--	--	--

			On Webpage 4.1.1 Add Venue
	<p>Step 5: The volunteer clicks the “Add a new venue” button.</p>		<p>Step 6: Using C#, HTML, CSS and JavaScript the system displays the Create Venue Webpage with the following:</p> <p>Header</p> <p>Navigation Bar Left to Right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members Drop Down List ➤ Events Drop Down List ➤ Reading Content Drop Down List ➤ Message Icon ➤ Profile picture box ➤ User Drop Down List <p>Body</p> <ul style="list-style-type: none"> ➤ H1 Header labelled “Add a new Venue” ➤ H3 Header labelled “Name” ➤ Name textbox ➤ H3 Header labelled “Capacity” ➤ Capacity textbox ➤ H3 Header labelled “Accessibility” ➤ Accessibility textbox ➤ H3 Header labelled “Street Number” ➤ Street Number Textbox ➤ H2 Header labelled “Street Name” ➤ Street Name Textbox ➤ H2 Header labelled “Suburb”

			<ul style="list-style-type: none"> ➤ Suburb Textbox ➤ H2 Header labelled “Province” ➤ Province dropdown list ➤ H2 Header “Venue Type” ➤ “Venue Type” dropdown list populated from <u>VenueType</u> table. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Return” button ➤ “Cancel”
	<p>Step 7: The volunteer types in all the required information and clicks the “Create” button.</p>		<p>Step 8: Using C# the system verifies whether the information entered is valid and complete, such that:</p> <ul style="list-style-type: none"> • Name (Maximum 35 characters) • Capacity (Maximum 10 numeric characters) • Accessibility (Maximum 20 characters) • Street Number (Max 10 numeric characters) • Street Name (Maximum 35 characters) • Suburb (Maximum 35 characters) • Province (Selected from the available options in the

			dropdown list) <ul style="list-style-type: none"> • VenueType (Selected from the available options in the dropdown list)
			Step 9: Using C# and SQL the system inserts the information entered by the user into the <u>Venue</u> and <u>Address</u> tables in the database.
			Step 10: Using JavaScript the system displays a Modal with the message “Congratulations! You have successfully added a new venue!”
ALTERNATE COURSES:	Alt Step 9: The information entered by the user is invalid, using JavaScript the system displays error messages for the incorrect information. → Go to Step 7		
CONCLUSION:	The volunteer has entered all the required information for the new venue, the system has verified that the information is complete and valid and has inserted the data into the database.		
POST-CONDITION:	A new venue has been created.		
BUSINESS RULES	<ul style="list-style-type: none"> • None 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • The user has a current password which they have forgotten. 		
OPEN ISSUES:	None		

USE CASE NAME:	Search Venue	USE CASE TYPE
USE CASE ID:	4.2	Business Requirements: <input type="checkbox"/>
PRIORITY:	High	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer (PBA)	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 	
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 	
DESCRIPTION:		
PRE-CONDITION:	The volunteer must be logged into the system.	
TRIGGER:	The volunteer wants to search a venue.	
TYPICAL COURSE OF EVENTS:	<p>Actor Action</p> <p>System Response</p>	
	<p>Step 1: The volunteer wants to search a venue and clicks the “Events” drop down list.</p>	<p>Manual Action</p> <p>Automated Action</p> <p>Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as:</p> <ul style="list-style-type: none"> ➤ Create Event ➤ Manage Events ➤ Manage Venues ➤ Log Attendance ➤ Manage Guest Speakers <p>On Webpage: Menu-Events</p>
	<p>Step 3: The volunteer clicks the “Manage Venues” list item from the drop down list.</p>	<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Venues webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p>

		<ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members Drop Down List ➤ Events Drop Down List ➤ Reading Content Drop Down List ➤ Message icon which displays notifications ➤ Profile picturebox ➤ User drop down list with “My Profile” and “Logout” list items. <p>Body</p> <ul style="list-style-type: none"> ➤ Venues <h1> header ➤ “Find by name:” textbox ➤ “Search” button ➤ “Venues” table with a list of current venues with the following headers above the table: <ul style="list-style-type: none"> ● Name ● Capacity ● Accessibility ● Street Number ● Suburb ● Province ● Venue Type <p>Four buttons beneath the venues table to the right of the body labelled “Add a new venue”, “Update”, “Delete”, “Return”</p> <p>The venues in the venue table have been retrieved from the Venue table using LINQ Queries.</p>
--	--	--

			On Webpage 4.2.1 Search Venue
	<p>Step 5: The volunteer types the details of the venue they are looking for into the “Find by name” textbox.</p> <p>On Webpage 4.2.1 Search Venue</p>		<p>Step 6: The system searches through the Venue and Address tables LINQ queries to find venues that match the search criteria.</p>
			<p>Step 7: Using JavaScript, the system replaces the venues in the venue table with only venues that match the search criteria.</p> <p>On Webpage 4.2.2 Search Venue</p>
ALTERNATE COURSES:	<p>Alt Step 5: The volunteer doesn't enter any search criteria but rather scrolls through the available venues.</p>		
	<p>Alt Step 6: There are no venues in the database that match the search criteria. Using C# and JavaScript the system displays an error message and prompts the user to enter valid search criteria</p> <p>→ Go to Step 5</p>		
CONCLUSION:	The volunteer has found the venue they were looking for and has seen the details of the venue.		
POST-CONDITION:	A new venue has been searched.		
BUSINESS RULES	<ul style="list-style-type: none"> • None 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Update Venue	USE CASE TYPE	
USE CASE ID:	4.3	Business Requirements: <input type="checkbox"/>	
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a volunteer updating a venue. The use case begins when the volunteer wants to update a venue and clicks the events navbar item. The volunteer must search and select the venue they would like to update. Once the volunteer has changed the relevant information, the system must verify whether the information entered is complete and valid. If the information is valid the system will update the venue in the database. The use case concludes when the venue has been successfully updated.</p>		
PRE-CONDITION:	The volunteer must be logged into the system.		
TRIGGER:	The volunteer wants to update a venue.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to update a venue and clicks the events navigation bar item.</p>		<p>Step 2: The system prompts the volunteer to search for the venue (Invoke Use Case 4.2 Search Venue)</p>
	<p>Step 3: The volunteer changes the relevant fields and clicks the update button. See Webpage 4.3.1 Update Venue</p>		<p>Step 4: Using C# the system verifies if all the information changed by the volunteer is complete and valid such that:</p> <ul style="list-style-type: none"> • Name (Maximum 35 characters) • Street Number

			<ul style="list-style-type: none"> (Max 10 numeric characters) • Street Name (Maximum 35 characters) • Suburb (Maximum 35 characters) • Province (Selected from the available options in the dropdown list) • VenueType (Selected from the available options in the dropdown list)
			<p>Step 5: Using JavaScript the system displays a modal with the message “Are you sure you want to update this venue?” with buttons underneath the message entitled “Confirm”, “Edit” and “Return”.</p>
	<p>Step 6: The volunteer clicks the “Confirm” button.</p>		<p>Step 7: Using C# and SQL the system updates the appropriate attributes in the <u>Venue</u> and <u>Address</u> tables.</p>
			<p>Step 8: Using JavaScript the system displays a modal with the message “Congratulations! You successfully updated this venue” with a button entitled “Confirm”</p>

ALTERNATE COURSES:

Alt Step 2: The volunteer updates the venue directly from the Venues webpage with the following layout:

Header

Navigation bar from left to right:

- TRWLA Management System
- Members Drop Down List
- Events Drop Down List
- Reading Content Drop Down List
- Gallery navigation bar item
- Message icon which displays notifications
- Profile picturebox
- User drop down list with “My Profile” and “Logout” list items.

Body

- Venue tab
- Types tab
- “Add a new venue” hyperlink
- “Find by name” textbox
- “Search” button
- “Venues” table with a list of current venues with the following headers above the table:
 - Name
 - Capacity
 - Accessibility
 - Street Number
 - Description
- There are three buttons in the table for each venue item, namely:
 - Update
 - Details
 - Delete

The venues in the venue table has been retrieved from the **Venue** table using Entity Framework and LINQ Queries.

The volunteer clicks the venue in the venues table and clicks the “Update” button.

→ Go to Step 3

Alt Step 5: The information the volunteer has entered is not valid, using JavaScript the system displays error messages below the relevant fields.
 → Go to Step 3

CONCLUSION:

The volunteer has changed the relevant details about the venue and the system has verified that the information is complete and valid and has updated the venue in the database.

POST-CONDITION:

A venue has been updated.

BUSINESS RULES	<ul style="list-style-type: none">• None
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none">• None
ASSUMPTIONS:	<ul style="list-style-type: none">• None
OPEN ISSUES:	None

USE CASE NAME:	Delete Venue		USE CASE TYPE
USE CASE ID:	4.4		Business Requirements: <input type="checkbox"/>
PRIORITY:	Low		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a volunteer deleting a venue. The use case begins when the volunteer wants to delete a venue and clicks the events navbar item. The volunteer must search and select the venue they would like to delete. The system will check whether the volunteer is sure they would like to delete the venue. Once the volunteer has confirmed deletion the system will remove the venue from the database. The use case concludes when the venue has been successfully deleted.</p>		
PRE-CONDITION:	The volunteer must be logged into the system.		
TRIGGER:	The volunteer wants to delete a venue.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to delete a venue and clicks the events navigation bar item. On Webpage Menu- Events</p>		<p>Step 2: The system prompts the volunteer to search for the venue → Invoke Use Case 4.2 Search Venue</p>
			<p>Step 4: Using JavaScript, CSS and HTML the system displays the Webpage 4.4.1 Delete Venue with the following layout: Header ➤ TRWLA Management System h1 header</p>

		<ul style="list-style-type: none"> ➤ Members drop down list ➤ Events drop down list ➤ Reading Content drop down list ➤ Gallery navigation bar item ➤ Message icon that displays notifications ➤ Profile picturebox ➤ User drop down list <p>Body</p> <ul style="list-style-type: none"> ➤ “Delete” h1 header ➤ “Are you sure you want to delete this?” h2 header ➤ “Venue” h3 header ➤ “Name” ➤ “Capacity” ➤ “Accessibility” ➤ “Street number” ➤ “Description” ➤ “Delete” button <p>Information displayed in the above webpage is populated from the Venue table, using LINQ Queries.</p> <p>On Webpage 4.4.1 Delete Venue</p>
	<p>Step 5: The volunteer clicks the “Delete” button.</p>	<p>Step 6: Using JavaScript the system displays a Modal with the following message “Are you sure? Once this venue is deleted it will be deleted forever)</p> <p>On webpage 4.4.2 Delete Venue</p>
		<p>Step 7: Using C# and SQL the system deletes the appropriate attributes in</p>

			the <u>Venue</u> and <u>Address</u> tables.
			Step 8: Using JavaScript the system displays a modal with the message “The venue has been deleted” with a button entitled “Confirm”
ALTERNATE COURSES:	<p>Alt Step 2: The volunteer deletes the venue directly from the Venues webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management system ➤ Members Drop Down List ➤ Events Drop Down List ➤ Reading Content Drop Down List ➤ Gallery navigation bar item ➤ Message icon which displays notifications ➤ Profile picturebox ➤ User drop down list with “My Profile” and “Logout” list items. <p>Body</p> <ul style="list-style-type: none"> ➤ Venue tab ➤ Types tab ➤ “Add a new venue” hyperlink ➤ “Find by name” textbox ➤ “Search” button ➤ “Venues” table with a list of current venues that contain the following headers above the table: ➤ Name ➤ Capacity ➤ Accessibility ➤ Street Number ➤ Description ➤ There is three buttons in the table for each venue in the list namely: ➤ Update ➤ Details ➤ Delete <p>The venues in the venue table has been retrieved from the <u>Venue</u> table using Entity Framework and LINQ Queries.</p> <p>The volunteer clicks the venue in the venues table and clicks the “Delete” button.</p> <p>→ Go to Step 4</p>		

CONCLUSION:	The volunteer has deleted the relevant venue from the system.
POST-CONDITION:	A venue has been deleted.
BUSINESS RULES	<ul style="list-style-type: none">• None
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none">• None
ASSUMPTIONS:	<ul style="list-style-type: none">• None
OPEN ISSUES:	None

USE CASE NAME:	Add Venue Type	USE CASE TYPE	
USE CASE ID:	4.5	Business Requirements: <input type="checkbox"/>	
PRIORITY:	Low	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	• None		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a volunteer adding a new venue type to the system to host for future events. The use case begins when the volunteer has logged onto the system via the static webpage and clicks on the “Events” drop down list on the navigation bar. The volunteer will then click on the “Manage Venues” link item and the Venues webpage will be displayed. The volunteer will click on the “Types” tab on the Venue webpage. The system will then display a webpage with a list of the current venue types that TRWLA has. The volunteer will then proceed to add the new venue type by clicking the “Add a new venue type” button and entering all of the relevant information needed pertaining to the event type. The use case concludes when the volunteer has successfully added a new venue type to the system for future events.</p>		
PRE-CONDITION:	The volunteer has logged into the system.		
TRIGGER:	The volunteer wants to add a new venue type to the system.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to add a new venue type and clicks the “Events” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as:</p> <ul style="list-style-type: none"> ➤ Create ➤ Manage Events ➤ Manage Venues ➤ Log Attendance ➤ Manage Guest Speakers

	<p>Step 3: The volunteer clicks the “Manage Venues” list item from the drop down list.</p>	<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Venues webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ Venue tab➤ Types tab➤ Residences tab➤ “Venues” h1 header➤ “Find by name” textbox➤ “Search” button➤ “Venues” container with a list of current venues under the following headers:<ul style="list-style-type: none">● Name● Capacity● Accessibility● StreetNumber
--	---	---

			<ul style="list-style-type: none"> • Description/Venue Type <p>➤ “Update” button as per MVC layout</p> <p>➤ “Details” button as per MVC layout</p> <p>➤ “Delete” button as per MVC layout</p> <p>Footer</p> <p>➤ “Return” button</p> <p>The venues in the venue container has been retrieved from the Venue and VenueType tables using Entity Framework and LINQ Queries.</p>
	<p>Step 5: The volunteer clicks the “Types” tab.</p>		<p>Step 6: The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox

			<ul style="list-style-type: none"> ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
	<p>Step 7: The volunteer clicks the “Add a New Venue Type” button.</p>		<p>Step 8: The system, using JavaScript, CSS and HTML, displays the Create Venue Type Webpage with the following layout:</p> <p>Header</p>

			<p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Add” h1 header ➤ “Venue Type” h2 header ➤ “Description” textbox ➤ “Create” button ➤ “Back to list” hyperlink button
	<p>Step 9: The volunteer enters the relevant information and clicks the “Create” button.</p>		<p>Step 10: The system, using C# and JavaScript, validates that the information entered is correct:</p> <ul style="list-style-type: none"> • Name (Max 50 characters) • Description (Max 300 characters)
			<p>Step 11: The system, using C# and JavaScript, displays a modal with the following message; “You</p>

			have successfully added a New Venue Type” and a “Confirm” button.
			<p>Step 12: The system using C#, adds the new venue type to the VenueType table using Entity Framework and LINQ queries.</p>
	<p>Step 13: The volunteer clicks the “Confirm” button.</p>		<p>Step 14: The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header

			<ul style="list-style-type: none"> ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
ALTERNATE COURSES:	<p>Alt Step 7a: The volunteer selects the specific venue type and clicks the “Edit” button. The system, using JavaScript, CSS and HTML, displays the Update Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Update Venue Type” h1 header ➤ “Venue Type” h2 header 		

	<ul style="list-style-type: none"> ➤ “Description” textbox ➤ “Save” button ➤ “Back to list” hyperlink button <p>→ Invoke 4.7 Update Venue Type On webpage: Update Venue Type</p>
	<p>Alt-Step 7b: The volunteer selects a venue type and clicks the “Delete” button. The system, using C# and JavaScript, displays a modal with the following message: “Are you sure you want to delete this venue type?” and two buttons namely, “Confirm” and “Cancel”.</p> <p>→ Invoke 4.8 Delete Venue Type On webpage: Delete Venue Type</p>
	<p>Alt- Step 10: The information entered is incorrect and does not meet the system standards. The system, using C# and JavaScript, displays a Modal with the following message: “The required fields have not been completed or the information entered is invalid.” with a “Try Again” button.</p> <p>→ Go back to step 9 On webpage: Create Venue Type</p>
CONCLUSION:	<p>The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout

	Footer ➤ “Return” button The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.
POST-CONDITION:	The volunteer has added a new venue type to the system.
BUSINESS RULES	<ul style="list-style-type: none"> • Only a volunteer or admin member can add a new venue type.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Search Venue Type	USE CASE TYPE	
USE CASE ID:	4.6	Business Requirements: <input type="checkbox"/>	
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a volunteer searching for a specific venue type. The use case begins when the volunteer has logged onto the system via the static webpage and clicks on the “Events” drop down list on the navigation bar. The volunteer will then click on the “Manage Venues” link item and the Venues webpage will be displayed. The volunteer will click on the “Types” tab on the Venue webpage. The system will then display a webpage with a list of the current venue types that TRWLA has. The volunteer will then proceed to search for the venue type by either entering search details into the search textbox or by scrolling through the venue type list. The use case concludes when the volunteer has successfully searched for a specific venue type and viewed its details.</p>		
PRE-CONDITION:	The volunteer has logged into the system.		
TRIGGER:	The volunteer wants to search for a venue type on the system.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to search for a venue type and clicks the “Events” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as:</p> <ul style="list-style-type: none"> ➢ Create ➢ Manage Events ➢ Manage Venues ➢ Log Attendance ➢ Manage Guest Speakers

	<p>Step 3: The volunteer clicks the “Manage Venues” list item from the drop down list.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Venues webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ Venue tab ➤ Types tab ➤ Residences tab ➤ “Venues” h1 header ➤ “Find by name” textbox ➤ “Search” button ➤ “Venues” container with a list of current venues under the following headers: <ul style="list-style-type: none"> • Name • Capacity • Accessibility • StreetNumber
--	---	--	---

			<ul style="list-style-type: none"> • Description/Venue Type <p>➤ “Update” button as per MVC layout</p> <p>➤ “Details” button as per MVC layout</p> <p>➤ “Delete” button as per MVC layout</p> <p>Footer</p> <p>➤ “Return” button</p> <p>The venues in the venue container has been retrieved from the Venue and VenueType tables using Entity Framework and LINQ Queries.</p>
	<p>Step 5: The volunteer clicks the “Types” tab.</p>		<p>Step 6: The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox

			<ul style="list-style-type: none"> ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
	<p>Step 7: The volunteer enters details in the “Find by name” textbox and clicks the “Search” button.</p>		<p>Step 8: The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the results from the search in the following layout:</p> <p>Header</p>

			<p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout
--	--	--	--

			<ul style="list-style-type: none"> ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
ALTERNATE COURSES:	<p>Alt Step 8: The system could not find results based on the volunteer’s search and displays a Modal with the following message: “No Venue Types match your search criteria.” with a “Try Again” button.</p> <p>→ Go back to Step 7</p> <p>On webpage: Venue Type Webpage</p>		
CONCLUSION:	<p>The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the results from the search in the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button 		

	The venue types in the venue type's container is populated from the VenueType using Entity Framework and LINQ queries.
POST-CONDITION:	The volunteer has successfully searched for a venue type.
BUSINESS RULES	<ul style="list-style-type: none"> • Only a volunteer or admin member can search for a venue type.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Update Venue Type	USE CASE TYPE
USE CASE ID:	4.7	Business Requirements: <input type="checkbox"/>
PRIORITY:	Low	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	• None	
OTHER INTERESTED STAKEHOLDERS:	None	
DESCRIPTION:	<p>This use case describes the event of a volunteer updating a specific venue type. The use case begins when the volunteer has logged onto the system via the static webpage and clicks on the “Events” drop down list on the navigation bar. The volunteer will then click on the “Manage Venues” link item and the Venues webpage will be displayed. The volunteer will click on the “Types” tab on the Venue webpage. The system will then display a webpage with a list of the current venue types that TRWLA has. The volunteer will then proceed to search for the venue type by either entering search details into the search textbox or by scrolling through the venue type list. Once the volunteer has clicked on a specific venue type, the venue type details webpage will be displayed. The volunteer will then update the details accordingly. The use case concludes when the volunteer has successfully updated the details of the venue type.</p>	
PRE-CONDITION:	The volunteer has logged into the system.	
TRIGGER:	The volunteer wants to update the details of a venue type.	
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: The volunteer wants to update the details of a venue type and clicks the “Events” drop down list.	System Response Manual Action Automated Action Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as: ➤ Create ➤ Manage Events ➤ Manage Venues ➤ Log Attendance

			<ul style="list-style-type: none"> ➤ Manage Guest Speakers
	<p>Step 3: The volunteer clicks the “Manage Venues” list item from the drop down list.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Venues webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ Venue tab ➤ Types tab ➤ Residences tab ➤ “Venues” h1 header ➤ “Find by name” textbox ➤ “Search” button ➤ “Venues” container with a list of current venues under the following headers: <ul style="list-style-type: none"> • Name • Capacity

			<ul style="list-style-type: none"> • Accessibility • StreetNumber • Description/Venue Type <p>➤ “Update” button as per MVC layout</p> <p>➤ “Details” button as per MVC layout</p> <p>➤ “Delete” button as per MVC layout</p> <p>Footer</p> <p>➤ “Return” button</p> <p>The venues in the venue container has been retrieved from the Venue and VenueType tables using Entity Framework and LINQ Queries.</p>
	<p>Step 5: The volunteer clicks the “Types” tab.</p>		<p>Step 6: The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <p>➤ TRWLA Management System h1 header</p> <p>➤ “Members” drop down list navigation bar item</p> <p>➤ “Events” drop down list navigation bar item</p> <p>➤ “Reading Content” drop down list navigation bar item</p> <p>➤ “Gallery” navigation bar item</p>

		<ul style="list-style-type: none"> ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
	<p>Step 7: The volunteer searches through the venue type list for a specific venue type and clicks on the “Update” button.</p>	<p>Step 8: The system, using JavaScript, CSS and HTML, displays the Update Venue Type</p>

			<p>Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Update Venues” h1 header ➤ “Venue Type” h2 header ➤ “Description” textbox ➤ “Save” button ➤ “Back to list” hyperlink button <p>The description is obtained from the VenueType table using Entity Framework and LINQ queries.</p>
	<p>Step 9: The volunteer updates the existing details and clicks on the “Save” button.</p>		<p>Step 10: The system, using C# and JavaScript, validates that the</p>

			<p>information updated is correct.</p> <ul style="list-style-type: none"> • Description (Max 300 characters)
			<p>Step 11: The system updates the <u>VenueType</u> table using Entity Framework and LINQ queries.</p>
			<p>Step 12: The system displays a modal with the following message: "You have successfully updated a New Venue Type." with a "Confirm" button.</p>
	<p>Step 13: The volunteer clicks the "Confirm" button.</p>		<p>Step 14: The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ "Members" drop down list navigation bar item ➤ "Events" drop down list navigation bar item ➤ "Reading Content" drop down list navigation bar item ➤ "Gallery" navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox

			<ul style="list-style-type: none"> ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
ALTERNATE COURSES:	Alt Step 7a: The volunteer enters details into the search textbox. The system, using C# and JavaScript, processes the details entered and displays an updated container with list items relevant to the search details entered, with the following layout: Header Navigation bar from left to right:		<ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item

	<ul style="list-style-type: none"> ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p> <p>Alt-Step 7b: Once the system displays the search results, the volunteer clicks on the relevant venue type. → Invoke UC 4.6 Search Venue Type On webpage: Venue Type Webpage</p>
	<p>Alt- Step 7d: The volunteer clicks on the “Delete” button. The system displays a modal with the following message: “Once this venue type is deleted it is unrecoverable” with a “Confirm” button. → Invoke UC 4.8 Delete Venue Type On webpage: Venue Type Details Webpage</p>
	<p>Alt-Step 10: The information entered is incorrect and does not meet the system standards. The system, using C# and JavaScript, displays viewbag error messages beneath the required textbox stating that the minimum requirements should be met as well as a Modal with the following message: “The required fields have not been completed or the information entered is invalid” with a “Try Again” button. → Go to step 9 On webpage: Update Venue Type Webpage</p>
	<p>Alt-Step 12: The volunteer clicks the “Cancel” button. The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage.</p>

	<p>→ Go to step 16</p> <p>On webpage: Venue Type Webpage</p>
CONCLUSION:	<p>The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the <u>VenueType</u> using Entity Framework and LINQ queries.</p>
POST-CONDITION:	The volunteer has successfully updated a venue type.
BUSINESS RULES	<ul style="list-style-type: none"> • Only a volunteer or admin member can update the details of a venue type.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Delete Venue Type		USE CASE TYPE
USE CASE ID:	4.8		Business Requirements: <input type="checkbox"/>
PRIORITY:	Low		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	<p>This use case describes the event of a volunteer deleting a specific venue type. The use case begins when the volunteer has logged onto the system via the static webpage and clicks on the “Events” drop down list on the navigation bar. The volunteer will then click on the “Manage Venues” link item and the Venues webpage will be displayed. The volunteer will click on the “Types” tab on the Venue webpage. The system will then display a webpage with a list of the current venue types that TRWLA has. The volunteer will then proceed to search for the venue type by either entering search details into the search textbox or by scrolling through the venue type list. Once the volunteer has selected a specific venue type, the volunteer will click on the “Delete” button. The use case concludes when the volunteer has successfully deleted a venue type.</p>		
PRE-CONDITION:	The volunteer has logged into the system.		
TRIGGER:	The volunteer wants to delete a venue type.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	Step 1: The volunteer wants to delete a venue type and clicks the “Events” drop down list.		Step 2: The system, using JavaScript, displays a list of available items on the drop down list such as: <ul style="list-style-type: none"> ➢ Create ➢ Manage Events ➢ Manage Venues ➢ Log Attendance

			<ul style="list-style-type: none"> ➤ Manage Guest Speakers
	<p>Step 3: The volunteer clicks the “Manage Venues” list item from the drop down list.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Venues webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ Venue tab ➤ Types tab ➤ Residences tab ➤ “Venues” h1 header ➤ “Find by name” textbox ➤ “Search” button ➤ “Venues” container with a list of current venues under the following headers: <ul style="list-style-type: none"> • Name • Capacity

			<ul style="list-style-type: none"> • Accessibility • StreetNumber • Description/Venue Type <p>➤ “Update” button as per MVC layout</p> <p>➤ “Details” button as per MVC layout</p> <p>➤ “Delete” button as per MVC layout</p> <p>Footer</p> <p>➤ “Return” button</p> <p>The venues in the venue container has been retrieved from the Venue and VenueType tables using Entity Framework and LINQ Queries.</p>
	<p>Step 5: The volunteer clicks the “Types” tab.</p>		<p>Step 6: The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <p>➤ TRWLA Management System h1 header</p> <p>➤ “Members” drop down list navigation bar item</p> <p>➤ “Events” drop down list navigation bar item</p> <p>➤ “Reading Content” drop down list navigation bar item</p> <p>➤ “Gallery” navigation bar item</p>

			<ul style="list-style-type: none"> ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
	<p>Step 7: The volunteer selects a venue type and clicks the “Delete” button.</p>		<p>Step 8: The system, using JavaScript, CSS and HTML, displays the Delete Venue Type</p>

			<p>Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Delete” h1 header ➤ “Are you sure you want to delete this?” h2 header ➤ “Venue Type” h2 header ➤ “Description” h4 header ➤ “Delete” button ➤ “Back to list” hyperlink button
	<p>Step 9: The volunteer clicks on the “Delete” button.</p>		<p>Step 10: The system, using JavaScript, displays a Modal with the following message: “Once this venue type is deleted it is</p>

			unrecoverable" with a "Confirm" button.
	Step 11: The volunteer clicks the "Confirm" button.		Step 12: The system, using C# and SQL, deletes the venue type from the VenueType table using Entity Framework and LINQ queries.
			<p>Step 11: The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ "Members" drop down list navigation bar item ➤ "Events" drop down list navigation bar item ➤ "Reading Content" drop down list navigation bar item ➤ "Gallery" navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ "User" drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ "Venues" tab ➤ "Types" tab ➤ "Residences" tab ➤ "Venue Type" h1 header

			<ul style="list-style-type: none"> ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
ALTERNATE COURSES:			<p>Alt Step 7a: The volunteer enters details into the search textbox. The system, using C# and JavaScript, processes the details entered and displays an updated container with list items relevant to the search details entered, with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab

	<ul style="list-style-type: none"> ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p> <p>Alt-Step 7b: Once the system displays the search results, the volunteer clicks on the relevant venue type. → Invoke UC 4.6 Search Venue Type On webpage: Venue Type Webpage</p>
	<p>Alt-Step 9: The volunteer clicks the “Back to list” button. → Go to step 13 On webpage: Venue Type Webpage</p>
CONCLUSION:	<p>The system, using C#, JavaScript, CSS and HTML, displays the Venue Type Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Venues” tab ➤ “Types” tab ➤ “Residences” tab ➤ “Venue Type” h1 header ➤ “Add a new Venue Type” hyperlink button ➤ “Find by name” textbox ➤ “Search” button

	<ul style="list-style-type: none"> ➤ “Description” container with a list of current venue types ➤ “Edit” button as per MVC layout ➤ “Details” button as per MVC layout ➤ “Delete” button as per MVC layout <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button <p>The venue types in the venue type’s container is populated from the VenueType using Entity Framework and LINQ queries.</p>
POST-CONDITION:	The volunteer has successfully deleted a venue type.
BUSINESS RULES	<ul style="list-style-type: none"> • Only a volunteer or admin member can delete a venue type.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • Once a venue type is deleted it must no longer show in the venue type container list.
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Add Residence		USE CASE TYPE
USE CASE ID:	4.9		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of admin wanting to add a residence to the system. Admin navigates to the venue webpage and click the residence tab. Admin clicks “Add” and the system will prompt admin to enter the required information. Admin will enter the information and click “Save” then the system will validate the information and save it to the Residence table. This use case concludes by a new residence being added to the system.</p>		
PRE-CONDITION:	Admin must be logged into the system.		
TRIGGER:	Admin wants to add a residence.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: Admin clicks the “Venues” dropdown list item under the “Events” navigation bar menu item. Events > Venues</p>		<p>Step 2: The system displays Screen: 4.2 Search Venue. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members

			<ul style="list-style-type: none"> ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Venues" and "Venue Type", the "Venue" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues' details populated by the Venue table. The list contains the following information about the venues as retrieved from the Venue table: <ul style="list-style-type: none"> • Name • Capacity • Accessibility • AddressID • VenueTypeID <p>Above the container is a textbox labelled</p>
--	--	--	--

			<p>“Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new venue”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the “Residence” tab. Screen: 4.2 Search Venue</p>		<p>Step 4: The system displays Screen: 4.10 Search Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items.

			<p>The body contains three tabs, “Venues”, “Venue Type” and “Residence”, the “Residence” tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Residences”.➤ A container below the header containing a list of all the venue types details populated by the <u>Residence</u> table. The list contains the following information about the Residence as retrieved from the <u>Residence</u> table:<ul style="list-style-type: none">● Description <p>Above the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new residence”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
--	--	--	---

	<p>Step 5: Admin clicks the add button. Screen: 4.10 Search Residence</p>	<p>Step 6: The system displays the Screen: 4.9.2 Add Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, "Venues" and "Venue Type" and "Residence", the "Residence" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled "Residences". ➤ A container below the
--	---	---

			<p>header containing a “Residence” (textbox), “ResType” (textbox), “Create” button</p> <p>➤ Below the container are two buttons entitled “Cancel” and “Return”</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks</p>
	<p>Step 7: Admin enters the required information into the textbox and clicks the “Create” button.</p> <p>On webpage: Residence</p>		<p>Step 8: The system validates that the “Residence Description” and “ResType Description” textboxes contain no more than 25 characters.</p>
			<p>Step 9: The system adds the following information to the Venue Type table:</p> <ul style="list-style-type: none"> • Description • ResType • ResID (automatically generated per addition)
			<p>Step 10: The system displays a modal with the following message, “You have successfully added a new residence.” with a button below the message entitled “Confirm”.</p>

	<p>Step 11: Admin clicks the “Confirm” button.</p>	<p>Step 12: The system displays Screen: 4.10 Search Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, “Venues”, “Venue Type” and “Residence”, the “Residence” tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Residences”.➤ A container below the header
--	---	---

			<p>containing a list of all the venue types details populated by the Residence table. The list contains the following information about the Residence as retrieved from the Residence table:</p> <ul style="list-style-type: none"> • Description <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are four buttons entitled "Add a new residence", "Update", "Delete" and "Return".</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
ALTERNATE COURSES:	Alt Step 7: Admin chooses not to create a residence and clicks the "Return" button. → Go to Step 4		
	Alt Step 8: The "Residence Description" textbox contains more than 25 characters. The system displays a viewbag error message below the textbox indicating the maximum characters allowed is 25. → Go to Step 6		
CONCLUSION:	Admin has created a residence.		
POST-CONDITION:	A student type is added to the Residence table.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can add a residence. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		

ASSUMPTIONS:	• None
OPEN ISSUES:	None

USE CASE NAME:	Search Residence	USE CASE TYPE	
USE CASE ID:	4.10	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of admin wanting to search for a specific residence. Admin navigates to the Venue webpage and clicks on the Residence tab. The system displays a container containing a list of residences, admin uses the search bar functionality to search for a specific residence. The system displays the search results. The use case concludes by admin viewing the specific residence information.</p>		
PRE-CONDITION:	Admin must be logged into the system.		
TRIGGER:	Admin wants to search for a residence.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: Admin clicks the “Venue” dropdown list item under the “Events” navigation bar menu item. Events > Venues</p>		<p>Step 2: The system displays Screen: 4.2 Search Venue. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content

			<ul style="list-style-type: none"> ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Venues" and "Venue Type", the "Venue" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues' details populated by the Venue table. The list contains the following information about the venues as retrieved from the Venue table: <ul style="list-style-type: none"> • Name • Capacity • Accessibility • AddressID • VenueTypeID <p>Above the container is a textbox labelled "Search". To the left of</p>
--	--	--	--

			<p>the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new venue”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the “Residence” tab. Screen: 4.2 Search Venue</p>		<p>Step 4: The system displays Screen: 4.10 Search Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, “Venues”, “Venue</p>

			<p>Type” and “Residence”, the “Residence” tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Residences”. ➤ A container below the header containing a list of all the venue types details populated by the Residence table. The list contains the following information about the Residence as retrieved from the Residence table: ● Description <p>Above the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new residence”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the</p>		<p>Step 6: The system performs a search and displays Screen: 4.10</p>

	<p>textbox and clicks the “search” button. Screen: 4.10 Search Residence</p>	<p>Search Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, “Venues”, “Venue Type” and “Residence”, the “Residence” tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Residences”. ➤ A container below the header containing a list of all the venue
--	---	--

			<p>types details populated by the <u>Residence</u> table. The list contains the following information about the Residence as retrieved from the <u>Residence</u> table:</p> <ul style="list-style-type: none"> • Description <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are four buttons entitled "Add a new residence", "Update", "Delete" and "Return".</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 7: Admin views the search results and clicks the "Return" button.</p> <p>Screen: 4.10 Search Residence</p>		<p>Step 8: The system displays the Main Menu webpage. This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items following elements from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Venues ➤ Events

		<ul style="list-style-type: none">➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of upcoming events populated by the Event table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the Event table:	<ul style="list-style-type: none">• Name• Date• Time• Venue• Summary <p>Each list item contains two buttons to the right of the item entitled "RSVP" and "View More Details". Above the</p>
--	--	---	---

			<p>container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. To the right of the body is another container with a list of upcoming events that the user has rsvp'd to, which is determined by the RSVP_Event table, and is populated by the Event table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the Event table:</p> <ul style="list-style-type: none"> • Name • Date • Time • Venue <p>Using CSS, JavaScript, C# and HTML.</p>
ALTERNATE COURSES:	Alt Step 5: Admin scrolls through the container containing a list of residences and finds the residence information that they are searching for and clicks the “Return” button. On webpage: Residence → Go to Step 8		
	Alt Step 6: The system has no matches. The following message will be displayed in the container, “Your search has no results”. On webpage: Search Residence		
CONCLUSION:	The system displays the search results to admin.		
POST-CONDITION:	Admin has searched for a residence.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can search a residence. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Update Residence	USE CASE TYPE	
USE CASE ID:	4.11	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Medium	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of admin updating a residence. The use case begins by admin navigating to the Venue webpage and clicking on the Residence tab. The system displays a container with a list of all the residences, which admin will search and click the “Update” button for the relevant list item. The system will display the list item in an editable format, which admin can update and save. This use case concludes by a residence being updated.</p>		
PRE-CONDITION:	Admin must be logged into the system.		
TRIGGER:	Admin wants to update a residence.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: Admin clicks the “Venue” dropdown list item under the “Events” navigation bar menu item. Events > Students On webpage: Main Menu Admin</p>		<p>Step 2: The system displays Screen: 4.2 Search Venue. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events

			<ul style="list-style-type: none"> ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Venues" and "Venue Type", the "Venue" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues' details populated by the <u>Venue</u> table. The list contains the following information about the venues as retrieved from the <u>Venue</u> table: <ul style="list-style-type: none"> • Name • Capacity • Accessibility • AddressID • VenueTypeID <p>Above the container is a textbox labelled "Search". To the left of</p>
--	--	--	---

			<p>the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new venue”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the “Residence” tab. Screen: 4.2 Search Venue</p>		<p>Step 4: The system displays Screen: 4.10 Search Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, “Venues”, “Venue</p>

			<p>Type” and “Residence”, the “Residence” tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Residences”. ➤ A container below the header containing a list of all the venue types details populated by the Residence table. The list contains the following information about the Residence as retrieved from the Residence table: ● Description <p>Above the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new residence”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the textbox and clicks the “search” button.</p>		<p>Step 6: The system performs a search and displays Screen: 4.10 Search Residence. This</p>

	Screen: 4.10 Search Residence		<p>webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, "Venues", "Venue Type" and "Residence", the "Residence" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled "Residences". ➤ A container below the header containing a list of all the venue types details
--	-------------------------------	--	---

			<p>populated by the <u>Residence</u> table. The list contains the following information about the Residence as retrieved from the <u>Residence</u> table:</p> <ul style="list-style-type: none"> • Description <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are four buttons entitled "Add a new residence", "Update", "Delete" and "Return".</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 7: Admin clicks the "Update" button for the specific list item in the container.</p> <p>Screen: 4.10 Search Residence</p>		<p>Step 8: The system displays the Screen: 4.11.1 Update Residence.</p> <p>This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content

			<ul style="list-style-type: none">➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, "Venues" and "Venue Type" and "Residence", the "Residence" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Update Residence".➤ A container below the header containing a "Residence" (textbox), "ResType" (textbox), "Create" button➤ Below the container are two buttons entitled "Cancel" and "Return" <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks</p>
--	--	--	--

	Step 9: Admin updates the residence description textbox and clicks “Update” 4.11.1 Update Residence		Step 10: The system validates the information updated by admin is no more than 25 characters.
			Step 11: The system updates the following attributes in the Residence table: <ul style="list-style-type: none"> • Description • ResType per the ResID
			Step 12: The system displays a modal with the following message, “You have successfully updated a Residence” with a button below the message entitled “Confirm”.
	Step 13: Admin clicks the “Confirm” button.		Step 14: The system performs a search and displays Screen: 4.10 Search Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively: <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content ➤ Gallery ➤ A message icon which implies the metaphor of notifications,

			<p>which contains the link to the user's notifications.</p> <ul style="list-style-type: none"> ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, "Venues", "Venue Type" and "Residence", the "Residence" tab is displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled "Residences". ➤ A container below the header containing a list of all the venue types details populated by the <u>Residence</u> table. The list contains the following information about the Residence as retrieved from the <u>Residence</u> table: ● Description <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are four</p>
--	--	--	--

			buttons entitled “Add a new residence”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
ALTERNATE COURSES:	Alt Step 5a: Admin chooses not to update a residence and clicks the “Return” button. → Go to Step 2		
	Alt Step 5b: Admin scrolls through the container containing a list of residences and finds the residence information that they are looking for and clicks the “Update” button. On webpage: Residence → Go to Step 8		
	Alt Step 8: The “Residence Description” textbox contains more than 25 characters. The system displays a viewbag error message below the textbox indicating the maximum characters allowed is 25. → Go to Step 6		
CONCLUSION:	Admin has updated a residence.		
POST-CONDITION:	The system updates the Residence table.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can update a residence. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Delete Residence	USE CASE TYPE	
USE CASE ID:	4.12	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Low	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of admin deleting a residence. The use case begins by admin navigating to the Events webpage and clicking on the Residence tab. The system displays a container with a list of all the residences, which admin will search and click the “Delete” button for the relevant list item. The system confirms if admin wants to delete that residence. This use case concludes by a residence being deleted.</p>		
PRE-CONDITION:	Admin must be logged into the system.		
TRIGGER:	Admin wants to delete a residence.		
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: Admin clicks the “Venue” dropdown list item under the “Events” navigation bar menu item. Events > Venue On webpage: Main Menu	System Response	
		Manual Action	Automated Action
			<p>Step 2: The system displays Screen: 4.2 Search Venue. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events

			<ul style="list-style-type: none">➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two tabs, "Venues" and "Venue Type", the "Venue" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the venues' details populated by the Venue table. The list contains the following information
--	--	--	---

			<p>about the venues as retrieved from the <u>Venue</u> table:</p> <ul style="list-style-type: none"> • Name • Capacity • Accessibility • AddressID • VenueTypeID <p>Above the container is a textbox labelled "Search". To the left of the textbox is a button labelled "Search". Below the container are four buttons entitled "Add a new venue", "Update", "Delete" and "Return". Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: Admin clicks the "Residence" tab. Screen: 4.2 Search Venue</p>		<p>Step 4: The system displays Screen: 4.10 Search Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p>

			<ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, "Venues", "Venue Type" and "Residence", the "Residence" tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Residence s".➤ A container below the
--	--	--	--

			<p>header containing a list of all the venue types details populated by the Residence table. The list contains the following information about the Residence as retrieved from the Residence table:</p> <ul style="list-style-type: none"> • Description <p>Above the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new residence”, “Update”, “Delete” and “Return”. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: Admin enters information into the textbox and clicks the “search” button.</p>		<p>Step 6: The system performs a search and displays</p> <p>Screen: 4.10 Search</p>

	Screen: 4.10 Search Residence	Residence. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:	<ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, "Venues", "Venue Type" and "Residence", the</p>
--	-------------------------------	---	---

			<p>“Residence” tab is displayed with the following elements:</p> <ul style="list-style-type: none">➤ A header entitled “Residence s”.➤ A container below the header containing a list of all the venue types details populated by the Residence table. The list contains the following information about the Residence as retrieved from the Residence table:<ul style="list-style-type: none">● DescriptionAbove the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new residence”,
--	--	--	---

			"Update", "Delete" and "Return". Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
	<p>Step 7: Admin clicks the "Delete" button for the specific list item in the container.</p> <p>Screen: 4.10 Search Residence</p>		<p>Step 8: The system displays a modal with the following message, "Once this residence is deleted it is unrecoverable)" with a button below the message entitled "Confirm".</p>
	<p>Step 9: Admin clicks the "Confirm" button on the modal.</p>		<p>Step 10: The system deleted the following attributes in the <u>Residence</u> table:</p> <ul style="list-style-type: none"> • Description • ResType • ResidenceID <p>Per the ResidenceID</p>
			<p>Step 11: The system displays a modal with the following message, "You have successfully deleted the residence" with a button below the message entitled "Confirm".</p>
	<p>Step 12: Admin clicks the "Confirm" button.</p>		<p>Step 13: The system performs a search and displays Screen: 4.10 Search Residence. This</p>

		<p>webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains three tabs, "Venues", "Venue Type" and "Residence", the "Residence" tab is</p>	
--	--	---	--

			<p>displayed with the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Residence s”. ➤ A container below the header containing a list of all the venue types details populated by the <u>Residence</u> table. The list contains the following information about the Residence as retrieved from the <u>Residence</u> table: • Description <p>Above the container is a textbox labelled “Search”. To the left of the textbox is a button labelled “Search”. Below the container are four buttons entitled “Add a new residence”, “Update”, “Delete” and “Return”.</p>
--	--	--	--

			Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
ALTERNATE COURSES:	Alt Step 9: Admin clicks the “Cancel” button. → Go to Step 6		
CONCLUSION:	Admin deletes a residence.		
POST-CONDITION:	The Residence table has been updated.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin can delete a residence. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

2.3.5 Events Narratives

USE CASE NAME:	Create Event	USE CASE TYPE
USE CASE ID:	AUC 2	Abstract: <input checked="" type="checkbox"/> Extension: <input type="checkbox"/>
PRIORITY:	High	
SOURCE:	TuksRes Women in Leadership Academy	
PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • Volunteer (PBA) 	
DESCRIPTION:	<p>This use case describes the event of a volunteer creating an event. The use case begins when the volunteer wants to create a new event on the TRWLA system and goes to the main menu. The system will prompt the volunteer to enter the required information pertaining to an event. The volunteer will enter all the required event information. The use case concludes when the volunteer has entered all the event information and wants to click an event type.</p>	
PRE-CONDITION:	The volunteer must be logged into the system.	
TYPICAL COURSE OF EVENTS:	<p>Step 1: The volunteer clicks the events dropdown list in the navigation bar at the top of the webpage: Events On Webpage: Main Menu Volunteer</p> <p>Step 2: The system, using JavaScript, displays the dropdown list options, namely: Create, Events, Venues and Guest Speakers On Webpage: Main Menu Volunteer</p> <p>Step 3: The volunteer clicks the Create list item. On Webpage: Main Menu Volunteer</p> <p>Step 4: The system, using JavaScript, CSS and HTML, displays the Events Type Selection Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Event Type Selection” h1 header ➤ “Please select an Event Type” h2 header ➤ “Function” button ➤ “Community Engagement” button 	

	<ul style="list-style-type: none"> ➤ “Lecture” button
	<p>Step 5: The volunteer clicks the “Function” button. Invoke UC 5.1 Create Function</p>
ALTERNATE COURSES:	<p>Alt Step 4: The volunteer does not want to create an event and clicks the “Return” button. The system displays the webpage: Main Menu Volunteer.</p>
	<p>Alt Step 5a: The volunteer clicks the “Lecture” button. →Invoke UC 5.2 Create Lecture</p>
	<p>Alt Step 5b: The volunteer clicks the “Community Engagement” button. →Invoke UC 5.3 Create Community Engagement</p>
POST-CONDITION:	The volunteer has selected an event type and has initiated one of the three following use cases: 5.1 Create Function, 5.2 Create Community Outreach or 5.3 Create Lecture.

USE CASE NAME:	Create Function	USE CASE TYPE									
USE CASE ID:	5.1	Business Requirements: <input type="checkbox"/>									
PRIORITY:	High	System Analysis: <input type="checkbox"/>									
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>									
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)										
PRIMARY SYSTEM ACTOR	None										
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • User: Volunteer and Student (ERA) 										
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Guest Speaker 										
DESCRIPTION:	<p>This use case describes the event of creating a function on the TRWLA System. The use case is initiated when AUC 2 Create Event has concluded. The system prompts the volunteer to enter the required function information. The volunteer enters the required information and clicks the “Create Event” button. The system validates the entered information, and if it is all valid and complete the system adds the new event to the Event and Function tables and confirms the successful creation of an event. The use case concludes when the system confirms creation.</p>										
PRE-CONDITION:	<p>The volunteer must be logged into the system AUC 2 Create Event must have concluded.</p>										
TRIGGER:	<p>AUC 2 Create Event has concluded and the volunteer has clicked the Function event type option.</p>										
TYPICAL COURSE OF EVENTS:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; width: 30%;">Actor Action</th> <th colspan="2" style="text-align: center;">System Response</th> </tr> <tr> <th style="text-align: center;"></th> <th style="text-align: center;">Manual Action</th> <th style="text-align: center;">Automated Action</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;"> Step 1: The volunteer wishes to create a new function on the system. </td><td></td><td> Step 2: The system, using JavaScript, CSS and HTML, displays the webpage: Event Type Selection with the following layout: → Invoke AUC 2 Create Event Header Navigation bar from left to right: </td></tr> </tbody> </table>	Actor Action	System Response			Manual Action	Automated Action	Step 1: The volunteer wishes to create a new function on the system.		Step 2: The system, using JavaScript, CSS and HTML, displays the webpage: Event Type Selection with the following layout: → Invoke AUC 2 Create Event Header Navigation bar from left to right:	
Actor Action	System Response										
	Manual Action	Automated Action									
Step 1: The volunteer wishes to create a new function on the system.		Step 2: The system, using JavaScript, CSS and HTML, displays the webpage: Event Type Selection with the following layout: → Invoke AUC 2 Create Event Header Navigation bar from left to right:									

		<ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Event Type Selection” h1 header ➤ “Please select an Event Type” h2 header ➤ “Function” button ➤ “Community Engagement” button ➤ “Lecture” button
	<p>Step 3: The volunteer clicks the “Function” button.</p>	<p>Step 4: The system, using JavaScript, CSS and HTML, displays the “Create Function” webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header

		<ul style="list-style-type: none"> ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Create a Function” h1 header ➤ “Event” h3 header ➤ “Name” textbox ➤ “Guest Speaker” drop down list as populated from the GuestSpeaker table using SQL, Entity Framework and LINQ Queries. ➤ “Register Guest Speaker” hyperlink button ➤ “Accepted invitation” radio button ➤ “Pending invitation” radio button
--	--	---

			<ul style="list-style-type: none"> ➤ “Time” time picker ➤ “Date” date picker ➤ “Venue” drop down list as populated from the Venue table using SQL, Entity Framework and LINQ Queries. ➤ “Add a Venue” hyperlink button ➤ “Summary” textbox ➤ Description” textbox ➤ Picturebox for posters ➤ “Browse” button <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Return” button ➤ “Cancel” button
	<p>Step 5: The volunteer enters all of the relevant details in the empty fields.</p>		<p>Step 6: The system, using C# and JavaScript, validates that the information entered is correct according to the following system standards:</p> <ul style="list-style-type: none"> • Name (Must be a string value of no more than 35 characters.) • Guest Speaker (Must be a value selected from the Guest Speaker dropdown list which is populated by the

			<p>GuestSpeaker table), either the “Pending invitation” or “Accepted invitation” radio button must have been clicked.</p> <ul style="list-style-type: none"> • Time (Must be a valid time format i.e HH:MM as clicked from the time picker) • Date (Must be a valid date in the format i.e. CCYY-MM-DD as clicked from the date picker) • Venue (Must be a value clicked from the Venue dropdown list which is populated by the Venue table) • Summary (Must be a string value of no more than 100 characters) • Description (Must be a string value of no more than 300 characters)
	Step 7: The volunteer clicks the “Browse” button.		Step 8: Step 10: The system, using C# and JavaScript, displays a file dialogue box which allows the volunteer to search for a picture on their machine.

	<p>Step 9: The volunteer searches and clicks the relevant picture and uploads it.</p>		<p>Step 10: The system, using C# and JavaScript, validates that the picture is in the correct format and displays the relevant picture clicked by the volunteer in the picturebox.</p> <ul style="list-style-type: none"> • Poster (Must be in a valid image format png or svg)
	<p>Step 11: The volunteer clicks the “Create” button.</p>		<p>Step 12: The system, using C# and JavaScript, displays a Modal with the following message: “Are you sure you want to create this event?” with three buttons namely, “Confirm”, “Edit” and “Return.”</p>
	<p>Step 13: The volunteer clicks the “Confirm” button.</p>		<p>Step 14: The system, using C# and JavaScript, displays a Modal with the following message: “Congratulations. Your event has been created. Please note: The guest speaker will display as ‘To be Announced’ until marked as accepted.” with a “Confirm” button.</p>
	<p>Step 15: The volunteer clicks the “Confirm” button</p>		<p>Step 16: The system, using C# and SQL, adds the new event to the Event and Function tables using entity framework and LINQ queries and updates the TRWLA timeline so that the event is visible to all relevant members. The</p>

			<p>tables share the same EventID primary key.</p> <p>Step 17: The system, using JavaScript, CSS and HTML, redirects to the Main Menu Volunteer webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header
--	--	--	--

			<ul style="list-style-type: none">➤ “Search upcoming events by Name” textbox➤ “Sort with the following:” h4 header➤ “Event name” button➤ “Date” button➤ “Upcoming TRWLA Events” h2 header➤ “My Upcoming Events” h2 header➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:<ul style="list-style-type: none">• “Event Name” h3 header• “Summary” h4 header• “Date of the event” h4 header• “Time of event” h4 header• “Venue name” h4 header• “RSVP” button• “Details” button• “Not going” button in the “My upcoming events” container.
--	--	--	--

			Footer <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
ALTERNATE COURSES:	<p>Alt Step 5a: The volunteer clicks the “Register Guest Speaker” hyperlink button.</p> <p>→ Invoke UC 6.1 Register Guest Speaker</p> <p>On webpage: Register Guest Speaker</p>		
	Alt Step 5b: The volunteer clicks the “None” list item in the guest speaker dropdown list. The system hides the “Pending” and “Accepted” radio buttons.		
	<p>Alt Step 6: The information entered are in the incorrect format and do not adhere to the system’s standards for the minimum and maximum characters. The system, using JavaScript, displays viewbag error messages beneath the relevant textboxes that indicate what the correct standard is.</p> <p>→ Go to Step 5</p>		
	<p>Alt Step 7: The volunteer does not click the “Browse” button to upload a poster.</p> <p>→ Go to Step 11</p>		
	<p>Alt Step 10: The photo uploaded is in the incorrect format and does not adhere to the system’s standards. The system, using JavaScript, displays viewbag error messages beneath the picture box that indicate what the correct standard is.</p> <p>→ Go to Step 9</p>		
	Alt Step 13a: The volunteer clicks the “Edit” button. The system returns to the current Create Function webpage.		
	<p>Alt Step 13b: The volunteer clicks the “Return” button. The system, using JavaScript, CSS and HTML, displays the Volunteer Main Menu.</p> <p>→ Go to Step 17</p>		
	<p>Alt Step 14: The volunteer has clicked the “Accepted Invitation” radio button. The system, using C# and JavaScript, displays a Modal with the following message: “Congratulations. Your event has been created.” with a “Confirm” button.</p> <p>→ Go to Step 15</p>		
CONCLUSION:	The system redirects to the Main Menu Volunteer webpage with the following layout: Header		

	<p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button ● “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
POST-CONDITION:	The volunteer has successfully created a function.
BUSINESS RULES	<ul style="list-style-type: none"> ● Only a volunteer can create a function event.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> ● The newly created function must be displayed on each member’s timeline in order to RSVP to the event.
ASSUMPTIONS:	<ul style="list-style-type: none"> ● None
OPEN ISSUES:	None

USE CASE NAME:	Create Community Outreach		USE CASE TYPE
USE CASE ID:	5.2		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Student (ERA) 		
DESCRIPTION:	<p>This use case describes the event of creating a community outreach event on the TRWLA System. The use case is initiated when AUC 2 Create Event has concluded. The system prompts the volunteer to enter the required community outreach information. The volunteer enters the required information and clicks the “Create Event” button. The system validates the entered information, and if it is all valid and complete the system adds the new event to the Event and CommServ tables and confirms the successful creation of an event. The use case concludes when the system confirms creation.</p>		
PRE-CONDITION:	<p>The volunteer must be logged into the system. AUC 2 Create Event must have concluded.</p>		
TRIGGER:	<p>AUC 2 Create Event has concluded and the volunteer has clicked the community outreach event type item.</p>		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wishes to create a new community outreach event on the system.</p>		<p>Step 2: The system displays the webpage: Event Type webpage → Invoke AUC 2 Create Event</p>
	<p>Step 3: The volunteer enters the required information Screen: 5.2.1Create Community Outreach VA</p>		<p>Step 8: The system displays the Screen: 5.2.1Create Community Outreach VA. This webpage contains a header and body.</p>

			<p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading Content➤ Gallery➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Create a community engagement". <p>A container below the header containing:</p> <ul style="list-style-type: none">➤ "Name" (textbox)➤ "Time" (textbox)➤ "Date" (textbox)➤ "Venue" (textbox)
--	--	--	--

			<ul style="list-style-type: none"> ➤ “Add Venue” hyperlink ➤ “Summary” (textbox) ➤ “Description” (textbox) ➤ “Content” (Dropdown list) ➤ “Upload Content” hyperlink ➤ Picturebox ➤ “browse” button <p>Below the container are three buttons entitled “Create”, “Return” and “Cancel”</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks</p>
	<p>Step 5: The volunteers enters the remaining required information and clicks the “Browse” button.</p> <p>Screen: 5.2.1Create Community Outreach VA</p>		<p>Step 6: The system displays a file dialogue box which allows the volunteer to search for a picture on their machine.</p> <p>Using C#.</p>
	<p>Step 7: The volunteer searches and clicks the relevant picture and uploads it.</p>		<p>Step 8: The system displays the relevant picture clicked by the volunteer in the picturebox.</p> <p>Using C#, HTML and JavaScript.</p>
	<p>Step 9: The volunteer clicks the “Create Event” button.</p> <p>Screen: 5.2.1Create Community Outreach VA</p>		<p>Step 10: The system validates that all the required information has been entered and is in the correct format, namely:</p>

			<ul style="list-style-type: none"> • Name (Must be a string value of no more than 35 characters.) • Date (Must be a valid date in the format i.e. CCYY-MM-DD as clicked from the date picker) • Time (Must be a valid time format i.e HH:MM as clicked from the time picker) • Venue (Must be a value clicked from the Venue dropdown list which is populated by the Venue table) • Summary (Must be a string value of no more than 100 characters) • Description (Must be a string value of no more than 300 characters) • Event Type (Must be clicked from the list items in the Event Type dropdown list) • Reading content Must be a value clicked from the Reading content dropdown list which is populated by the CommServContent table)
--	--	--	--

			<ul style="list-style-type: none"> Poster (Must be in a valid image format png or svg)
			<p>Step 11: The system adds the new event to the <u>Event</u>, <u>Content</u> and <u>CommServ</u> tables using entity framework and LINQ queries and updates the TRWLA timeline so that the event is visible to all relevant members. The tables share the same EventID primary key.</p>
			<p>Step 12: The system displays a modal with the following message: “Your event has been created” and a button in the bottom right corner entitled “Confirm” Using C#, CSS, HTML and JavaScript.</p>
	<p>Step 13: The volunteer clicks the “Confirm” button.</p>		<p>Step 14: The system displays the Main Menu Volunteer webpage. This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items following elements from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading Content

			<ul style="list-style-type: none"> ➤ Gallery ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains the following elements displayed vertically:</p> <ul style="list-style-type: none"> ➤ A container at the top of the body with the following message "Welcome (Name retrieved from <u>Person</u> table)" ➤ A textbox with a button under entitled "Search upcoming events by name". ➤ To the right of the textbox is a label entitled "Sort with the following" with two buttons below the label "Event" and "Date"
--	--	--	--

			<p>➤ A container containing a list of upcoming events populated by the Event table using entity framework and LINQ queries. The list contains the following information about events as retrieved from the Event table:</p> <ul style="list-style-type: none">• Name• Date• Time• Venue• Summary <p>Each list item contains two buttons under the item entitled “RSVP” and “Details”. Below the container are four buttons labelled “Create” “Update” and “Delete”.</p> <p>➤ To the right of the body is another container with a list of upcoming events that the user has rsvp'd to, which is determined by the RSVP_Event table, and is populated by the Event table using entity framework</p>
--	--	--	--

			<p>and LINQ queries. The list contains the following information about events as retrieved from the Event table:</p> <ul style="list-style-type: none"> • Name • Date • Time • Venue <p>Using CSS, JavaScript, C#, SQL and HTML.</p>
ALTERNATE COURSES:	<p>Alt Step 5: The volunteer does not wish to upload a poster. → Go to Step 9</p>		
	<p>Alt Step 11: The validation of event information was not successful and the system displays viewbag error messages beneath each field that is not valid. → Go to Step 5</p>		
CONCLUSION:	The volunteer created a community outreach event by first invoking Abstract Use Case 2. The system prompted the volunteer to provide the required information for a community outreach event. The volunteer entered the relevant information into the system and the system verified the information.		
POST-CONDITION:	Event table, Content table and CommServ table have been updated.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only volunteers can add a community outreach event. • Only members that are invited to an event may attend unless it is specified to be an open event. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • The newly created community outreach must be displayed on the relevant member's timeline to RSVP to the event. 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Log Event Attendance	USE CASE TYPE
USE CASE ID:	5.8	Business Requirements: <input type="checkbox"/>
PRIORITY:	High	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 	
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Student 	
DESCRIPTION:	<p>This use case describes the event of a volunteer logging the attendance of a student when a student has attended a function, lecture or community outreach event. The volunteer will click the “Log Attendance” list item under the event navigation bar item. The system will display a container with the details of all the previous events and a hyperlink under each event. The volunteer will then search through the list of events and click the “Log Attendance” hyperlink under the event. The system will display a webpage with a container containing a list of students that have RSVP’d to the specific event with a hyperlink labelled “Log Attendance” next to each list item and an item to add a student’s attendance who has not RSVP’d for that event, but who has attended the event. The volunteer will find the student and click the “Log Attendance” hyperlink. The volunteer will confirm the attendance of the student. The system will update the Attendance table and the studentmilestone table according to the milestone type that is linked to each event.</p>	
PRE-CONDITION:	The volunteer must be logged into the system.	
TRIGGER:	The volunteer wants to log attendance for an event that has taken place.	
TYPICAL COURSE OF EVENTS:	Actor Action	System Response
		Manual Action Automated Action
	Step 1: The volunteer clicks the “Log Attendance” button. On webpage: Main Menu Volunteer	Step 2: The system displays the Log Attendance webpage. The webpage contains a header and body. The header contains a navigation bar with the

			<p>following menu items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading content➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two containers. The first container contains a header entitled Log Event Attendance. Below the first container is an editable textbox with a label above entitled "Past Events" and a button beneath entitled "Search Past Events". Next to the textbox is a label entitled "Sort with the following" and two buttons under labelled "Event name" and "Date". Below this is a container containing a list of past events with</p>
--	--	--	--

			<p>the following details retrieved from the <u>Event</u> table:</p> <ul style="list-style-type: none"> • Name • Date • Time • Summary <p>and each list item has a hyperlink at the bottom labelled “Log attendance”.</p>
	<p>Step 3: The volunteer enters search details relating to the specific event in the “Search” textbox and clicks the “Search Past Events” button.</p> <p>On webpage; Log Event Attendance</p>		<p>Step 4: The system displays the Log Attendance webpage. The webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two containers. The first</p>

			<p>container contains a header entitled Log Event Attendance. Below the first container is an editable textbox with a label above entitled “Past Events” and a button beneath entitled “Search Past Events”. Next to the textbox is a label entitled “Sort with the following” and two buttons under labelled “Event name” and “Date”. Below this is a container containing a list of the relevant past events based on the volunteers search with the following details retrieved from the <u>Event</u> table:</p> <ul style="list-style-type: none"> • Name • Date • Time • Summary <p>and each list item has a hyperlink at the bottom labelled “Log attendance”.</p>
	<p>Step 5: The volunteer clicks the “Log Attendance” hyperlink for the specific event list item. Screen: 5.8.1 Log Event Attendance VA</p>		<p>Step 6: The system displays Screen: 5.8.1 Log Event Attendance VA. The webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p>

			<ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading content➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two containers. The first container at the top contains a header entitled Log Attendance. Below the container is a header entitled Log the attendance of a student, with an editable textbox beneath it and a button beneath the textbox labelled "Search for a Student". The second container contains a list of students populated by the Event table and Person table connected by RSVP table according to the students' that have RSVP'd for a specific event. Each list item contains the following information</p>
--	--	--	--

			<p>retrieved from the <u>Person</u> table:</p> <ul style="list-style-type: none"> • Name • Surname • StudentNumber <p>And a hyperlink labelled “Log Attendance”. Below to the left of the container are two buttons next to one another labelled “Add new student” and “Return”.</p>
	<p>Step 7: The volunteer enters search details relating to the specific student in the “Search” textbox and clicks the “Search for a Student” button.</p> <p>On webpage: Log Attendance of a Student</p>		<p>Step 8: The system displays the Log Attendance of a Student webpage. The webpage contains a header and a body.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User

			<p>dropdown list items.</p> <p>The body contains two containers. The first container at the top contains a header entitled Log Attendance. Below the container is a header entitled Log the attendance of a student, with an editable textbox beneath it and a button beneath the textbox labelled "Search for a Student". The second container contains a list of students populated by the Event table and Person table connected by RSVP table according to the students' that have RSVP'd for a specific event according to the relevant search results. Each list item contains the following information retrieved from the Person table:</p> <ul style="list-style-type: none"> • Name • Surname • StudentNumber <p>And a hyperlink labelled "Log Attendance".</p> <p>Below to the left of the container are two buttons next to one another labelled "Add new student" and "Return".</p>
	<p>Step 9: The volunteer finds the specific student and</p>		<p>Step 10: The system displays a modal with header entitled "Log</p>

	<p>clicks the “Log Attendance” hyperlink. On webpage: Log Attendance of a Student.</p>		<p>Attendance” and a paragraph below with the following text “Are you sure that you want to log the attendance of this student?” with two buttons below to the bottom right labelled “Confirm” and “Cancel”.</p>
	<p>Step 11: The volunteer clicks the “Confirm” button.</p>		<p>Step 12: The system updates the Attendance table with the following information:</p> <ul style="list-style-type: none"> • PersonID • EventID <p>And the StudentMilestone table with the following information:</p> <ul style="list-style-type: none"> • PersonID • MilestoneTypeID <p>According to the type of event (Community Outreach, Lecture or Function) that attendance is logged for.</p>
			<p>Step 13: The system displays the Log Attendance of a Student webpage. The webpage contains a header and a body.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events

			<ul style="list-style-type: none">➤ Reading content➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two containers. The first container at the top contains a header entitled Log Attendance. Below the container is a header entitled Log the attendance of a student, with an editable textbox beneath it and a button beneath the textbox labelled "Search for a Student". The second container contains a list of students populated by the Event table and Person table connected by RSVP table according to the students' that have RSVP'd for a specific event. Each list item contains the following information retrieved from the Person table:</p> <ul style="list-style-type: none">• Name• Surname• StudentNumber
--	--	--	--

			And a hyperlink labelled “Log Attendance”. Below to the left of the container are two buttons next to one another labelled “Add new student” and “Return”.
ALTERNATE COURSES:	<p>Alt Step 7: The volunteer is logging attendance for a student that did not RSVP for the specific event, but did attend the event.</p> <ol style="list-style-type: none"> 1. The volunteer will click the “Add a new student” button. 2. The system will display the Log attendance of a student who hasn’t RSVP’d. <p>The webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two containers. The first container at the top contains a header entitled Log Attendance. Below the container is a header entitled Log the attendance of a student who hasn’t RSVP’d, with an editable textbox beneath it and a button beneath the textbox labelled “Search for a Student”. The second container contains a list of students Person table. Each list item contains the following information retrieved from the <u>Person</u> table:</p> <ul style="list-style-type: none"> • Name • Surname • StudentNumber <p>And a hyperlink labelled “Log Attendance”. Below to the left of the container is one button labelled “Return”.</p> <ol style="list-style-type: none"> 3. The volunteer finds the specific student and clicks the “Log Attendance” hyperlink. <p>On webpage: Log Attendance of a Student who hasn’t RSVP’d. → Go to Step 10</p>		
	<p>Alt Step 11: The volunteer selects the “cancel” button. → Go to Step 8</p>		

CONCLUSION:	The attendance of a student is successfully logged.
POST-CONDITION:	The Attendance table and StudentMilestone table is updated
BUSINESS RULES	<ul style="list-style-type: none"> • Only past event attendance can be logged • Only authorised volunteers may log attendance of students
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Send Notification		USE CASE TYPE
USE CASE ID:	5.9		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Users: Admin, Volunteers and Students (ERA) 		
DESCRIPTION:	<p>This use case describes the event of a volunteer sending a notification to the TRWLA system users. The volunteer will navigate to the Send Notification webpage by clicking the list item under the Events navigation bar item. The webpage will display a dropdown list and a textbox. The dropdown list contains a list of users on the system for the volunteer to select recipients and the textbox is for the volunteer to type the notification. The volunteer will select the intended recipients, type the notification and click send notification. The system will validate the information and send the notification. This use case concludes by a notification being sent out by a volunteer.</p>		
PRE-CONDITION:	The volunteer must be logged into the system.		
TRIGGER:	The volunteer wants to send a notification.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1: The volunteer clicks the Send Notification item under the Events navigation bar item. Events > Send Notification On webpage: Main Menu Volunteer</p>		<p>Step 2: The system displays Screen: 5.9.1 Send Notification VA. The webpage contains a header and a body. The header contains a navigation bar with the following menu</p>

			<p>items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading content➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two containers. The first container at the top contains a header entitled "Send a Notification". Below the container is a header entitled Please select your recipients, with an editable textbox</p>
--	--	--	---

		beneath it and a button beneath the textbox labelled “Search recipients by residence”. The second container contains a list of people part of TRWLA populated by the Person table. Each list item contains the following information retrieved from the <u>Person</u> table: <ul style="list-style-type: none"> • Name • Surname • StudentNumber Below to the left of the container are two buttons next to one another labelled “Confirm Recipients” and “Return”.
	<p>Step 3: The volunteer enters information into to textbox and clicks the “Search recipients by residence” button.</p> <p>On webpage: Send a Notification.</p>	<p>Step 4: The system retrieves the information according to the search information entered by the volunteer and displays the Send a Notification webpage. The webpage contains a header and a body. The header contains a navigation bar with</p>

			<p>the following menu items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading content➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two containers. The first container at the top contains a header entitled "Send a Notification". Below the container is a header entitled Please select your recipients, with an</p>
--	--	--	---

		<p>editable textbox beneath it and a button beneath the textbox labelled “Search recipients by residence”. The second container contains a list of people part of TRWLA populated by the Person table according to the search performed. Each list item contains the following information retrieved from the Person table:</p> <ul style="list-style-type: none"> • Name • Surname • StudentNumber <p>Below to the left of the container are two buttons next to one another labelled “Confirm Recipients” and “Return”.</p>
	<p>Step 5: The volunteer selects the recipients and clicks “Confirm Recipients”.</p>	<p>Step 6: The system displays Screen: 5.9.3 Send Notification VA. The webpage contains a header and a body. The header contains a navigation bar with the following menu</p>

			<p>items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body contains two containers. The first container at the top contains a header entitled "Send a Notification". Below the container is a header entitled Please fill in your message. The second container</p>
--	--	--	--

			contains an editable textbox in the centre. Below to the left of the container are two buttons next to one another labelled “Send notification” and “Return”.
	Step 7: The volunteer types the notification in the textbox and clicks the “send notification” button.		Step 8: The system displays a modal with header entitled “Send message” and a paragraph below with the following text “Are you sure that you want to send this message?” with two buttons below to the bottom right labelled “Confirm” and “Edit”. Using CSS,HTML,JavaScript and C#.
	Step 9: The volunteer clicks the “Confirm” button.		Step 10: The system validates that the notification textbox is no more than 255 characters.
			Step 11: The system updates the Notification table with the following attributes: <ul style="list-style-type: none"> • Notification • NotificationID

			(autogenerated)
			<p>Step 12: The system displays a modal with header entitled “Congratulations!” and a paragraph below with the following text “You have sent the message” with a button below to the bottom right labelled “Confirm”.</p>
	<p>Step 13: The volunteer clicks the “Confirm” button.</p>		<p>Step 14: The system displays the Send a Notification webpage. The webpage contains a header and a body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains

			<p>the link to the user's notifications.</p> <p>➤ A profile picture which contains a link to the User dropdown list items.</p> <p>The body contains two containers. The first container at the top contains a header entitled "Send a Notification". Below the container is a header entitled Please select your recipients, with an editable textbox beneath it and a button beneath the textbox labelled "Search recipients by residence". The second container contains a list of people part of TRWLA populated by the Person table. Each list item contains the following information retrieved from the <u>Person</u> table:</p> <ul style="list-style-type: none">• Name
--	--	--	---

			<ul style="list-style-type: none"> • Surname • StudentNumber <p>Below to the left of the container are two buttons next to one another labelled “Confirm Recipients” and “Return”.</p>
ALTERNATE COURSES:	Alt Step 9: The volunteer clicks the “edit” button. → Go to step 8		
	Alt Step 10: The characters in the notification textbox is more than 255. The system will display a viewbag error message that the maximum characters allowed are 255. → Go to step 8		
CONCLUSION:	A message is successfully sent to the intended recipients.		
POST-CONDITION:	The Notification table is updated.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only a volunteer may send out notifications • Notifications will only be sent out to notify TRWLA members with important information. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Create Lecture	USE CASE TYPE	
USE CASE ID:	5.3	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • User: Volunteer and Student (ERA) 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Guest Speaker 		
DESCRIPTION:	<p>This use case describes the event of creating a lecture on the TRWLA System. The use case is initiated when AUC 2 Create Event has concluded. The system prompts the volunteer to enter the required lecture information. The volunteer enters the required information and clicks the “Create Event” button. The system validates the entered information, and if it is all valid and complete the system adds the new event to the Event and Lecture tables and confirms the successful creation of an event. The use case concludes when the system confirms creation.</p>		
PRE-CONDITION:	<p>The volunteer must be logged into the system AUC 2 Create Event must have concluded.</p>		
TRIGGER:	<p>AUC 2 Create Event has concluded and the volunteer has clicked the Lecture event type option.</p>		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wishes to create a new lecture event on the system.</p>		<p>Step 2: The system displays the webpage: Create Event → Invoke AUC 2 Create Event</p>
	<p>Step 3: The volunteer clicks the Lecture button On webpage: 5.0.2 VA</p>		<p>Step 4: Using C#, HTML, CSS and JavaScript the system displays webpage 5.3.1 VA which contains the following layout: Header:</p>

			<ul style="list-style-type: none"> ➤ A navigation bar containing the following items from left to right: <ul style="list-style-type: none"> ○ TRWLA Management System Label ○ Members dropdown list ○ Events dropdown list ○ Reading Content Dropdown list ○ Gallery item ○ A message icon that indicates notifications ○ A profile picture icon with a User dropdown list <p>Body</p> <p>The following required lecture fields down the left hand side of the body:</p> <ul style="list-style-type: none"> ➤ “Create Lecture Event” h1 Header ➤ “Name” textbox ➤ “Time” time picker ➤ “Date” date picker ➤ “Venue” dropdown list (As populated by the <u>Venue</u> table) ➤ “Add Venue” hyperlink ➤ “Summary” textbox ➤ “Description” textbox ➤ “Content” dropdown list (As populated by
--	--	--	--

			<p>the <u>LectureContent</u> table)</p> <ul style="list-style-type: none"> ➤ “Upload Content” hyperlink ➤ “Residence” dropdown list list (As populated by the <u>Residence</u> table) <p>On the right hand side of the body:</p> <ul style="list-style-type: none"> ➤ An empty picturebox on the right hand side of the body of the webpage with a button entitled “Browse” below it. To the right of the label is a cloud icon used as a metaphor to imply that a picture can be uploaded here. <p>Footer</p> <p>At the bottom right of the footer are the following three buttons from left to right:</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Return” button ➤ “Cancel” button
	<p>Step 5: The volunteers enters the required information and clicks the “Browse” button.</p>		<p>Step 6: The system displays a file dialogue box which allows the volunteer to search for a picture on their machine. On webpage: 5.3.2 VA</p>
	<p>Step 7: The volunteer searches and clicks the relevant picture and uploads it.</p>		<p>Step 8: The system displays the relevant picture clicked by the volunteer in the picturebox.</p>

	Step 9: The volunteer clicks the “Create” button		Step 10: Using JavaScript the system displays a modal with the following message: “Are you sure that you want to create this event?” with 3 buttons namely “Confirm”, “Edit” and “Cancel” On webpage 5.3.3 VA
	Step 11: The volunteer clicks the “Confirm” button		Step 12: The system validates that all the required information has been entered and is in the correct format, namely: <ul style="list-style-type: none"> • Name (Must be a string value of no more than 35 characters.) • Time (Must be a valid time format i.e HH:MM as clicked from the time picker) • Date (Must be a valid date in the format i.e. CCYY-MM-DD as clicked from the date picker) • Venue (Must be a value clicked from the Venue dropdown list which is populated by the Venue table) • Summary (Must be a string value of no more than 100 characters) • Description (Must be a string value of no

			<p>more than 300 characters)</p> <ul style="list-style-type: none"> Content Must be a value clicked from the Content dropdown list which is populated by the <u>LectureContent</u> table) Residence Must be a value clicked from the Content dropdown list which is populated by the <u>Residence</u> table) Poster (Must be in a valid image format png or svg)
			<p>Step 13: The system adds the new event to the <u>Event</u> and <u>Lecture</u> tables using LINQ queries and updates the TRWLA timeline so that the event is visible to all relevant members. The tables share the same EventID primary key.</p>
			<p>Step 14: The system displays a modal with the following message: "Congratulations Your Event has been created" and a button in the bottom right corner entitled "Confirm" On webpage 5.3.4 VA</p>
ALTERNATE COURSES:	<p>Alt Step 5: The volunteer does not wish to upload a poster. → Go to Step 9</p>		

	<p>Alt Step 12: The validation of event information was not successful and the system displays viewbag messages beneath each field that is either not filled in or in the incorrect format.</p> <p>→ Go to Step 5</p>
CONCLUSION:	The volunteer created a lecture event by first invoking Abstract Use Case 2 and then selecting the lecture event type. The system prompted the volunteer to provide the required information for a lecture event. The volunteer entered the relevant information into the system and the system verified that the information was in fact valid and complete. Once the information was verified, the system inserted the information onto the database in the <u>Event</u> and <u>Lecture</u> tables.
POST-CONDITION:	A lecture event has been created on the TRWLA system and the relevant members can view it on their timelines.
BUSINESS RULES	<ul style="list-style-type: none"> • Only members that are invited to an event may attend unless it is specified to be an open event. • Only a volunteer may create an event.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • The newly created lecture must be displayed on the relevant member's timeline in order to RSVP to the event.
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Search Event	USE CASE TYPE	
USE CASE ID:	5.4	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a user searching for current events on the system. The use case begins when the user clicks on the “Events” drop down list on their main menu/ home page. The system will then display the Events Webpage with a list of current events that the user must still respond to as well as a list of events that the user has already responded to. The user will then either enter search details in the “Search” textbox or locate the event in the containers. Once the user has located the specific event, the user will click on the event to view more details. The system will then display the Events Details webpage with all the relevant information pertaining to the event. The use case concludes when the user has successfully searched for an event on the system.</p>		
PRE-CONDITION:	The user must be logged into the system		
TRIGGER:	The user wants to search for an event on the system.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The user wants to search for an event on the system and clicks the “Events” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the drop down list items:</p> <ul style="list-style-type: none"> ➤ Create Event ➤ Manage Events ➤ Manage Venues ➤ Log Attendance ➤ Manage Guest Speakers

	<p>Step 3: The user clicks on the “Manage Events” list item.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Events webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon that is used to display notifications➤ Profile Picturebox➤ “User” drop down list navigation bar item <p>Body</p>
--	---	--	--

			<ul style="list-style-type: none">➤ “Welcome User Name” h1 header➤ “Search upcoming events by Name” textbox➤ “Here is your progress” button (If the user is a student)➤ “Sort with the following:” h4 header➤ “Event name” button➤ “Date” button➤ “Upcoming TRWLA Events” h2 header➤ “My Upcoming Events” h2 header➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:<ul style="list-style-type: none">• “Event Name” h3 header• “Summary” h4 header• “Date of the event” h4 header• “Time of event” h4 header• “Venue name” h4 header• “RSVP” button
--	--	--	---

			<ul style="list-style-type: none"> • “Details” button • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button (if the user is a volunteer) ➤ “Update” button (if the user is a volunteer) ➤ “Delete” button (if the user is a volunteer) ➤ “Return” button ➤ “Log Attendance” button (if the user is a volunteer)
	Step 5: The user enters search details relating to the specific event in the “Search” textbox.		Step 6: The system, using C# and JavaScript, displays the relevant event containers based on the text entered in the “Search” textbox.
	Step 7: The user clicks on the “Details” button of a specific events container.		<p>Step 8: The system, using JavaScript, CSS and HTML, displays the Event Details webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/“Volunteers” (if the user is a student) drop down list navigation bar item

		<ul style="list-style-type: none"> ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Event Name Details” h1 header ➤ “Name” (textbox editable if user is admin/volunteer) ➤ “Date” (textbox editable if user is admin/volunteer) ➤ “Time” (textbox editable if user is admin/volunteer) ➤ “Summary” (textbox editable if user is admin/volunteer) ➤ “Description” (textbox editable if user is admin/volunteer) ➤ “Reading Content” (if a community engagement or lecture event)
--	--	--

			<ul style="list-style-type: none"> ➤ “Guest Speaker” (if a function event) ➤ “Residence” (if dedicated to a specific residence only, such as for lectures) ➤ Picturebox for poster <p>Footer</p> <ul style="list-style-type: none"> ➤ “RSVP” button ➤ “Return” button ➤ “Delete” button (if user is admin/volunteer) ➤ “Update” button (if user is admin/volunteer) <p>All fields are populated from the respected “event” tables namely, Function, CommunityOutreach, Lecture and Event. This is done using Entity Framework and LINQ queries.</p>
	<p>Step 9: The user clicks the “Return” button.</p>		<p>Step 10: The system, using C#, JavaScript, CSS and HTML, displays the Events Webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/“Volunteers” (if the user is a

			<p>student) drop down list navigation bar item</p> <ul style="list-style-type: none"> ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Here is your progress” button (If the user is a student) ➤ “Sort with the following:” h4 header
--	--	--	---

			<ul style="list-style-type: none">➤ “Event name” button➤ “Date” button➤ “Upcoming TRWLA Events” h2 header➤ “My Upcoming Events” h2 header➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:<ul style="list-style-type: none">• “Event Name” h3 header• “Summary” h4 header• “Date of the event” h4 header• “Time of event” h4 header• “Venue name” h4 header• “RSVP” button• “Details” button• “Not going” button in the “My upcoming events” container.
		Footer	<ul style="list-style-type: none">➤ “Create” button (if the user is a volunteer)➤ “Update” button (if the user is a volunteer)

			<ul style="list-style-type: none"> ➤ “Delete” button (if the user is a volunteer) ➤ “Return” button ➤ “Log Attendance” button (if the user is a volunteer)
ALTERNATE COURSES:	<p>Alt Step 3: If the user is a student, the student will click on the “Home” hyperlink in the navigation bar.</p> <p>→ Go to Step 4</p> <p>On webpage: Student Main Menu</p>		
	<p>Alt Step 5: The user does not enter search details but rather double clicks on a specific event container.</p> <p>→ Go to Step 7</p> <p>On webpage: Events Webpage</p>		
	<p>Alt Step 6: The system could not locate an Event based on the search details, display a viewbag message: “Your search returned 0 results”.</p> <p>→ Go back to Step 5</p> <p>On webpage: Events Webpage</p>		
	<p>Alt Step 9a: The user clicks on the “RSVP” button.</p> <p>→ Invoke UC 5.5 RSVP to an Event</p> <p>On webpage: Event Details Webpage</p>		
	<p>Alt Step 9b: The volunteer clicks on the “Update button”.</p> <p>→ Invoke UC 5.6 Update Event Information</p> <p>On webpage: Event Details Webpage</p>		
	<p>Alt Step 9c: The volunteer clicks the “Delete” button.</p> <p>→ Invoke UC 5.7 Cancel Event</p>		
CONCLUSION:	<p>The system, using C#, JavaScript, CSS and HTML, displays the Events Webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox 		

	<ul style="list-style-type: none"> ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Here is your progress” button (If the user is a student) ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button ● “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button (if the user is a volunteer) ➤ “Update” button (if the user is a volunteer) ➤ “Delete” button (if the user is a volunteer) ➤ “Return” button ➤ “Log Attendance” button (if the user is a volunteer)
POST-CONDITION:	The user has successfully searched for an event.
BUSINESS RULES	<ul style="list-style-type: none"> ● Only a volunteer/admin member can update or delete/cancel an event. ● The event details cannot be editable if the user is a student.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> ● None
ASSUMPTIONS:	<ul style="list-style-type: none"> ● None
OPEN ISSUES:	None

USE CASE NAME:	RSVP to an Event		USE CASE TYPE
USE CASE ID:	5.5		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • Volunteer (ERA) 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Admin 		
DESCRIPTION:	<p>This use case describes the event of how a user RSVP's to an event on the system. The use case begins when the user has successfully logged in and their main menu is displayed. Looking through the upcoming events section, the user will see the timeline of TRWLA and the upcoming events that the organization has. The user will then either click the RSVP button in the event container or the user will double click on the event to view the details and then click on the RSVP button. Once the user has RSVP'd, the system will display a modal message to confirm that the user want move the event responded to, under the "My Events" h1 header which is displayed on the user's main menu. The system will then add the user's name to the RSVP Event table, in order to be retrieved when logging the event attendance. The use case concludes when the user successfully RSVP's to an event.</p>		
PRE-CONDITION:	The user must be logged into the system		
TRIGGER:	The user wants to RSVP to an event.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	Step 1: The user wants to RSVP to an event and views their main menu.		Step 2: The system, using C#, JavaScript, CSS and HTML, displays the user's main menu depending on their admin rights. The main menu has the following general layout: Header

			<p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox
--	--	--	---

			<ul style="list-style-type: none">➤ “Here is your progress” button (If the user is a student)➤ “Sort with the following:” h4 header➤ “Event name” button➤ “Date” button➤ “Upcoming TRWLA Events” h2 header➤ “My Upcoming Events” h2 header➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:<ul style="list-style-type: none">• “Event Name” h3 header• “Summary” h4 header• “Date of the event” h4 header• “Time of event” h4 header• “Venue name” h4 header• “RSVP” button• “Details” button• “Not going” button in the “My upcoming events” container.
--	--	--	---

			Footer <ul style="list-style-type: none"> ➤ “Create” button (if the user is a volunteer) ➤ “Update” button (if the user is a volunteer) ➤ “Delete” button (if the user is a volunteer) ➤ “Return” button ➤ “Log Attendance” button (if the user is a volunteer)
	<p>Step 3: The user clicks on the “RSVP” button.</p>		<p>Step 4: The system, using C# and JavaScript, displays a modal with the following message; “Are you sure you want to RSVP to this event?” with a “RSVP” and “Cancel” button.</p>
	<p>Step 5: The user clicks on the “RSVP” button.</p>		<p>Step 6: The system, using C#,SQL and JavaScript, displays a modal with the following message; “You have successfully RSVP’d for the following event: xxxxxxxx” (where xxxxxx is the name of the event) and a “Confirm” button and adds the user’s details to the RSVP_Event table using SQL, Entity Framework and LINQ queries.</p>
	<p>Step 7: The user clicks on the “Confirm” button.</p>		<p>Step 8: The system, using C# and JavaScript, updates the user’s main menu and places the event RSVP’d to,</p>

		underneath the “My Events” h1 header.
ALTERNATE COURSES:	<p>Alt Step 3a: The user clicks on the “Not Going” button. The system displays a modal with the following message: “Are you sure you do not want to attend this event?” with two buttons, namely, “Confirm” and “Cancel”.</p> <p>Alt Step 3b: The user clicks on the “Confirm” button. The system does not place the event underneath the “My Events” h1 header but rather leaves the event on the main menu in case the user changes their mind. On webpage: Student/ Volunteer/Admin Main Menu</p> <p>Alt Step 3c: The user decides to view the details of the event and clicks on the event container. The system, using C#, JavaScript, CSS and HTML, displays the Event Details Webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ “Members”/“Volunteers” drop down list ➤ “Events” drop down list ➤ “Reading Content” drop down list ➤ “Gallery” hyperlink button (if the user is admin) ➤ Message icon that displays notifications ➤ Profile picturebox ➤ “Users” drop down list <p>Body</p> <ul style="list-style-type: none"> ➤ “Event Name Details” h1 header ➤ “Name” (textbox editable if user is admin/volunteer) ➤ “Date” (textbox editable if user is admin/volunteer) ➤ “Time” (textbox editable if user is admin/volunteer) ➤ “Summary” (textbox editable if user is admin/volunteer) ➤ “Description” (textbox editable if user is admin/volunteer) ➤ “Reading Content” (if a community engagement or lecture event) ➤ “Guest Speaker” (if a function event) ➤ “Residence” (if dedicated to a specific residence only, such as for lectures) ➤ Picturebox for poster <p>Footer</p> <ul style="list-style-type: none"> ➤ “RSVP” button ➤ “Return” button ➤ “Delete” button (if user is admin/volunteer) ➤ “Update” button (if user is admin/volunteer) 	

	<p>All fields are populated from the respected “event” tables namely, Function, CommunityOutreach, Lecture and Event. This is done using Entity Framework and LINQ queries.</p> <p>→ Go to Step 4</p> <p>On Webpage: Event Details Webpage</p> <p>Alt Step 3e: The user searches for an event instead of viewing it from their main menu.</p> <p>→ Invoke UC 5.4 Search Event</p> <p>On Webpage: Student/Volunteer/Admin Main Menu Webpage</p>
	<p>Alt Step 5: The user clicks on the “Cancel” button. The system displays the main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Here is your progress” button (If the user is a student) ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button

	<ul style="list-style-type: none"> “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> “Create” button (if the user is a volunteer) “Update” button (if the user is a volunteer) “Delete” button (if the user is a volunteer) “Return” button “Log Attendance” button (if the user is a volunteer) <p>On webpage: Events Webpage</p>
CONCLUSION:	<ul style="list-style-type: none"> The system, using C# and JavaScript, updates the user’s main menu and places the event RSVP’d to, underneath the “My Events” h1 header.
POST-CONDITION:	The user has successfully RSVP’d to an upcoming event.
BUSINESS RULES	<ul style="list-style-type: none"> All users can RSVP to an upcoming event.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> If a user chooses not to attend an event, the system must not add the user to the <u>RSVP Event</u> table.
ASSUMPTIONS:	<ul style="list-style-type: none"> None
OPEN ISSUES:	None

USE CASE NAME:	Update Event Information		USE CASE TYPE
USE CASE ID:	5.6		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Admin members • Students 		
DESCRIPTION:	<p>This use case describes the event of a user updating the information for a current event on the system. The use case begins when the volunteer clicks on the “Events” drop down list on their main menu/ home page. The system will then display the Events Webpage with a list of current events that the user must still respond to as well as a list of events that the user has already responded to. The user will then either enter search details in the “Search” textbox or locate the event in the containers. Once the user has located the specific event, the user will click on the event to view more details. The system will then display the Events Details webpage with all the relevant information pertaining to the event. The volunteer will update the information accordingly and the system will update the event information on every user’s profile. The use case concludes when the user has successfully updated an event’s information.</p>		
PRE-CONDITION:	The volunteer must be logged into the system		
TRIGGER:	The volunteer wants to update an event’s information.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to update an event’s information on the system and clicks the “Events” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the drop down list items:</p> <ul style="list-style-type: none"> ➤ Create Event ➤ Manage Events ➤ Manage Venues ➤ Log Attendance

			<ul style="list-style-type: none"> ➤ Manage Guest Speakers
	<p>Step 3: The volunteer clicks on the “Manage Events” list item.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Events webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list

			<p>navigation bar item</p> <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> • “Event Name” h3 header • “Summary” h4 header • “Date of the event” h4 header • “Time of event” h4 header • “Venue name” h4 header • “RSVP” button • “Details” button
--	--	--	---

			<ul style="list-style-type: none"> • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
	<p>Step 5: The volunteer locates the event and clicks on the “Details” button of the specific events container.</p>		<p>Step 6: The system, using JavaScript, CSS and HTML, displays the Event Details webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ “Members” drop down list ➤ “Events” drop down list ➤ “Reading Content” drop down list ➤ “Gallery” hyperlink button ➤ Message icon that displays notifications ➤ Profile picturebox ➤ “Users” drop down list <p>Body</p> <ul style="list-style-type: none"> ➤ “Event Name Details” h1 header ➤ “Name” (textbox editable if user is admin/volunteer)

			<ul style="list-style-type: none"> ➤ “Date” (textbox editable if user is admin/volunteer) ➤ “Time” (textbox editable if user is admin/volunteer) ➤ “Summary” (textbox editable if user is admin/volunteer) ➤ “Description” (textbox editable if user is admin/volunteer) ➤ “Reading Content” (if a community engagement or lecture event) ➤ “Guest Speaker” (if a function event) ➤ “Residence” (if dedicated to a specific residence only, such as for lectures) ➤ Picturebox for poster <p>Footer</p> <ul style="list-style-type: none"> ➤ “RSVP” button ➤ “Return” button ➤ “Delete” button (if user is admin/volunteer) ➤ “Update” button (if user is admin/volunteer) <p>All fields are populated from the respected “event” tables namely, <u>Function,</u></p>
--	--	--	---

			CommunityOutreach, Lecture and Event. This is done using Entity Framework and LINQ queries.
	<p>Step 9: The volunteer updates the information and clicks the “Update” button.</p>		<p>Step 10: The system, using C# and JavaScript, validates that the information updated is in the following format:</p> <ul style="list-style-type: none"> • Name (Must be a string value of no more than 35 characters.) • Guest Speaker (Must be a value selected from the Guest Speaker dropdown list which is populated by the GuestSpeaker table), either the “Pending invitation” or “Accepted invitation” radio button must have been clicked. • Time (Must be a valid time format i.e HH:MM as clicked from the time picker) • Date (Must be a valid date in the format i.e. CCYY-MM-DD as clicked from the date picker) • Venue (Must be a value clicked)

			<p>from the Venue dropdown list which is populated by the Venue table)</p> <ul style="list-style-type: none"> • Summary (Must be a string value of no more than 100 characters) • Description (Must be a string value of no more than 300 characters) • Content (Must be an item clicked from the Content drop down list which is populated by the Content table)
			<p>Step 11: The system, using C# and JavaScript, displays a Modal with the following message: "Are you sure that you want to update this event?" with a "Confirm" and "Cancel" button.</p>
	<p>Step 12: The volunteer clicks on the "Confirm" button.</p>		<p>Step 13: The system, using C# and SQL, updates the CommunityOutreach, Function, Lecture and Event tables respectively, according to which event is updated, using Entity Framework and LINQ Queries.</p>
			<p>Step 14: The system, using C# and JavaScript, displays a Modal with</p>

			<p>the following message: “The event has been successfully updated.” with a “Confirm” button.</p>
	<p>Step 15: The volunteer clicks the “Confirm” button.</p>		<p>Step 16: The system, using JavaScript, CSS and HTML, displays the Events Webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox

			<ul style="list-style-type: none"> ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ +- “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> • “Event Name” h3 header • “Summary” h4 header • “Date of the event” h4 header • “Time of event” h4 header • “Venue name” h4 header
--	--	--	--

			<ul style="list-style-type: none"> • “RSVP” button • “Details” button • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➢ “Create” button ➢ “Update” button ➢ “Delete” button ➢ “Return” button ➢ “Log Attendance” button
ALTERNATE COURSES:	Alt Step 5: The volunteer enters search details in the “Search by name” textbox. → Invoke UC 5.4 Search Event On webpage: Events Webpage		
	Alt Step 9a: The volunteer clicks the “Return” button. The system displays the Events Webpage. → Go to Step 14 On webpage: Events Webpage		
	Alt Step 9b: The volunteer clicks on the “RSVP” button. → Invoke UC 5.5 RSVP to an Event On webpage: Event Details Webpage		
	Alt Step 9c: The volunteer clicks the “Delete” button. → Invoke UC 5.7 Cancel Event On webpage: Event Details Webpage		
	Alt Step 10: The information is incorrect and not in the correct format. The system, using C# and JavaScript, displays a Modal with the following message: “There is a problem with the information that you entered. Please enter the information in the correct format.” with a “Confirm” button. The system also shows viewbag error messages beneath the required textboxes, stating what the correct format is. → Go to Step 9 On webpage: Events Details Webpage		
	Alt Step 12: The volunteer clicks the “Cancel” button. The system displays the Events webpage. → Go to Step 14 On webpage: Events Webpage		

CONCLUSION:

The system, using C#, JavaScript, CSS and HTML, displays the Events Webpage with the following layout:

Header

Navigation Bar from left to right:

- TRWLA Management System h1 header
- “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item
- “Events”/ “Home” (if the user is a student) drop down list navigation bar item
- “Reading Content” drop down list navigation bar item
- “Gallery” navigation bar item
- Message icon that is used to display notifications
- Profile Picturebox
- “User” drop down list navigation bar item

Body

- “Welcome User Name” h1 header
- “Search upcoming events by Name” textbox
- “Sort with the following:” h4 header
- “Event name” button
- “Date” button
- “Upcoming TRWLA Events” h2 header
- “My Upcoming Events” h2 header
- Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:
 - “Event Name” h3 header
 - “Summary” h4 header
 - “Date of the event” h4 header
 - “Time of event” h4 header
 - “Venue name” h4 header
 - “RSVP” button
 - “Details” button
 - “Not going” button in the “My upcoming events” container.

Footer

- “Create” button
- “Update” button
- “Delete” button
- “Return” button
- “Log Attendance” button

POST-CONDITION:

The volunteer has successfully updated an event's information.

BUSINESS RULES

- Only a volunteer/admin member can update or delete/cancel an event.

	<ul style="list-style-type: none">• The event details cannot be editable if the user is a student.• Event Details on each User's profile must be updated accordingly.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	
ASSUMPTIONS:	<ul style="list-style-type: none">• None
OPEN ISSUES:	None

USE CASE NAME:	Cancel Event		USE CASE TYPE
USE CASE ID:	5.7		Business Requirements: <input type="checkbox"/>
PRIORITY:	Low		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Admin members • Students 		
DESCRIPTION:	<p>This use case describes the event of a volunteer cancelling an event for a specific reason. The use case begins when the volunteer clicks on the “Events” drop down list on their main menu/ home page. The system will then display the Events Webpage with a list of current events that the user must still respond to as well as a list of events that the user has already responded to. The user will then either enter search details in the “Search” textbox or locate the event in the containers. Once the user has located the specific event, the user will click on the event to view more details. The system will then display the Events Details webpage with all the relevant information pertaining to the event. The volunteer will then click on the “Cancel Event” button and the system will delete the event from the relevant tables and the members that have already RSVP’d will be notified of the cancellation. The use case concludes when the user has successfully cancelled an event on the system.</p>		
PRE-CONDITION:	The volunteer must be logged into the system		
TRIGGER:	The volunteer wants to cancel an event.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer wants to cancel an event on the system and clicks the “Events” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the drop down list items:</p> <ul style="list-style-type: none"> ➤ Create Event ➤ Manage Events ➤ Manage Venues ➤ Log Attendance

			<ul style="list-style-type: none"> ➤ Manage Guest Speakers
	<p>Step 3: The volunteer clicks on the “Manage Events” list item.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Events webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list

			<p>navigation bar item</p> <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> • “Event Name” h3 header • “Summary” h4 header • “Date of the event” h4 header • “Time of event” h4 header • “Venue name” h4 header • “RSVP” button • “Details” button
--	--	--	---

			<ul style="list-style-type: none"> • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
	<p>Step 5: The volunteer locates the event and clicks on the “Details” button of the specific events container.</p>		<p>Step 6: The system, using JavaScript, CSS and HTML, displays the Event Details webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Event Name Details” h1 header

			<ul style="list-style-type: none"> ➤ “Name” (textbox editable if user is admin/volunteer) ➤ “Date” (textbox editable if user is admin/volunteer) ➤ “Time” (textbox editable if user is admin/volunteer) ➤ “Summary” (textbox editable if user is admin/volunteer) ➤ “Description” (textbox editable if user is admin/volunteer) ➤ “Reading Content” (if a community engagement or lecture event) ➤ “Guest Speaker” (if a function event) ➤ “Residence” (if dedicated to a specific residence only, such as for lectures) ➤ Picturebox for poster <p>Footer</p> <ul style="list-style-type: none"> ➤ “RSVP” button ➤ “Return” button ➤ “Cancel Event” button (if user is admin/volunteer) ➤ “Update” button (if user is admin/volunteer)
--	--	--	---

			All fields are populated from the respected “event” tables namely, <u>Function</u> , <u>CommunityOutreach</u> , <u>Lecture</u> and <u>Event</u> . This is done using Entity Framework and LINQ queries.
	Step 9: The volunteer clicks on the “Cancel Event” button.		Step 10: The system, using C# and JavaScript, displays a Modal with the following message: “Are you sure that you want to cancel this event? Note: All users who have RSVP’d to the event will be notified of the cancellation.” with a “Confirm” and “Cancel” button.
	Step 11: The volunteer clicks the “Confirm” button.		Step 12: The system, using C# and SQL, deletes the event from the <u>CommunityOutreach</u> , <u>Function</u> , <u>Lecture</u> and <u>Event</u> tables respectively, according to which event is cancelled, using Entity Framework and LINQ Queries.
			Step 13: The system, using C# and JavaScript, displays a Modal with the following message: “The event was successfully cancelled” with a “Confirm” button.

	Step 14: The volunteer clicks on the “Confirm” button.		Step 15: The system, using C#, notifies all members that the event has been cancelled. Invoke UC 5.9 Send Notification
			<p>Step 16: The system, using JavaScript, CSS and HTML, displays the Events Webpage with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications

			<ul style="list-style-type: none">➤ Profile Picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Welcome User Name” h1 header➤ “Search upcoming events by Name” textbox➤ “Sort with the following:” h4 header➤ “Event name” button➤ “Date” button➤ “Upcoming TRWLA Events” h2 header➤ “My Upcoming Events” h2 header➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries:<ul style="list-style-type: none">● “Event Name” h3 header● “Summary” h4 header● “Date of the event” h4 header● “Time of event” h4 header
--	--	--	--

			<ul style="list-style-type: none"> • “Venue name” h4 header • “RSVP” button • “Details” button • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
ALTERNATE COURSES:	<p>Alt Step 5a: The volunteer enters search details in the “Search by name” textbox.</p> <p>→ Invoke UC 5.4 Search Event</p> <p>On webpage: Events Webpage</p>		
	<p>Alt Step 5b: The volunteer locates the event and clicks the “Delete” button on the Events Webpage.</p> <p>→ Go to Step 10</p> <p>On webpage: Events Webpage</p>		
	<p>Alt Step 9a: The volunteer clicks the “Return” button. The system displays the Events Webpage.</p> <p>→ Go to Step 16</p> <p>On webpage: Events Webpage</p>		
	<p>Alt Step 9b: The volunteer clicks on the “RSVP” button.</p> <p>→ Invoke UC 5.5 RSVP to an Event</p> <p>On webpage: Event Details Webpage</p>		
	<p>Alt Step 9c: The volunteer clicks the “Update” button.</p> <p>→ Invoke UC 5.6 Update Event Information</p> <p>On webpage: Event Details Webpage</p>		
	<p>Alt Step 11: The volunteer clicks the “Cancel” button. The system displays the Events webpage.</p> <p>→ Go to Step 16</p> <p>On webpage: Events Webpage</p>		
CONCLUSION:	<p>The system, using C#, JavaScript, CSS and HTML, displays the Events Webpage with the following layout:</p> <p>Header</p>		

	<p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button ● “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
POST-CONDITION:	The volunteer has successfully cancelled an event.
BUSINESS RULES	<ul style="list-style-type: none"> ● Only a volunteer/admin member can update or delete/cancel an event. ● The event details cannot be editable if the user is a student. ● An event cannot be cancelled within 24 hours before it is supposed to take place.

IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none">• Event Details on each User's profile must be updated accordingly.• All members who have already RSVP'd must receive a notification that the event has been cancelled.
ASSUMPTIONS:	<ul style="list-style-type: none">• The event is cancelled for a valid reason.
OPEN ISSUES:	None

2.3.6 Guest Speaker Narratives

USE CASE NAME:	Register Guest Speaker		USE CASE TYPE
USE CASE ID:	6.1	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design:	<input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Admin 		
DESCRIPTION:	<p>This use case describes the event of a volunteer wanting to register a guest speaker to the system. The volunteer navigates to the Guest Speaker webpage under the Events navigation bar item. The volunteer clicks “Register” and the system will prompt volunteer to enter the required information. Volunteer will enter the information and click “Save” then the system will validate the information and save it to the Guest speaker table. This use case concludes by a new guest speaker being registered to the system.</p>		
PRE-CONDITION:	The volunteer must be logged into the system.		
TRIGGER:	The volunteer wants to add a guest speaker to the system.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1: The volunteer clicks the “Guest Speakers” dropdown list item under the “Events” navigation bar menu item. Events > Guest Speakers</p>		<p>Step 2: The system displays the Guest Speaker Webpage. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System

			<ul style="list-style-type: none"> ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues' details populated by the <u>Guest Speaker</u> table. The list contains the following information about the venues as retrieved from the <u>Guest Speaker</u> table: <ul style="list-style-type: none"> • FirstName • LastName • Phone • EmailAddress • PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the</p>
--	--	--	--

			<p>container is a textbox labelled “Search for a guest speaker by first name”. Below to the right of the container are four buttons labelled “Register Guest Speaker”, “Update”, “Delete” and Return. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: The volunteer clicks the “Register Guest Speaker” button. On webpage: Guest speaker</p>		<p>Step 4: The system displays Screen: 6.1.1 Register Guest Speaker VA.</p> <p>This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User

			<p>dropdown list items.</p> <p>The body contains the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Register Guest Speaker”. ➤ A container below the header containing a “Name” (textbox) “Surname” (textbox) “Phone Number” (textbox) “Email Address” (textbox) “Picture URL” (textbox) ➤ Below the container are two buttons entitled “Create” and “Return” <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 5: The volunteer enters the required information into the textboxes and clicks the “Create” button. On webpage: Register Guest Speaker</p>		<p>Step 6: The system validates that the textboxes against the following requirements:</p> <ul style="list-style-type: none"> • Speaker_Name (35 characters) • Speaker_Surname (35 characters) • Phone Number(integer) • Email (255 characters)

			<ul style="list-style-type: none"> • PictureLink (100 characters)
			<p>Step 7: The system adds the following information to the Guest Speaker table:</p> <ul style="list-style-type: none"> • Speaker_Name • Speaker_Surname • Phone Number • Email • PictureLink
			<p>Step 8: The system displays a modal with the following message, “You have successfully created a new guest speaker: (Speaker_Name and Speaker_Surname from Guest speaker table)” with a button below the message entitled “Confirm”.</p>
	<p>Step 8: The volunteer clicks the “Confirm” button.</p>		<p>Step 9: The system displays the Guest Speaker Webpage. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content

			<ul style="list-style-type: none"> ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues' details populated by the <u>Guest Speaker</u> table. The list contains the following information about the venues as retrieved from the <u>Guest Speaker</u> table: <ul style="list-style-type: none"> • FirstName • LastName • Phone • EmailAddress • PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the container is a textbox labelled "Search for a</p>
--	--	--	--

			guest speaker by first name". Below to the right of the container are four buttons labelled "Register Guest Speaker", "Update", "Delete" and Return. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.
ALTERNATE COURSES:	Alt Step 6: The information entered is not valid. The system displays viewbag error messages for the appropriate fields. → Go to Step 5		
CONCLUSION:	A volunteer has successfully registered a new guest speaker to the system.		
POST-CONDITION:	Guest Speaker details have been added to the <u>Guest Speaker</u> table.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised volunteers can add a guest speaker 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Search Guest Speaker		USE CASE TYPE
USE CASE ID:	6.2	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Student		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a student wanting to search for a specific guest speaker. Student navigates to the Guest Speaker webpage under the Events navigation bar menu item. The system displays a container containing a list of guest speakers, the student uses the search bar functionality to search for a specific guest speaker. The system displays the search results. The use case concludes by the student viewing the specific guest speaker information.</p>		
PRE-CONDITION:	The student is logged into the system.		
TRIGGER:	A student wants to search for a guest speaker.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The student clicks the “Guest Speakers” dropdown list item under the “Events” navigation bar menu item. Events > Guest Speakers On webpage: Main Menu</p>		<p>Step 2: The system displays the Guest Speaker Webpage. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Volunteers ➤ Events

			<ul style="list-style-type: none">➤ Reading content➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the Student dropdown list items. <p>The body the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the guest speakers' details populated by the <u>Guest Speaker</u> table. The list contains the following information about the guest speakers as retrieved from the <u>Guest Speaker</u> table: <ul style="list-style-type: none">• FirstName• LastName• Phone• EmailAddress• PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the container is a textbox</p>
--	--	--	---

			<p>labelled “Search for a guest speaker by first name”. Below to the right of the container are four buttons labelled “Register Guest Speaker”, “Update”, “Delete” and Return. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: The student enters information into the textbox above the container clicks on the “Search for a guest speaker by first name” button. On webpage: Guest Speaker</p>		<p>Step 4: The system performs a “fuzzy” search and displays the Guest Speaker Webpage according to the relevant search results. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the student’s notifications. ➤ A profile picture which contains a link to the user’s

			<p>dropdown list items.</p> <p>The body the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the guest speakers' details populated by the <u>Guest Speaker</u> table. <p>The list contains the following information about the guest speakers as retrieved from the <u>Guest Speaker</u> table:</p> <ul style="list-style-type: none">• FirstName• LastName• Phone• EmailAddress• PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the container is a textbox labelled "Search for a guest speaker by first name".</p> <p>Below to the right of the container are four buttons labelled "Register Guest Speaker", "Update", "Delete" and Return.</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
--	--	--	---

	<p>Step 5: The student finds the relevant speaker and clicks on the hyperlink labelled “details”. On webpage: Guest Speaker</p>	<p>Step 6: The system displays the Guest Speaker Details webpage. This webpage contains a header, body and footer. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Home ➤ Volunteers ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the Student dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Guest Speaker Details”. ➤ A container below the header containing a
--	--	--

			<p>“Name” (noneditable textbox) “Surname” (noneditable textbox) “Phone Number” (noneditable textbox) “Email Address” (noneditable textbox) “Picture URL” (noneditable textbox)</p> <ul style="list-style-type: none"> ➤ Below the container is a button entitled “Return” <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 7: The student looks at the details of the guest speaker and clicks the “Return” button. On webpage: Guest Speaker Details</p>		<p>Step 8: The system displays the Guest Speaker Webpage. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Volunteers ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the student’s notifications.

			<p>➤ A profile picture which contains a link to the Student dropdown list items.</p> <p>The body the following elements:</p> <p>➤ A container containing a list of all the guest speakers' details populated by the <u>Guest Speaker</u> table. The list contains the following information about the guest speakers as retrieved from the <u>Guest Speaker</u> table:</p> <ul style="list-style-type: none">• FirstName• LastName• Phone• EmailAddress• PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the container is a textbox labelled "Search for a guest speaker by first name". Below to the right of the container is one button labelled "Return". Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
--	--	--	--

ALTERNATE COURSES:

Alt Step 2: The PBA (Primary Business Actor) is a Volunteer or Admin, the system will display the following:

The system displays the Guest Speaker Webpage. This webpage contains a header and body.

The header contains a navigation bar with the following menu items from left to right respectively:

- TRWLA Management System
- Members
- Events
- Reading content
- A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.
- A profile picture which contains a link to the User dropdown list items.

The body the following elements:

- A container containing a list of all the venues' details populated by the **Guest Speaker** table. The list contains the following information about the venues as retrieved from the **Guest Speaker** table:
 - FirstName
 - LastName
 - Phone
 - EmailAddress
 - PictureLink

Each list item contains a button to the right of the item entitled "Details".

Above the container is a textbox labelled "Search for a guest speaker by first name".

Below to the right of the container are four buttons labelled "Register Guest Speaker", "Update", "Delete" and Return.

Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.

Alt Step 6: The PBA (Primary Business Actor) is a Volunteer or Admin, the system will display the following:

The system displays the Guest Speaker Details webpage.

This webpage contains a header, body and footer.

The header contains a navigation bar with the following menu items from left to right respectively:

- TRWLA Management System
- Members
- Events
- Reading content
- A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.

	<ul style="list-style-type: none"> ➤ A profile picture which contains a link to the Student dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled “Guest Speaker Details”. ➤ A container below the header containing a “Name” (textbox) “Surname” (textbox) “Phone Number” (textbox) “Email Address” (textbox) “Picture URL” (textbox) ➤ Below the container are three buttons entitled “Update”, “Delete” and “Return” <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Alt Step 8: The PBA (Primary Business Actor) is a Volunteer or Admin, the system will display the following:</p> <p>The system displays the Guest Speaker Webpage. This webpage contains a header and body.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user’s notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues’ details populated by the Guest Speaker table. The list contains the following information about the venues as retrieved from the Guest Speaker table: • FirstName • LastName • Phone • EmailAddress • PictureLink <p>Each list item contains a button to the right of the item entitled “Details”. Above the container is a textbox labelled “Search for a guest speaker by first name”.</p> <p>Below to the right of the container are four buttons labelled “Register Guest Speaker”, “Update”, “Delete” and Return.</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>

CONCLUSION:	A student has successfully searched for a guest speaker.
POST-CONDITION:	A Guest Speaker's information from the Guest Speaker table is retrieved.
BUSINESS RULES	<ul style="list-style-type: none">• All registered students can search for a guest speaker.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none">• None
ASSUMPTIONS:	<ul style="list-style-type: none">• None

OPEN ISSUES: None

USE CASE NAME:	Update Guest Speaker	USE CASE TYPE	
USE CASE ID:	6.3	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Medium	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of volunteer updating a guest speaker. The use case begins by volunteer navigating to the Guest Speaker webpage under the Events navigation bar menu item. The system displays a container with a list of all the guest speakers, which volunteer will search and click the “Update” button for the relevant list item. The system will display the list item in an editable format, which volunteer can update and save. The guest speaker details are updated in the Guest Speaker table. This use case concludes by a guest speaker being updated on the system.</p>		
PRE-CONDITION:	The volunteer is logged into the system.		
TRIGGER:	The volunteer wants to update a guest speaker’s details.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer clicks the “Guest Speakers” dropdown list item under the “Events” navigation bar menu item. Events > Guest Speakers On webpage: Main Menu Volunteer</p>		<p>Step 2: The system displays the Guest Speaker Webpage. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System

			<ul style="list-style-type: none"> ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues' details populated by the <u>Guest Speaker</u> table. The list contains the following information about the venues as retrieved from the <u>Guest Speaker</u> table: <ul style="list-style-type: none"> • FirstName • LastName • Phone • EmailAddress • PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the</p>
--	--	--	--

			<p>container is a textbox labelled “Search for a guest speaker by first name”. Below to the right of the container are four buttons labelled “Register Guest Speaker”, “Update”, “Delete” and Return. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: The volunteer enters information into the textbox above the container clicks on the “Search for a guest speaker by first name” button. On webpage: Guest Speaker</p>		<p>Step 4: The system performs a “fuzzy” search and displays the Guest Speaker Webpage according to the relevant search results. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the student’s notifications. ➤ A profile picture which contains a

			<p>link to the user's dropdown list items.</p> <p>The body the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the guest speakers' details populated by the <u>Guest Speaker</u> table. <p>The list contains the following information about the guest speakers as retrieved from the <u>Guest Speaker</u> table:</p> <ul style="list-style-type: none">• FirstName• LastName• Phone• EmailAddress• PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the container is a textbox labelled "Search for a guest speaker by first name".</p> <p>Below to the right of the container are four buttons labelled "Register Guest Speaker", "Update", "Delete" and Return.</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
--	--	--	--

	<p>Step 5: The volunteer finds the relevant speaker and clicks on the button labelled "update".</p> <p>On webpage: Guest Speaker</p>	<p>Step 6:</p> <p>The system displays the Guest Speaker Details webpage.</p> <p>This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading content➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the Student dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Guest Speaker Details".➤ A container below the header containing a
--	---	--

			<p>“Name” (textbox) “Surname” (textbox) “Phone Number” (textbox) “Email Address” (textbox) “Picture URL” (textbox)</p> <p>➤ Below the container are three buttons entitled “Update”, “Delete” and “Return”</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 7: The volunteer updates the guest speaker details and clicks the “Update” button. On webpage: Guest Speaker details</p>		<p>Step 8: The system displays a modal with header entitled “Update (Speaker_Name)’s information” and a paragraph below with the following text “Are you sure that you want to update(Speaker_Name)’s information?” with two buttons below to the bottom right labelled “Confirm” and “Cancel”. Using CSS,HTML,JavaScript and C#.</p>
	<p>Step 9: The volunteer clicks the “Confirm” button.</p>		<p>Step 10: The system validates that the textboxes against the following requirements:</p> <ul style="list-style-type: none"> • Speaker_Name (35 characters)

			<ul style="list-style-type: none"> • Speaker_Surname (35 characters) • Phone Number(integer) • Email (255 characters) • PictureLink (100 characters) <p>Using C#.</p>
			<p>Step 11: The system updates the following information according to the changes made to the Guest Speaker table:</p> <ul style="list-style-type: none"> • Speaker_Name • Speaker_Surname • Phone Number • Email • PictureLink <p>Using C# and SQL.</p>
			<p>Step 12:</p> <p>The system displays the Guest Speaker Details webpage.</p> <p>This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of

			<p>notifications, which contains the link to the user's notifications.</p> <ul style="list-style-type: none"> ➤ A profile picture which contains a link to the Student dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none"> ➤ A header entitled "Guest Speaker Details". ➤ A container below the header containing a "Name" (textbox) "Surname" (textbox) "Phone Number" (textbox) "Email Address" (textbox) "Picture URL" (textbox) ➤ Below the container are three buttons entitled "Update", "Delete" and "Return" <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
ALTERNATE COURSES:	Alt Step 9: The volunteer clicks the "Cancel" button. Go to step 7		

	Alt Step 10: The information entered is not valid. The system displays viewbag error messages for the appropriate fields. Go to Step 7
CONCLUSION:	A guest speaker's details have been successfully updated.
POST-CONDITION:	The Guest Speaker table has been updated.
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised volunteers can update a guest speaker
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Delete Guest Speaker		USE CASE TYPE
USE CASE ID:	6.4	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Low	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of volunteer deleting a guest speaker. The use case begins by volunteer navigating to the Guest Speaker webpage under the Events navigation bar item. The system displays a container with a list of all the guest speakers, which volunteer will search and click the “Delete” button for the relevant list item. The system confirms if volunteer wants to delete that guest speaker. The guest speaker is deleted from the Guest Speaker table. This use case concludes by a guest speaker being deleted from the system.</p>		
PRE-CONDITION:	The volunteer must be logged into the system.		
TRIGGER:	The volunteer wants to delete a guest speaker from the system.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The volunteer clicks the “Guest Speakers” dropdown list item under the “Events” navigation bar menu item. Events > Guest Speakers On webpage: Main Menu Volunteer</p>		<p>Step 2: The system displays the Guest Speaker Webpage. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members

			<ul style="list-style-type: none"> ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues' details populated by the <u>Guest Speaker</u> table. The list contains the following information about the venues as retrieved from the <u>Guest Speaker</u> table: <ul style="list-style-type: none"> • FirstName • LastName • Phone • EmailAddress • PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the container is a textbox</p>
--	--	--	--

			<p>labelled “Search for a guest speaker by first name”. Below to the right of the container are four buttons labelled “Register Guest Speaker”, “Update”, “Delete” and Return. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 3: The volunteer enters information into the textbox above the container clicks on the “Search for a guest speaker by first name” button. On webpage: Guest Speaker</p>		<p>Step 4: The system performs a “fuzzy” search and displays the Guest Speaker Webpage according to the relevant search results. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the student’s notifications. ➤ A profile picture which contains a link to the user’s

			<p>dropdown list items.</p> <p>The body the following elements:</p> <ul style="list-style-type: none">➤ A container containing a list of all the guest speakers' details populated by the <u>Guest Speaker</u> table. <p>The list contains the following information about the guest speakers as retrieved from the <u>Guest Speaker</u> table:</p> <ul style="list-style-type: none">• FirstName• LastName• Phone• EmailAddress• PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the container is a textbox labelled "Search for a guest speaker by first name".</p> <p>Below to the right of the container are four buttons labelled "Register Guest Speaker", "Update", "Delete" and Return.</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
--	--	--	---

	<p>Step 5: The volunteer finds the relevant speaker and clicks on the button labelled "details".</p> <p>On webpage: Guest Speaker</p>	<p>Step 6:</p> <p>The system displays the Guest Speaker Details webpage.</p> <p>This webpage contains a header, body and footer.</p> <p>The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading content➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications.➤ A profile picture which contains a link to the Student dropdown list items. <p>The body contains the following elements:</p> <ul style="list-style-type: none">➤ A header entitled "Guest Speaker Details".➤ A container below the header containing a
--	--	--

			<p>“Name” (textbox) “Surname” (textbox) “Phone Number” (textbox) “Email Address” (textbox) “Picture URL” (textbox)</p> <p>➤ Below the container are three buttons entitled “Update”, “Delete” and “Return”</p> <p>Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
	<p>Step 7: The volunteer clicks the “Delete” button. On webpage: Guest Speaker details</p>		<p>Step 8: The system displays a modal with header entitled “Delete (Speaker_Name)’s information” and a paragraph below with the following text “Are you sure that you want to delete (Speaker_Name)’s information?” with two buttons below to the bottom right labelled “Confirm” and “Cancel”. Using CSS, HTML, JavaScript and C#.</p>
	<p>Step 9: The volunteer clicks the “Confirm” button.</p>		<p>Step 11: The system deletes the following information from the Guest Speaker table:</p> <ul style="list-style-type: none"> • Speaker_Name Speaker_Surname • Phone Number

			<ul style="list-style-type: none"> • Email • PictureLink <p>Using C# and SQL.</p>
			<p>Step 2: The system displays the Guest Speaker Webpage. This webpage contains a header and body. The header contains a navigation bar with the following menu items from left to right respectively:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ A message icon which implies the metaphor of notifications, which contains the link to the user's notifications. ➤ A profile picture which contains a link to the User dropdown list items. <p>The body the following elements:</p> <ul style="list-style-type: none"> ➤ A container containing a list of all the venues' details populated by the Guest Speaker table. <p>The list contains</p>

			<p>the following information about the venues as retrieved from the <u>Guest Speaker</u> table:</p> <ul style="list-style-type: none"> • FirstName • LastName • Phone • EmailAddress • PictureLink <p>Each list item contains a button to the right of the item entitled "Details". Above the container is a textbox labelled "Search for a guest speaker by first name". Below to the right of the container are four buttons labelled "Register Guest Speaker", "Update", "Delete" and Return. Using C#, CSS, HTML, JavaScript, SQL and Entity Frameworks.</p>
ALTERNATE COURSES:	Alt Step 5: The volunteer scrolls through the list and finds the relevant speaker and clicks on the button labelled "Delete". On webpage: Guest Speaker → Go to Step 8		
	Alt Step 7: The volunteer clicks the "Cancel" button. → Go to step 5		
CONCLUSION:	A guest speaker is successfully deleted from the system.		
POST-CONDITION:	Guest Speaker information is removed from the <u>Guest Speaker</u> table		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised volunteers can delete a guest speaker 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		

ASSUMPTIONS:	• None
OPEN ISSUES:	None

2.3.7 Content Narratives

USE CASE NAME:	Upload Content	USE CASE TYPE
USE CASE ID:	7.1	Business Requirements: <input type="checkbox"/>
PRIORITY:	High	System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Volunteer (PBA)	
PRIMARY SYSTEM ACTOR	None	
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • Student (ERA) 	
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 	
DESCRIPTION:	<p>This use case describes the event of a volunteer uploading content. The use case begins when the volunteer wants to upload content and clicks on the reading content dropdown menu. The volunteer will click on the content item in the navigation bar and then click on the “Manage Content” dropdown list item. The volunteer will select the option to upload new lecture content and the system will prompt the volunteer to enter the required information. Once the volunteer has added all the required information the system will verify that the information entered is valid and complete and will add the content to the database. The use case concludes when the system confirms that the content has been added.</p>	
PRE-CONDITION:	The volunteer must be logged into the system.	
TRIGGER:	The volunteer wants to upload lecture content and clicks the reading content dropdown list.	
TYPICAL COURSE OF EVENTS:	Actor Action Step 1: The volunteer wants to upload content and clicks the reading content dropdown list in the navigation bar.	System Response Manual Action Automated Action Step 2: Using JavaScript the system displays the reading content dropdown list which contains the following items: <ul style="list-style-type: none"> • Content • E-Library

	<p>Step 3: The volunteer clicks on the “Content” dropdown list item.</p>	<p>Step 4: Using HTML, CSS, JavaScript and C# the system displays Webpage 7.0.1 VA which contains the following:</p> <p>Header:</p> <p>The header contains a navigation bar which contains the following items from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members Drop Down List➤ Events Drop Down List➤ Reading Content Drop Down List➤ Gallery Navigation bar item➤ Message icon which displays notifications➤ Profile picturebox➤ User drop down list <p>Body:</p> <p>The body contains an h1 header labelled “Content Management” to the top left of the body. Beneath the header is a textbox button below which is a button labelled “Search for a Content by Name”. Beneath this is a container with a list of all the current existing content with the following attributes as retrieved from the Content table from left to right:</p> <ul style="list-style-type: none">• ContentType
--	---	---

			<ul style="list-style-type: none"> • Name • Author • DatePublished • Link • Theme • Status • Description • A “Details” hyperlink <p>Below the container to the right of the body are four buttons labelled “Upload Content”, “Update”, “Delete” and “Return”</p> <p>Footer: Nothing On Webpage 7.0.1 VA</p>
	<p>Step 5: The volunteer clicks the “Upload Content” button.</p>		<p>Step 6: Using HTML, CSS, JavaScript and C# the system displays Webpage 7.1.1 VA which contains the following:</p> <p>Header: The header contains a navigation bar which contains the following items from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members Drop Down List ➤ Events Drop Down List ➤ Reading Content Drop Down List ➤ Gallery Navigation bar item ➤ Message icon which displays notifications ➤ Profile picturebox ➤ User drop down list <p>Body:</p>

			<p>The body contains fields for the required content information, which is to the left of the body, as follows:</p> <ul style="list-style-type: none"> ➤ “Create Content” h1 header ➤ Content Type Label with a content type textbox to its right ➤ Name Label with a name textbox to its right ➤ Author Label with a author textbox to its right ➤ Date Published Label with a date published date picker to its right ➤ Content Link Label with a content link textbox to its right ➤ Theme Label with a theme textbox to its right ➤ Status Label with an on/off toggle button to its right ➤ Description Label with a description textbox to its right <p>Below this to the right of the body are two buttons labelled “Create” and “Return”</p> <p>On Webpage 7.1.1 VA</p>
	<p>Step 7: The volunteer enters all of the required information and clicks the “Create” button</p>		<p>Step 8: Using C# and SQL the system verifies whether the information entered by the volunteer</p>

			<p>is complete and valid, such that:</p> <ul style="list-style-type: none"> • Content Type must be a string • Name must be a string up to 35 character long. • Author must be a string up to 70 characters long • Date Published must be in the valid date format - 10 characters in CCYY-MM-DD format • Content Link may be no longer than 100 characters • Theme must be a string no longer than 35 characters. • Status either locked or unlocked. • Description must be a string.
			<p>Step 9: Using C# and SQL the system adds the data to the <u>Content</u> table in the database.</p>
			<p>Step 10: Using JavaScript the system displays a modal with the following message “You have successfully uploaded ‘Content Type’ content.</p>
ALTERNATE COURSES:	<p>Alt Step 9: The information entered by the volunteer is not valid. Using C# and JavaScript the system displays error messages below the invalid fields. On webpage 7.1.1 VA Error → Go to Step 7</p>		
CONCLUSION:	<p>The volunteer has uploaded a new content. The system has verified that the information entered by the volunteer is complete and valid and has</p>		

	inserted the data into the Content table in the database. The use case concluded when the system displayed a Modal confirming that the content has been successfully uploaded.
POST-CONDITION:	New content has been uploaded to the database.
BUSINESS RULES	<ul style="list-style-type: none"> • Only enrolled students at TRWLA may have access to content.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Search Content	USE CASE TYPE	
USE CASE ID:	7.2	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User: Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a user searching for content on the system. The use case begins when the user wants to search for content and clicks on the “Reading Content” navigation bar item. The user will view all the available content and search until they find the particular content they are looking for. The use case concludes when the user has successfully viewed the relevant content details.</p>		
PRE-CONDITION:	The user must be logged on to the TRWLA system.		
TRIGGER:	The user wants to search for content and clicks the “Reading Content” navigation bar item.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The user wants to search for content and clicks the “Reading Content” navigation bar item.</p>		<p>Step 2: Using JavaScript the system displays the reading content dropdown list which contains the following items:</p> <ul style="list-style-type: none"> • Content • E-Library
	<p>Step 3: The volunteer clicks on the “Content” dropdown list item.</p>		<p>Step 4: Using HTML, CSS, JavaScript and C# the system displays Webpage 7.0.1 VA which contains the following:</p> <p>Header:</p>

		<p>The header contains a navigation bar which contains the following items from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members Drop Down List➤ Events Drop Down List➤ Reading Content Drop Down List➤ Gallery Navigation bar item➤ Message icon which displays notifications➤ Profile picturebox➤ User drop down list <p>Body:</p> <p>The body contains an h1 header labelled “Content Management” to the top left of the body. Beneath the header is a textbox button below which is a button labelled “Search for a Content by Name”. Beneath this is a container with a list of all the current existing content with the following attributes as retrieved from the Content table from left to right:</p> <ul style="list-style-type: none">• ContentType• Name• Author• DatePublished• Link• Theme• Status	
--	--	---	--

		<ul style="list-style-type: none"> • Description • A “Details” hyperlink <p>Below the container to the right of the body are four buttons labelled “Upload Content”, “Update”, “Delete” and “Return”</p> <p>Footer: Nothing On Webpage 7.0.1 VA</p>
	<p>Step 5: The user scrolls through the content in the content container and clicks on the relevant content they are looking for.</p>	<p>Step 6: Using HTML, CSS, JavaScript and C# the system displays Webpage 7.2.2 VA which contains the following:</p> <p>Header: The header contains a navigation bar which contains the following items from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members Drop Down List ➤ Events Drop Down List ➤ Reading Content Drop Down List ➤ Gallery Navigation bar item ➤ Message icon which displays notifications ➤ Profile picturebox ➤ User drop down list <p>Body: The body contains fields for the current content information, which is to the left of the body, as follows:</p>

			<ul style="list-style-type: none"> ➤ Content Type Label with a content type textbox to its right ➤ Name Label with a name textbox to its right ➤ Author Label with a author textbox to its right ➤ Date Published Label with a date published date picker to its right ➤ Content Link Label with a content link textbox to its right ➤ Theme Label with a theme textbox to its right ➤ Status Label with an on/off switch to its right ➤ Description Label with a description textbox to its right <p>Below this to the right of the body are three buttons labelled "Update", "Delete" and "Return". Each of the above fields are populated with the data of the selected content from the Content table using LINQ queries</p>
ALTERNATE COURSES:			<p>Alt Step 5a: The user types a name into the search textbox and clicks on the "Search by Name" button. The system uses C# and LINQ to run through the Content table in the database and searches for content that has a name that matches the text entered into the textbox. The system then uses JavaScript to display a new list in the content container with only content that has the same name entered into the textbox.</p> <p>On Webpage 7.2.1 VA</p>

	→ Go to Step 6
	Alt Step 5b: The content the user is looking for does not exist in the system or is not available to them. The system displays a Modal using JavaScript with the message “Search Content Error There is no content that matches your search inputs” with a cancel button. On webpage 7.2.2 VA Error
CONCLUSION:	The user has searched for specific content on the TRWLA system. The system has displayed all the relevant details of the searched content and the user has successfully been able to view the content details.
POST-CONDITION:	Content has been searched by a user.
BUSINESS RULES	<ul style="list-style-type: none"> • Only content that is unlocked may be displayed to student but volunteers can view all content on the system.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Update Content	USE CASE TYPE	
USE CASE ID:	7.3	Business Requirements: <input type="checkbox"/>	
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • Student (ERA) 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a volunteer updating content information. The use case begins when the volunteer wants to update the content information and clicks the “Reading Content” navigation bar item. The volunteer will search for the relevant content, update its details and click the update button. The system will verify whether the updated information is complete and accurate and will then update the Content table in the database. The use case concludes when the system displays a message confirming that the content has been updated.</p>		
PRE-CONDITION:	The volunteer must be logged onto the TRWLA system		
TRIGGER:	The user wants to search for content and clicks the “Reading Content” navigation bar item.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The user wants to search for content and clicks the “Reading Content” navigation bar item.</p>		<p>Step 2: The system prompts the volunteer to search the relevant content</p> <p>→ Invoke Use Case 7.2 Search Content</p>
	<p>Step 3: The volunteer changes all the necessary information in the fields provided by the system and clicks on the “Update” button.</p>		<p>Step 4: Using JavaScript the system displays a modal with the message “Update ‘Content Name’ Are you sure you want to update ‘Content Name’ with buttons beneath</p>

			the message labelled “Confirm” and “Cancel” On webpage 7.3.2 VA
	Step 5: The volunteer clicks on the “Confirm” button.		Step 6: Using C# and SQL the system verifies whether the information entered by the volunteer is complete and valid, such that: <ul style="list-style-type: none"> • Content Type must be a string character • Name must be a string up to 35 character long. • Author must be a string up to 70 characters long • Date Published must be in the valid date format - 10 characters in CCYY-MM-DD format • Content Link may be no longer than 100 characters • Theme must be a string no longer than 35 characters. • Status either locked or unlocked. • Description must be a string.
			Step 7: Using C# and LINQ the system updates the data to the Content table in the database.
			Step 8: Using JavaScript the system displays a modal with the following message “You have

		successfully updated 'Content Name'.
ALTERNATE COURSES:	<p>Alt Step 7: The information entered by the volunteer is not valid. Using JavaScript the system displays a modal with the message: "Update 'Content Name' Error There is a problem with the information you have entered. Please enter the information in the correct format".</p> <p>On webpage 7.3.1 VA Error → Go to Step 3</p>	
CONCLUSION:	The volunteer has updated the content in the database. After updating the Content table in the database the system displays a message confirming that the content has been updated.	
POST-CONDITION:	Content has been updated	
BUSINESS RULES	<ul style="list-style-type: none"> • Only volunteers can update content. 	
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 	
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 	
OPEN ISSUES:	None	

USE CASE NAME:	Delete Content	USE CASE TYPE	
USE CASE ID:	7.4	Business Requirements: <input type="checkbox"/>	
PRIORITY:	Low	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	Volunteer (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a volunteer deleting content information. The use case begins when the volunteer wants to delete the content information and clicks the “Reading Content” navigation bar item. The volunteer will search for the relevant content, delete its details and click the delete button. The system will verify whether the deleted information is complete and accurate and will then delete the Content table in the database. The use case concludes when the system displays a message confirming that the content has been deleted.</p>		
PRE-CONDITION:	The volunteer must be logged onto the TRWLA system		
TRIGGER:	The user wants to search for content and clicks the “Reading Content” navigation bar item.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The user wants to search for content and clicks the “Reading Content” navigation bar item.</p>		<p>Step 2: The system prompts the volunteer to search the relevant content → Invoke Use Case 7.2 Search Content</p>
	<p>Step 3: The volunteer clicks on the “Delete” button.</p>		<p>Step 4: Using JavaScript the system displays a modal with the message “Are you sure you want to delete ‘Content Name’ with buttons beneath</p>

			the message labelled "Confirm" and "Cancel"
	Step 5: The volunteer clicks on the "Confirm" button.		Step 6: Using C# and SQL the system deletes the data from the <u>Content</u> table in the database.
			Step 7: Using JavaScript the system displays a modal with the following message "You have successfully deleted 'Content Name'".
ALTERNATE COURSES:	Alt Step 3: The volunteer deletes the content on the Content Management Page by selecting the relevant content in the table and clicking the delete button. Using JavaScript the system displays a message: "Delete Content Are you sure you want to delete this content?" with "On second thought" and "Confirm" buttons. → Go to Step 5		
CONCLUSION:	The volunteer has deleted the content in the database. After deleting the <u>Content</u> table in the database the system displays a message confirming that the content has been deleted.		
POST-CONDITION:	Content has been deleted		
BUSINESS RULES	<ul style="list-style-type: none"> • Only volunteers can delete content. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Review Lecture Content		USE CASE TYPE
USE CASE ID:	7.5		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Student (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Volunteers 		
DESCRIPTION:	<p>The use case describes the event of a student making a review on the content of the lecture attended. The use case begins when the student clicks on the “Reading Content” drop down list on their navigation bar. The system will then display the various list items that are available for the student. The student will then click on the “Content” list item. The system will then display the Content Management Screen where the student can view a list of existing content on the system as well as search for specific content. The student will then search for specific content and click on the “Details” button to view the details of the content. The system will then display the details of the content selected, as well as an empty field for the student to write a review about the content specified. The student will then write the review and rate the content. The use case concludes when the student has successfully written a review, as well as rated the content specified.</p>		
PRE-CONDITION:	The student must be logged into the system.		
TRIGGER:	The student wants to review the lecture content.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1: The student wants to review the lecture content and clicks the “Reading Content” drop down list on their navigation bar.</p>		<p>Step 2: The system, using JavaScript, displays the drop down list items as follows:</p> <ul style="list-style-type: none"> ➤ Content ➤ E-Library

	<p>Step 3: The student clicks on the “Content” list item in the drop down list.</p>	<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Content Management Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ System name h1 header➤ “Home” drop down list navigation bar item➤ “Volunteers” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message box icon that displays notifications➤ Profile picturebox➤ “Users” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Content Management” h1 header➤ “Search” textbox
--	--	--

			<ul style="list-style-type: none"> ➤ “Search for a content by name” button ➤ A container that displays a list of current content available on the system with the following headers above the Container: ➤ Content Type ➤ Name ➤ Author ➤ Date Published ➤ Link ➤ Theme ➤ Status ➤ Description ➤ “Details” button for each content item in the container ➤ “Return” button <p>The information stated in the container is populated from the Content table using Entity Framework and LINQ Queries.</p>
	<p>Step 5: The student clicks on the “Details” button of a specific content item in the container.</p>		<p>Step 6: The system, using JavaScript, CSS and HTML, displays the Content Details Webpage with the following layout:</p> <p>Header Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ System name h1 header

			<ul style="list-style-type: none">➤ “Home” drop down list navigation bar item➤ “Volunteers” drop down list navigation bar item➤ “Reading Content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message box icon that displays notifications➤ Profile picturebox➤ “Users” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Name of content Details” h1 header➤ “Content type” h4 header➤ “Name” h4 header➤ “Author” h4 header➤ “Date Published” h4 header➤ “Link” h4 header➤ “Theme” h4 header➤ “Status” h4 header
--	--	--	--

			<ul style="list-style-type: none"> ➤ “Description” h4 header ➤ “Review the content of this lecture” h3 header ➤ “Leave feedback on the content” h3 header ➤ Textbox for the review ➤ “Rate the content out of 5” h3 header ➤ Rating drop down list with numbers 1-5 <p>Footer</p> <ul style="list-style-type: none"> ➤ “Review Content” button ➤ “Return” button <p>The details displayed are obtained from the Content table using Entity Framework and LINQ Queries.</p>
	<p>Step 7: The student enters their review in the available textbox and selects their rating in the drop down list.</p>		
	<p>Step 8: The student clicks the “Review Content” button.</p>		<p>Step 10: The system, using C# and JavaScript, validates that the information entered is in the correct format as follows:</p> <ul style="list-style-type: none"> • Review (Maximum 300 characters) • Rating (Must be selected from the drop down list)

			Step 11: The system, using C# and JavaScript, displays a Modal with the following message: "Are you sure you want to review this lecture content?" with two buttons namely, "Confirm" and "Cancel"
	Step 12: The student clicks the "Confirm" button.		Step 13: The system using C# and SQL, adds the review to the <u>Review</u> table using Entity Framework and LINQ queries.
			<p>Step 14: The system, using JavaScript, CSS and HTML, displays the Content Management Webpage with the following layout:</p> <p>Header Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ "Home" drop down list navigation bar item ➤ "Volunteers" drop down list navigation bar item ➤ "Reading Content" drop down list navigation bar item

			<ul style="list-style-type: none"> ➤ “Gallery” navigation bar item ➤ Message box icon that displays notifications ➤ Profile picturebox ➤ “Users” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Content Management” h1 header ➤ “Search” textbox ➤ “Search for a content by name” button ➤ A container that displays a list of current content available on the system with the following headers above the Container: <ul style="list-style-type: none"> ➤ Content Type ➤ Name ➤ Author ➤ Date Published ➤ Link ➤ Theme ➤ Status ➤ Description ➤ “Details” button for each content item in the container ➤ “Return” button <p>The information stated in the container is</p>
--	--	--	--

			populated from the Content table using Entity Framework and LINQ Queries.
ALTERNATE COURSES:	Alt Step 5: The student searches for a specific content by entering details in the “Search” textbox → Invoke UC 7.2 Search Content On Webpage: Content Management		
	Alt Step 7: The student clicks the “Return” button and does not make a review. → Go to Step 14 On Webpage : Content Management		
	Alt Step 10: The information entered in the textbox is incorrect and does not meet the specific system standard. The system displays a Modal with the following message: “The review you have entered exceeds the allotted characters. Please edit your message.” with a “Confirm” button. The system also displays viewbag error messages below the textbox indicating that the maximum amount of characters is 300. → Go back to Step 9 On Webpage: Content Details		
	Alt Step 12: The student clicks the “Cancel” button. → Go to Step 14 On Webpage: Content Management		
CONCLUSION:	The system, using JavaScript, CSS and HTML, displays the Content Management Webpage with the following layout: Header Navigation bar from left to right: ➤ System name h1 header ➤ “Home” drop down list navigation bar item ➤ “Volunteers” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message box icon that displays notifications ➤ Profile picturebox ➤ “Users” drop down list navigation bar item Body ➤ “Content Management” h1 header ➤ “Search” textbox ➤ “Search for a content by name” button ➤ A container that displays a list of current content available on the system with the following headers above the Container:		

	<ul style="list-style-type: none"> ➤ Content Type ➤ Name ➤ Author ➤ Date Published ➤ Link ➤ Theme ➤ Status ➤ Description ➤ “Details” button for each content item in the container ➤ “Return” button <p>The information stated in the container is populated from the <u>Content</u> table using Entity Framework and LINQ Queries.</p>
POST-CONDITION:	The student has successfully reviewed and rated lecture content.
BUSINESS RULES	<ul style="list-style-type: none"> • Only a student can rate lecture content. • A student cannot edit content details on the system. • A student cannot rate the same lecture content more than once.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • Once the student has rated lecture content, the student should not be able to rate it again.
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

2.3.8 Marketing Narratives

USE CASE NAME:	Upload Photo	USE CASE TYPE	
USE CASE ID:	8.1	Business Requirements: <input type="checkbox"/>	
PRIORITY:	High	System Analysis: <input type="checkbox"/>	
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>	
PRIMARY BUSINESS ACTOR	User (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • Admin (ERA) 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a user uploading a photo to the gallery. The use case begins when a user clicks on the “Gallery” menu item in the navigation bar. The Gallery webpage will then be displayed with existing gallery photos as well as blank fields where the user can upload a photo. Once the user clicks on the “Search” button, a file dialog box will be displayed based to enable the user to search for a photo on their computer and upload it to the system. Once the user has selected a photo to upload, the user will provide a suitable caption and click the “Upload” button. The system will then display a modal message confirming the user’s actions. The use case concludes when the user has successfully uploaded the photo to the system and the system notifies the admin members in order to post the photo to the static webpage.</p>		
PRE-CONDITION:	The user must be logged into the system.		
TRIGGER:	The user wants to upload a photo to the gallery.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action Automated Action	
	Step 1: The user wants to upload a photo to the gallery and clicks on the “Gallery” menu hyperlink item on the navigation bar.		Step 2: The system, using JavaScript, CSS and HTML, displays the Gallery Webpage with the following layout: Header Navigation bar from left to right:

			<ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Gallery” h1 header ➤ Thumbnails for photos ➤ “Upload a Photo” h2 header ➤ “File” textbox ➤ “Search” button ➤ “Caption” textbox ➤ “Upload” button <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button
	<p>Step 3: The user clicks on the “Browse” button.</p>		<p>Step 4: The system, using C# and JavaScript, displays a file dialog box related to the user’s operating system.</p>
	<p>Step 5: The user selects the photo they wish to upload to the system.</p>		<p>Step 6: The system, using JavaScript, CSS and HTML, displays the</p>

			<p>Gallery Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” / “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Gallery” h1 header ➤ Thumbnails for photos ➤ “Upload a Photo” h2 header ➤ “File” textbox with the selected location of the photo ➤ “Search” button ➤ “Caption” textbox ➤ “Upload” button <p>Footer</p> <p>“Return” button</p>
--	--	--	---

	Step 7: The user enters a caption for the photo and clicks the “Upload” button.		Step 8: The system, using C# and JavaScript, validates that the correct information has been entered, such as: <ul style="list-style-type: none"> • File (Must be in png or svg format) • Caption (Max 50 characters)
			Step 9: The system, using C# and JavaScript, displays a modal with the following message: “Are you sure you want to upload this file?” with two buttons namely, “Yes” and “No”.
	Step 10: The user clicks the “Yes” button.		Step 11: The system, using C# and JavaScript, displays a modal with the following message: “Photo upload successful” with a “Confirm” button.
	Step 12: The user clicks the “Confirm” button.		Step 13: The system, using C# and JavaScript, save the photo to the Gallery Webpage and send a notification to the admin members that a photo has been uploaded. Invoke UC 5.9 Send Notification
ALTERNATE COURSES:	Alt Step 3: The user clicks the “Return” button. The system, using C#, JavaScript, CSS and HTML, displays the user’s home page/ main menu with the following layout: Header Navigation Bar from left to right: ➤ TRWLA Management System h1 header		

- “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item
- “Events”/ “Home” (if the user is a student) drop down list navigation bar item
- “Reading Content” drop down list navigation bar item
- “Gallery” navigation bar item
- Message icon that is used to display notifications
- Profile Picturebox
- “User” drop down list navigation bar item

Body

- “Welcome User Name” h1 header
- “Search upcoming events by Name” textbox
- “Here is your progress” button (If the user is a student)
- “Sort with the following:” h4 header
- “Event name” button
- “Date” button
- “Upcoming TRWLA Events” h2 header
- “My Upcoming Events” h2 header
- Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:
 - “Event Name” h3 header
 - “Summary” h4 header
 - “Date of the event” h4 header
 - “Time of event” h4 header
 - “Venue name” h4 header
 - “RSVP” button
 - “Details” button
 - “Not going” button in the “My upcoming events” container.

Footer

- “Create” button (if the user is a volunteer)
- “Update” button (if the user is a volunteer)
- “Delete” button (if the user is a volunteer)
- “Return” button
- “Log Attendance” button (if the user is a volunteer)

Alt Step 8a: The photo uploaded is in the incorrect format, display viewbag error messages beneath the required textboxes stating what format the photo should be in, namely, “png”or “svg”.

→ Go to **Step 5**

Alt Step 8b: The caption entered is more than the maximum 50 characters, the system, using JavaScript, will display a viewbag that says the characters entered exceeds 50.

	<p>→ Go to Step 7 On webpage: Gallery Webpage</p>
	<p>Alt Step 10: The user clicks the “No” button. The system displays the Gallery Webpage. → Go to Step 2 On webpage: Gallery Webpage</p>
CONCLUSION:	<p>The system, using C# and JavaScript, save the photo to the Gallery Webpage and send a notification to the admin members that a photo has been uploaded. Invoke UC 5.9 Send Notification</p>
POST-CONDITION:	A photo has been successfully uploaded to the system.
BUSINESS RULES	<p>➤ Only admin members can post a photo to the static webpage.</p>
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • Once a user uploads a photo to the system, the system must notify admin members so that they can post the photo to the static webpage.
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Post Photo	USE CASE TYPE			
USE CASE ID:	8.2	Business Requirements: <input type="checkbox"/>			
PRIORITY:	Medium	System Analysis: <input type="checkbox"/>			
SOURCE:	TuksRes Women in Leadership Academy	System Design: <input checked="" type="checkbox"/>			
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)				
PRIMARY SYSTEM ACTOR	None				
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 				
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 				
DESCRIPTION:	<p>This use case describes the event of an admin member posting a photo to the static webpage. The use case begins when an admin member clicks on the “Gallery” menu item in the navigation bar. The Gallery webpage will then be displayed with existing gallery photos as well as recently uploaded photos by other users. The admin member will then review the photos and select one or many photos and then choose to either “Post” the photos to the static webpage or “Reject” the photos. Once the admin member clicks the “Post” button, the system will display a modal to confirm the admin member’s actions. The use case concludes when the admin member has successfully posted a photo to the Gallery on the static webpage.</p>				
PRE-CONDITION:	The admin member must be logged into the system.				
TRIGGER:	The admin member wants to post a photo to the static webpage.				
TYPICAL COURSE OF EVENTS:	Actor Action	System Response			
		Manual Action	Automated Action		
	<p>Step 1: The admin member wants to post a photo to the gallery on the static webpage and clicks on the “Gallery” menu hyperlink item on the navigation bar.</p>		<p>Step 2: The system, using JavaScript, CSS and HTML, displays the Gallery Webpage with the following layout:</p> <p>Header Navigation bar from left to right: ➤ TRWLA Management System h1 header</p>		

		<ul style="list-style-type: none"> ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Gallery” h1 header ➤ Thumbnails for photos ➤ “Upload a Photo” h2 header hyperlink ➤ “Post a Photo” h2 header ➤ Thumbnails of recently uploaded photos ➤ “Post” button ➤ “Reject” button <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button
	<p>Step 3: The admin member selects a photo and clicks the “Post” button.</p>	<p>Step 4: The system, using C# and JavaScript, displays a modal with the following message: “Are you sure you want to post this photo?” with a “Yes” and “No” button.</p>
	<p>Step 5: The admin member clicks the “Yes” button.</p>	<p>Step 6: The system, using C# and JavaScript,</p>

			displays a modal with the following message: "Photo post successful" with a "Confirm" button.
	Step 7: The admin member clicks the "Confirm" button.		Step 8: The system, using C#, JavaScript, CSS and HTML, saves the photos and uploads them to the Gallery on the static webpage.
ALTERNATE COURSES:	<p>Alt Step 3a: The admin member clicks the "Return" button. The system, using C#, JavaScript, CSS and HTML, displays the admin member's home page/ main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ "Members"/ "Volunteers" (if the user is a student) drop down list navigation bar item ➤ "Events"/ "Home" (if the user is a student) drop down list navigation bar item ➤ "Reading Content" drop down list navigation bar item ➤ "Gallery" navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ "User" drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ "Welcome User Name" h1 header ➤ "Search upcoming events by Name" textbox ➤ "Sort with the following:" h4 header ➤ "Event name" button ➤ "Date" button ➤ "Upcoming TRWLA Events" h2 header ➤ "My Upcoming Events" h2 header ➤ Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● "Event Name" h3 header ● "Summary" h4 header ● "Date of the event" h4 header ● "Time of event" h4 header ● "Venue name" h4 header ● "RSVP" button ● "Details" button 		

	<ul style="list-style-type: none"> • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➢ “Create” button ➢ “Update” button ➢ “Delete” button ➢ “Return” button • “Log Attendance” button
	<p>Alt Step 3b: The admin member clicks the “Reject” button. → Invoke UC 8.3 Delete Photo On webpage: Gallery Webpage</p>
	<p>Alt Step 5: The admin member clicks the “No” button. The system displays the Gallery Webpage. → Go to Step 2 On webpage: Gallery Webpage</p>
CONCLUSION:	The system, using C#, JavaScript, CSS and HTML, saves the photos and uploads them to the Gallery on the static webpage.
POST-CONDITION:	A photo has been successfully posted to the static webpage.
BUSINESS RULES	<ul style="list-style-type: none"> ➢ Only admin members can post a photo to the static webpage.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • Photos that are rejected must be deleted.
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Delete Photo		USE CASE TYPE
USE CASE ID:	8.3	Business Requirements:	<input type="checkbox"/>
PRIORITY:	Low	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of an admin member deleting a photo from the gallery. The use case begins when an admin member clicks on the “Gallery” menu item in the navigation bar. The Gallery webpage will then be displayed with existing gallery photos as well as recently uploaded photos by other users. The admin member will then select the photo they wish to delete and click the “Delete Photo” button. The system will then display a modal confirming the admin member’s action. The use case concludes when a photo has been successfully deleted.</p>		
PRE-CONDITION:	The admin member must be logged into the system.		
TRIGGER:	The admin member wants to delete a photo from the gallery.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1: The admin member wants to delete a photo from the gallery on the static webpage and clicks on the “Gallery” menu hyperlink item on the navigation bar.</p>		<p>Step 2: The system, using JavaScript, CSS and HTML, displays the Gallery Webpage with the following layout:</p> <p>Header Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item

		<ul style="list-style-type: none"> ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Gallery” h1 header ➤ Thumbnails for photos ➤ “Upload a Photo” h2 header hyperlink ➤ “Post a Photo” h2 header ➤ Thumbnails of recently uploaded photos ➤ “Post” button ➤ “Reject” button ➤ “Delete” button. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Return” button
	<p>Step 3: The admin member selects a photo and clicks the “Delete” button.</p>	<p>Step 4: The system, using C# and JavaScript, displays a modal with the following message: “Are you sure you want to delete this photo?” with a “Yes” and “No” button.</p>
	<p>Step 5: The admin member clicks the “Yes” button.</p>	<p>Step 6: The system, using C# and JavaScript, displays a modal with the following message:</p>

			"Photo deletion successful" with a "Confirm" button.
	Step 7: The admin member clicks the "Confirm" button.		Step 8: The system, using C#, JavaScript, CSS and HTML, deletes the photo from the Gallery on the static webpage.
ALTERNATE COURSES:	<p>Alt Step 3a: The admin member clicks the "Return" button. The system, using C#, JavaScript, CSS and HTML, displays the admin member's home page/ main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ "Members"/ "Volunteers" (if the user is a student) drop down list navigation bar item ➤ "Events"/ "Home" (if the user is a student) drop down list navigation bar item ➤ "Reading Content" drop down list navigation bar item ➤ "Gallery" navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ "User" drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ "Welcome User Name" h1 header ➤ "Search upcoming events by Name" textbox ➤ "Sort with the following:" h4 header ➤ "Event name" button ➤ "Date" button ➤ "Upcoming TRWLA Events" h2 header ➤ "My Upcoming Events" h2 header ➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● "Event Name" h3 header ● "Summary" h4 header ● "Date of the event" h4 header ● "Time of event" h4 header ● "Venue name" h4 header ● "RSVP" button ● "Details" button ● "Not going" button in the "My upcoming events" container. <p>Footer</p>		

	<ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ● “Log Attendance” button
	<p>Alt Step 3b: The admin member clicks the “Reject” button. → Go to Step 4 On webpage: Gallery Webpage</p>
	<p>Alt Step 5: The admin member clicks the “No” button. The system displays the Gallery Webpage. → Go to Step 2 On webpage: Gallery Webpage</p>
CONCLUSION:	The system, using C#, JavaScript, CSS and HTML, deletes the photo from the Gallery on the static webpage.
POST-CONDITION:	A photo has been successfully deleted from the Gallery.
BUSINESS RULES	<ul style="list-style-type: none"> ➤ Only admin members can delete a photo from the Gallery.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> ● None
ASSUMPTIONS:	<ul style="list-style-type: none"> ● All photos are deleted with appropriate reasoning.
OPEN ISSUES:	None

2.3.9 Reports Narratives

USE CASE NAME:	Generate Class Attendance Report	USE CASE TYPE	
USE CASE ID:	9.1	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design:	<input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Directors • Management 		
DESCRIPTION:	<p>This use case describes the event of an admin member generating a class attendance report on an ad hoc basis. The use case begins when the admin member clicks on the “Reports” list item under the “User” drop down list. The system displays a list of reports that the admin member can choose to generate. The admin member will click on the “Class Attendance” tab, which will then display a graph that shows which students attend their lectures. The use case concludes when the admin member has successfully generated a class attendance report.</p>		
PRE-CONDITION:	Lectures must have taken place during the year.		
TRIGGER:	The admin member wants to generate a class attendance report.		
TYPICAL COURSE	Actor Action	System Response	
OF EVENTS:		Manual Action	Automated Action
	<p>Step 1 The admin member wants to generate a class attendance report and clicks on the “User” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the User List Items:</p> <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Logout
	<p>Step 3: The admin member clicks the “Reports” list item.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Reports webpage with the following layout:</p> <p>Header</p>

			<p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Class Attendance” tab ➤ “Community Engagement” tab ➤ “Demographics” tab ➤ “Event Popularity” tab ➤ “Feedback” tab ➤ “Function Attendance” tab ➤ “User Statistics” tab ➤ “Student Progress” tab <p>Footer</p> <p>“Return” button</p>
	<p>Step 5: The admin member clicks on the “Class Attendance” tab.</p>		<p>Step 6: The system, using C# and SQL, accesses the <u>Attendance</u>, <u>Event</u> and <u>Lecture</u> tables and</p>

			<p>retrieves the following information from each table:</p> <p>Attendance table:</p> <ul style="list-style-type: none"> • PersonID <p>Event table:</p> <ul style="list-style-type: none"> • EventID • EventDate <p>Lecture table:</p> <ul style="list-style-type: none"> • LectureID • LectureName <p>This is done by using Entity Framework and LINQ queries, it also calculates the total attendance per student per lecture, up to the current date.</p>
			<p>Step 7: The system, using JavaScript, displays the Class Attendance Report graph with the following layout:</p> <ul style="list-style-type: none"> ➤ Container with the following layout: ➤ “Class Attendance of Registered TRWLA Members” h3 header graph title ➤ “Students” vertical axis header ➤ “Total classes” horizontal axis header ➤ “Class of 2017” legend title

ALTERNATE COURSES:

Alt Step 5: The admin member clicks on the “Return” button. The system, using C#, JavaScript, CSS and HTML, displays the admin member’s main menu with the following layout:

Header

Navigation Bar from left to right:

- TRWLA Management System h1 header
- “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item
- “Events”/ “Home” (if the user is a student) drop down list navigation bar item
- “Reading Content” drop down list navigation bar item
- “Gallery” navigation bar item
- Message icon that is used to display notifications
- Profile Picturebox
- “User” drop down list navigation bar item

Body

- “Welcome User Name” h1 header
- “Search upcoming events by Name” textbox
- “Sort with the following:” h4 header
- “Event name” button
- “Date” button
- “Upcoming TRWLA Events” h2 header
- “My Upcoming Events” h2 header
- Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:
 - “Event Name” h3 header
 - “Summary” h4 header
 - “Date of the event” h4 header
 - “Time of event” h4 header
 - “Venue name” h4 header
 - “RSVP” button
 - “Details” button
 - “Not going” button in the “My upcoming events” container.

Footer

- “Create” button
- “Update” button
- “Delete” button
- “Return” button
- “Log Attendance” button

CONCLUSION:

The system, using JavaScript, displays the Class Attendance Report graph with the following layout:

- Container with the following layout:

	<ul style="list-style-type: none"> ➤ “Class Attendance of Registered TRWLA Members” h3 header graph title ➤ “Students” vertical axis header ➤ “Total classes” horizontal axis header ➤ “Class of 2017” legend title
POST-CONDITION:	A class attendance report is successfully generated.
BUSINESS RULES	<ul style="list-style-type: none"> • Only admin members can generate a report.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Generate Function Attendance Report	USE CASE TYPE	
USE CASE ID:	9.2	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design:	<input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Directors • Management 		
DESCRIPTION:	<p>This use case describes the event of an admin member generating a function attendance report on an ad hoc basis. The use case begins when the admin member clicks on the “Reports” list item under the “User” drop down list. The system displays a list of reports that the admin member can choose to generate. The admin member will click on the “Function Attendance” tab, which will then display a graph that shows how many students attended a specific function. The use case concludes when the admin member has successfully generated a function attendance report.</p>		
PRE-CONDITION:	Functions must have taken place during the year.		
TRIGGER:	The admin member wants to generate a function attendance report.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1 The admin member wants to generate a function attendance report and clicks on the “User” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the “User” drop down list items:</p> <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Logout
	<p>Step 3: The admin member clicks on the “Reports” list item.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Reports webpage with the following layout:</p>

			<p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Class Attendance” tab ➤ “Community Engagement” tab ➤ “Demographics” tab ➤ “Event Popularity” tab ➤ “Feedback” tab ➤ “Function Attendance” tab ➤ “User Statistics” tab ➤ “Student Progress” tab <p>Footer</p> <p>“Return” button</p>
	<p>Step 5: The admin member clicks on the “Function Attendance” tab.</p>		<p>Step 6: The system, using C# and SQL, accesses the <u>Attendance</u>, <u>Event</u> and</p>

			<p>Function tables and retrieves the following information from each table:</p> <p>Attendance table:</p> <ul style="list-style-type: none"> • PersonID <p>Event table:</p> <ul style="list-style-type: none"> • EventID • EventDate <p>Function table:</p> <ul style="list-style-type: none"> • FunctionName <p>This is done by using Entity Framework and LINQ queries, it also calculates the total attendance of students per function event.</p>
			<p>Step 7: The system, using JavaScript, displays the Function Attendance Report graph with the following layout:</p> <ul style="list-style-type: none"> ➤ Container with the following layout: ➤ “Function Attendance of Registered TRWLA Members” h3 header graph title ➤ “Function Events” vertical axis header ➤ “Number of Students” horizontal axis header ➤ “Class of 2017” legend title

ALTERNATE COURSES:

Alt Step 5: The admin member clicks on the “Return” button. The system, using C#, JavaScript, CSS and HTML, displays the admin member’s main menu with the following layout:

Header

Navigation Bar from left to right:

- TRWLA Management System h1 header
- “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item
- “Events”/ “Home” (if the user is a student) drop down list navigation bar item
- “Reading Content” drop down list navigation bar item
- “Gallery” navigation bar item
- Message icon that is used to display notifications
- Profile Picturebox
- “User” drop down list navigation bar item

Body

- “Welcome User Name” h1 header
- “Search upcoming events by Name” textbox
- “Sort with the following:” h4 header
- “Event name” button
- “Date” button
- “Upcoming TRWLA Events” h2 header
- “My Upcoming Events” h2 header
- Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:
 - “Event Name” h3 header
 - “Summary” h4 header
 - “Date of the event” h4 header
 - “Time of event” h4 header
 - “Venue name” h4 header
 - “RSVP” button
 - “Details” button
 - “Not going” button in the “My upcoming events” container.

Footer

- “Create” button
- “Update” button
- “Delete” button
- “Return” button
- “Log Attendance” button

CONCLUSION:

The system, using JavaScript, displays the Function Attendance Report graph with the following layout:

- Container with the following layout:

	<ul style="list-style-type: none"> ➤ “Function Attendance of Registered TRWLA Members” h3 header graph title ➤ “Function Events” vertical axis header ➤ “Number of Students” horizontal axis header ➤ “Class of 2017” legend title
POST-CONDITION:	A function attendance report is successfully generated.
BUSINESS RULES	<ul style="list-style-type: none"> • Only admin members can generate a report.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Generate Community Engagement Attendance Report		USE CASE TYPE
USE CASE ID:	9.3		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Directors • Management 		
DESCRIPTION:	<p>This use case describes the event of an admin member generating a community engagement attendance report on an ad hoc basis. The use case begins when the admin member clicks on the “Reports” list item under the “User” drop down list. The system displays a list of reports that the admin member can choose to generate. The admin member will click on the “Community Engagement” tab, which will then display a graph that shows how many students attended a specific function. The use case concludes when the admin member has successfully generated a function attendance report.</p>		
PRE-CONDITION:	Community Engagements must have taken place during the year.		
TRIGGER:	The admin member wants to generate a community engagement attendance report.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1 The admin member wants to generate a community engagement attendance report and clicks on the “User” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the “User” drop down list items:</p> <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Logout
	<p>Step 3: The admin member clicks on the “Reports” list item.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the</p>

			<p>Reports webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Class Attendance” tab ➤ “Community Engagement” tab ➤ “Demographics” tab ➤ “Event Popularity” tab ➤ “Feedback” tab ➤ “Function Attendance” tab ➤ “User Statistics” tab ➤ “Student Progress” tab <p>Footer</p> <p>“Return” button</p>
--	--	--	---

	<p>Step 5: The admin member clicks on the “Community Engagement” tab.</p>		<p>Step 6: The system, using C# and SQL, accesses the <u>Attendance</u>, <u>Event</u> and <u>CommunityOutreach</u> tables and retrieves the following information from each table:</p> <p>Attendance table:</p> <ul style="list-style-type: none"> • PersonID <p>Event table:</p> <ul style="list-style-type: none"> • EventID • EventDate <p>CommunityOutreach table:</p> <ul style="list-style-type: none"> • OutreachName <p>This is done by using Entity Framework and LINQ queries, it also calculates the total attendance of students per community engagement event.</p>
			<p>Step 7: The system, using JavaScript, displays the Community Engagement Attendance Report graph with the following layout:</p> <ul style="list-style-type: none"> ➤ Container with the following layout: ➤ “Community Engagement Attendance of Registered TRWLA Members” h3 header graph title ➤ “Community Engagement Events” vertical axis header

			<ul style="list-style-type: none"> ➤ “Number of Students” horizontal axis header ➤ “Class of 2017” legend title
ALTERNATE COURSES:	<p>Alt Step 5: The admin member clicks on the “Return” button. The system, using C#, JavaScript, CSS and HTML, displays the admin member’s main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> • “Event Name” h3 header • “Summary” h4 header • “Date of the event” h4 header • “Time of event” h4 header • “Venue name” h4 header • “RSVP” button • “Details” button • “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button 		

	<ul style="list-style-type: none"> ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
CONCLUSION:	<p>The system, using JavaScript, displays the Community Engagement Attendance Report tab with the following layout:</p> <ul style="list-style-type: none"> ➤ Container with the following layout: ➤ “Community Engagement Attendance of Registered TRWLA Members” h3 header graph title ➤ “Community Engagement Events” vertical axis header ➤ “Number of Students” horizontal axis header ➤ “Class of 2017” legend title
POST-CONDITION:	A community engagement attendance report is successfully generated.
BUSINESS RULES	<ul style="list-style-type: none"> • Only admin members can generate a report.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

USE CASE NAME:	Generate Demographics Report	USE CASE TYPE	
USE CASE ID:	9.4	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy	System Design:	<input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Directors • Management 		
DESCRIPTION:	<p>This use case describes the event of an admin member generating a demographics report on an ad hoc basis. The use case begins when the admin member clicks on the “Reports” list item under the “User” drop down list. The system displays a list of reports that the admin member can choose to generate. The admin member will click on the “Demographics” tab, which will then display graph that shows how many students belong to a certain race. The use case concludes when the admin member has successfully generated a demographics report.</p>		
PRE-CONDITION:	The race of each student in the academy must have been captured in the year.		
TRIGGER:	The admin member wants to generate a demographics report.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	Step 1 The admin member wants to generate a demographics report and clicks on the “User” drop down list.		Step 2: The system, using JavaScript, displays the “User” drop down list items: <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Logout
	Step 3: The admin member clicks on the “Reports” list item.		Step 4: The system, using JavaScript, CSS and HTML, displays the Reports webpage with the following layout:

			<p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Class Attendance” tab ➤ “Community Engagement” tab ➤ “Demographics” tab ➤ “Event Popularity” tab ➤ “Feedback” tab ➤ “Function Attendance” tab ➤ “User Statistics” tab ➤ “Student Progress” tab <p>Footer</p> <p>“Return” button</p>
	<p>Step 5: The admin member clicks on the “Demographics” tab.</p>		<p>Step 6: The system, using C# and SQL, accesses the <u>Student</u> table and</p>

			<p>retrieves the following information from the table:</p> <p>Student table:</p> <ul style="list-style-type: none"> • Race <p>This is done by using Entity Framework and LINQ queries, it also calculates the total amount of students per race.</p>
			<p>Step 7: The system, using JavaScript, CSS and HTML, displays the Demographics Report graph with the following layout:</p> <ul style="list-style-type: none"> ➤ Container with the a pie chart that contains the following layout: ➤ “Demographic Make Up of TRWLA Students” h3 header graph title ➤ Legend with the following h4 headers: ➤ Black ➤ White ➤ Colored ➤ Indian ➤ Asian
ALTERNATE COURSES:	<p>Alt Step 5: The admin member clicks on the “Return” button. The system, using C#, JavaScript, CSS and HTML, displays the admin member’s main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item 		

	<ul style="list-style-type: none"> ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button ● “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
CONCLUSION:	<p>The system, using JavaScript, CSS and HTML, displays the Demographics Report graph with the following layout:</p> <ul style="list-style-type: none"> ➤ Container with the a pie chart that contains the following layout: ➤ “Demographic Make Up of TRWLA Students” h3 header graph title ➤ Legend with the following h4 headers: <ul style="list-style-type: none"> ➤ Black ➤ White ➤ Colored ➤ Indian ➤ Asian

POST-CONDITION:	A demographics report is successfully generated.
BUSINESS RULES	<ul style="list-style-type: none">Only admin members can generate a report.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none">None
ASSUMPTIONS:	<ul style="list-style-type: none">None
OPEN ISSUES:	None

USE CASE NAME:	Generate Event Popularity Report		USE CASE TYPE
USE CASE ID:	9.5		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of an Event Popularity Report being generated. The Event Popularity information will be retrieved from the Event table, Function table, CommServ table, Lecture table, Attendance table and RSVP table, which contains the event Popularity of the different events and types of events part of TRWLA. This Report will be generated on an ad hoc basis as needed by admin. Admin will choose the Reports item under the User dropdown list. The system will display the Reports Webpage and admin will choose to generate an Event Popularity Report. The system will retrieve the event information from the <u>Event</u> table, <u>Function</u> table, <u>CommServ</u> table, <u>Lecture</u> table, <u>Attendance</u> table and <u>RSVP</u> table and calculate the most popular event type and the most popular event for each event type (Function, Lecture and Community Service). The system will then display a report with the calculations to admin. This use case concludes by a report being generated for admin to view and save.</p>		
PRE-CONDITION:	Admin must be logged into the system.		
TRIGGER:	Admin wants to generate an Event Popularity Report.		
TYPICAL COURSE OF EVENTS:	Actor Action Step 1 The admin member wants to generate a class attendance report and clicks on the "User" drop down list. On webpage: Main Menu	System Response Manual Action	Automated Action
			Step 2: The system, using JavaScript, displays the User List Items: ➤ Reports ➤ Register Admin ➤ User Type

	<p>Step 3: The admin member clicks the “Reports” item. On webpage: Report</p>		<ul style="list-style-type: none"> ➤ Logout <p>Step 4: The system displays the Reports webpage with a header and a body.</p> <p>The header contains the following items:</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System ➤ Members ➤ Events ➤ Reading content ➤ Gallery ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ User drop down list navigation bar item <p>The body contains the following tabs at the top:</p> <ul style="list-style-type: none"> ➤ Class Report ➤ Community Engagement ➤ Demographics ➤ Event Popularity ➤ Feedback ➤ Function Attendance ➤ User Statistics
--	--	--	---

			<p>➤ Student Progress using JavaScript, C#, CSS and HTML.</p>
	<p>Step 5: Admin clicks on the “Event Popularity Report” item. On webpage Reports</p>		<p>Step 6: The system, using C# and SQL, accesses the <u>Attendance</u>, <u>Event</u>, <u>CommServ</u>, <u>Function</u> and <u>Lecture</u> tables and retrieves the following information from each table:</p> <ul style="list-style-type: none"> Attendance table: <ul style="list-style-type: none"> • PersonID Event table: <ul style="list-style-type: none"> • EventID • EventDate Lecture table: <ul style="list-style-type: none"> • EventID • Name Function table: <ul style="list-style-type: none"> • EventID • Name CommServ table: <ul style="list-style-type: none"> • EventID • Name <p>using Entity Framework and LINQ queries, it also calculates the total attendance per student per lecture, up to the current date.</p>
			<p>Step 7: The system, using JavaScript, CSS and HTML, displays the Event Popularity Report</p>

			<p>Webpage with the following layout:</p> <p>Header</p> <p>Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System h1 header➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Reading content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Event Popularity Report” h1 header
--	--	--	---

			<ul style="list-style-type: none"> ➤ Container with the following layout: ➤ “Event Popularity of TRWLA events” h3 header graph title ➤ “Students” vertical axis header ➤ “Types of Events” horizontal axis header ➤ “Class of 2017” legend title
ALTERNATE COURSES:	None		
CONCLUSION:	An event popularity report is successfully generated.		
POST-CONDITION:	The system has generated and displayed an event popularity report.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin will have access to the report. • Only Admin can generate reports. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Generate Feedback Report		USE CASE TYPE
USE CASE ID:	9.6		Business Requirements: <input type="checkbox"/>
PRIORITY:	High		System Analysis: <input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	Admin		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • None 		
DESCRIPTION:	<p>This use case describes the event of a Feedback Report being generated. The feedback information will be retrieved from the Feedback table, which contains the feedback of volunteers' part of TRWLA. This Report will only contain certain values from that table and will be generated on an ad hoc basis as needed by admin. Admin will choose the Reports item under the User dropdown list. The system will display the Reports Webpage and admin will choose to generate a Feedback Report. The system will retrieve the Feedback information from the table and calculate the average rating of the volunteers for certain attributes. The system will then display a report with the calculations to admin. This use case concludes by a report being generated for admin to view and save.</p>		
PRE-CONDITION:	Admin is logged into the system.		
TRIGGER:	Admin wants to generate a Feedback Report.		
TYPICAL COURSE OF EVENTS:	Actor Action Step 1 Admin wants to generate a feedback report and clicks on the "User" drop down list. On webpage: Main menu	System Response Manual Action	Automated Action
	Step 3: Admin clicks the "Reports" list item.		Step 4: The system displays the Reports webpage

		<p>with a header and a body. The header contains the following items: Navigation bar from left to right:</p> <ul style="list-style-type: none">➤ TRWLA Management System➤ Members➤ Events➤ Reading content➤ Gallery➤ Message icon which displays notifications.➤ Profile picturebox➤ User drop down list navigation bar item <p>The body contains the following tabs at the top:</p> <ul style="list-style-type: none">➤ Class Report➤ Community Engagement➤ Demographics➤ Event Popularity➤ Feedback➤ Function Attendance➤ User Statistics➤ Student Progress <p>using JavaScript, C#, CSS and HTML.</p>	
--	--	--	--

	<p>Step 5: Admin clicks on the “Feedback Report” item. On webpage: Reports</p>		<p>Step 6: The system, using C# and SQL, accesses the VolunteerFeedback table and retrieves the following information:</p> <ul style="list-style-type: none">• Enjoyment• Prepared• Expectations• Communication• Growth• Impact• Coping• FutureVision• Confidence• Challenged <p>This is done by using Entity Framework and LINQ queries, it calculates the average rating per attribute, up to the current date.</p>
			<p>Step 7: The system, using JavaScript, CSS and HTML, displays the Feedback Report Webpage with the following layout:</p> <p>Header Navigation bar from left to right: ➤ TRWLA Management System h1 header</p>

			<ul style="list-style-type: none">➤ “Members” drop down list navigation bar item➤ “Events” drop down list navigation bar item➤ “Reading Reading content” drop down list navigation bar item➤ “Gallery” navigation bar item➤ Message icon which displays notifications.➤ Profile picturebox➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none">➤ “Feedback Report” h1 header➤ Container with the following layout:➤ “Feedback of Volunteers” h3 header graph title➤ “Average rating”
--	--	--	---

			vertical axis header ➤ Feedback Attribute names on from VolunteerFeedback table horizontal axis header ➤ “Class of 2017” legend title
ALTERNATE COURSES:	None		
CONCLUSION:	A feedback report is generated.		
POST-CONDITION:	A Feedback Report is generated for admin to view and save.		
BUSINESS RULES	<ul style="list-style-type: none"> • Only authorised admin will have access to the report. • Only Admin can generate reports. 		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None 		
ASSUMPTIONS:	<ul style="list-style-type: none"> • None 		
OPEN ISSUES:	None		

USE CASE NAME:	Generate User Statistics Report		USE CASE TYPE
USE CASE ID:	9.7	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Directors • Management 		
DESCRIPTION:	<p>This use case describes the event of an admin member generating a user statistics report on an ad hoc basis. The use case begins when the admin member clicks on the “Reports” list item under the “User” drop down list. The system displays a list of reports that the admin member can choose to generate. The admin member will click on the “User Statistics” tab, which will then display a graph that shows how many users have been active on the system. The use case concludes when the admin member has successfully generated a user statistics report.</p>		
PRE-CONDITION:	Users must have logged into the system up until the current date.		
TRIGGER:	The admin member wants to generate a user statistics report.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	<p>Step 1 The admin member wants to generate a user statistics report and clicks on the “User” drop down list.</p>		<p>Step 2: The system, using JavaScript, displays the User drop down list items:</p> <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Logout
	<p>Step 3: The admin member clicks the “Reports” list item.</p>		<p>Step 4: The system, using JavaScript, CSS and HTML, displays the Reports webpage with the following layout:</p> <p>Header</p>

			<p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Class Attendance” tab ➤ “Community Engagement” tab ➤ “Demographics” tab ➤ “Event Popularity” tab ➤ “Feedback” tab ➤ “Function Attendance” tab ➤ “User Statistics” tab ➤ “Student Progress” tab <p>Footer</p> <p>“Return” button</p>
	<p>Step 5: The admin member clicks on the “User Statistics” tab.</p>		<p>Step 6: The system, using C# and SQL, accesses the Person table and retrieves the following</p>

			<p>information from the table:</p> <p>Person table:</p> <ul style="list-style-type: none"> • PersonID <p>This is done by using Entity Framework and LINQ queries, it then calculates the total amount of users that has logged into the system until the current date.</p>
			<p>Step 7: The system, using JavaScript, , displays the User Statistics Report graph with the following layout:</p> <ul style="list-style-type: none"> ➤ Container with the following layout: ➤ “Report Generated for the month of year” h3 header graph title ➤ “Number of Users” vertical axis header ➤ “Month” horizontal axis header ➤ “Registered TRWLA Users” legend title
ALTERNATE COURSES:	<p>Alt Step 5a: The admin member clicks on the “Return” button. The system, using C#, JavaScript, CSS and HTML, displays the admin member’s main menu with the following layout:</p> <p>Header</p> <p>Navigation Bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item 		

	<ul style="list-style-type: none"> ➤ “Events”/ “Home” (if the user is a student) drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon that is used to display notifications ➤ Profile Picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Welcome User Name” h1 header ➤ “Search upcoming events by Name” textbox ➤ “Sort with the following:” h4 header ➤ “Event name” button ➤ “Date” button ➤ “Upcoming TRWLA Events” h2 header ➤ “My Upcoming Events” h2 header ➤ Containers with the following Event information as populated from the <u>Event</u> table using Entity Framework and LINQ queries: <ul style="list-style-type: none"> ● “Event Name” h3 header ● “Summary” h4 header ● “Date of the event” h4 header ● “Time of event” h4 header ● “Venue name” h4 header ● “RSVP” button ● “Details” button ● “Not going” button in the “My upcoming events” container. <p>Footer</p> <ul style="list-style-type: none"> ➤ “Create” button ➤ “Update” button ➤ “Delete” button ➤ “Return” button ➤ “Log Attendance” button
CONCLUSION:	The system, using JavaScript, , displays the User Statistics Report graph with the following layout: <ul style="list-style-type: none"> ➤ Container with the following layout: ➤ “Report Generated for the month of year” h3 header graph title ➤ “Number of Users” vertical axis header ➤ “Month” horizontal axis header ➤ “Registered TRWLA Users” legend title
POST-CONDITION:	A user statistics report is successfully generated.
BUSINESS RULES	<ul style="list-style-type: none"> ➤ Only admin members can generate a report.

IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none">• None
ASSUMPTIONS:	<ul style="list-style-type: none">• None
OPEN ISSUES:	None

USE CASE NAME:	Generate Student Progress Report		USE CASE TYPE
USE CASE ID:	9.8	Business Requirements:	<input type="checkbox"/>
PRIORITY:	High	System Analysis:	<input type="checkbox"/>
SOURCE:	TuksRes Women in Leadership Academy		System Design: <input checked="" type="checkbox"/>
PRIMARY BUSINESS ACTOR	User: Volunteer (Admin) (PBA)		
PRIMARY SYSTEM ACTOR	None		
OTHER PARTICIPATING ACTORS:	<ul style="list-style-type: none"> • None 		
OTHER INTERESTED STAKEHOLDERS:	<ul style="list-style-type: none"> • Directors • Management 		
DESCRIPTION:	<p>This use case describes the event of an admin member generating a student progress report on an ad hoc basis. The use case begins when the admin member clicks on the “Reports” list item under the “User” drop down list. The system displays a list of reports that the admin member can choose to generate. The admin member will click on the “Student Progress” tab, which will then display a list of registered TRWLA students with their progress percentage. The use case concludes when the admin member has successfully generated a student progress report.</p>		
PRE-CONDITION:	The admin member must be logged into the system.		
TRIGGER:	The admin member wants to generate a student progress report.		
TYPICAL COURSE OF EVENTS:	Actor Action	System Response	
		Manual Action	Automated Action
	Step 1 The admin member wants to generate a student progress report and clicks on the “User” drop down list.		Step 2: The system, using JavaScript, displays the User drop down list items: <ul style="list-style-type: none"> ➤ Reports ➤ Register Admin ➤ User Type ➤ Logout
	Step 3: The admin member clicks the “Reports” list item.		Step 4: The system, using JavaScript, CSS and HTML, displays the Reports webpage with the following layout: Header

			<p>Navigation bar from left to right:</p> <ul style="list-style-type: none"> ➤ TRWLA Management System h1 header ➤ “Members” drop down list navigation bar item ➤ “Events” drop down list navigation bar item ➤ “Reading Content” drop down list navigation bar item ➤ “Gallery” navigation bar item ➤ Message icon which displays notifications. ➤ Profile picturebox ➤ “User” drop down list navigation bar item <p>Body</p> <ul style="list-style-type: none"> ➤ “Class Attendance” tab ➤ “Community Engagement” tab ➤ “Demographics” tab ➤ “Event Popularity” tab ➤ “Feedback” tab ➤ “Function Attendance” tab ➤ “User Statistics” tab ➤ “Student Progress” tab <p>Footer</p> <p>“Return” button</p>
	<p>Step 5: The admin member clicks on the “Student Progress” tab.</p>		<p>Step 6: The system, using C# and SQL, accesses the <u>Student</u> and <u>StudentMilestone</u> tables</p>

			<p>and retrieves the following information from the tables:</p> <p><u>Student</u> table:</p> <ul style="list-style-type: none"> • StudentNumber • First Name • Last Name <p><u>StudentMilestone</u> table</p> <ul style="list-style-type: none"> • ProgressStatus • ProgressPercentage <p>This is done by using Entity Framework and LINQ queries, it then calculates the total amount of progress based from the student's attendance from the <u>Attendance</u> table and converts it into a percentage.</p>
			<p>Step 7: The system, using JavaScript, CSS and HTML, displays the Student Progress Report graph with the following layout:</p> <ul style="list-style-type: none"> ➤ Container with the following layout: ➤ “Student Number” h3 header ➤ “First Name” h3 header ➤ “Last Name” h3 header ➤ “Progress Status” h3 header ➤ “Progress Score” h3 header

ALTERNATE COURSES:

Alt Step 5a: The admin member clicks on the “Return” button. The system, using C#, JavaScript, CSS and HTML, displays the admin member’s main menu with the following layout:

Header

Navigation Bar from left to right:

- TRWLA Management System h1 header
- “Members”/ “Volunteers” (if the user is a student) drop down list navigation bar item
- “Events”/ “Home” (if the user is a student) drop down list navigation bar item
- “Reading Content” drop down list navigation bar item
- “Gallery” navigation bar item
- Message icon that is used to display notifications
- Profile Picturebox
- “User” drop down list navigation bar item

Body

- “Welcome User Name” h1 header
- “Search upcoming events by Name” textbox
- “Sort with the following:” h4 header
- “Event name” button
- “Date” button
- “Upcoming TRWLA Events” h2 header
- “My Upcoming Events” h2 header
- Containers with the following Event information as populated from the Event table using Entity Framework and LINQ queries:
 - “Event Name” h3 header
 - “Summary” h4 header
 - “Date of the event” h4 header
 - “Time of event” h4 header
 - “Venue name” h4 header
 - “RSVP” button
 - “Details” button
 - “Not going” button in the “My upcoming events” container.

Footer

- “Create” button
- “Update” button
- “Delete” button
- “Return” button
- “Log Attendance” button

CONCLUSION:

The system, using JavaScript, CSS and HTML, displays the Student Progress Report graph with the following layout:

- Container with the following layout:

	<ul style="list-style-type: none"> ➤ “Student Number” h3 header ➤ “First Name” h3 header ➤ “Last Name” h3 header ➤ “Progress Status” h3 header ➤ “Progress Score” h3 header
POST-CONDITION:	A student progress report is successfully generated.
BUSINESS RULES	<ul style="list-style-type: none"> • Only admin members can generate a report.
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	<ul style="list-style-type: none"> • None
ASSUMPTIONS:	<ul style="list-style-type: none"> • None
OPEN ISSUES:	None

2.4 Conclusion

In this section, the group depicted the technical specifications of the to-be system by means of a use case diagram. The use cases were then further decomposed and explained in the use case narratives. Both of the above provide a clearer explanation to the client as to what their system functionality and processes will be like.

3. Process-Oriented Design

3.1 Introduction

3.1.1 The following process models are depicted below; context diagram, fully functional decomposition diagram and high-level, mid-level and primitive-level data flow diagrams. The context diagram depicts how the external actors interact with the system and how the system responds to the external actor. The decomposition diagram displays the hierarchical order that flows throughout the system from the system level, function level, activity level to task level. The data flow diagrams, i.e. High-level, Mid-level and Primitive-Level, are used to describe the process and its data flow from the use case narratives above. Each diagram also depicts technically how the data will be processed from the database to the system. Under each primitive diagram, the pseudo code or structured English is depicted in order to explain the logic behind every process in the system.

3.2 Context Diagram

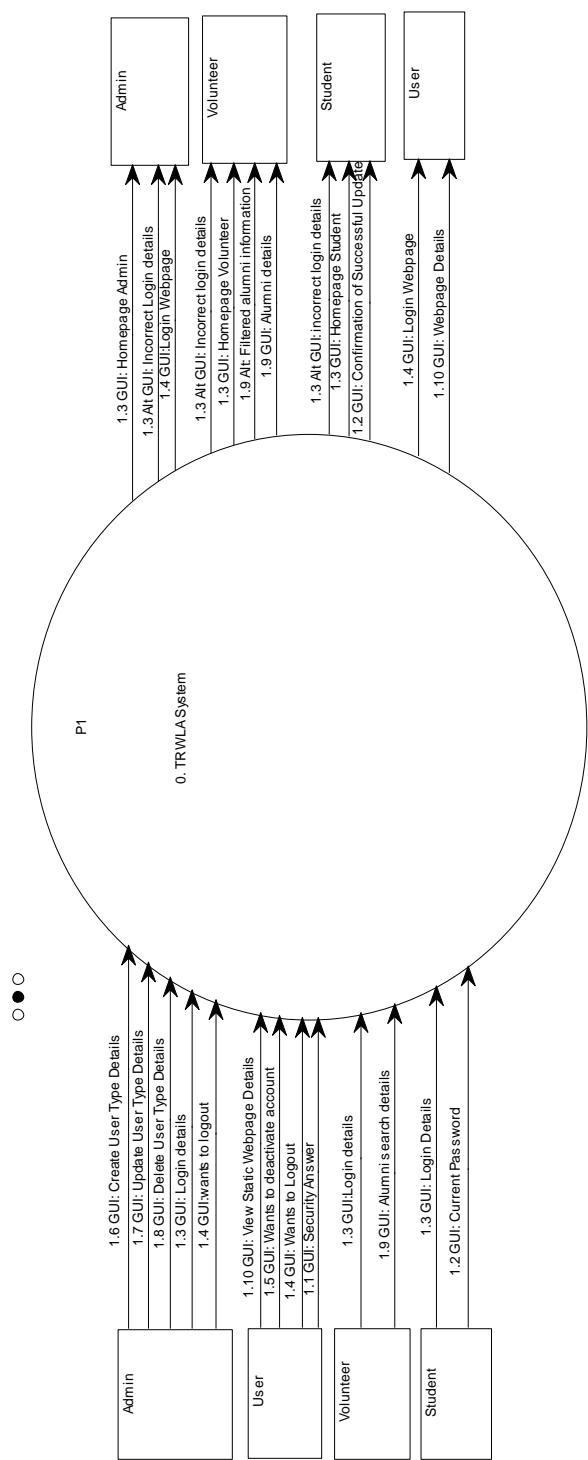


Figure 10: Context Diagram Subsystem 1

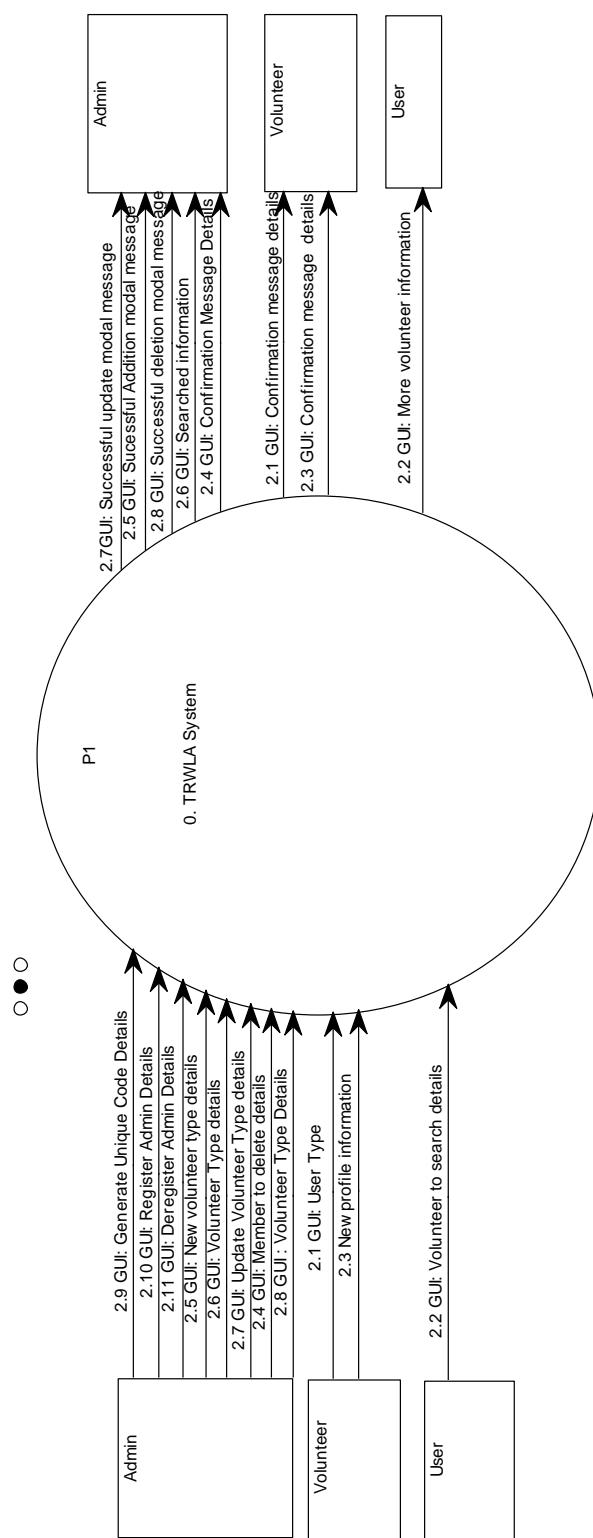


Figure 11: Context Diagram Subsystem 2

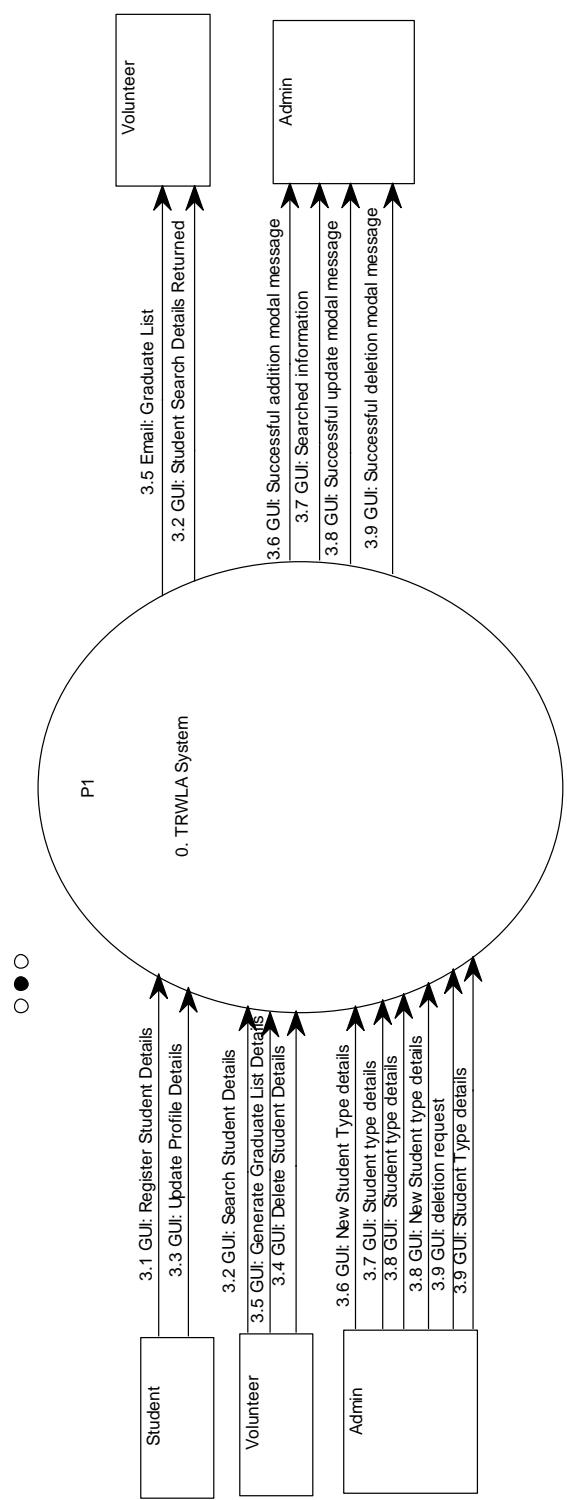


Figure 12: Context Diagram Subsystem 3

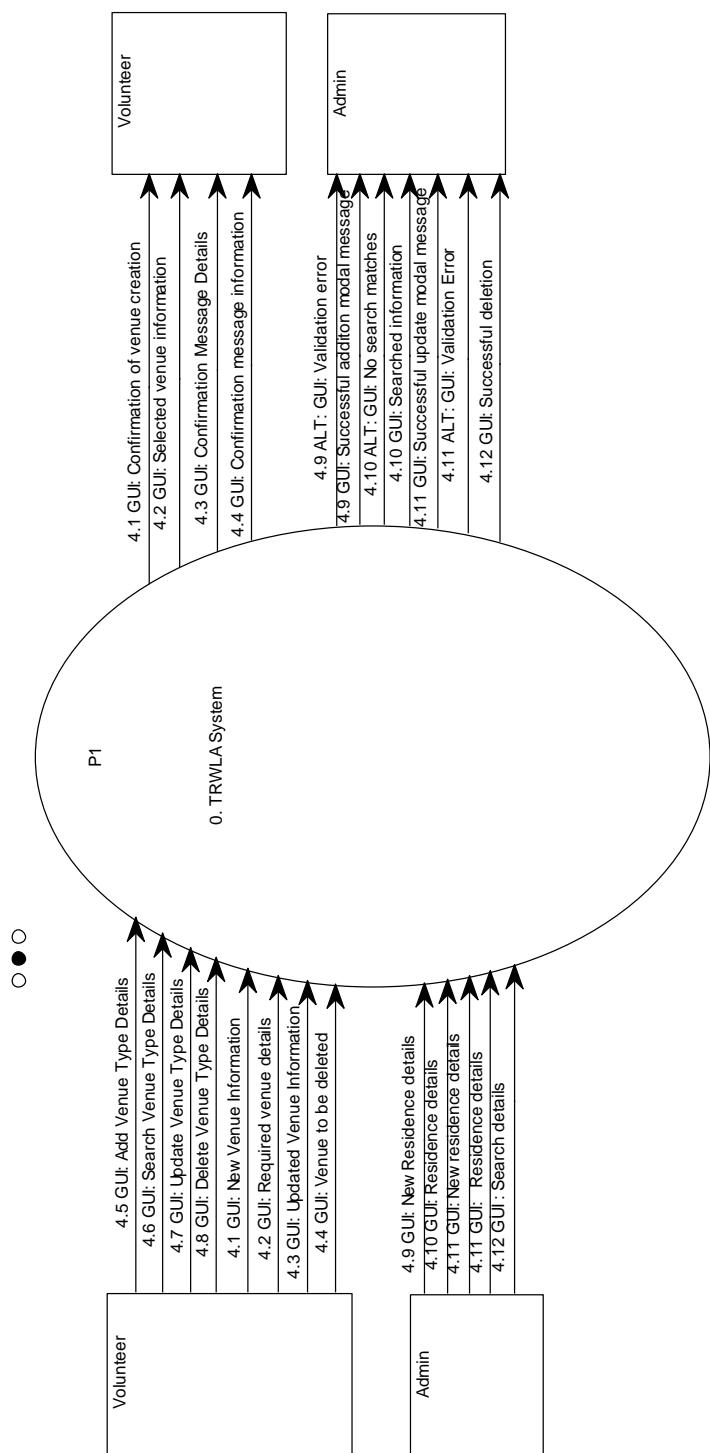


Figure 13: Context Diagram Subsystem 4

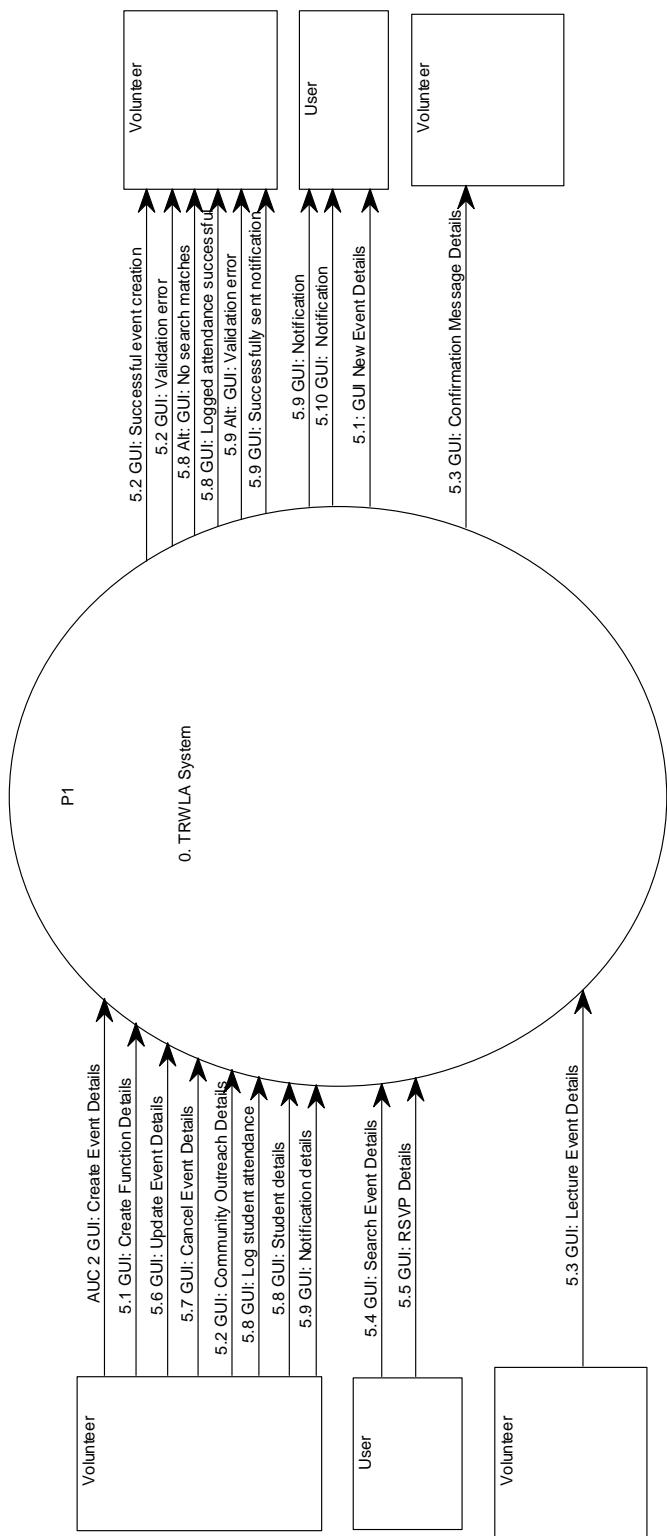


Figure 14: Context Diagram Subsystem 5

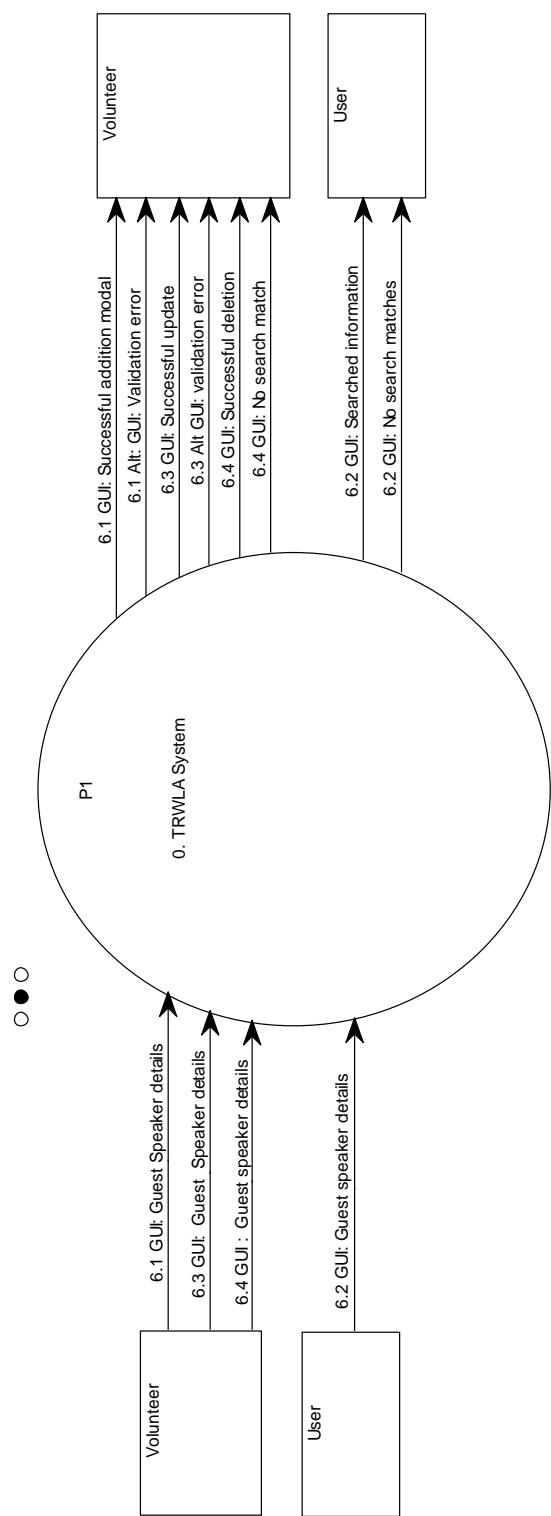


Figure 15: Context Diagram Subsystem 6

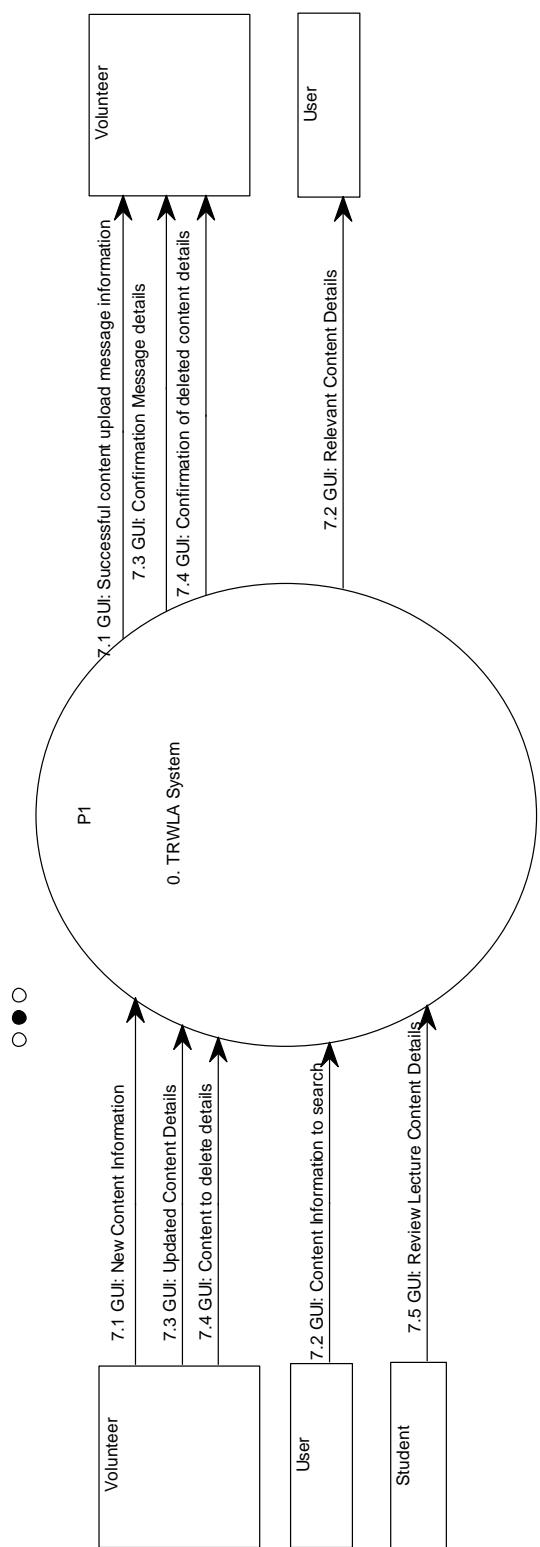


Figure 16: Context Diagram Subsystem 7

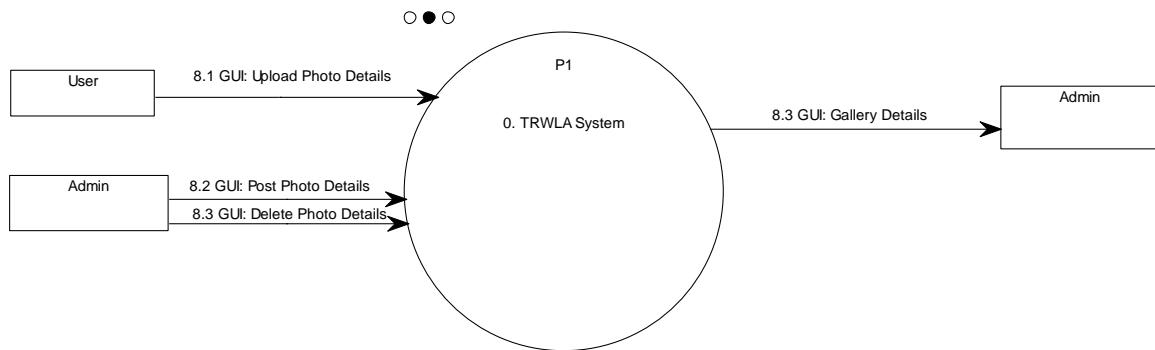


Figure 17: Context Diagram Subsystem 8

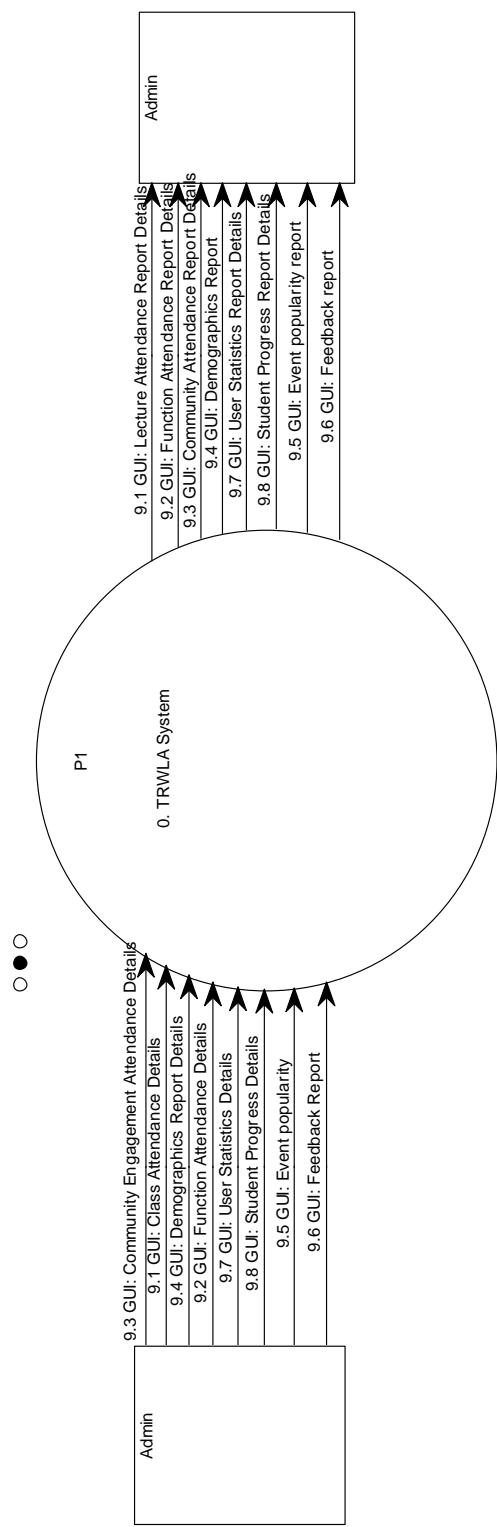


Figure 18: Context Diagram Subsystem 9

3.3 Decomposition Diagram

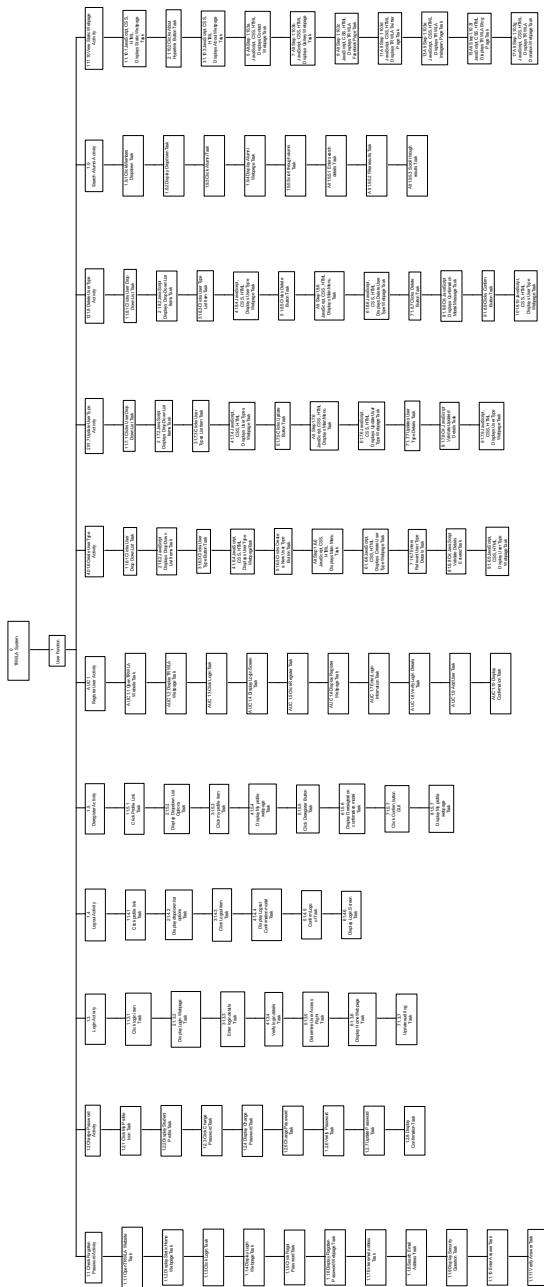


Figure 19: Function 1 Decomposition Diagram

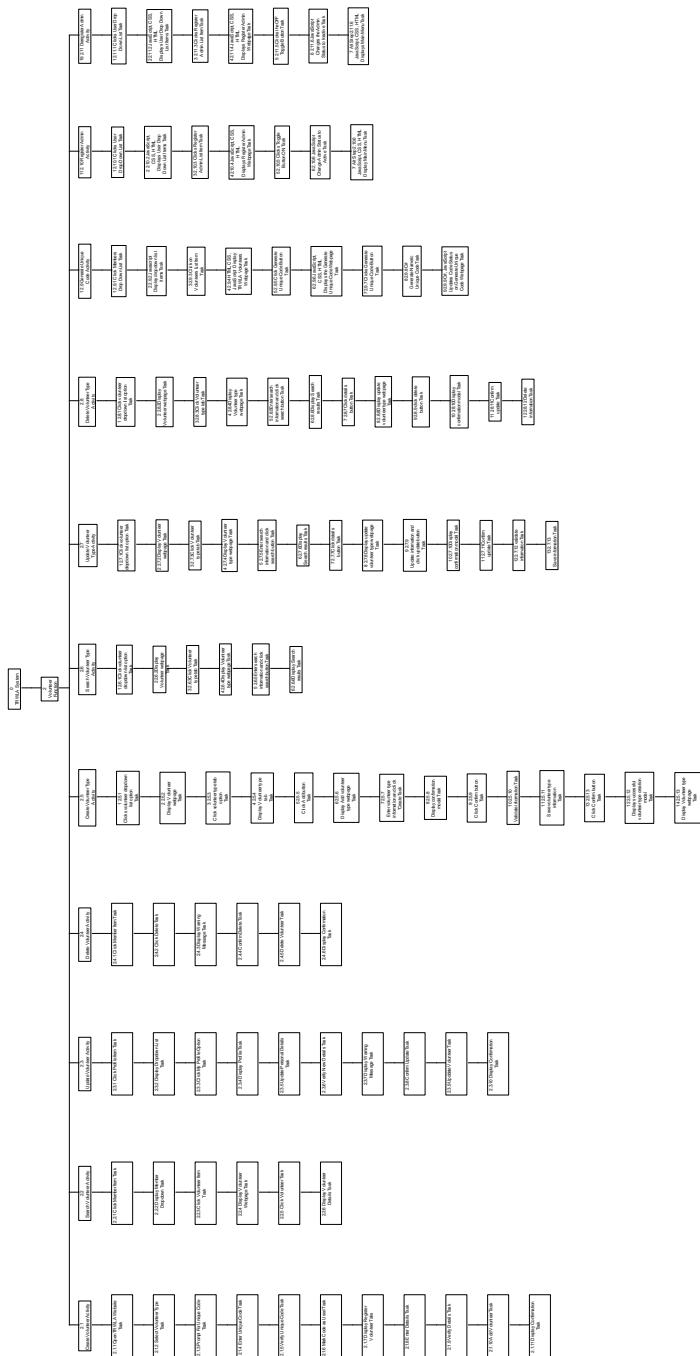


Figure 20: Function 2 Decomposition Diagram

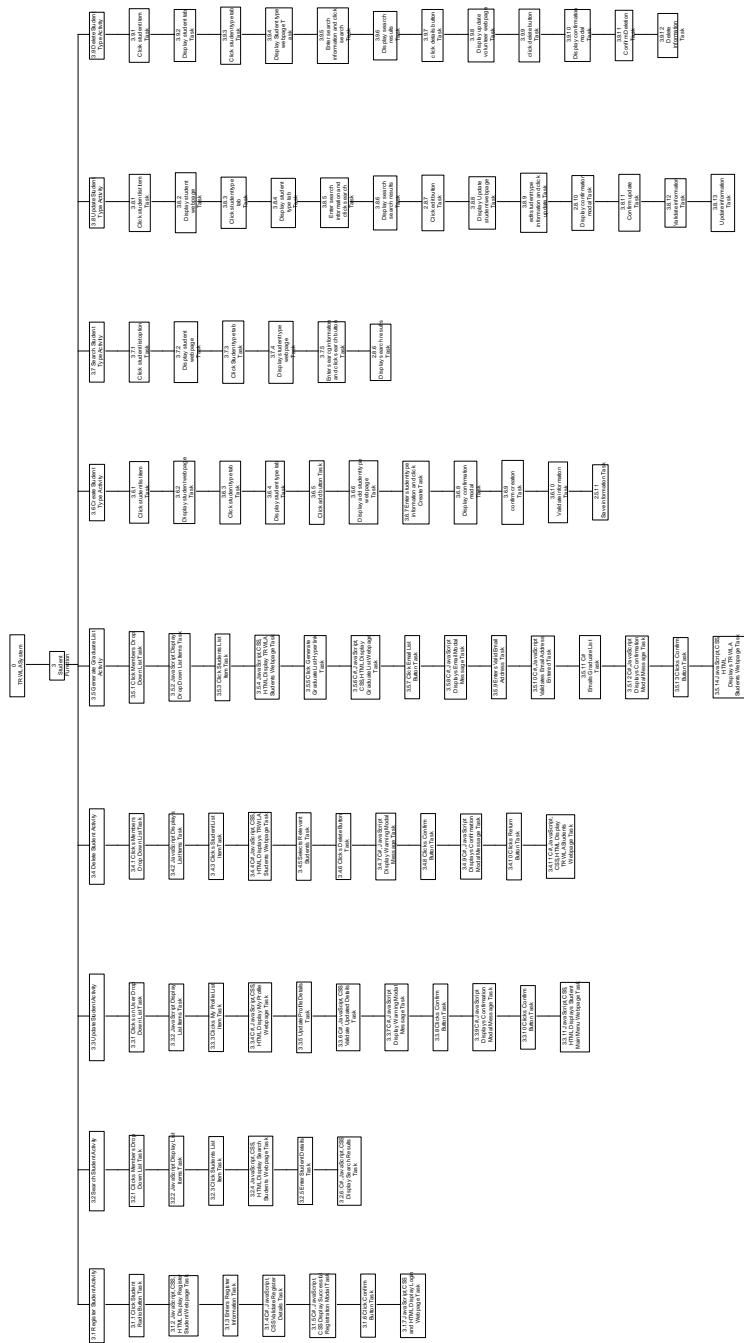


Figure 21: Function 3 Decomposition Diagram

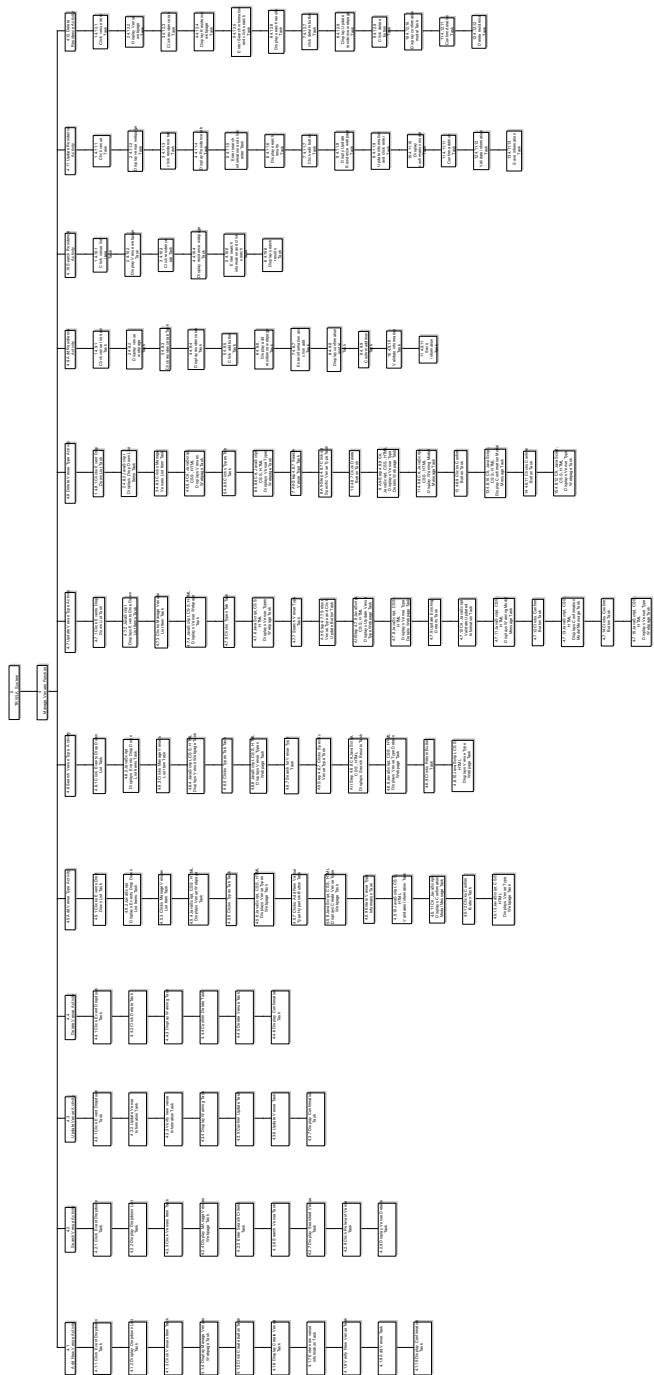


Figure 22: Function 4 Decomposition Diagram

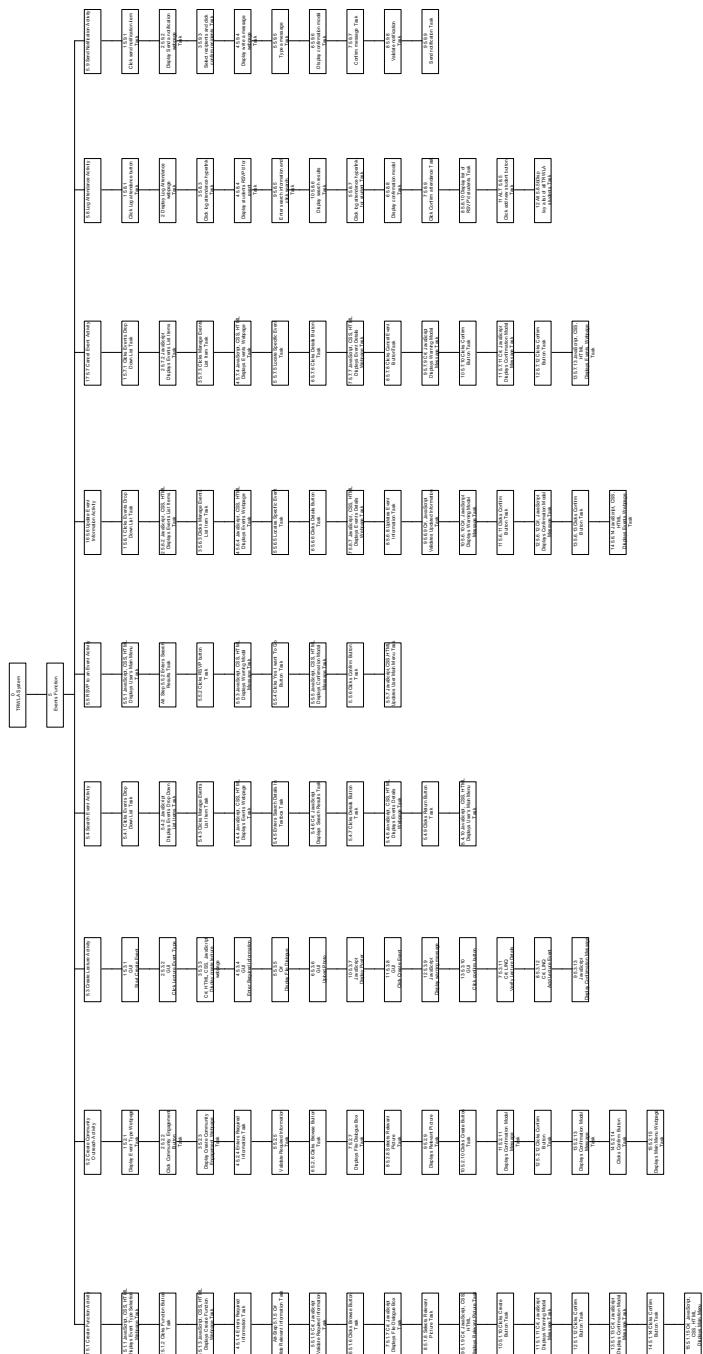


Figure 23: Function 5 Decomposition Diagram

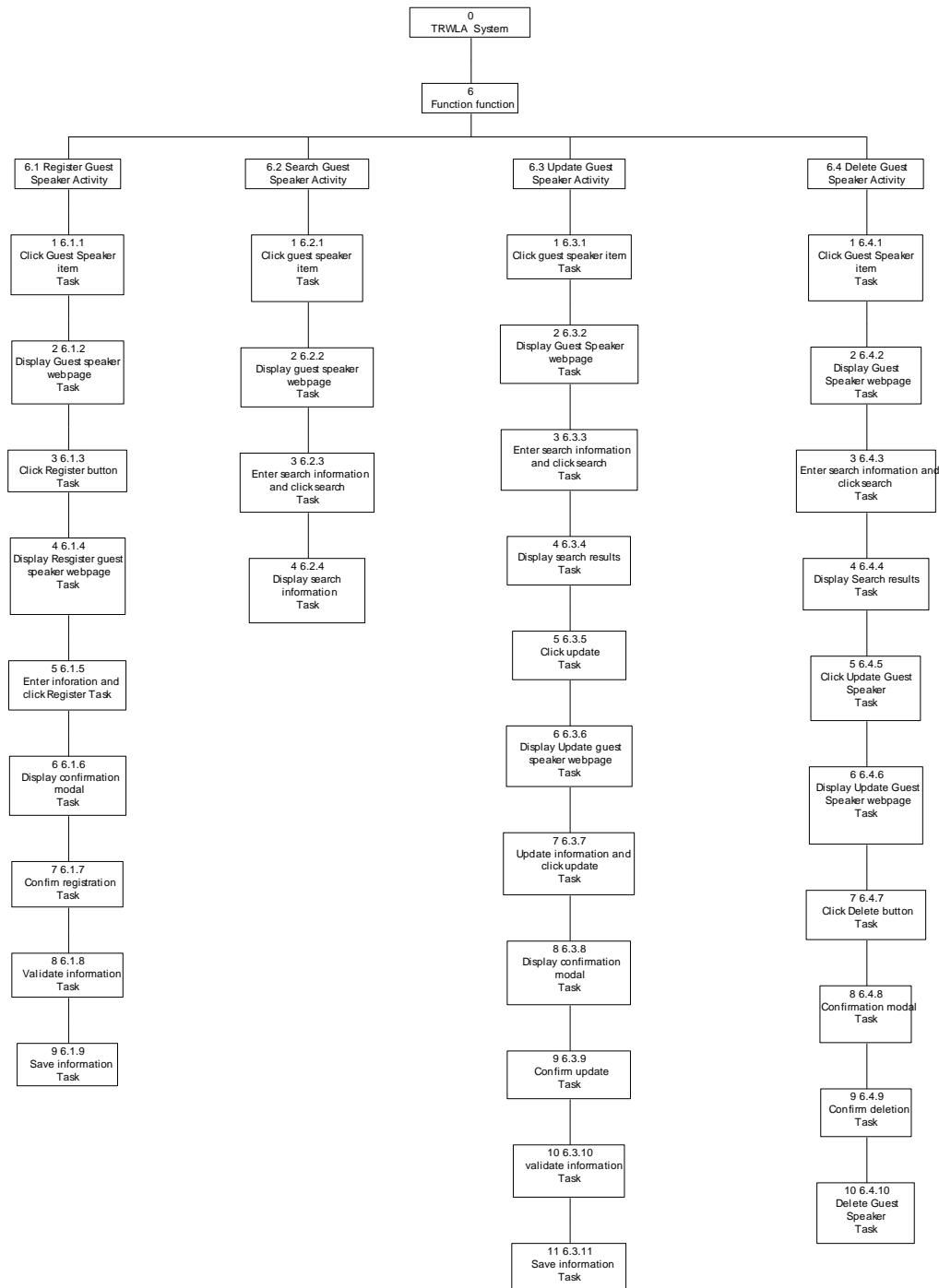


Figure 24: Function 6 Decomposition Diagram

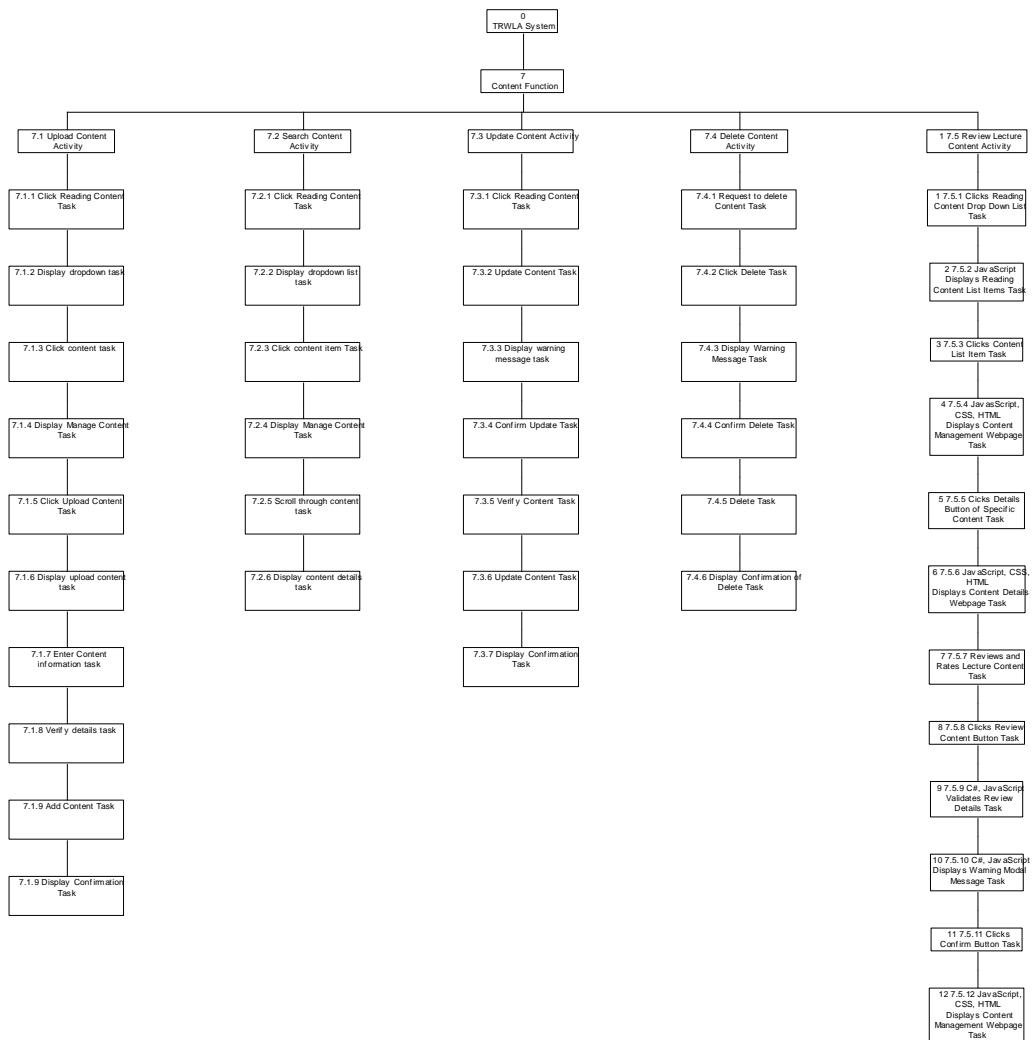


Figure 25: Function 7 Decomposition Diagram

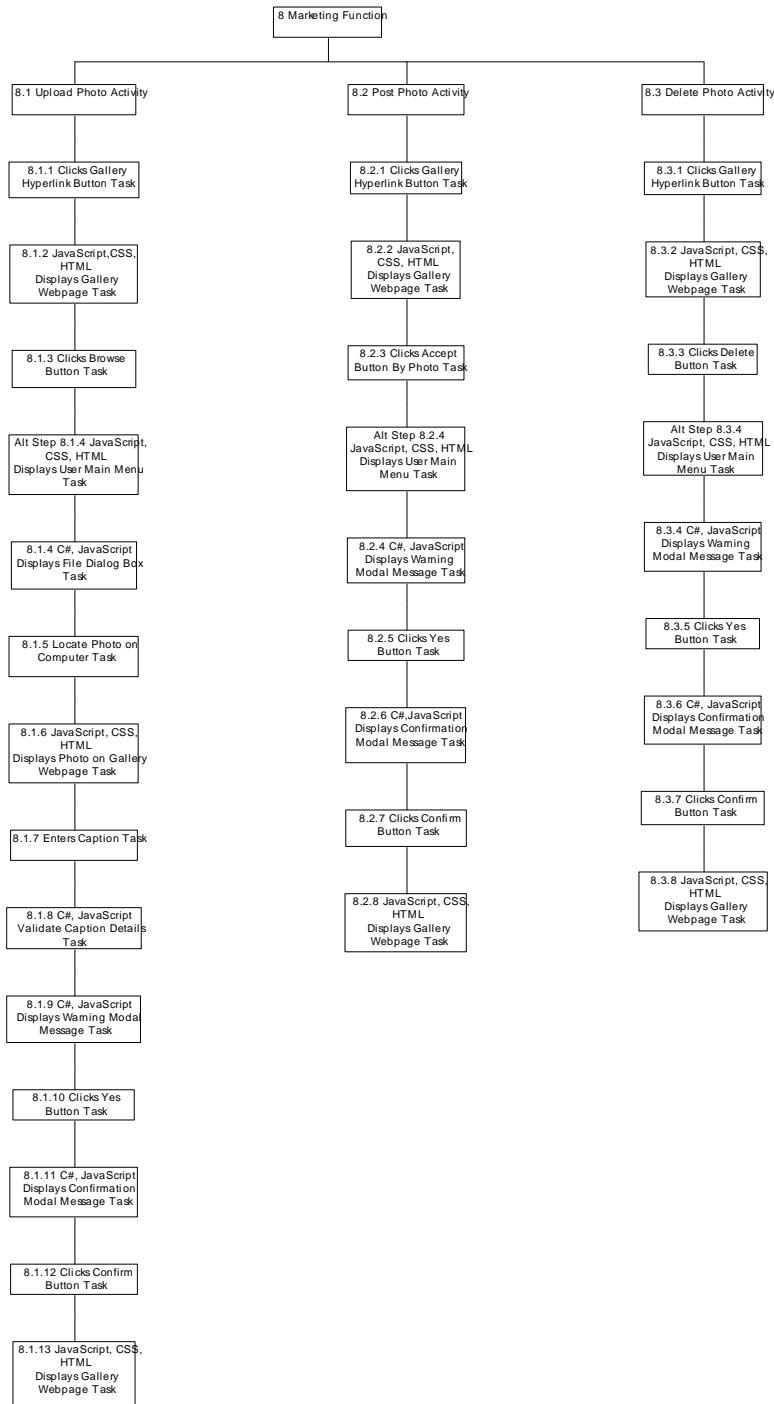


Figure 26: Function 8 Decomposition Diagram

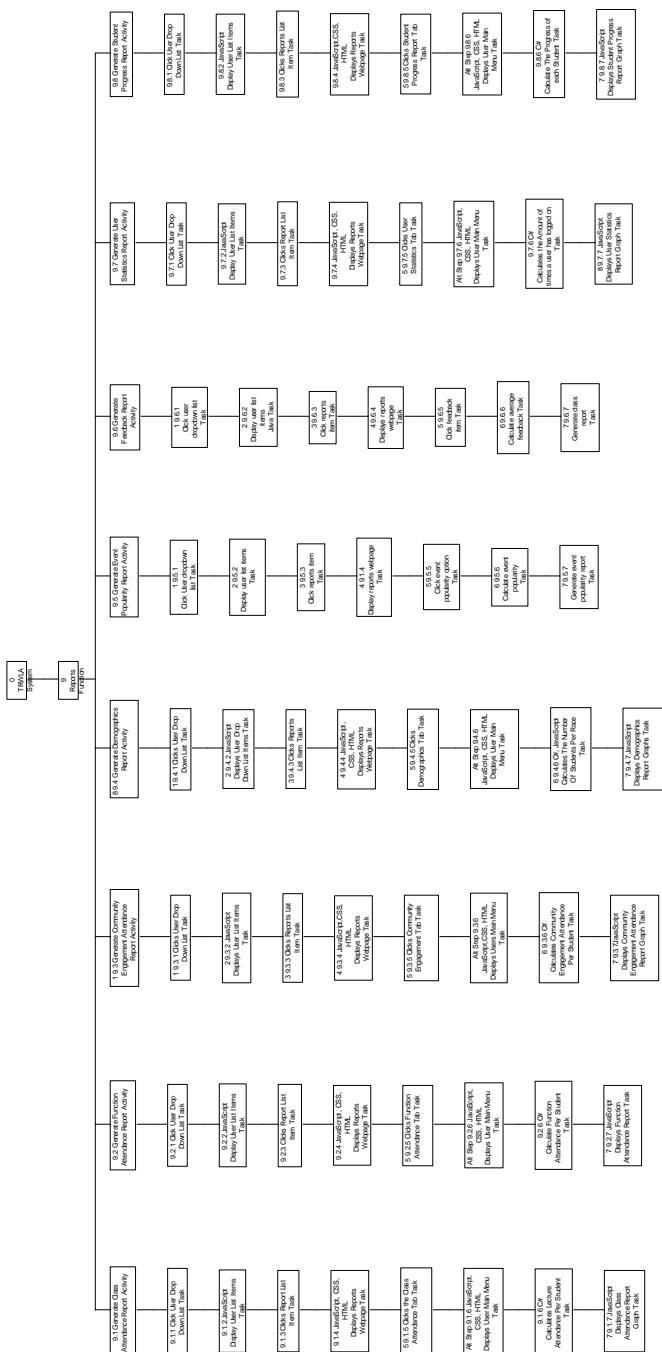


Figure 27: Function 9 Decomposition Diagram

3.4 Physical Data Flow Diagrams

3.4.1 High Level Data Flow Diagrams

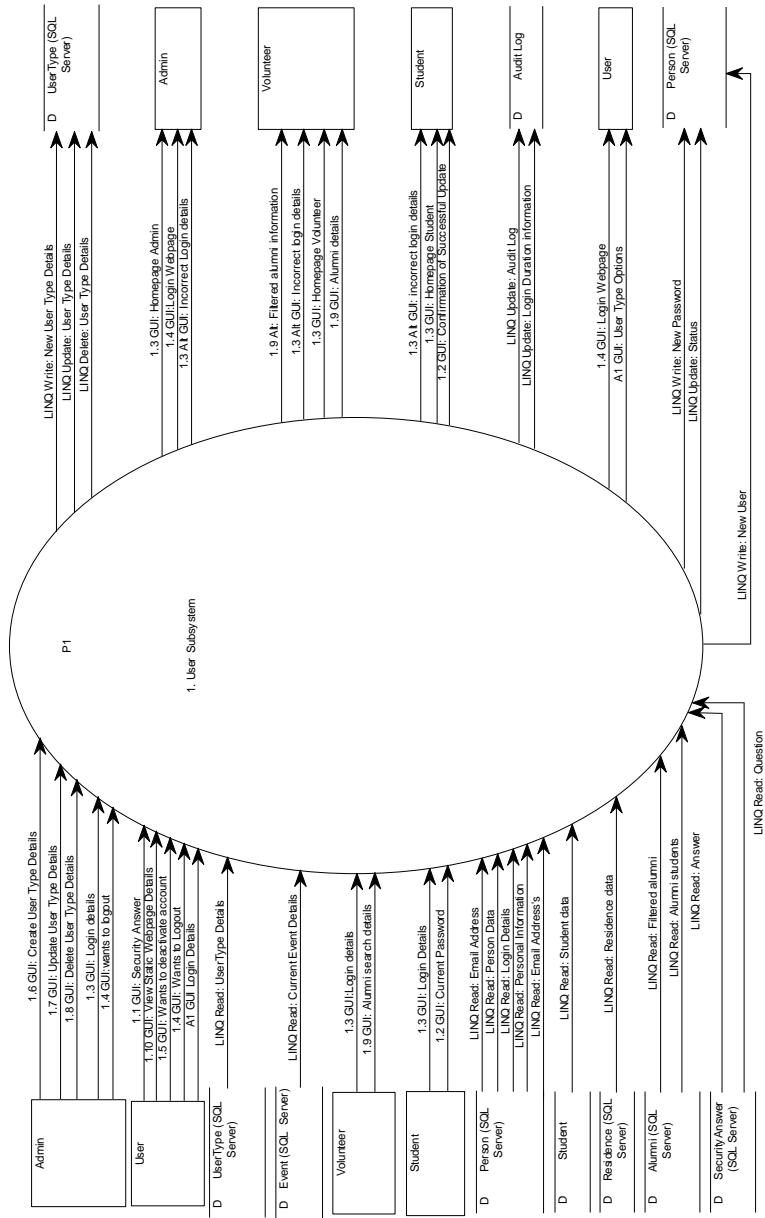


Figure 28: 1 High Level Data Flow Diagram

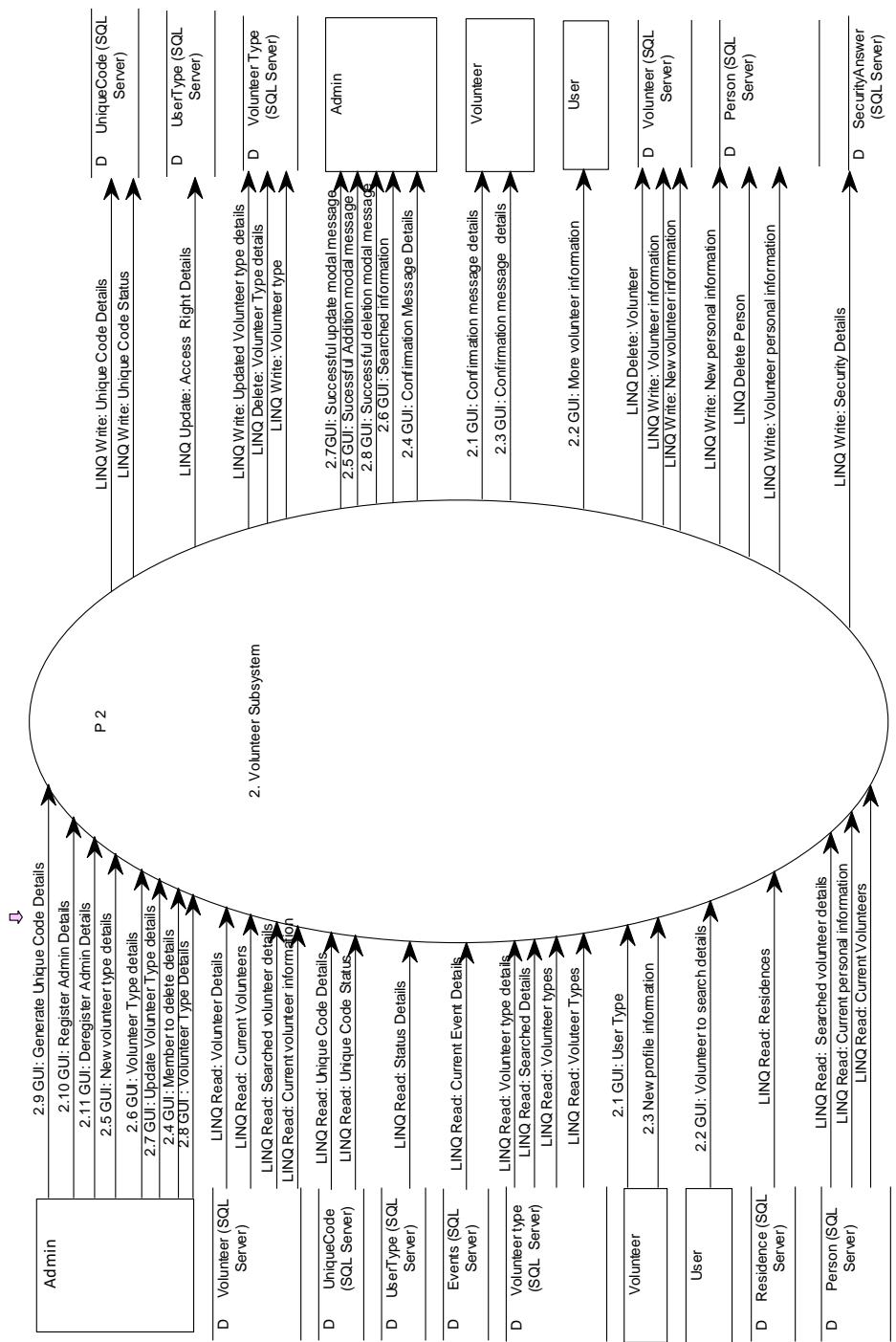


Figure 29: 2 High Level Data Flow Diagram

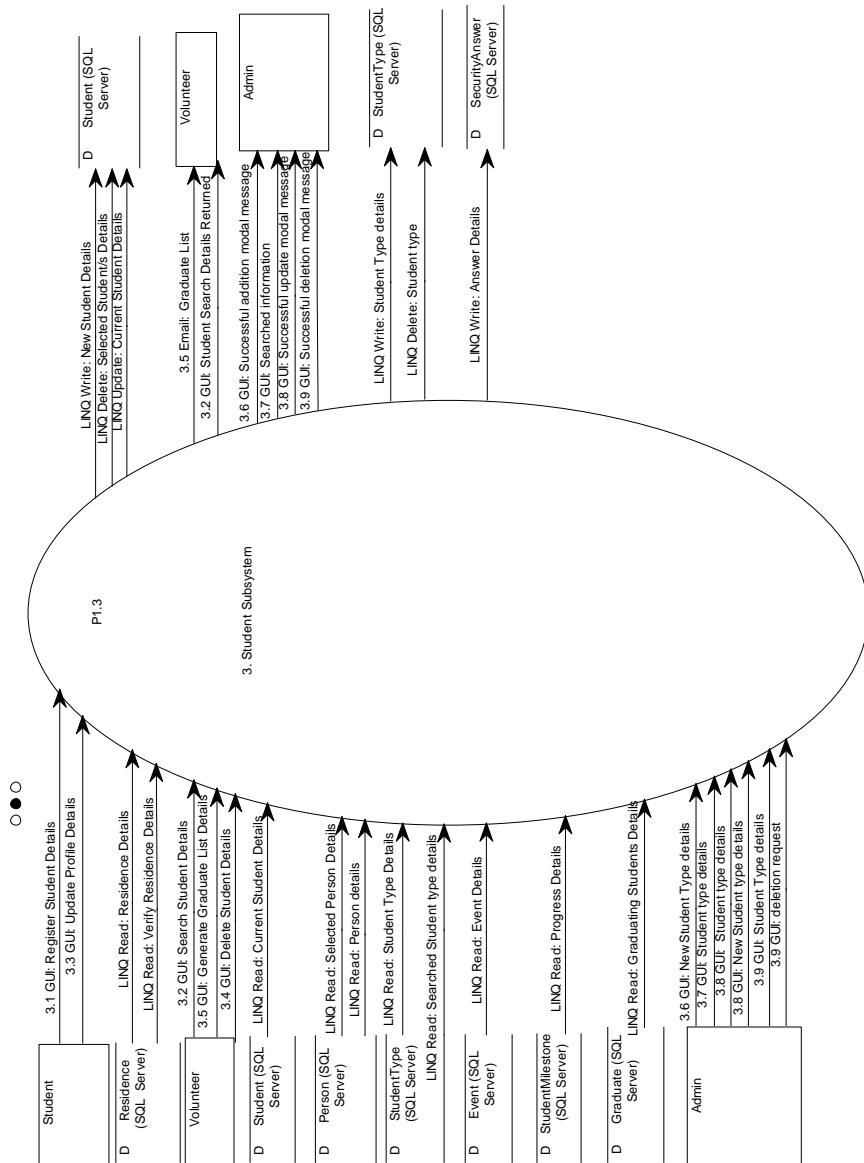


Figure 30: 3 High Level Data Flow Diagram

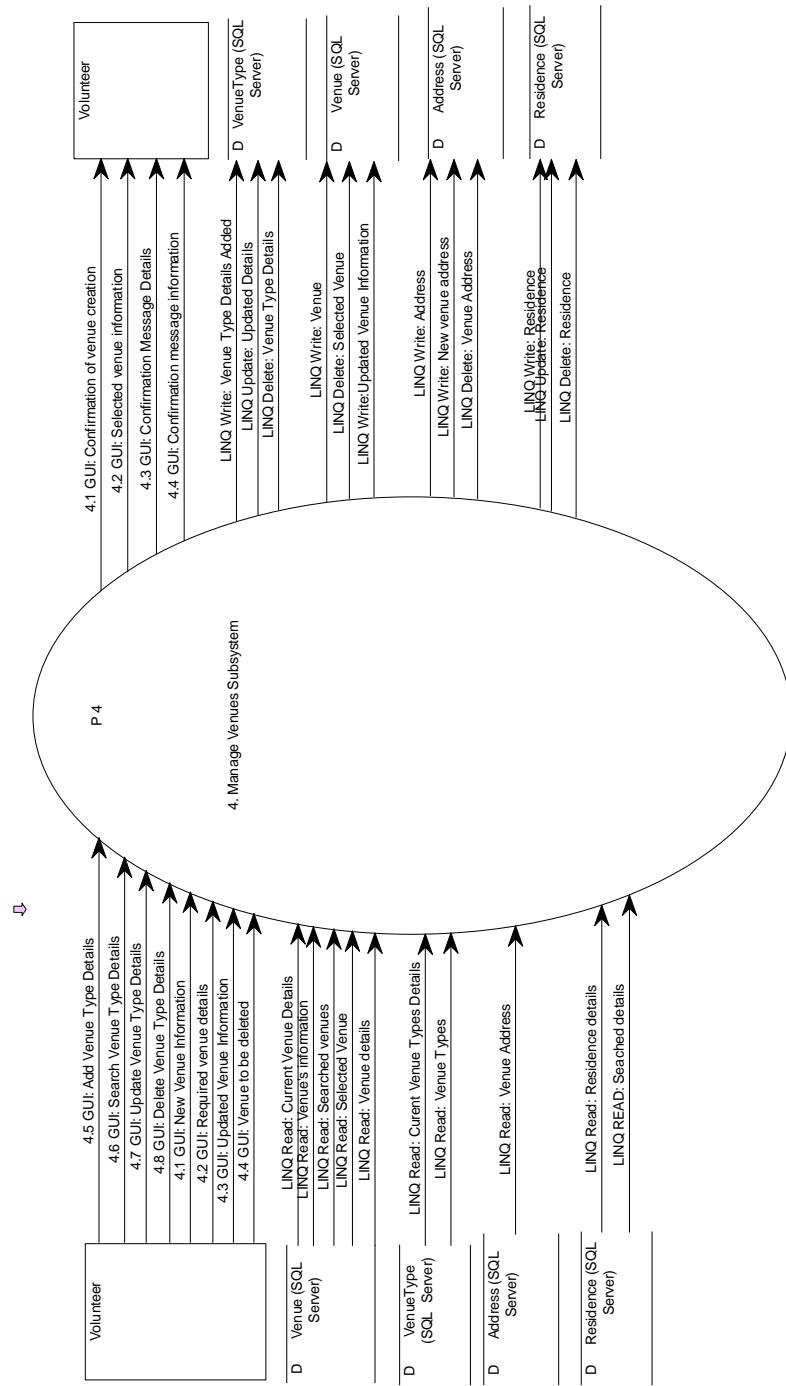


Figure 31: 4 High Level Data Flow Diagram

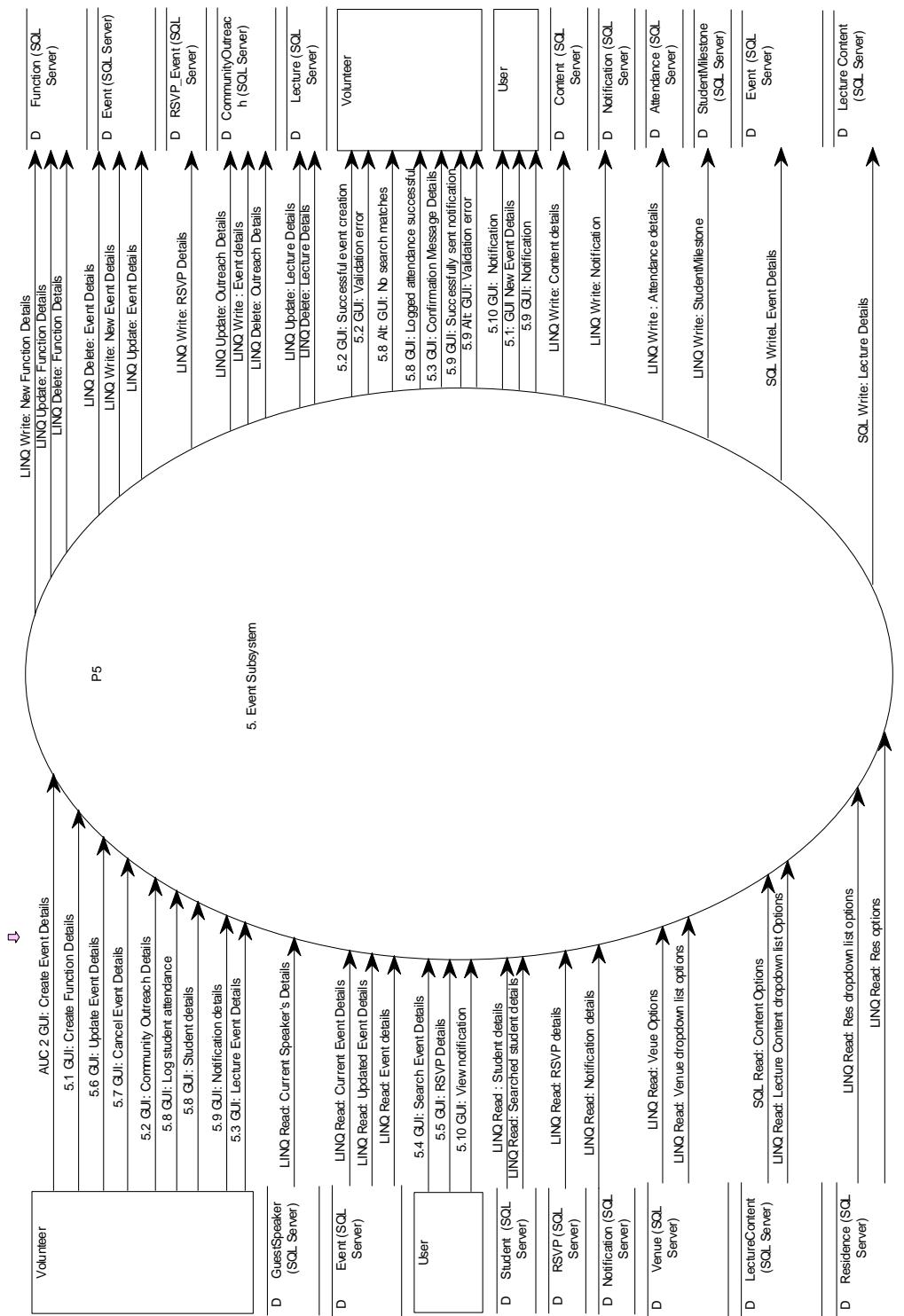


Figure 32: 5 High Level Data Flow Diagram

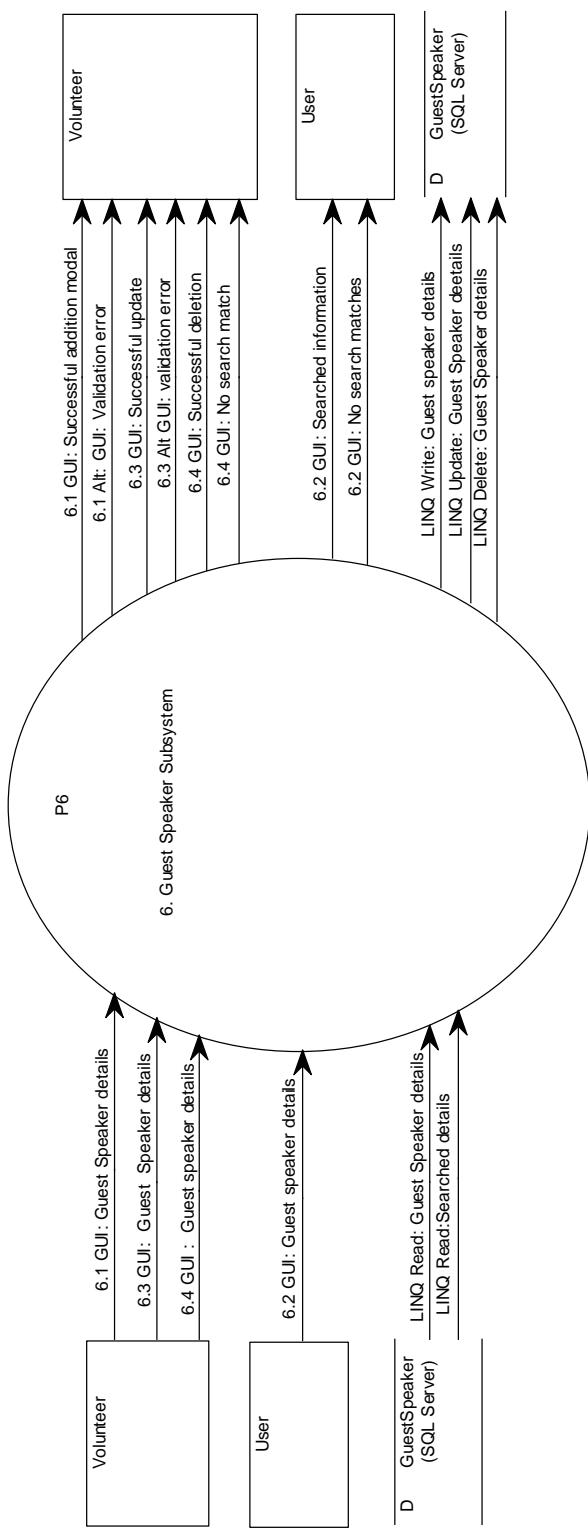


Figure 33: 6 High Level Data Flow Diagrams

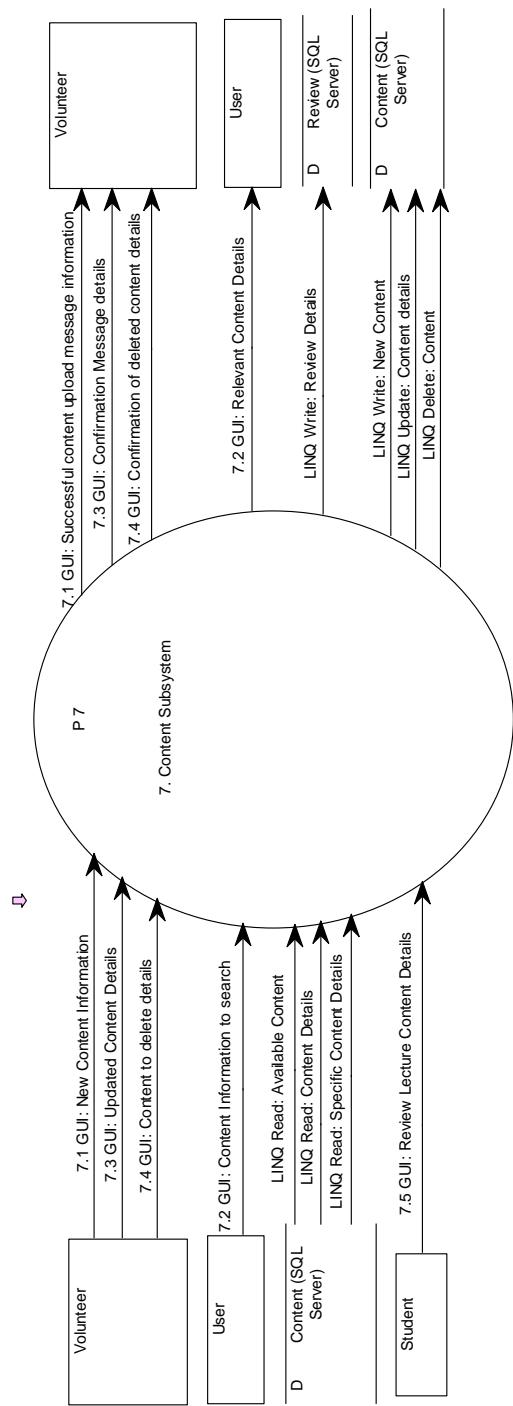


Figure 34: 7 High Level Data Flow Diagram

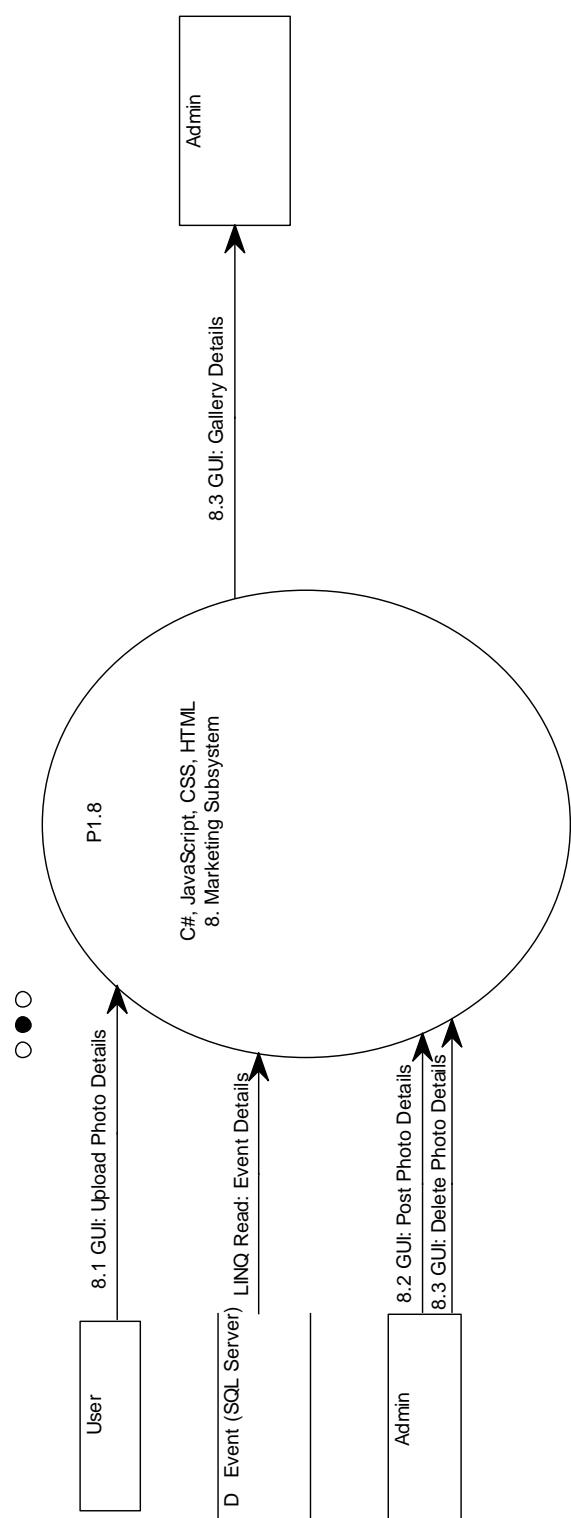


Figure 35: 8 High Level Data Flow Diagram

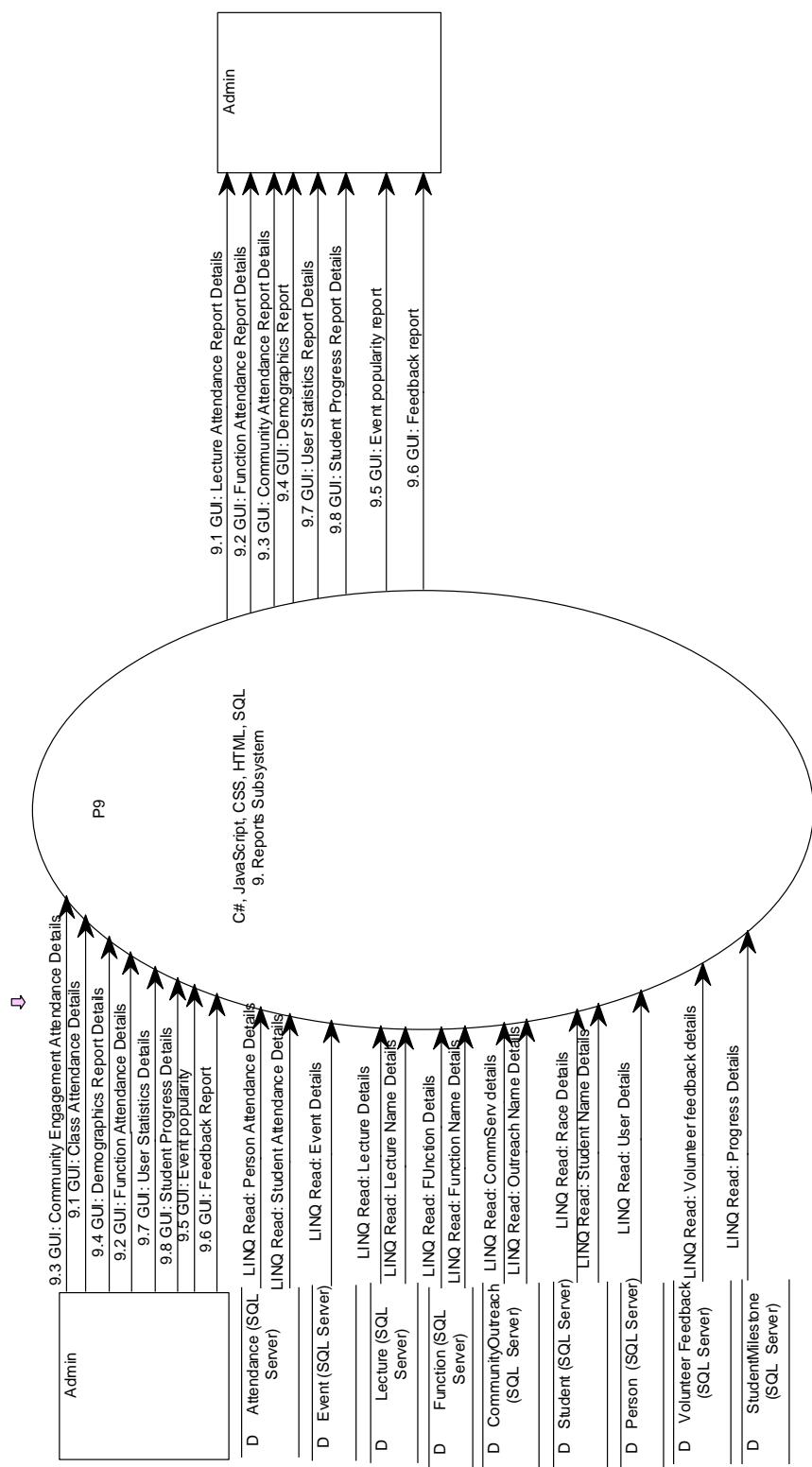


Figure 36: 9 High Level Data Flow Diagrams

3.4.2 Mid-Level Data Flow Diagrams

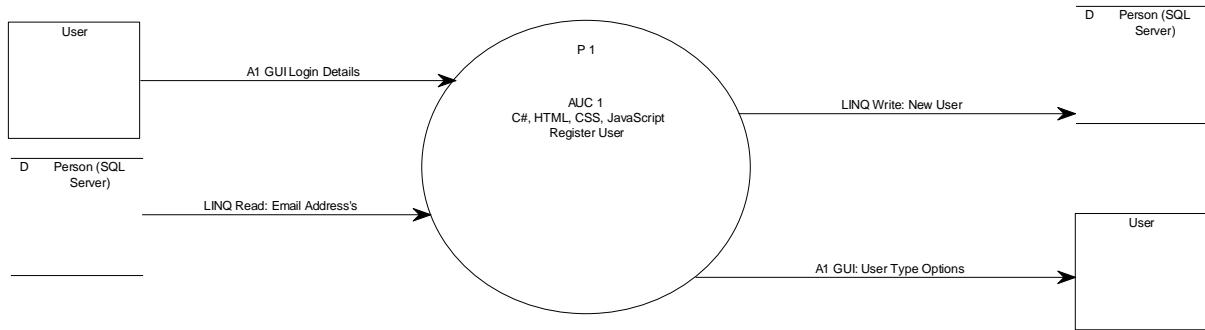


Figure 37: AUC 1 Mid-Level Data Flow Diagram

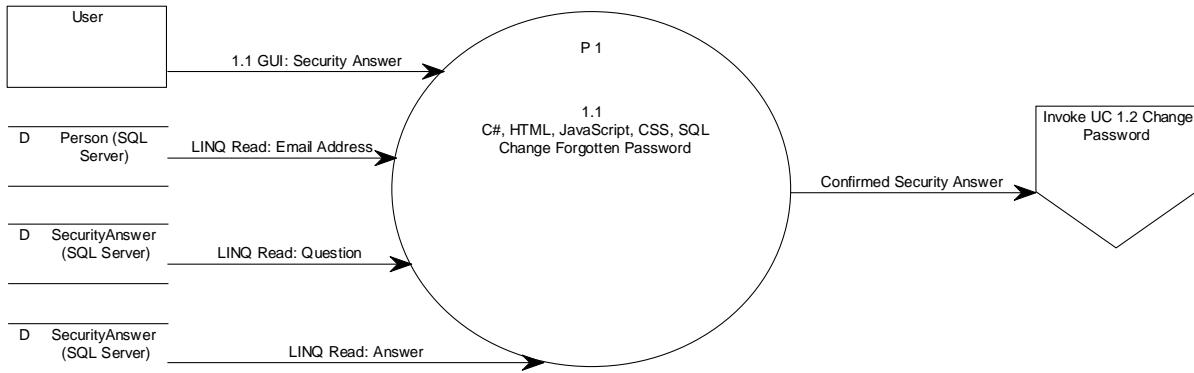


Figure 38: 1.1 Mid-Level Data Flow Diagram

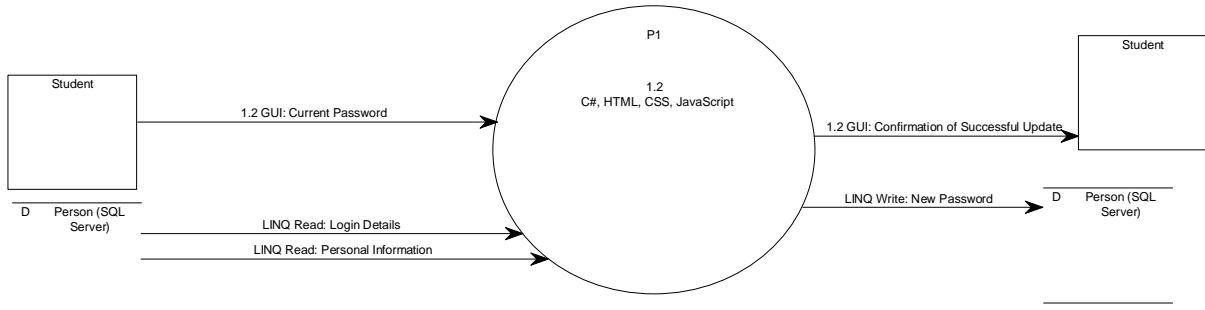


Figure 39: 1.2 Mid-Level Data Flow Diagram

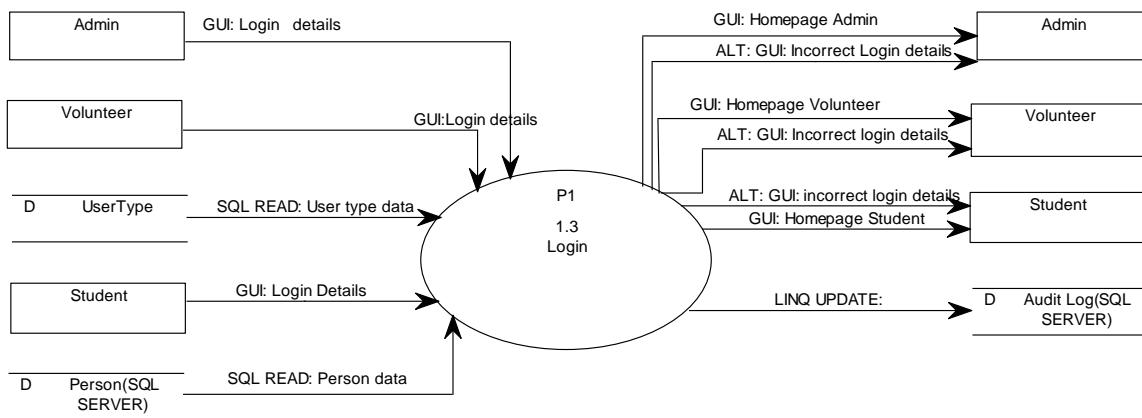


Figure 40: 1.3 Mid-Level Data Flow Diagram

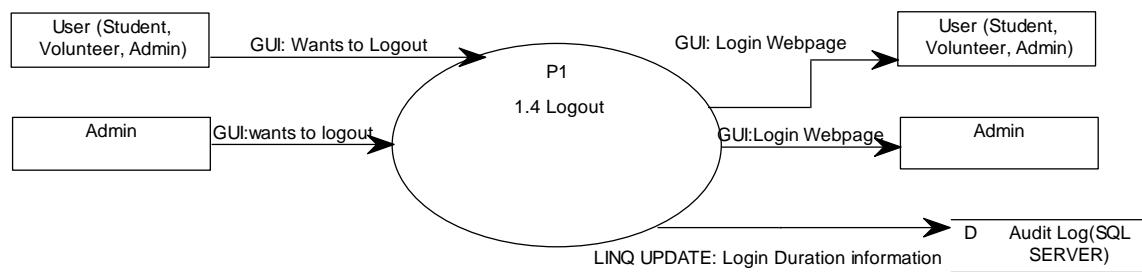


Figure 41: 1.4 Mid-Level Data Flow Diagram

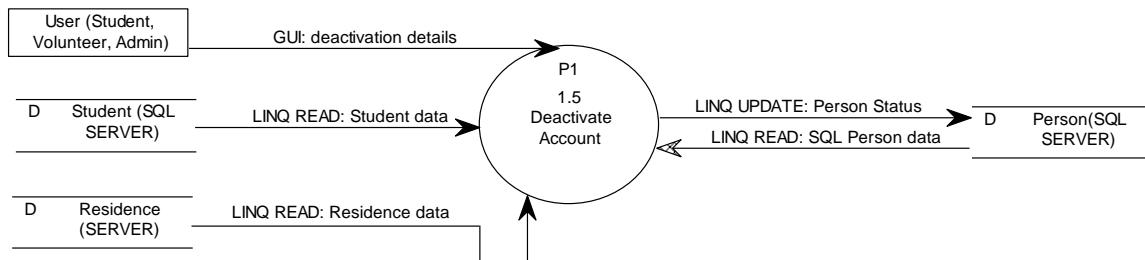


Figure 42: 1.5 Mid-Level Data Flow Diagram

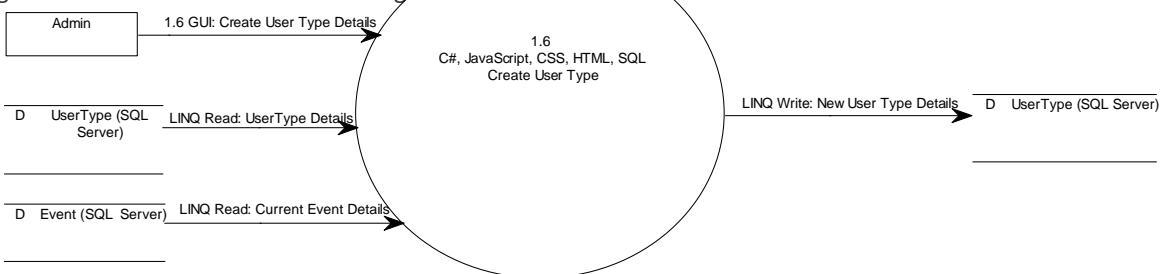


Figure 43: 1.6 Mid-Level Data Flow Diagram

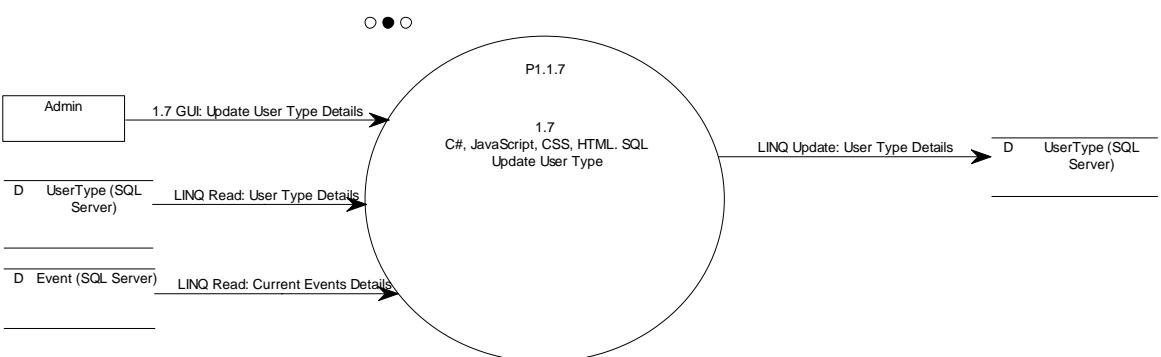


Figure 44: 1.7 Mid-Level Data Flow Diagram

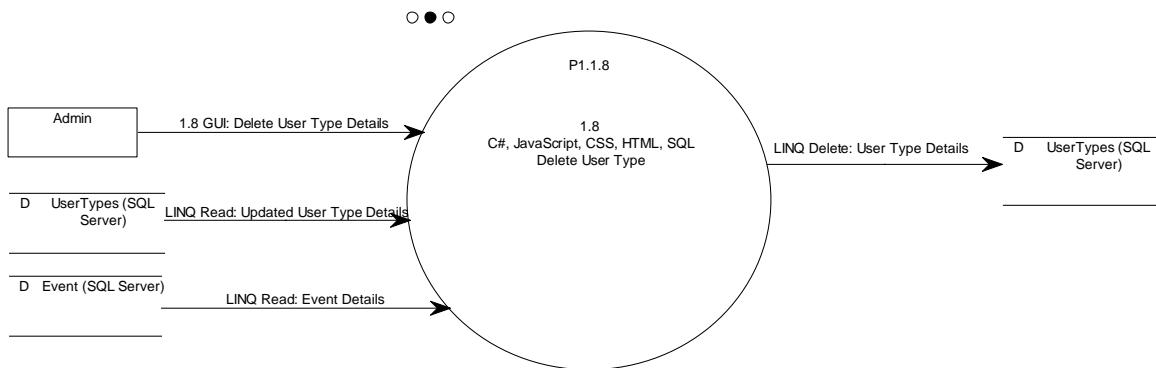


Figure 45: 1.8 Mid-Level Data Flow Diagram

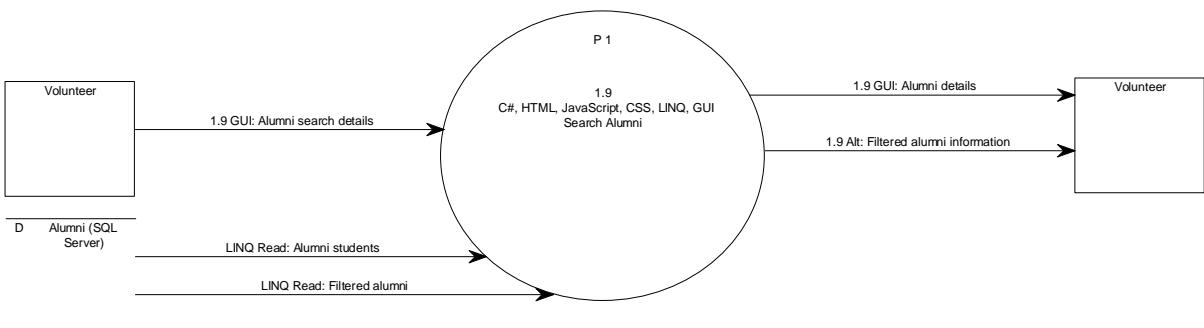


Figure 46: 1.9 Mid-Level Data Flow Diagram

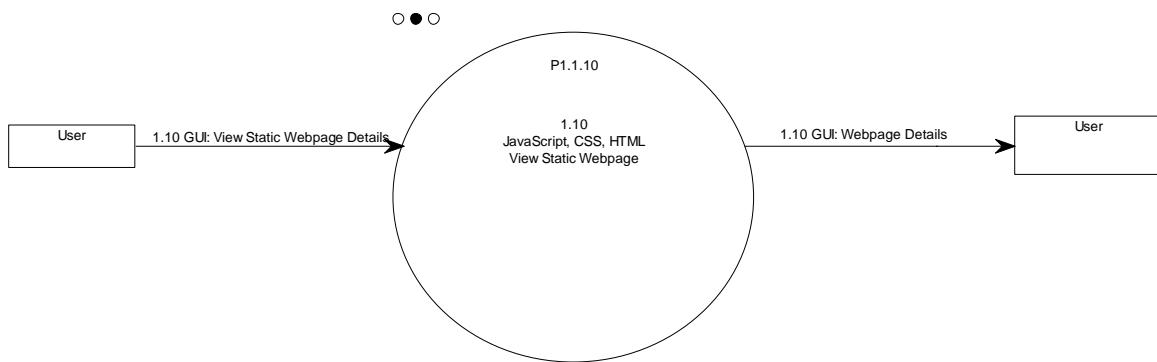


Figure 47: 1.10 Mid-Level Data Flow Diagram

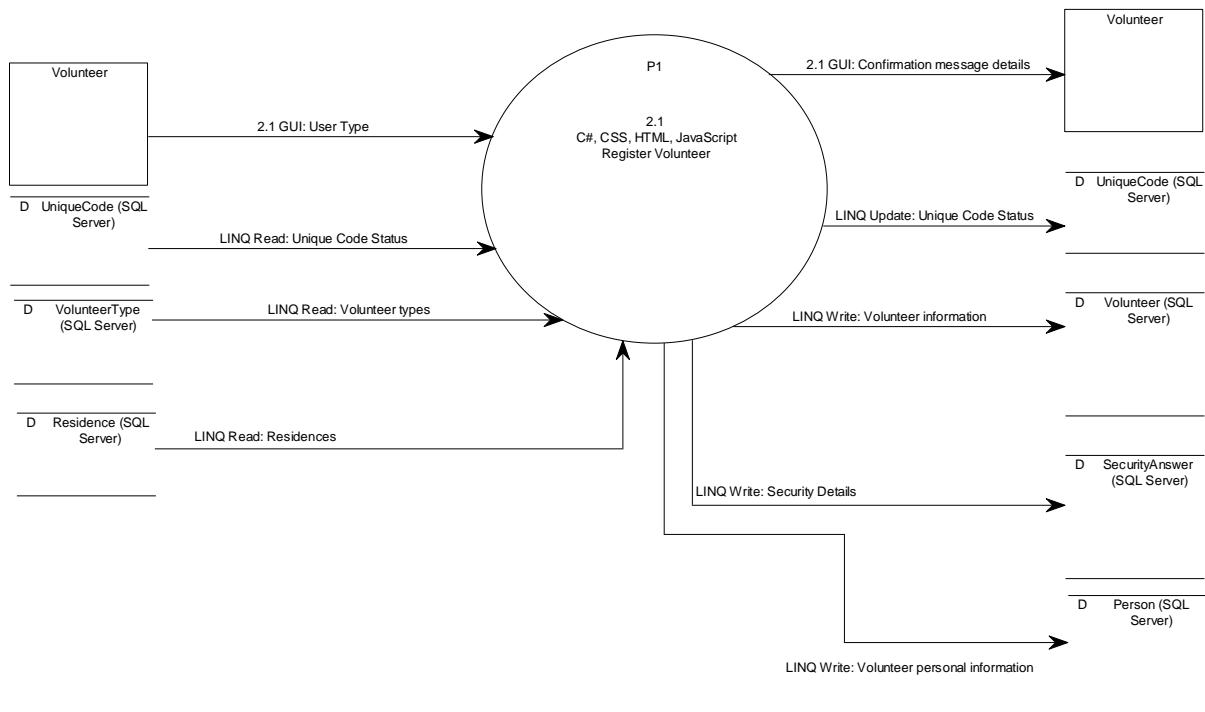


Figure 48: 2.1 Mid-Level Data Flow Diagram

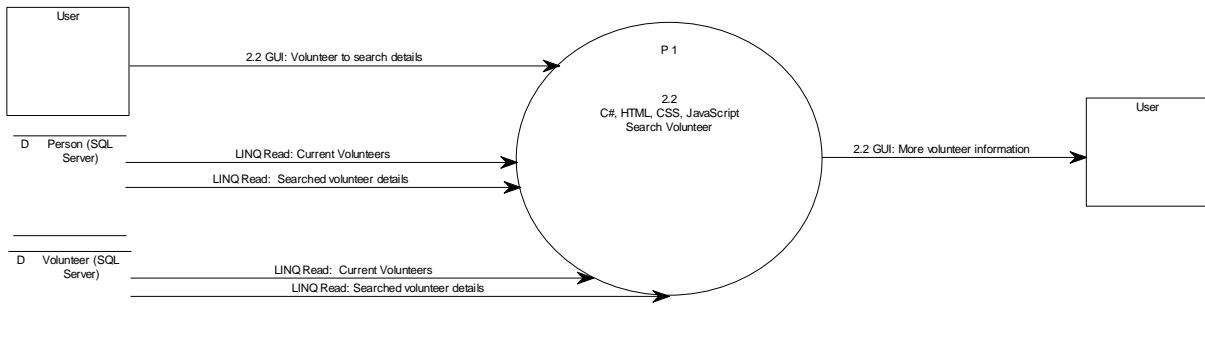


Figure 49: 2.2 Mid-Level Data Flow Diagram

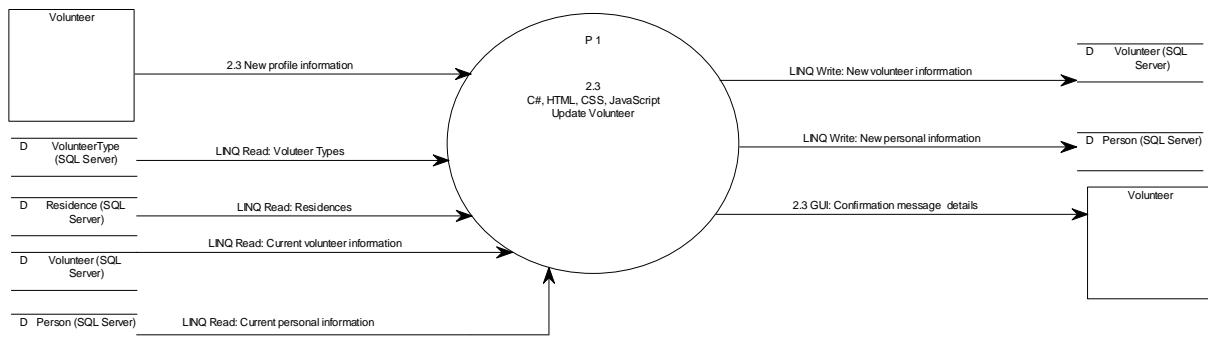


Figure 50: 2.3 Mid-Level Data Flow Diagram

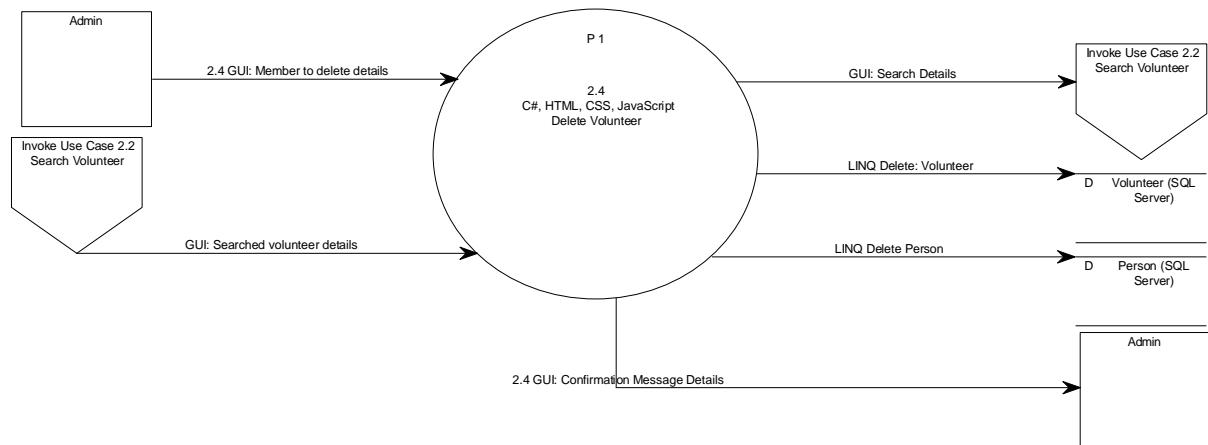


Figure 51: 2.4 Mid-Level Data Flow Diagram

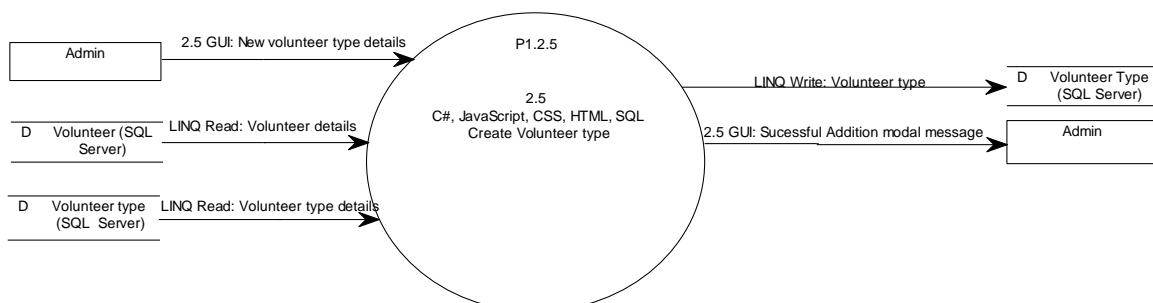


Figure 52: 2.5 Mid-Level Data Flow Diagram

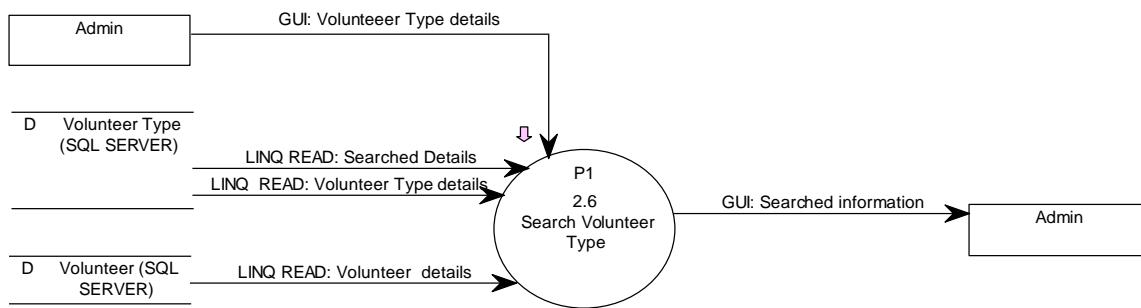


Figure 53: 2.6 Mid-Level Data Flow Diagram

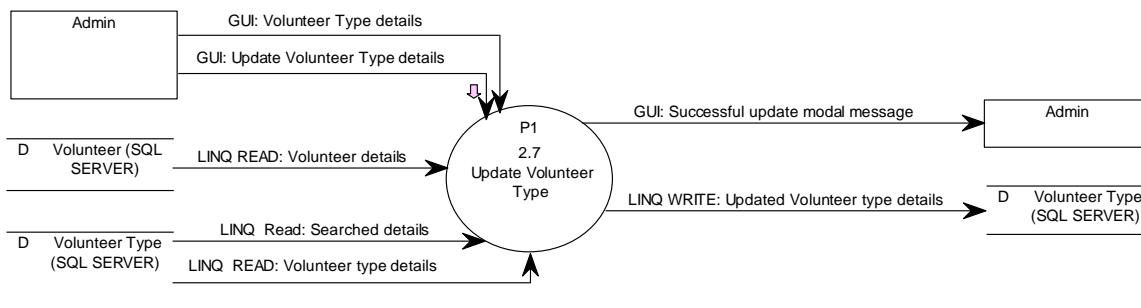


Figure 54: 2.7 Mid-Level Data Flow Diagram

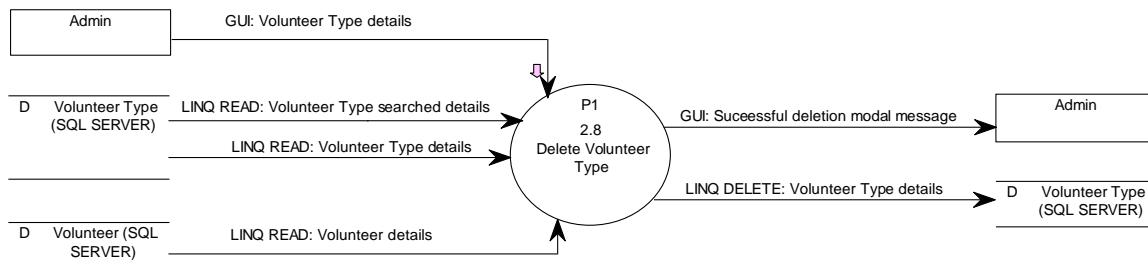


Figure 55: 2.8 Mid-Level Data Flow Diagram

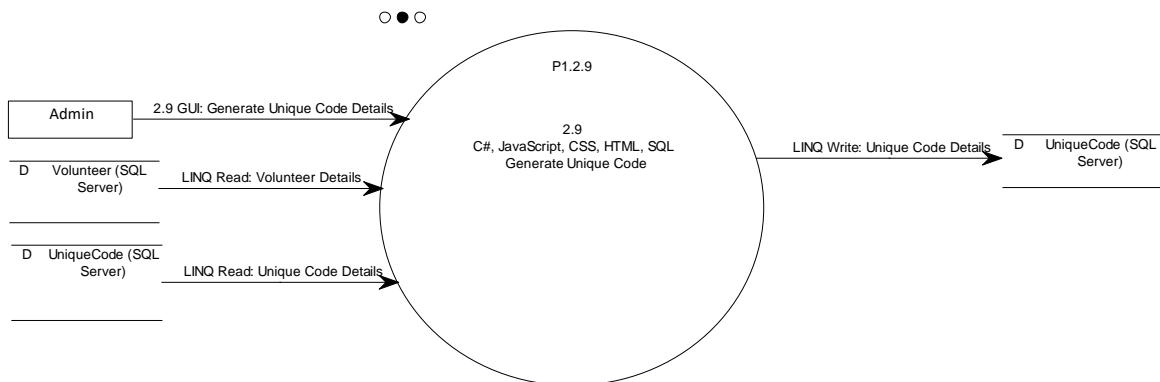


Figure 56: 2.9 Mid-Level Data Flow Diagram

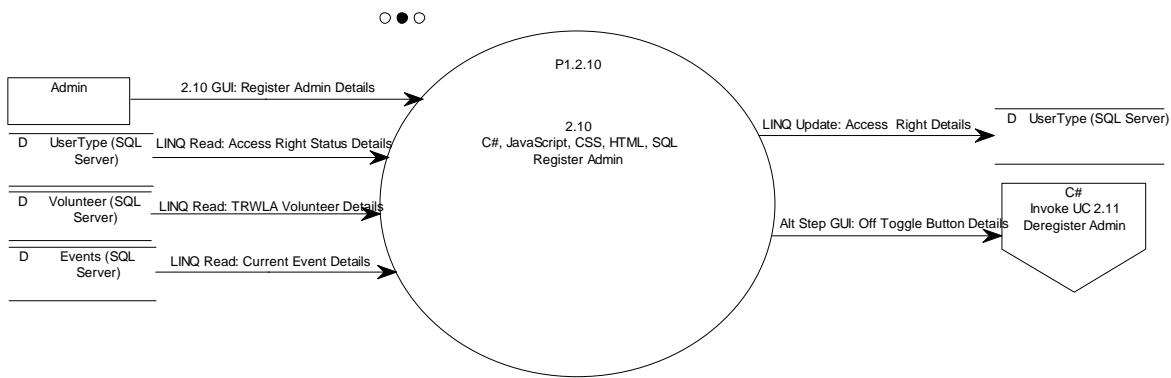


Figure 57: 2.10 Mid-Level Data Flow Diagram

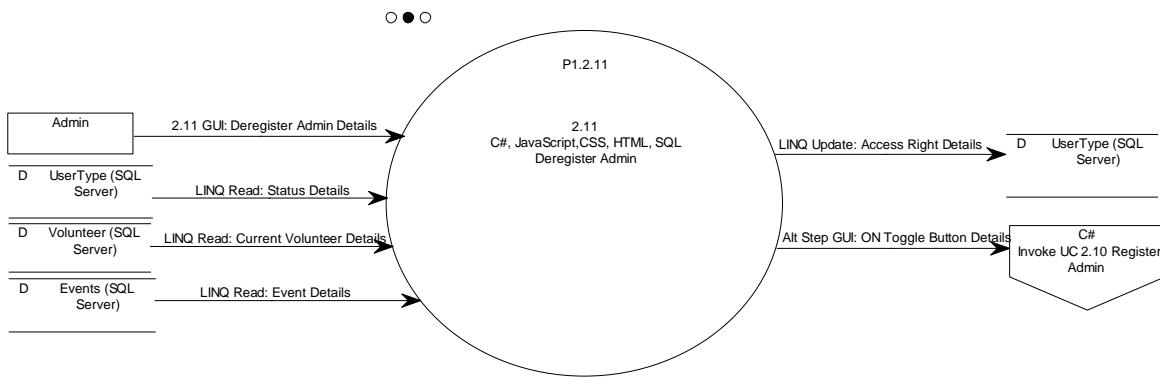


Figure 58: 2.11 Mid-Level Data Flow Diagram

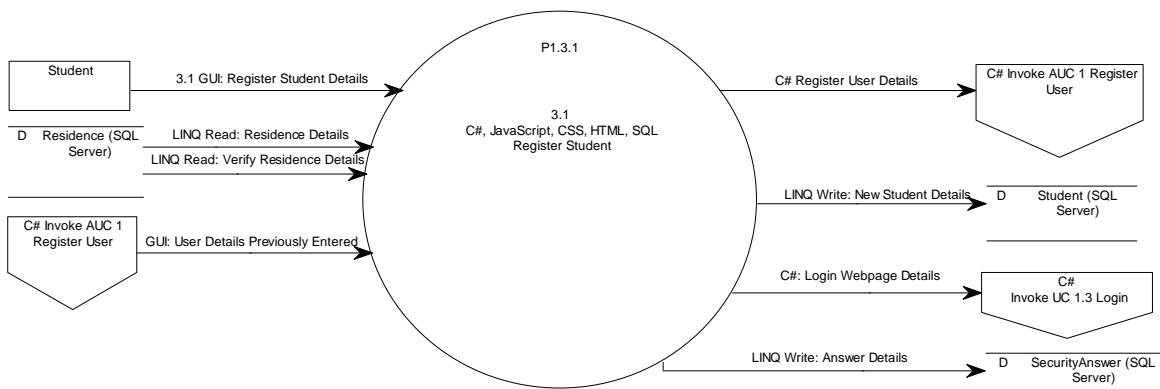


Figure 59: 3.1 Mid-Level Data Flow Diagram

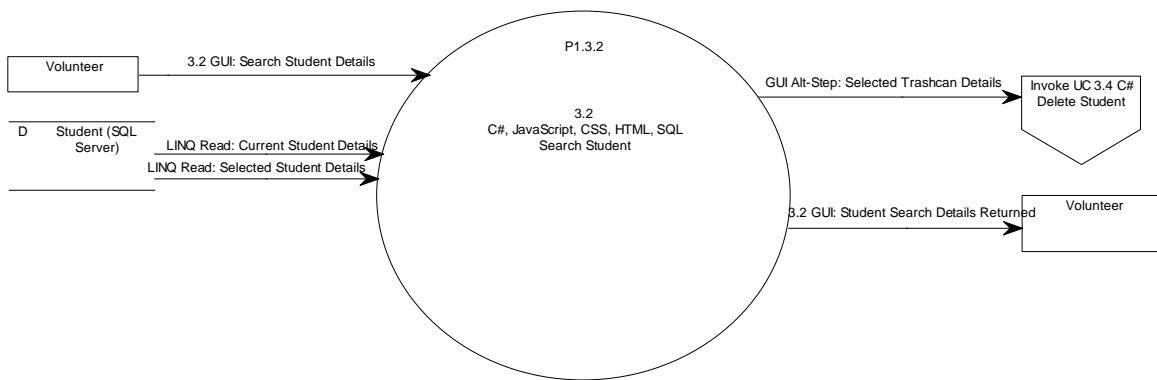


Figure 60: 3.2 Mid-Level Data Flow Diagram

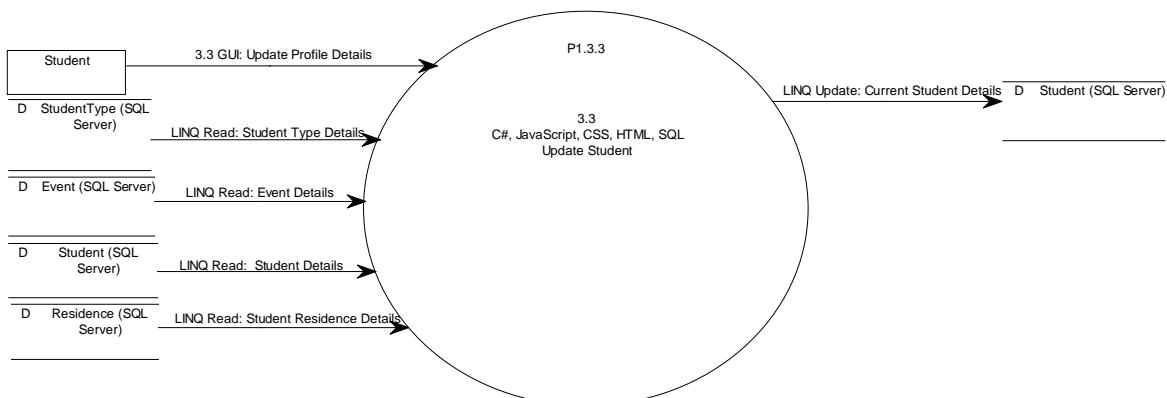


Figure 61: 3.3 Mid-Level Data Flow Diagram

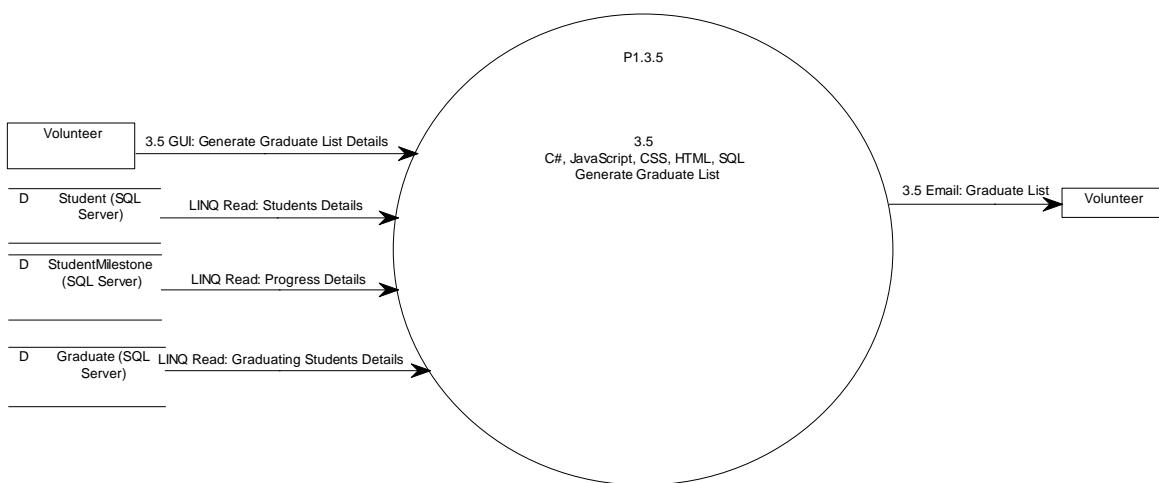
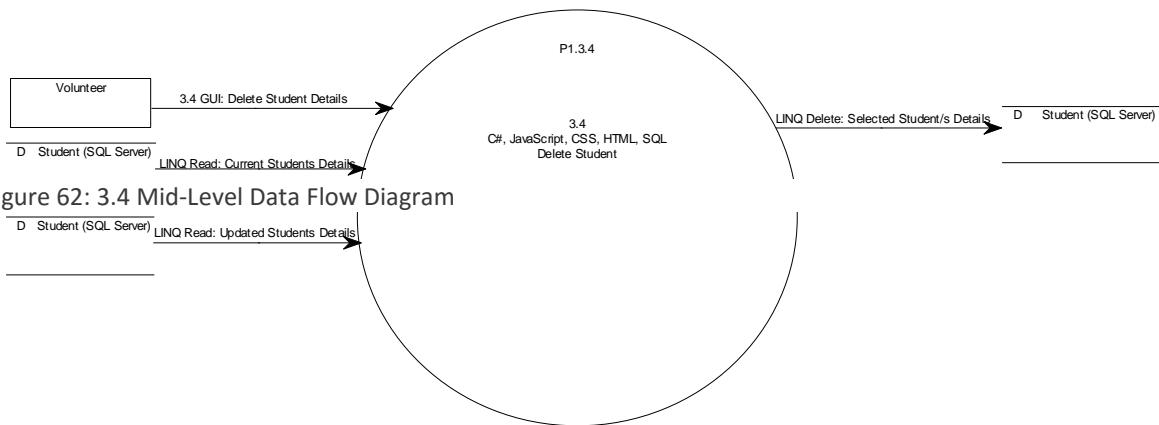


Figure 63: 3.5 Mid-Level Data Flow Diagram

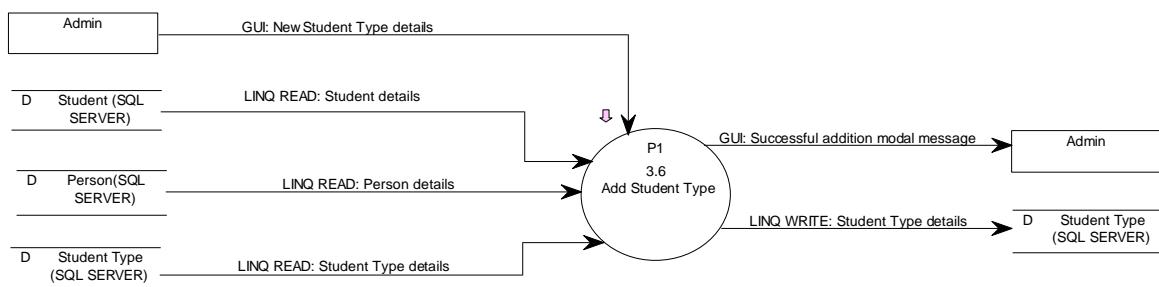


Figure 64: 3.6 Mid-Level Data Flow Diagram

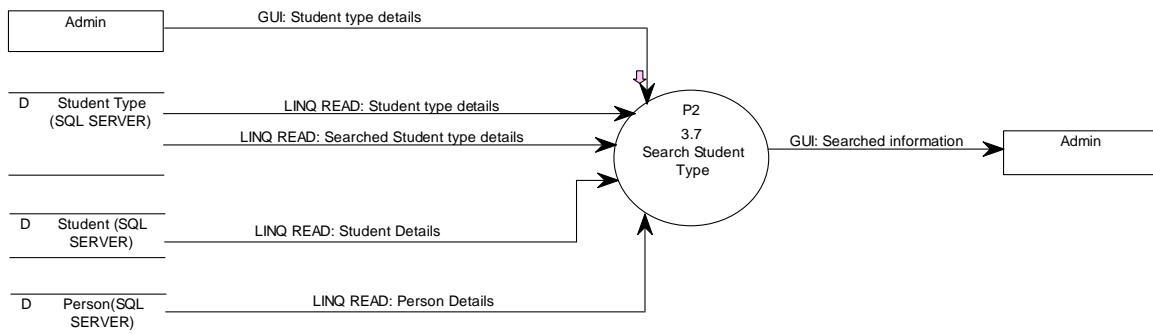


Figure 65: 3.7 Mid-Level Data Flow Diagram

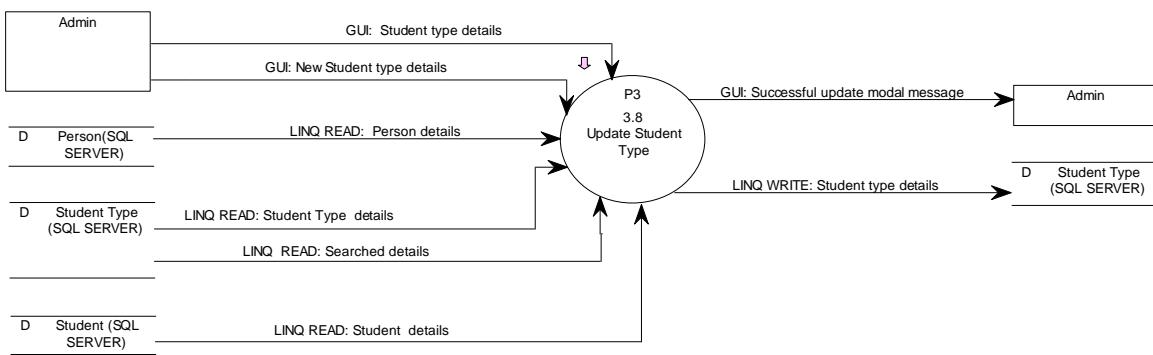


Figure 66: 3.8 Mid-Level Data Flow Diagram

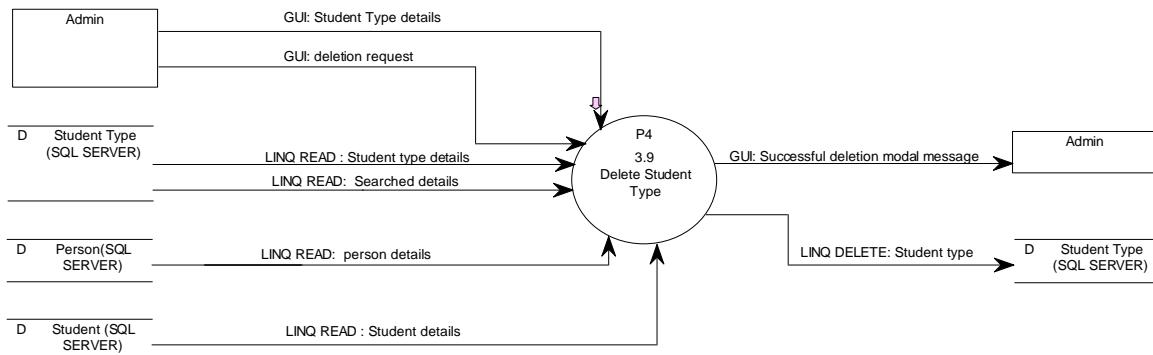


Figure 67: 3.9 Mid-Level Data Flow Diagram

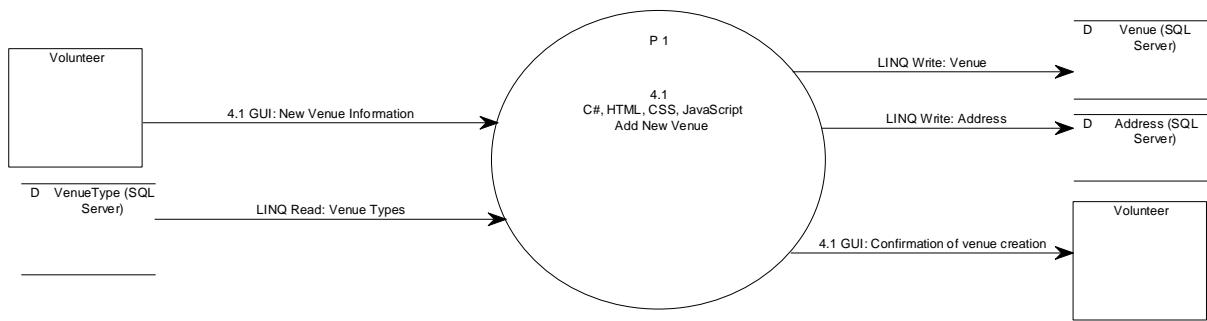


Figure 68: 4.1 Mid-Level Data Flow Diagram

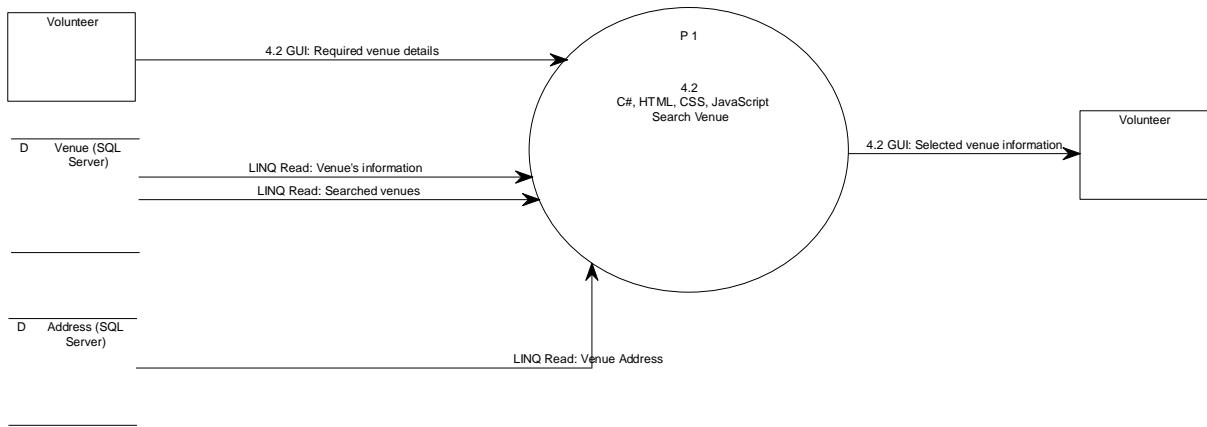


Figure 69: 4.2 Mid-Level Data Flow Diagram

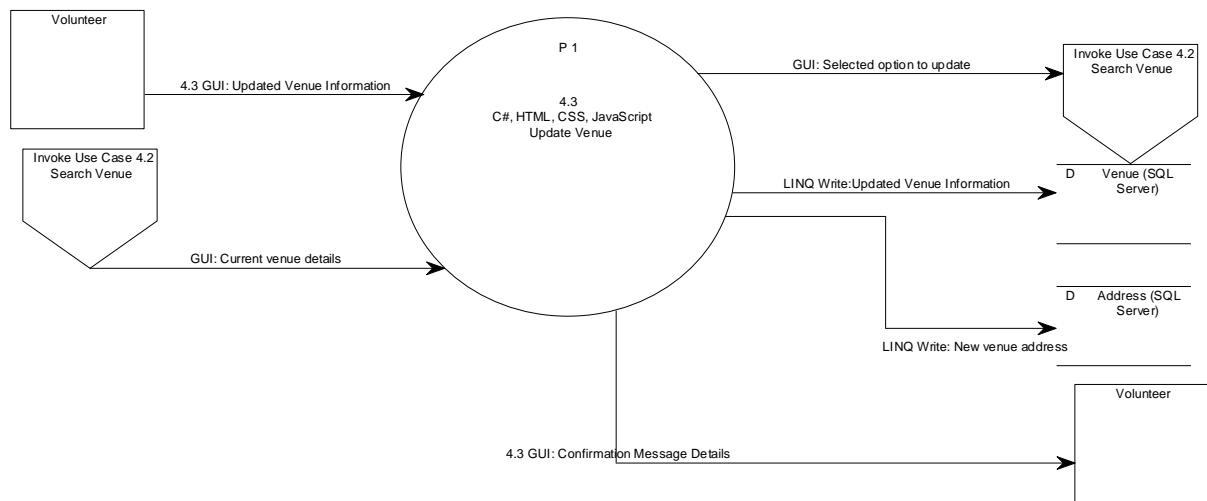


Figure 70: 4.3 Mid-Level Data Flow Diagram

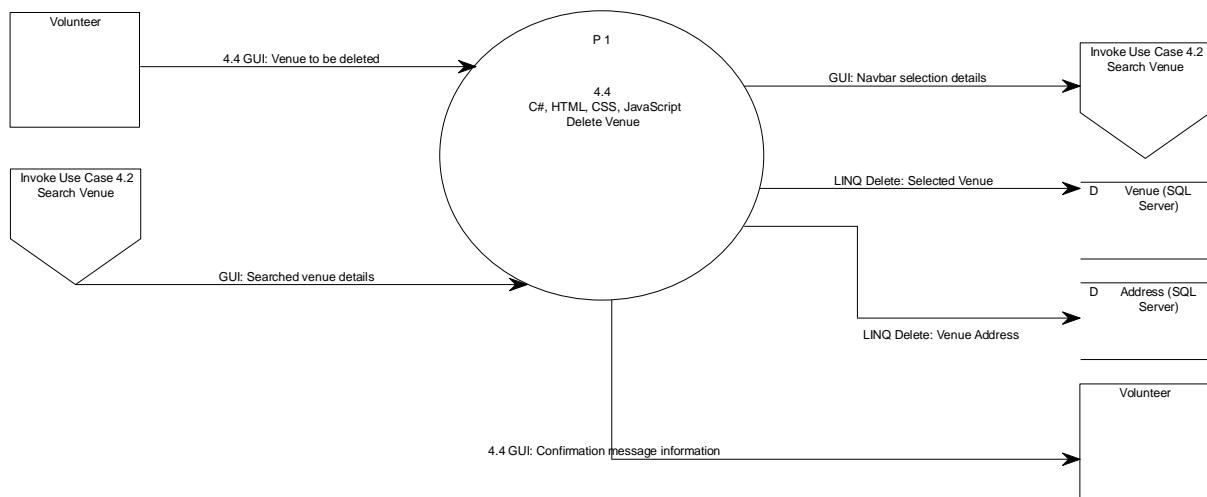


Figure 71: 4.4 Mid-Level Data Flow Diagram

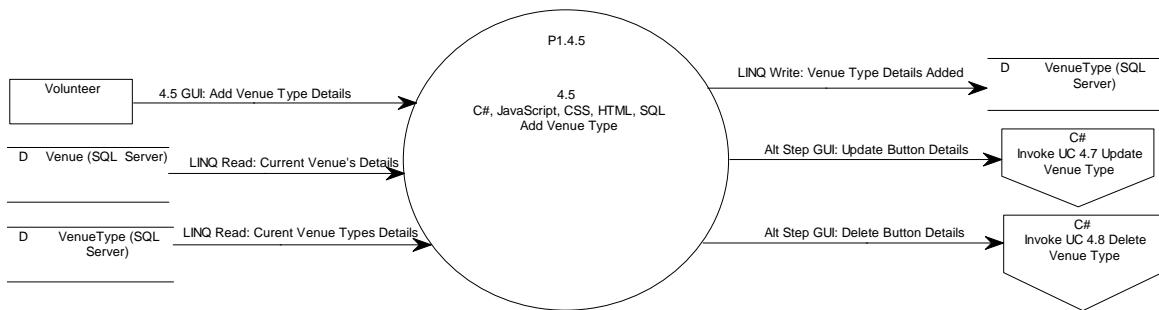


Figure 72: 4.5 Mid-Level Data Flow Diagram

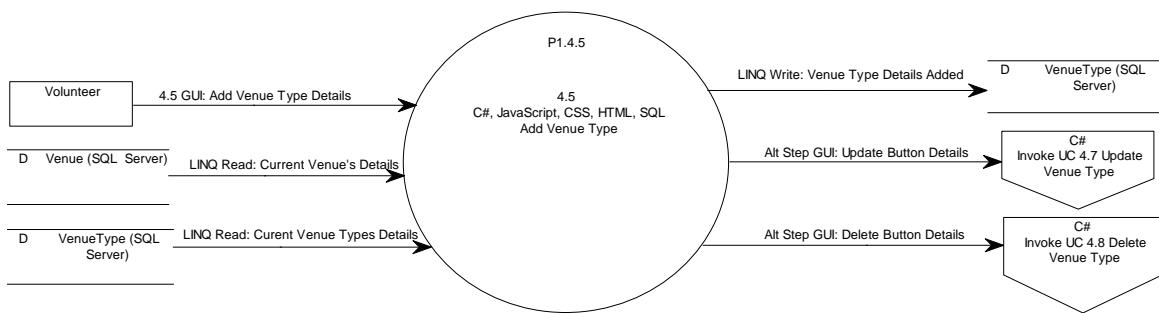


Figure 73: 4.6 Mid-Level Data Flow Diagram

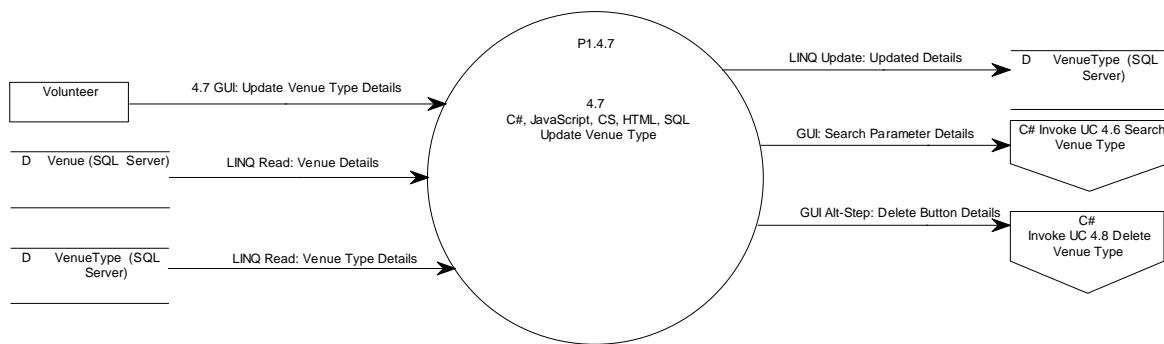


Figure 74: 4.7 Mid-Level Data Flow Diagram

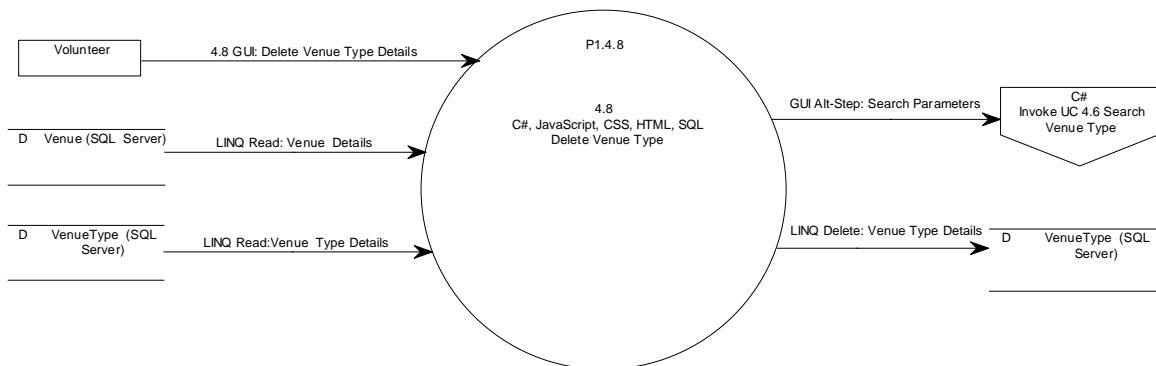


Figure 75: 4.8 Mid-Level Data Flow Diagram

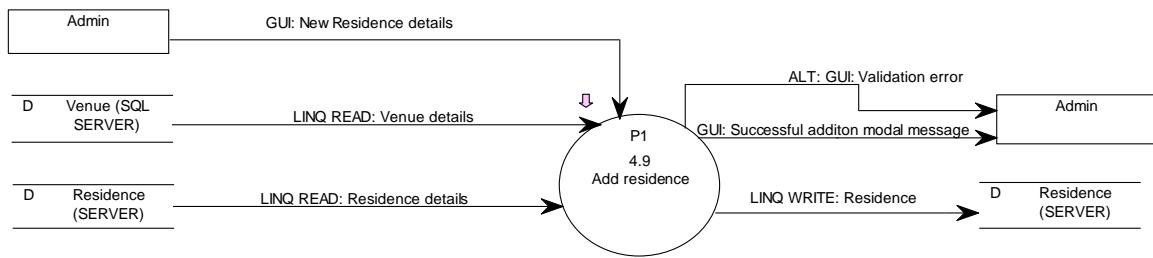


Figure 76: 4.9 Mid-Level Data Flow Diagram

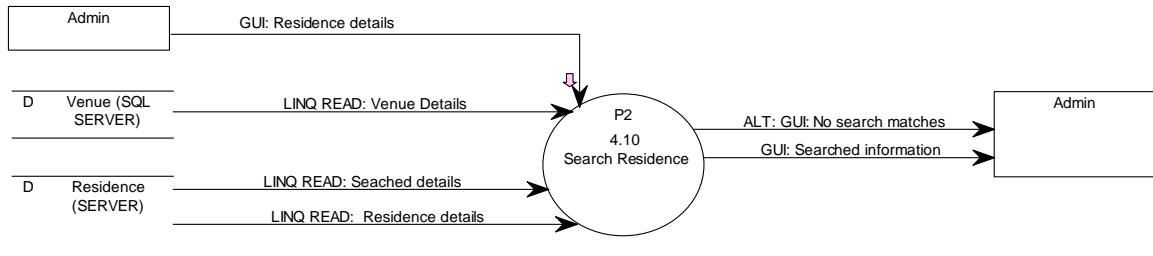


Figure 77: 4.10 Mid-Level Data Flow Diagram

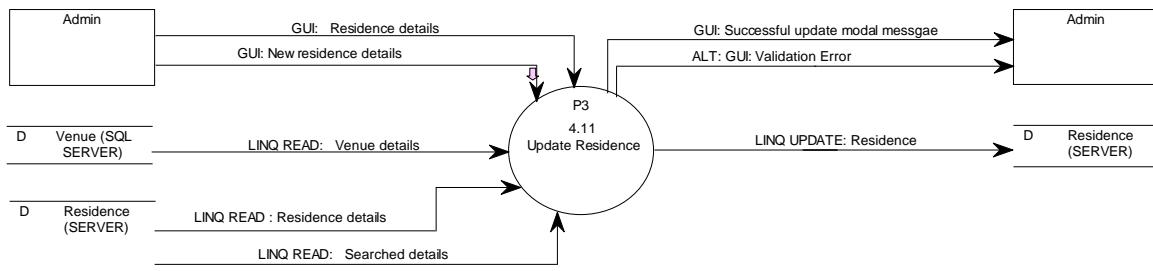


Figure 78: 4.11 Mid-Level Data Flow Diagram

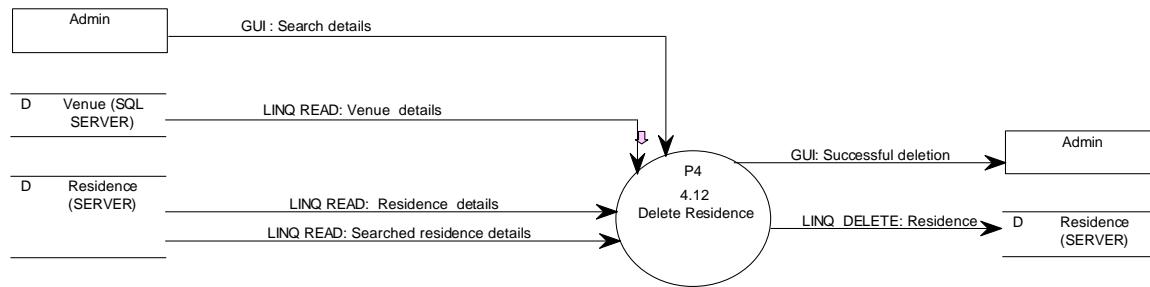


Figure 79: 4.12 Mid-Level Data Flow Diagram

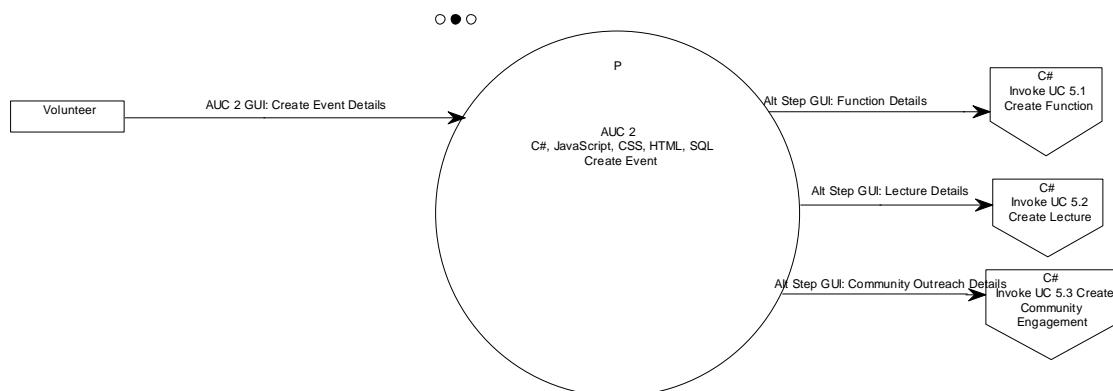


Figure 80: AUC 2 Mid-Level Data Flow Diagram

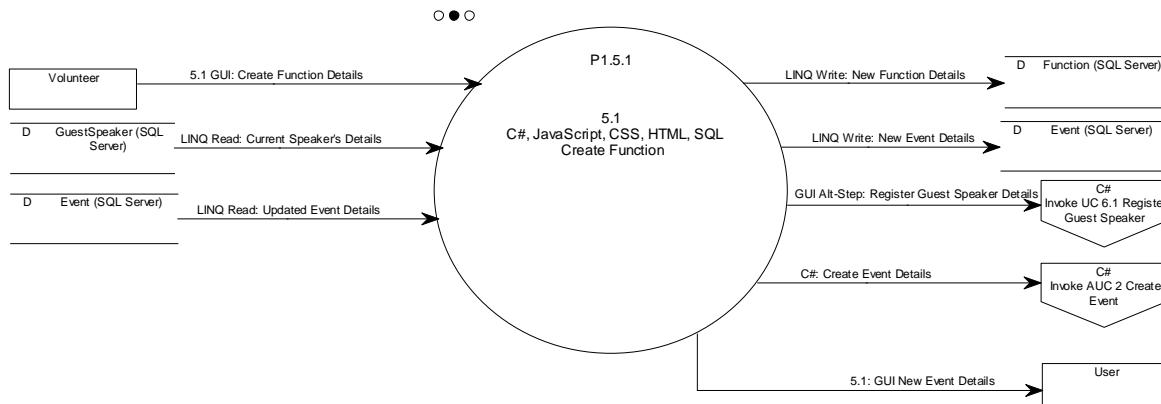


Figure 81: 5.1 Mid-Level Data Flow Diagram

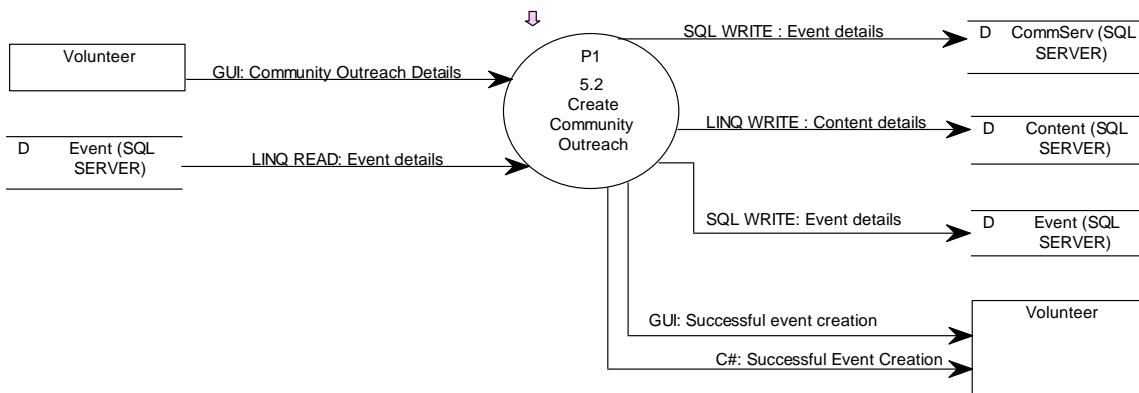


Figure 82: 5.2 Mid-Level Data Flow Diagram

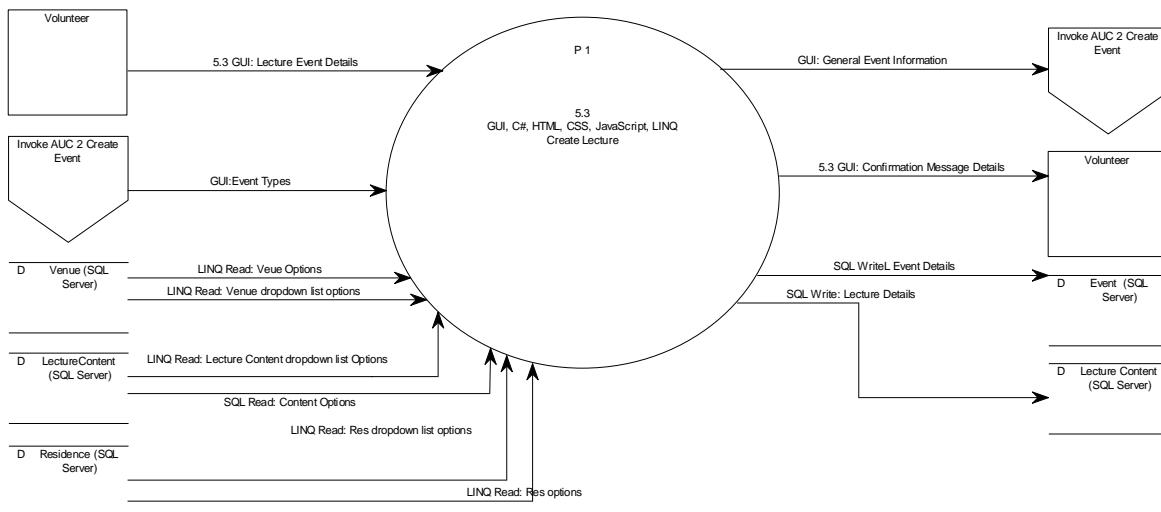


Figure 83: 5.3 Mid-Level Data Flow Diagram

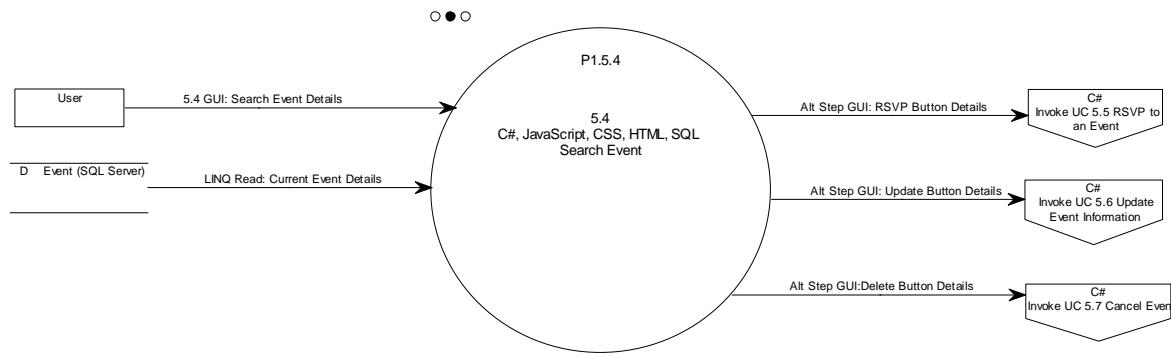


Figure 84: 5.4 Mid-Level Data Flow Diagram

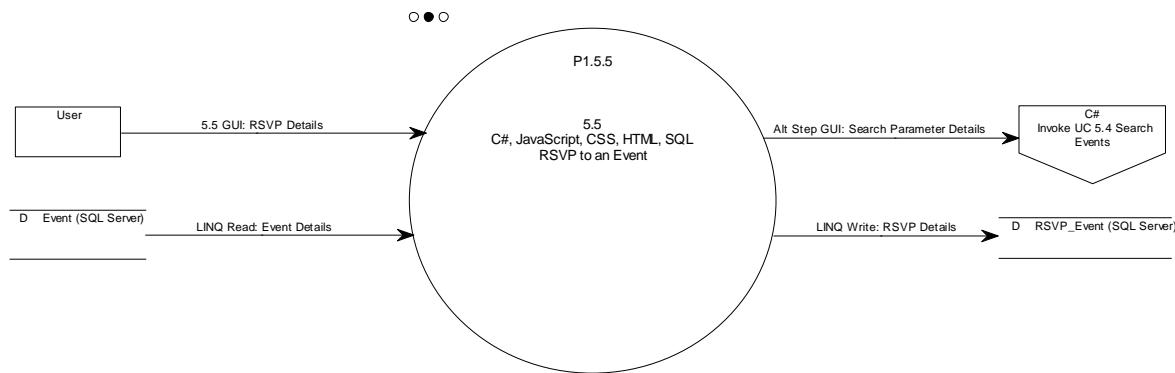


Figure 85: 5.5 Mid-Level Data Flow Diagram

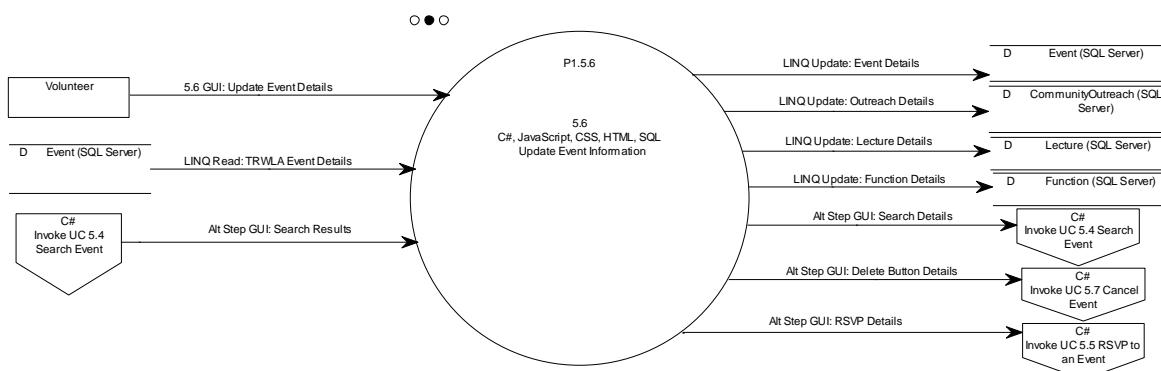


Figure 86: 5.6 Mid-Level Data Flow Diagram

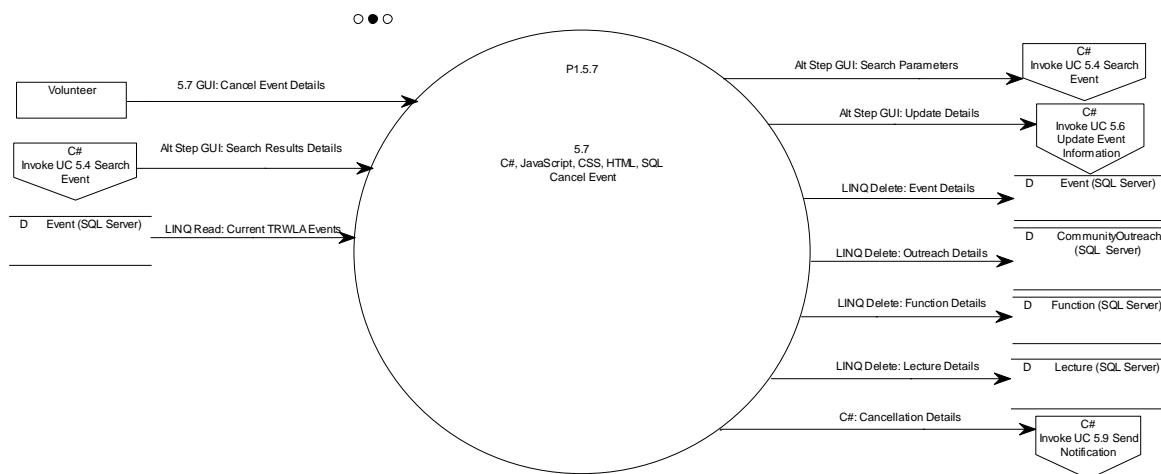


Figure 87: 5.7 Mid-Level Data Flow Diagram

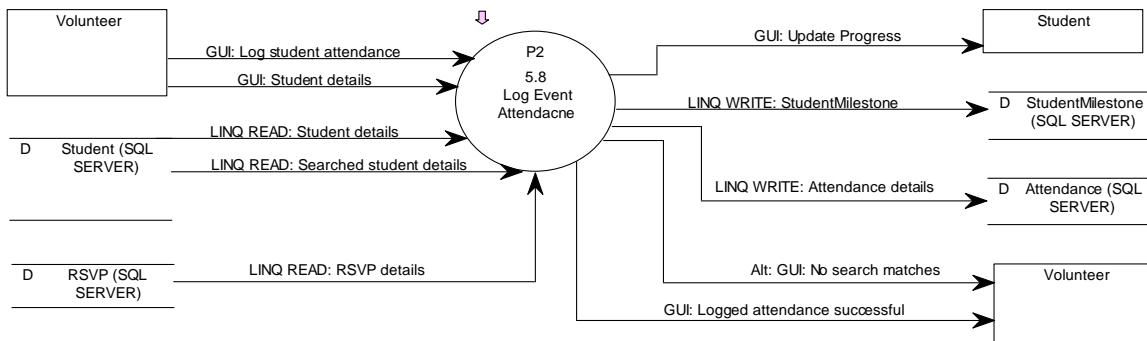


Figure 88: 5.8 Mid-Level Data Flow Diagram

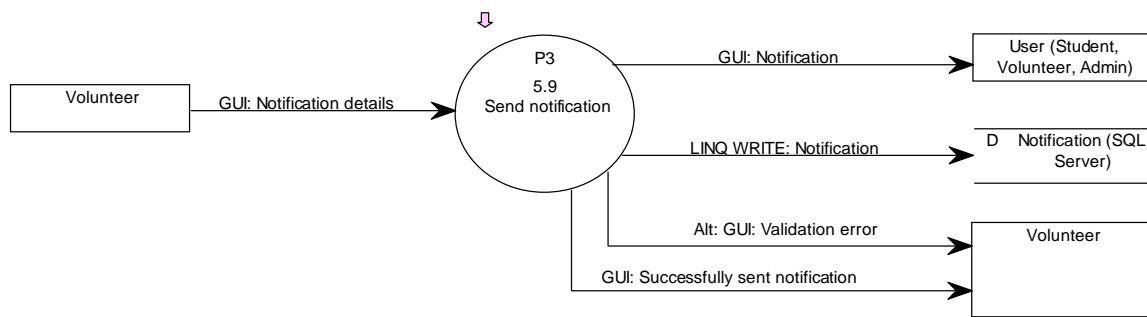


Figure 89: 5.9 Mid-Level Data Flow Diagram

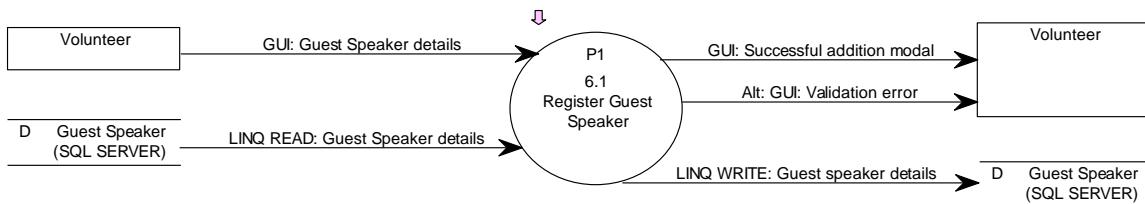


Figure 90: 6.1 Mid-Level Data Flow Diagram

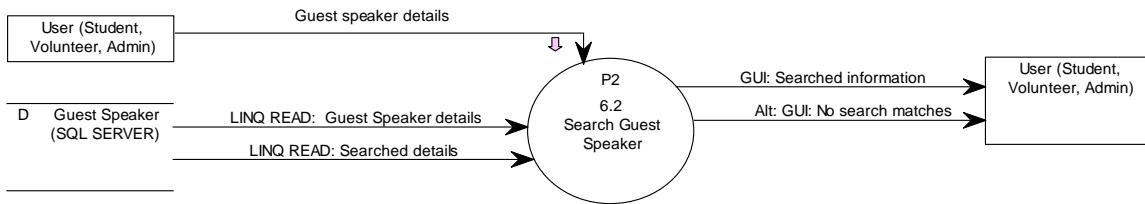


Figure 91: 6.2 Mid-Level Data Flow Diagram

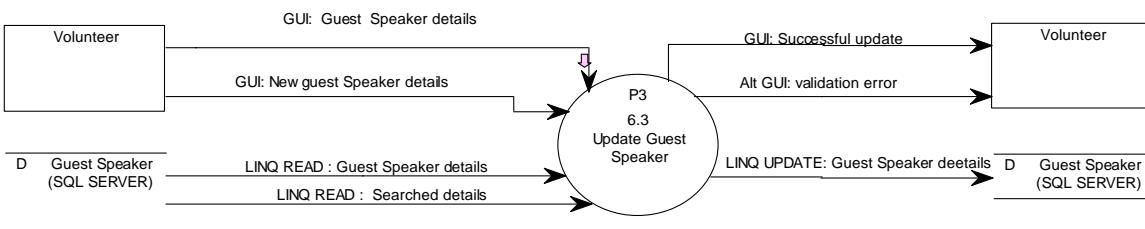


Figure 92: 6.3 Mid-Level Data Flow Diagram

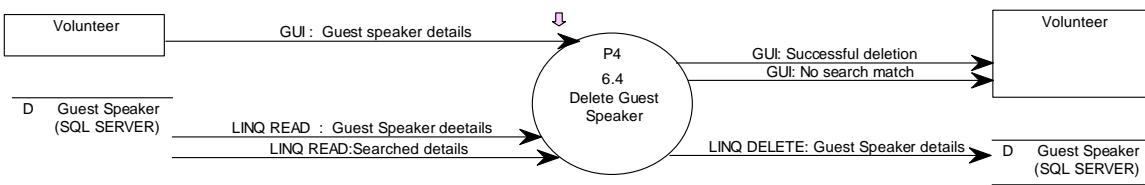


Figure 93: 6.4 Mid-Level Data Flow Diagram

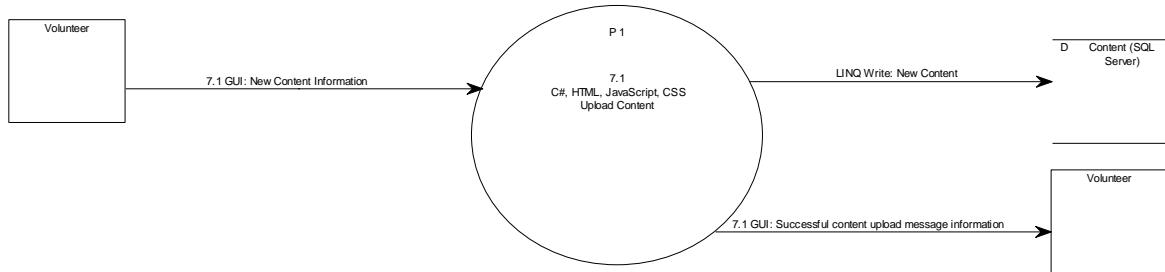


Figure 94: 7.1 Mid-Level Data Flow Diagram

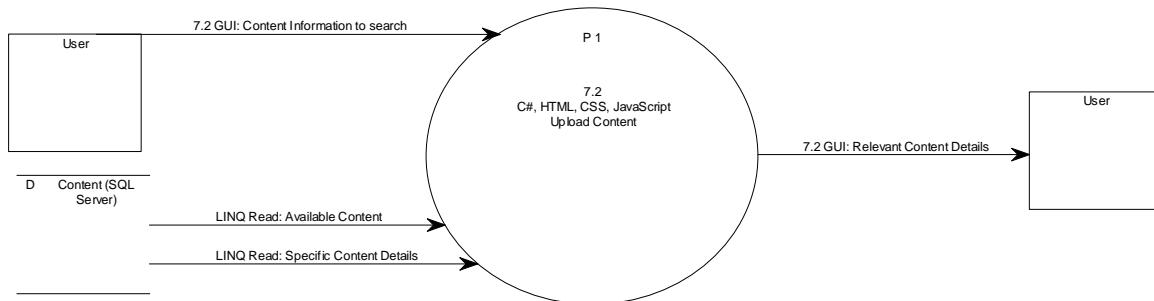


Figure 95: 7.2 Mid-Level Data Flow Diagram

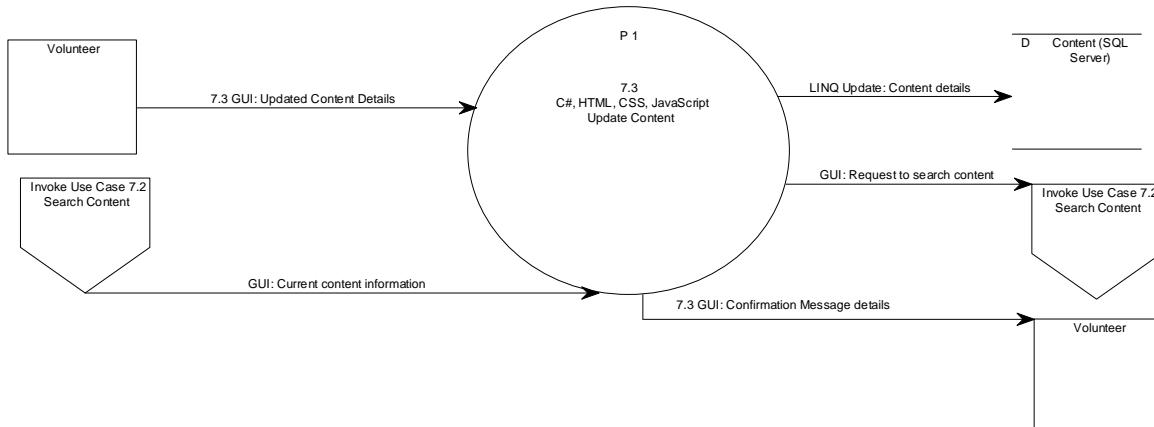


Figure 96: 7.3 Mid-Level Data Flow Diagram

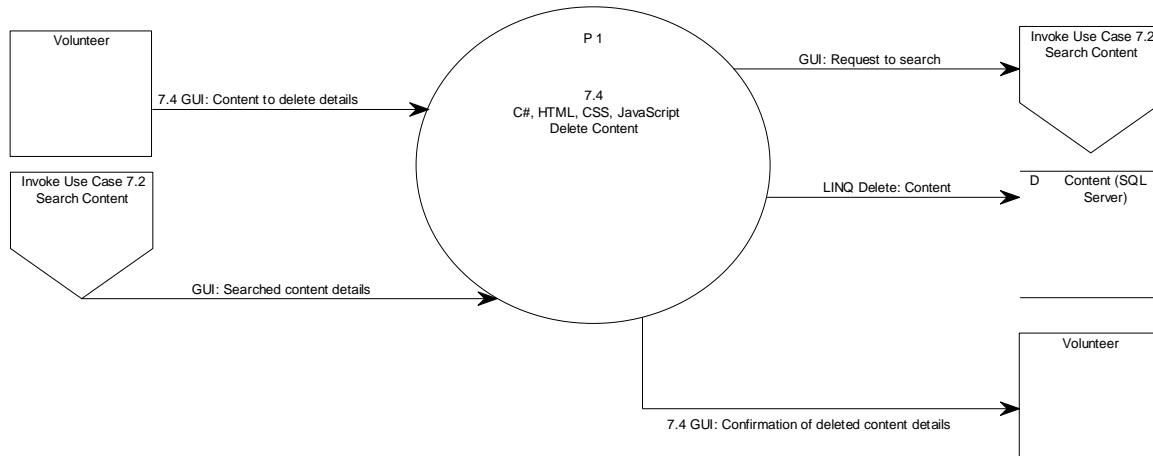


Figure 97: 7.4 Mid-Level Data Flow Diagram

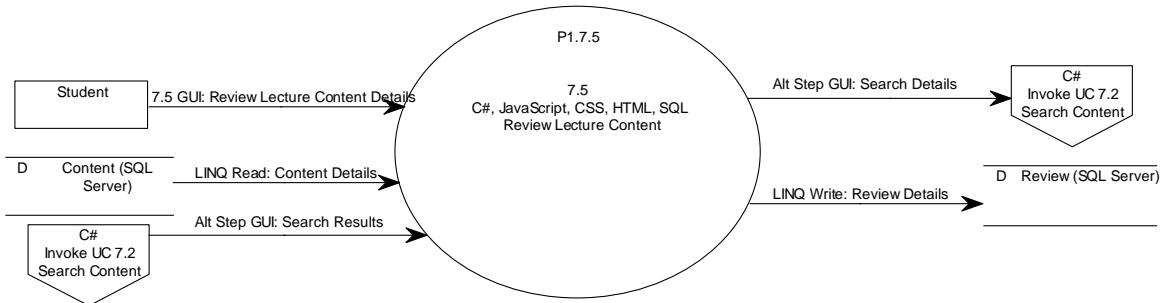
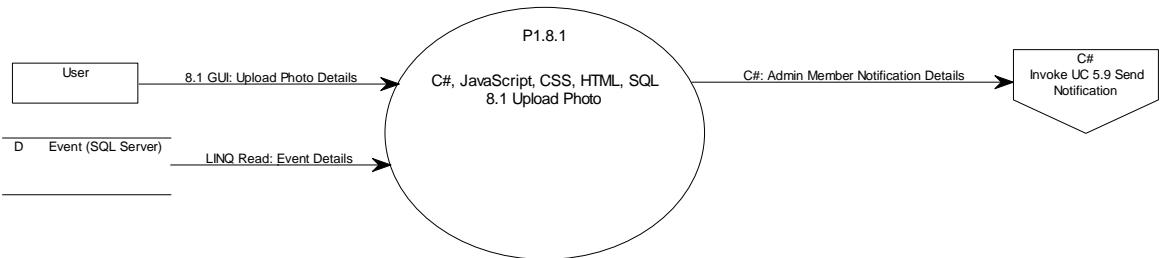


Figure 98: 7.5 Mid-Level Data Flow Diagram



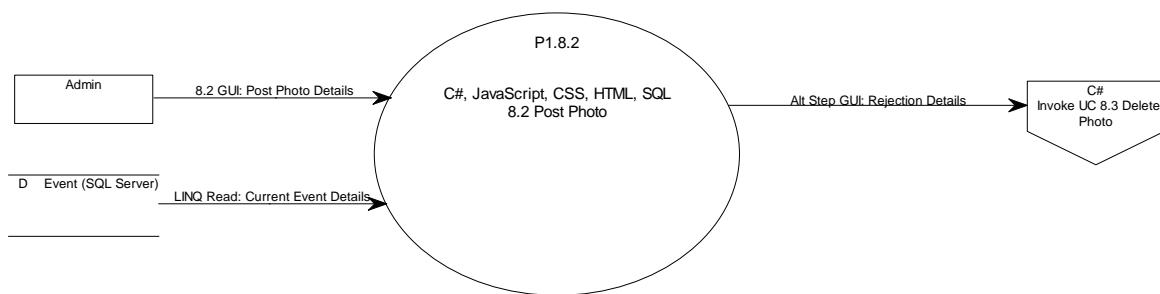


Figure 100: 8.2 Mid-Level Data Flow Diagram

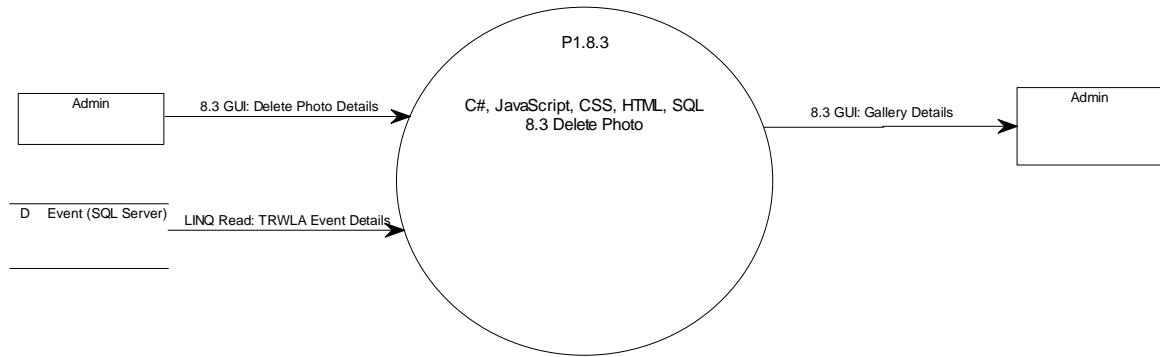


Figure 101: 8.3 Mid-Level Data Flow Diagram

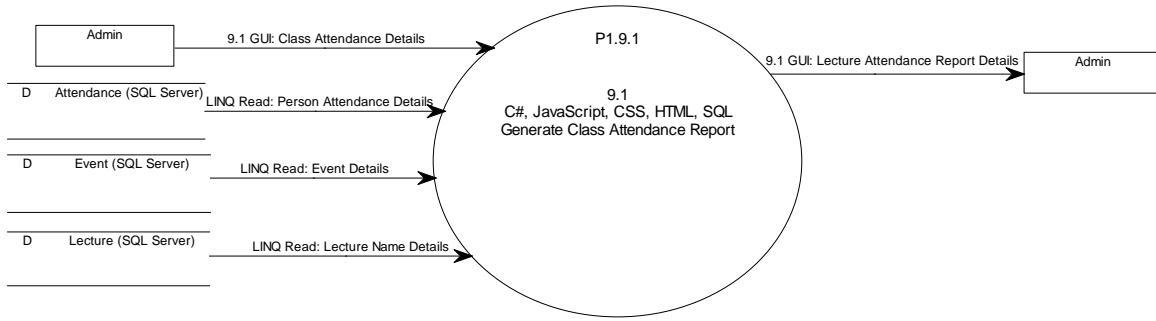


Figure 102: 9.1 Mid-Level Data Flow Diagram

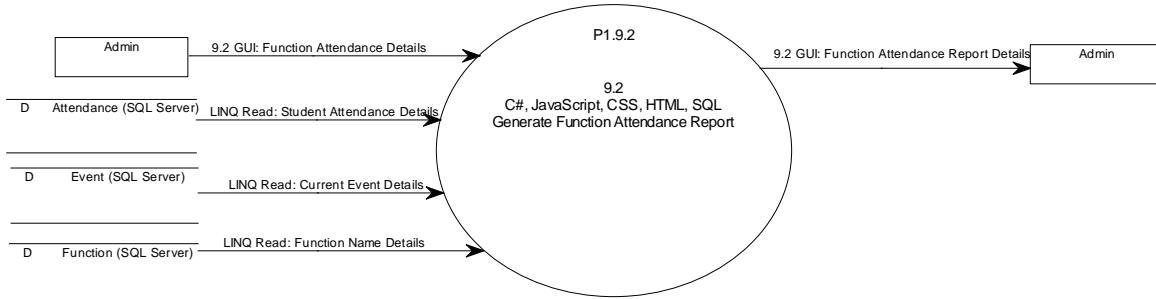


Figure 103: 9.2 Mid-Level Data Flow Diagram

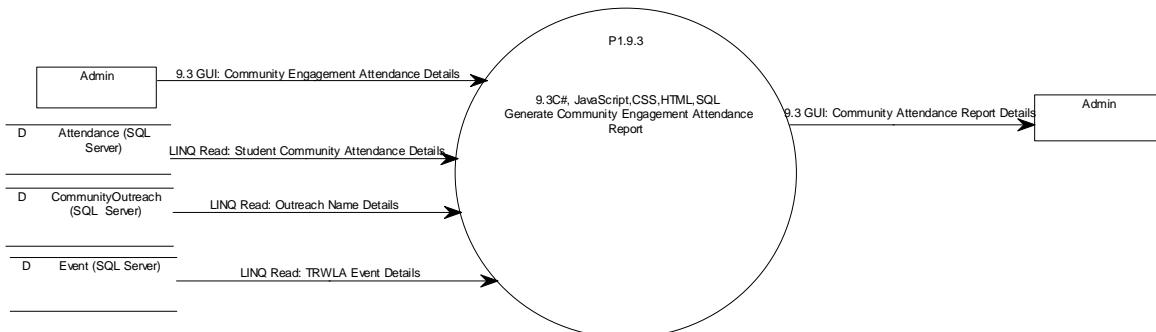


Figure 104: 9.3 Mid-Level Data Flow Diagram

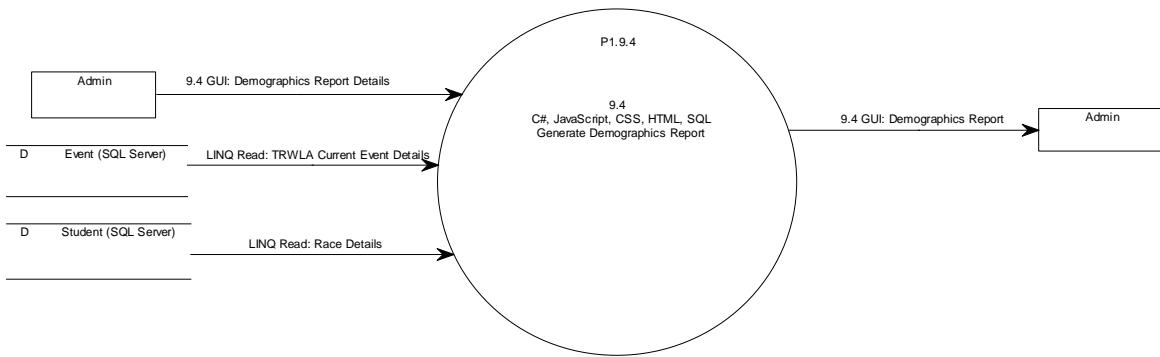


Figure 105: 9.4 Mid-Level Data Flow Diagram

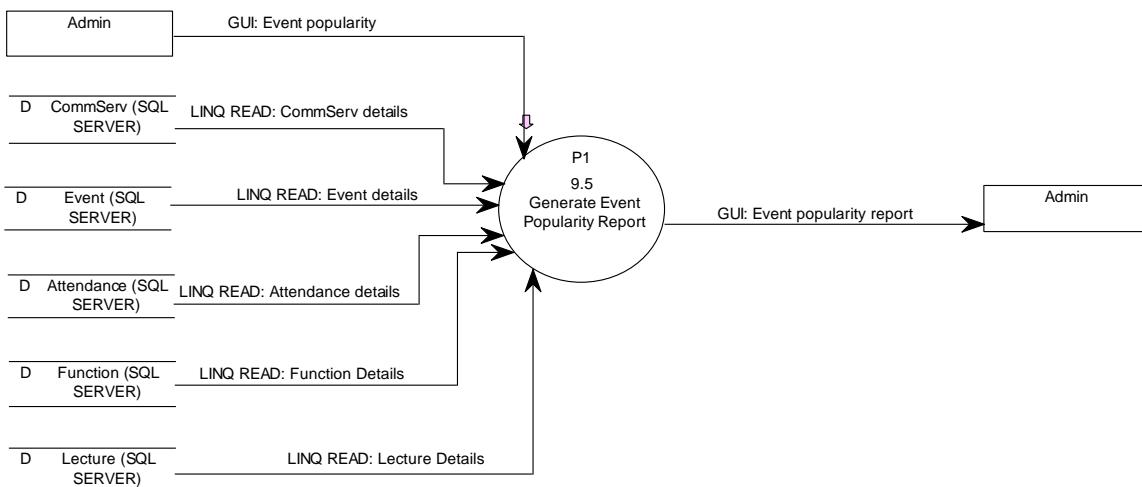


Figure 106: 9.5 Mid-Level Data Flow Diagram

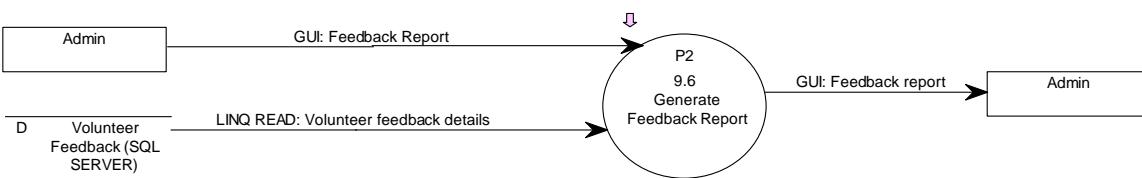


Figure 107: 9.6 Mid-Level Data Flow Diagram

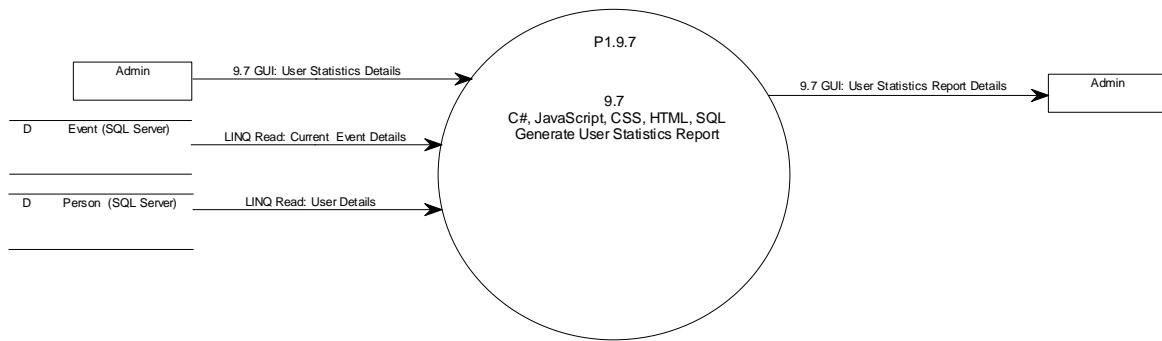


Figure 108: 9.7 Mid-Level Data Flow Diagram

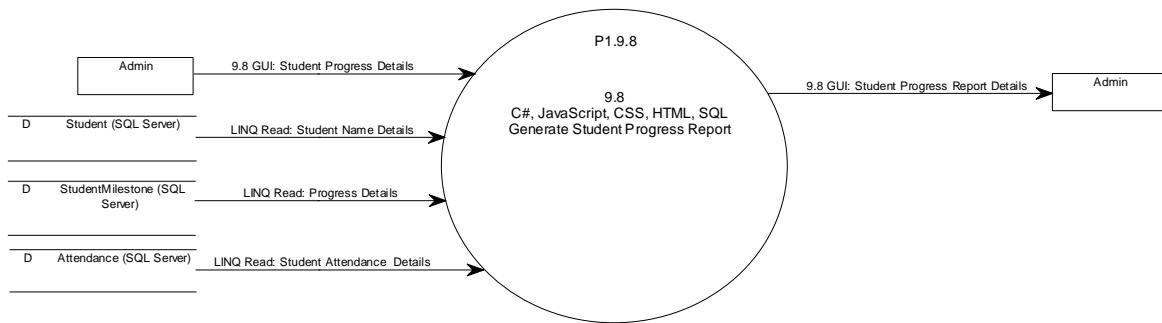


Figure 109: 9.8 Mid-Level Data Flow Diagram

3.4.3 Primitive Level Data Flow Diagrams

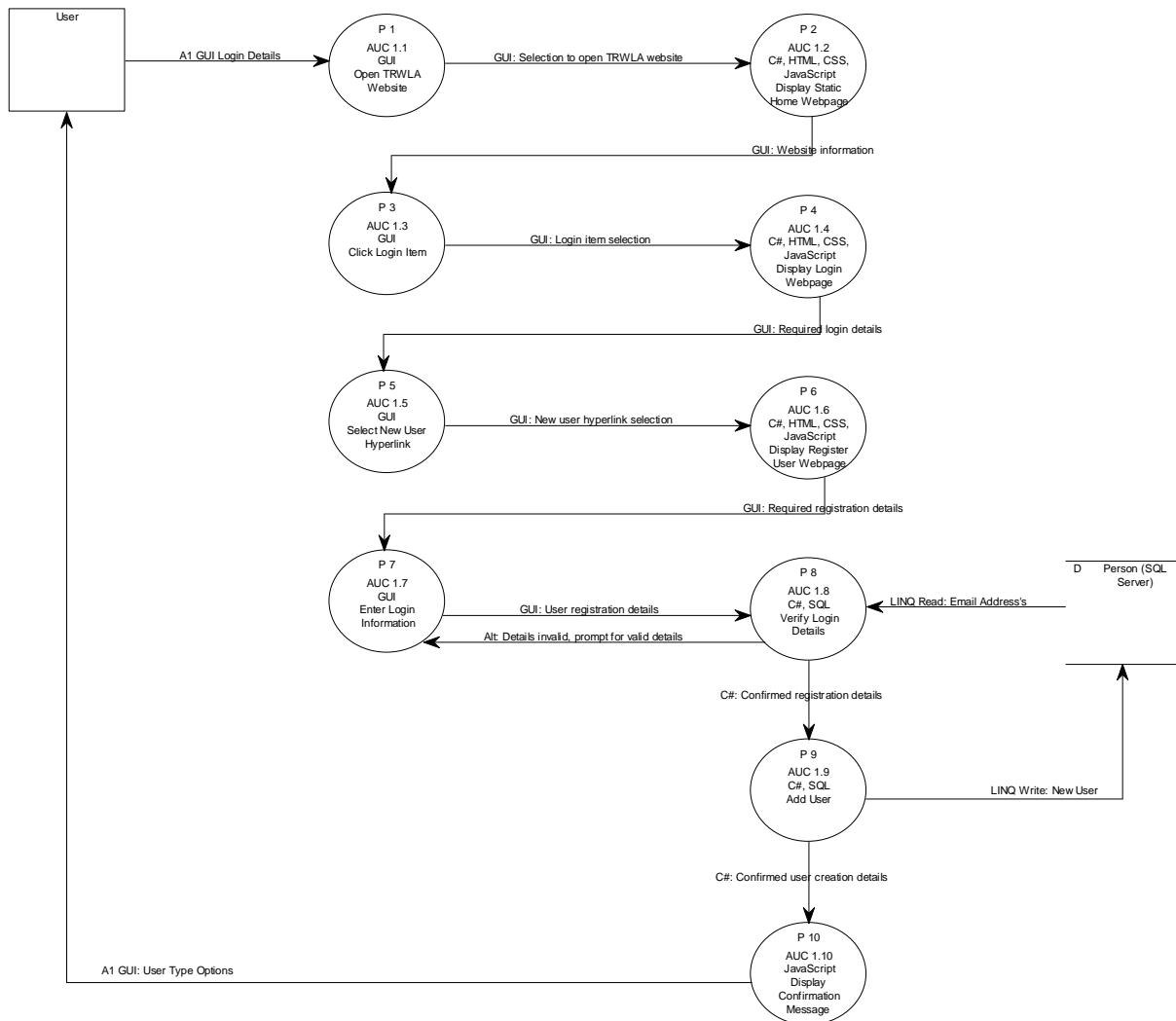


Figure 110: AUC 1 Primitive Level Data Flow Diagram

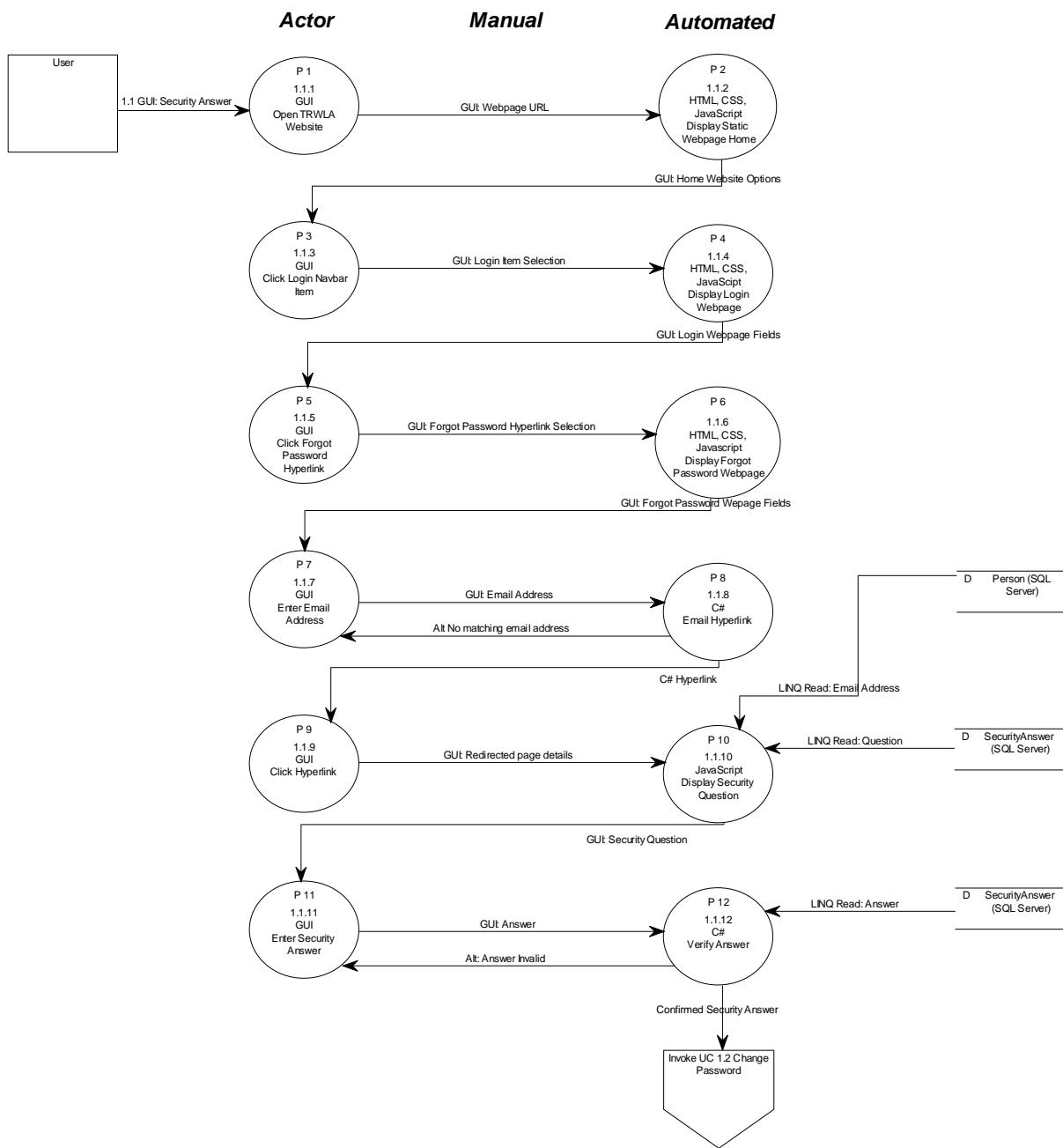


Figure 111: 1.1 Primitive Level Data Flow Diagram

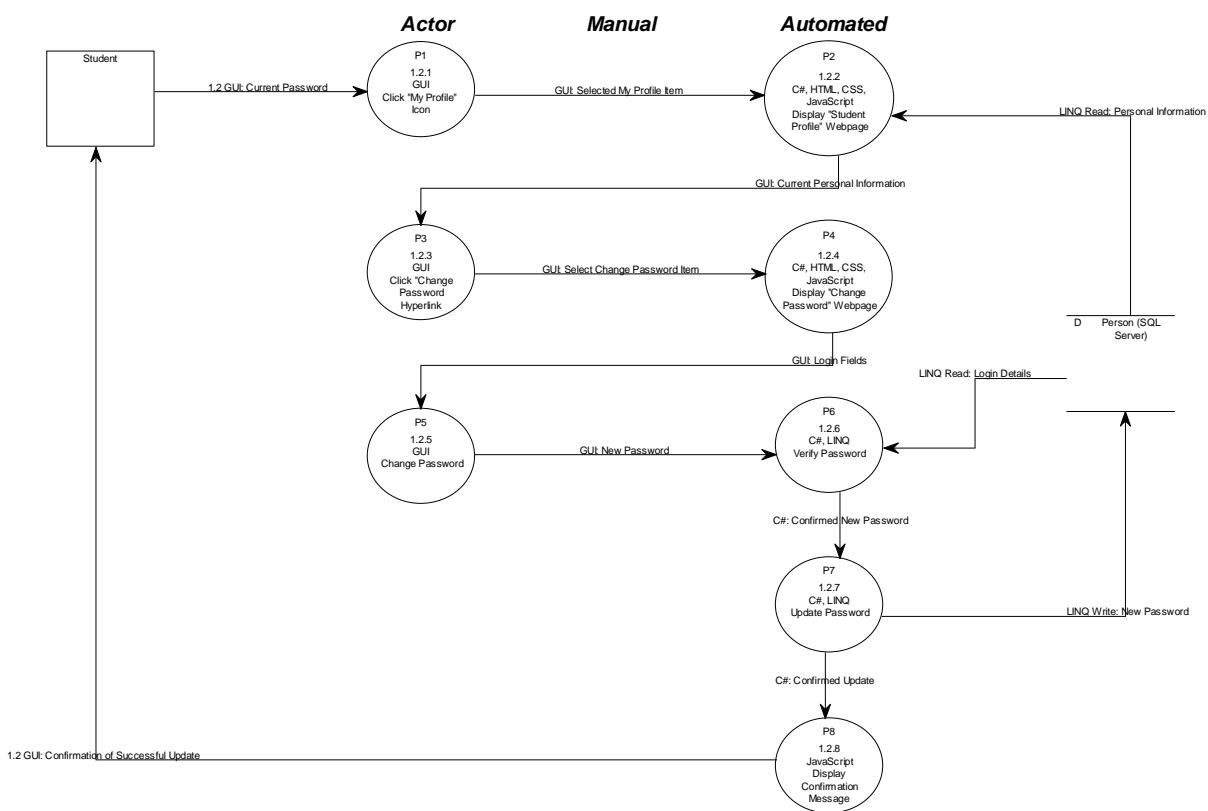


Figure 112: 1.2 Primitive Level Data Flow Diagram

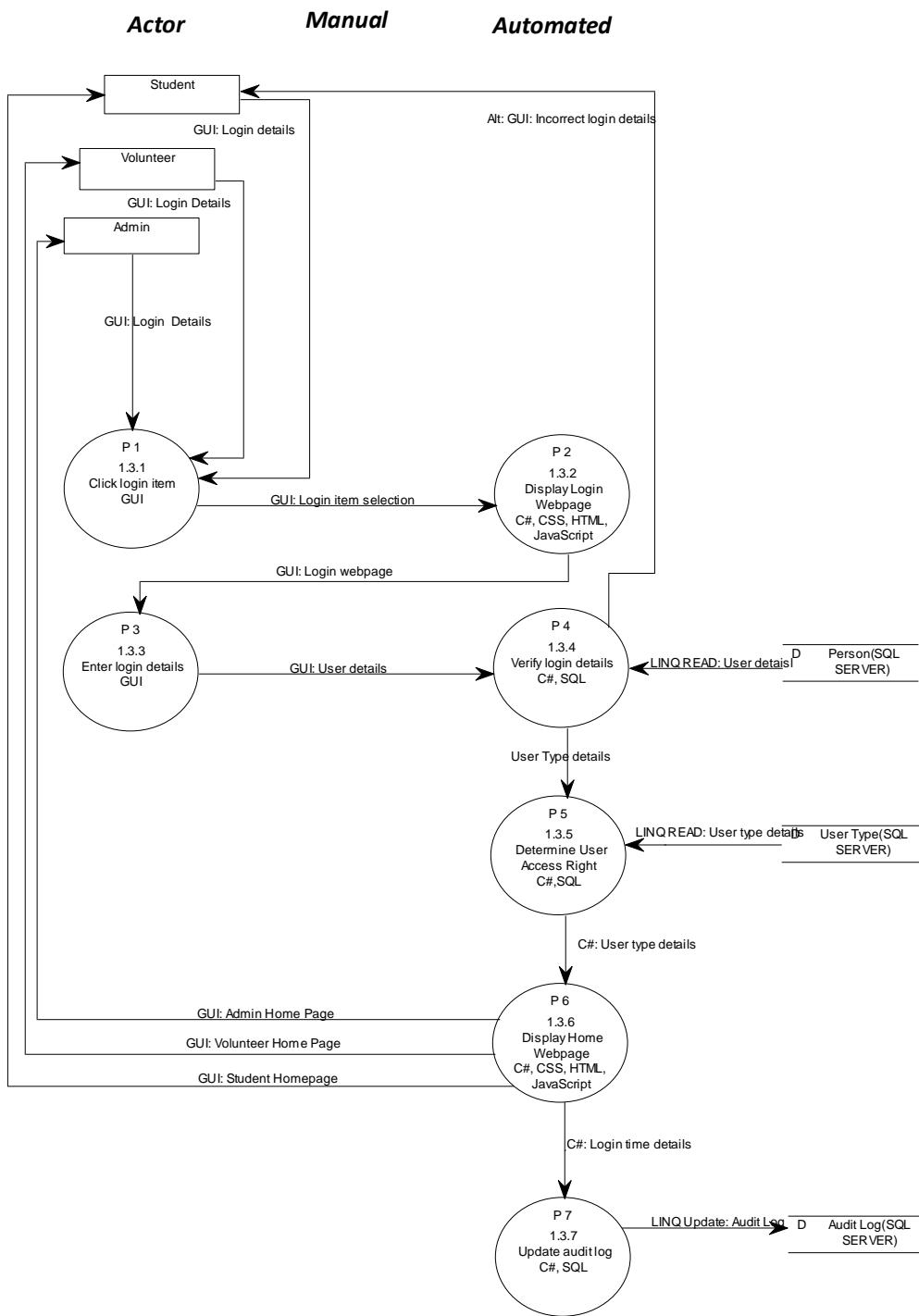


Figure 113: 1.3 Primitive Level Data Flow Diagram

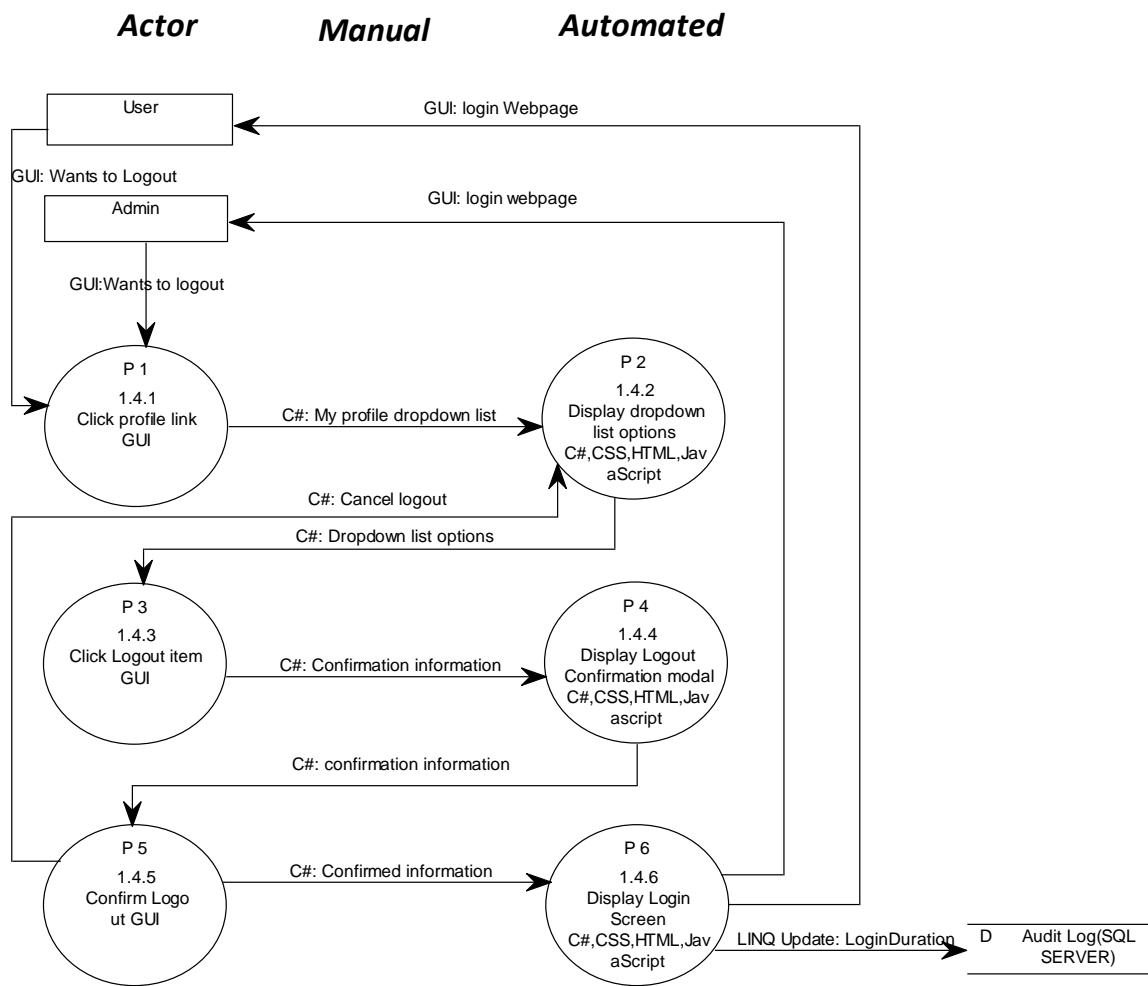


Figure 114: 1.4 Primitive Level Data Flow Diagram

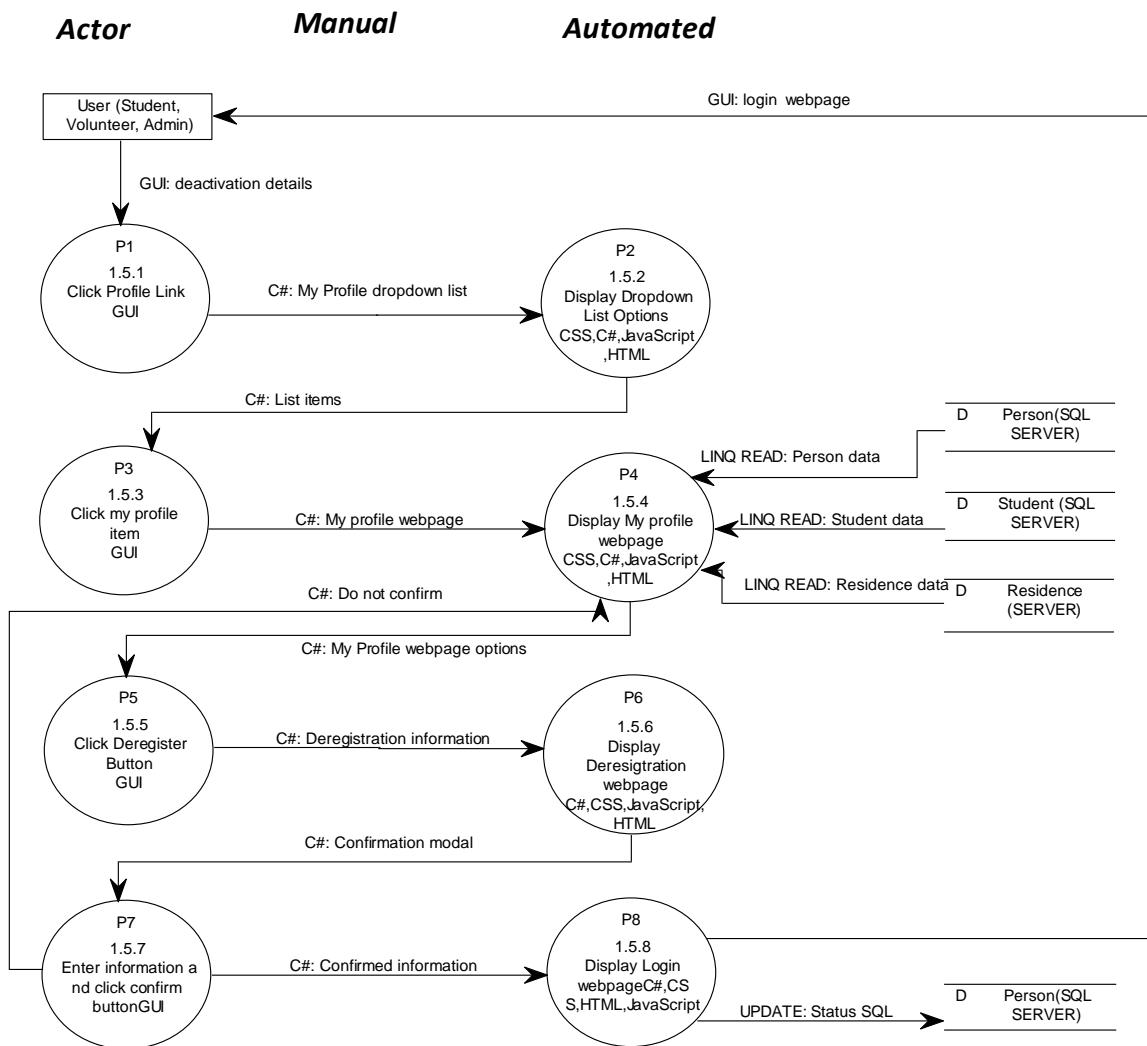


Figure 115: 1.5 Primitive Level Data Flow Diagram

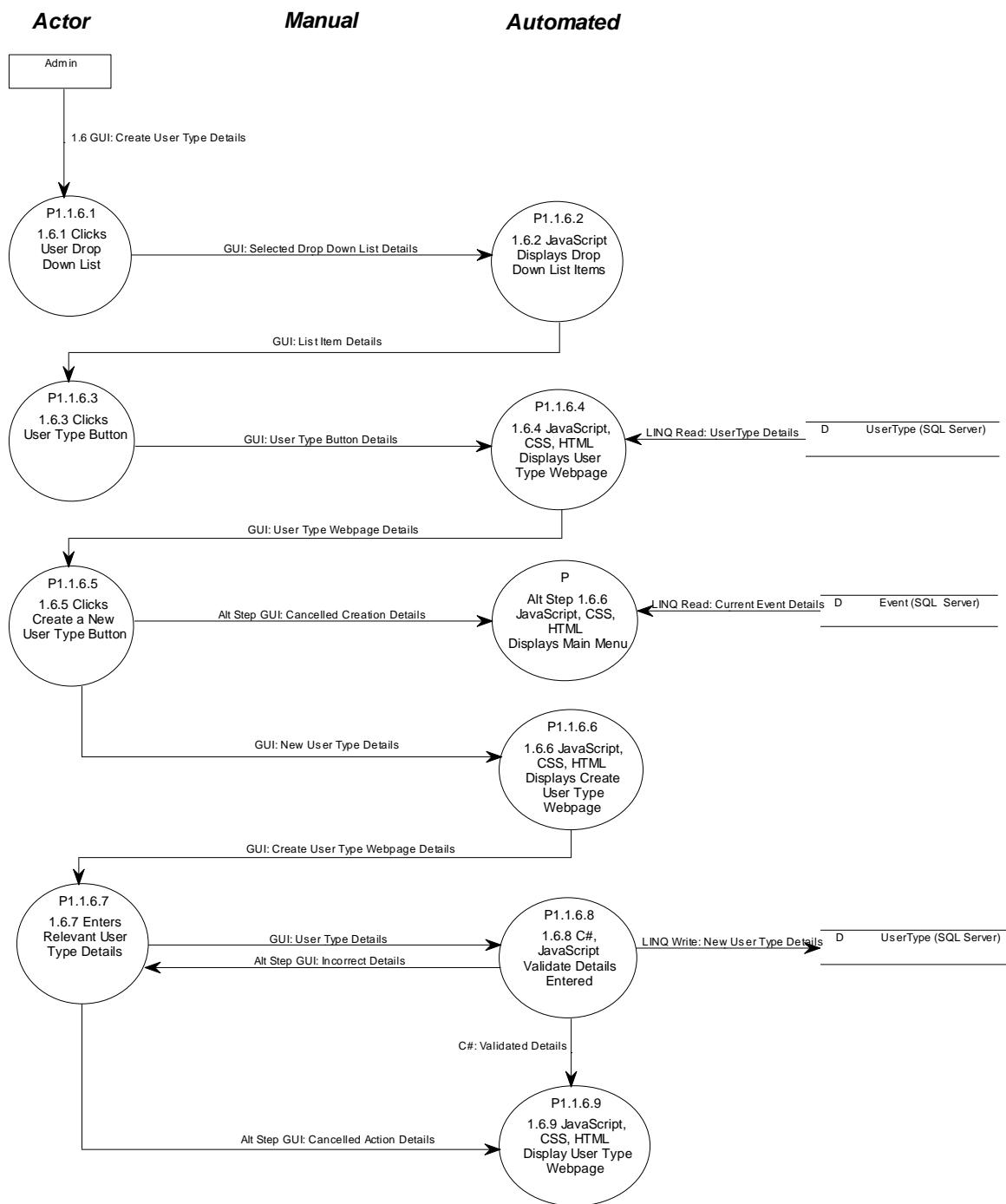


Figure 116: 1.6 Primitive Level Data Flow Diagram

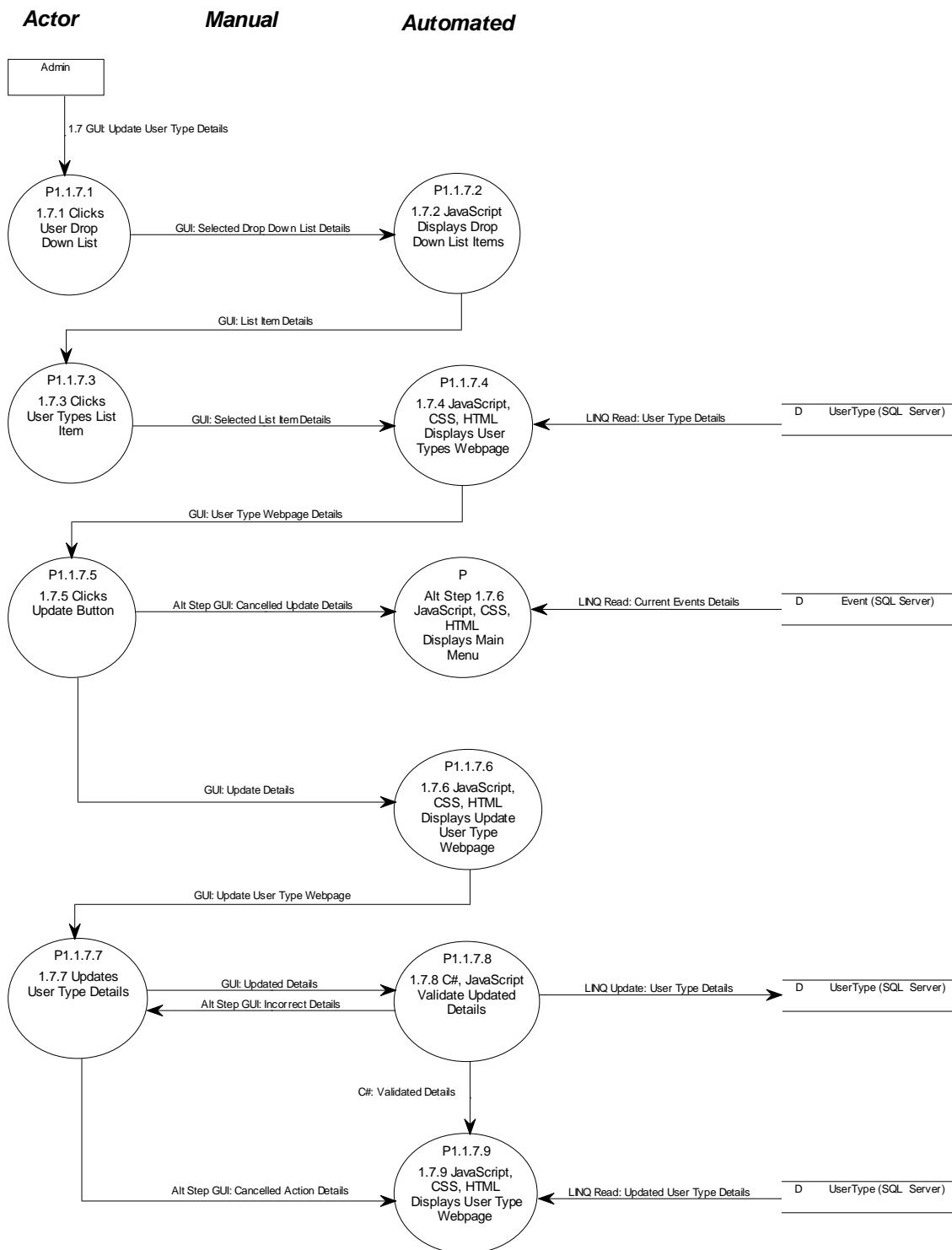


Figure 117: 1.7 Primitive Level Data Flow Diagram

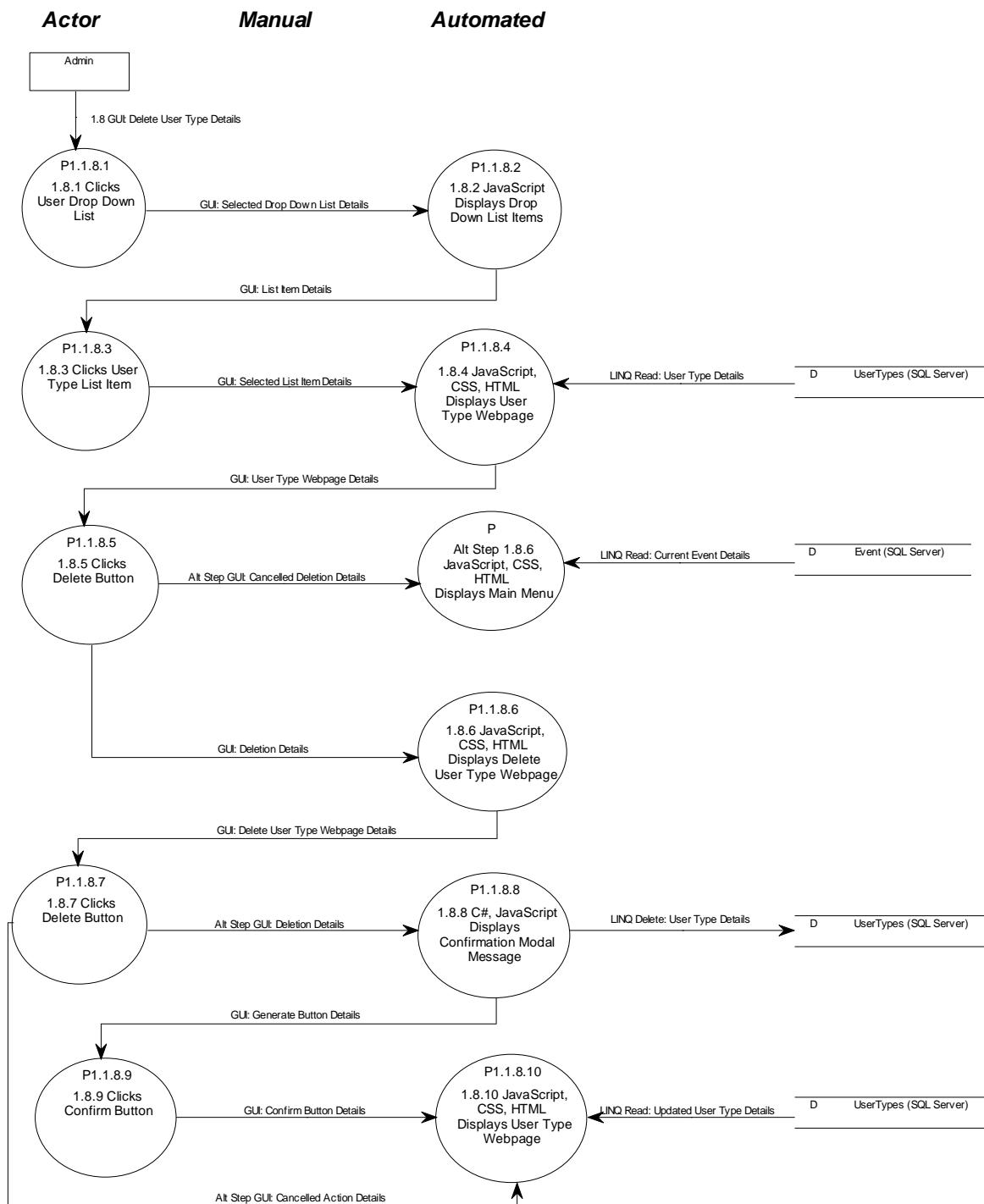


Figure 118: 1.8 Primitive Level Data Flow Diagram

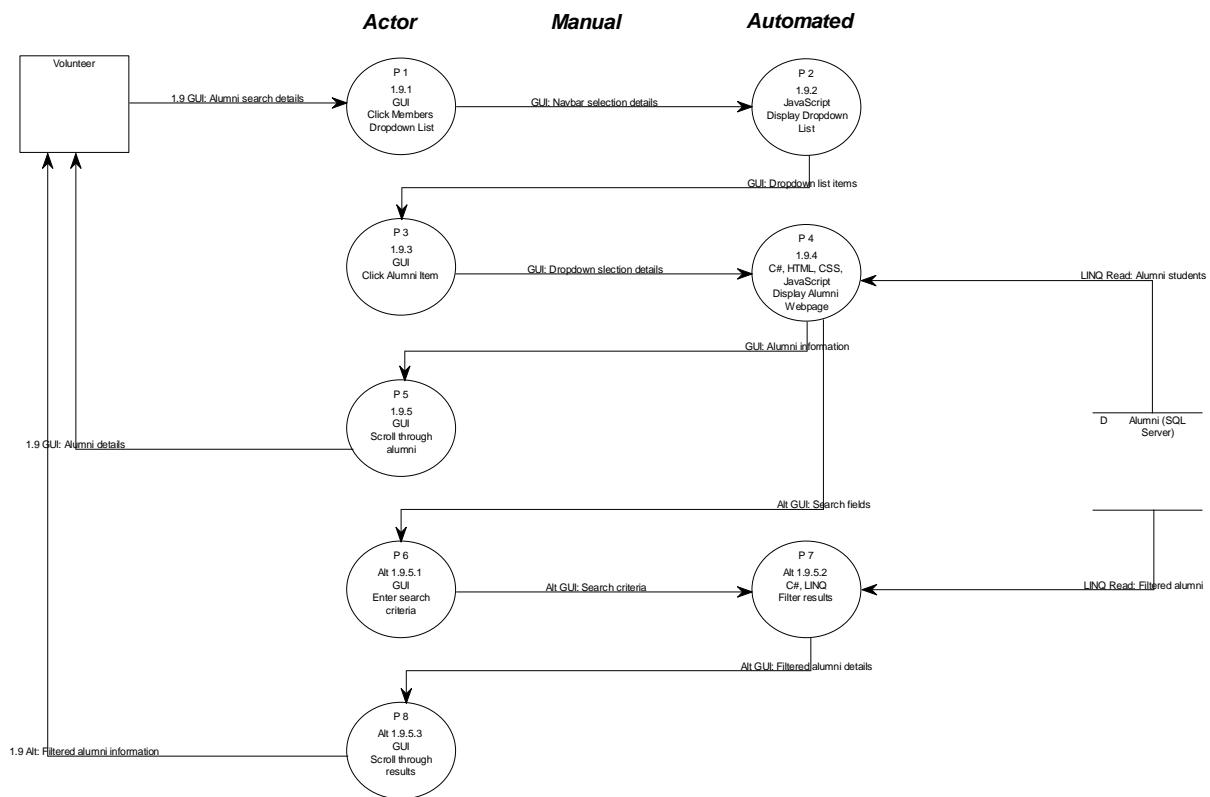


Figure 119: 1.9 Primitive Level Data Flow Diagram

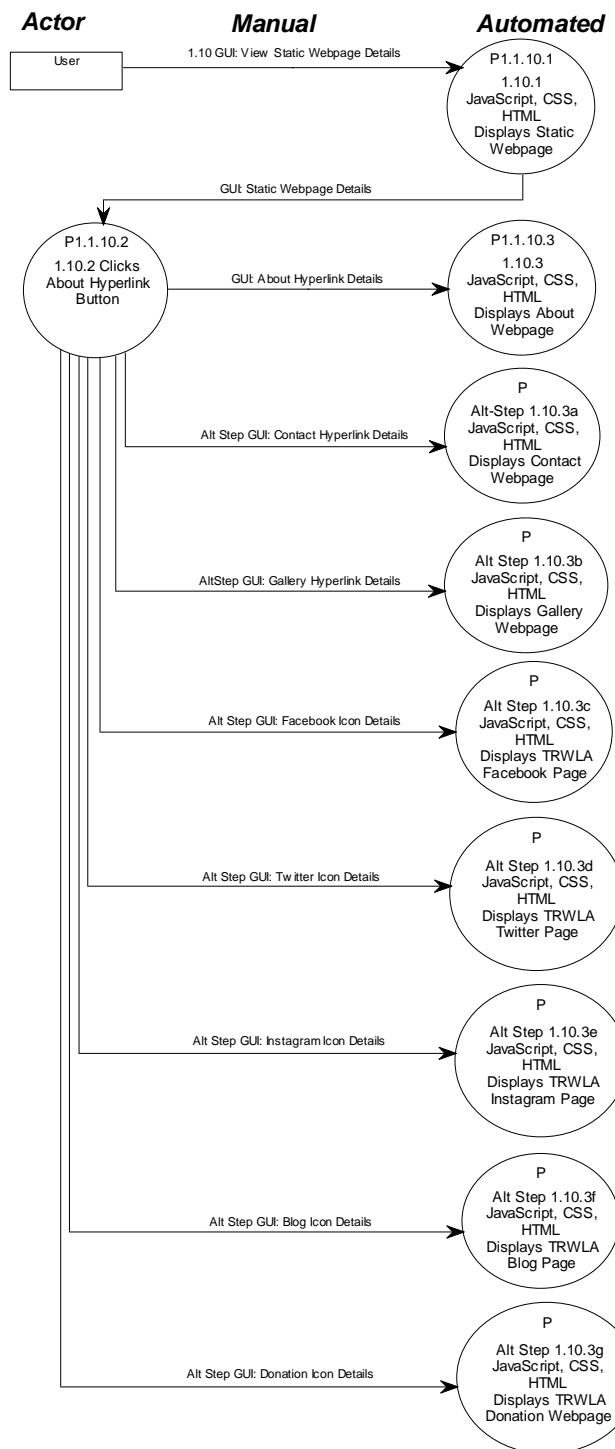


Figure 120: 1.10 Primitive Level Data Flow Diagram

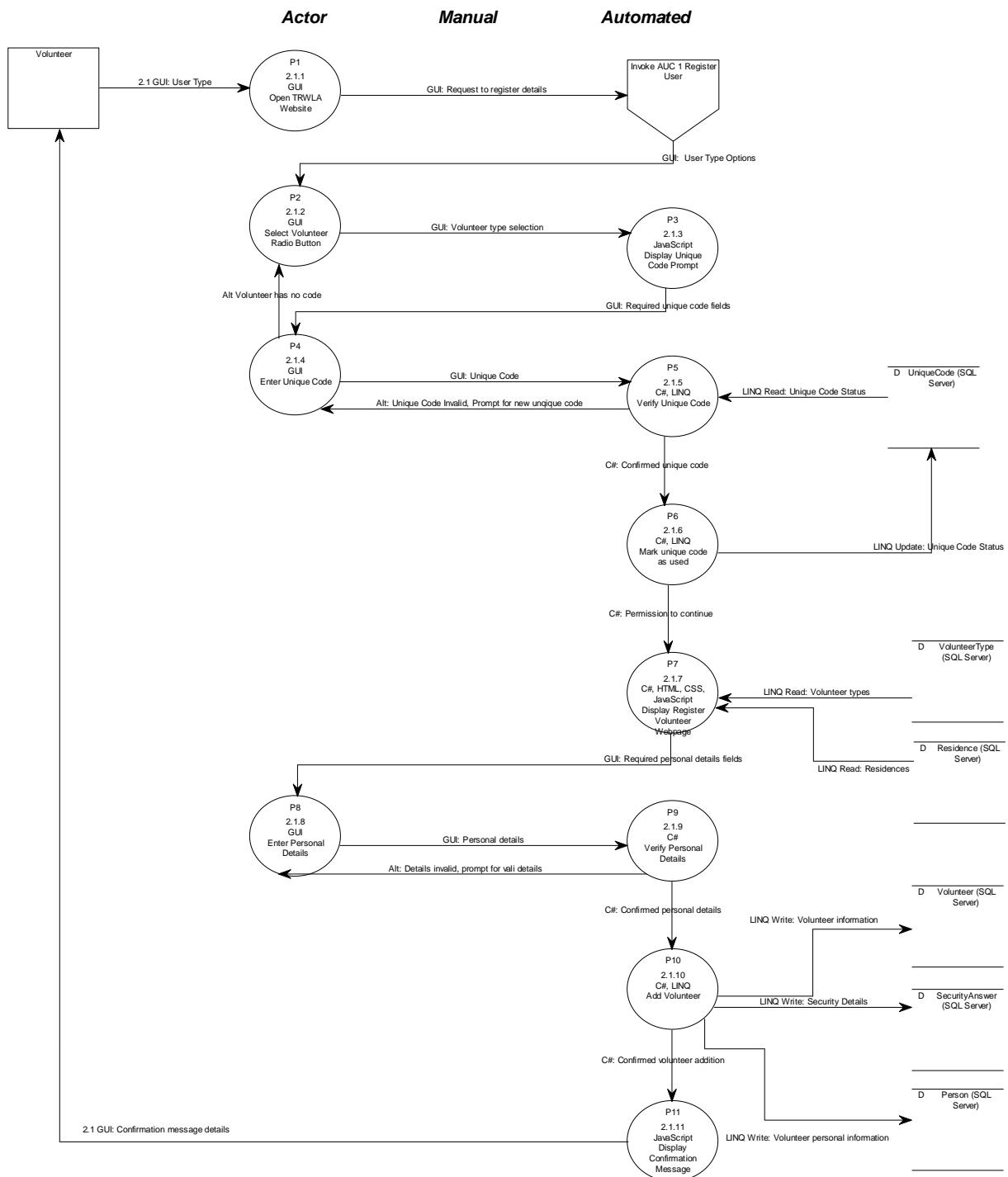


Figure 121: 2.1 Primitive Level Data Flow Diagram

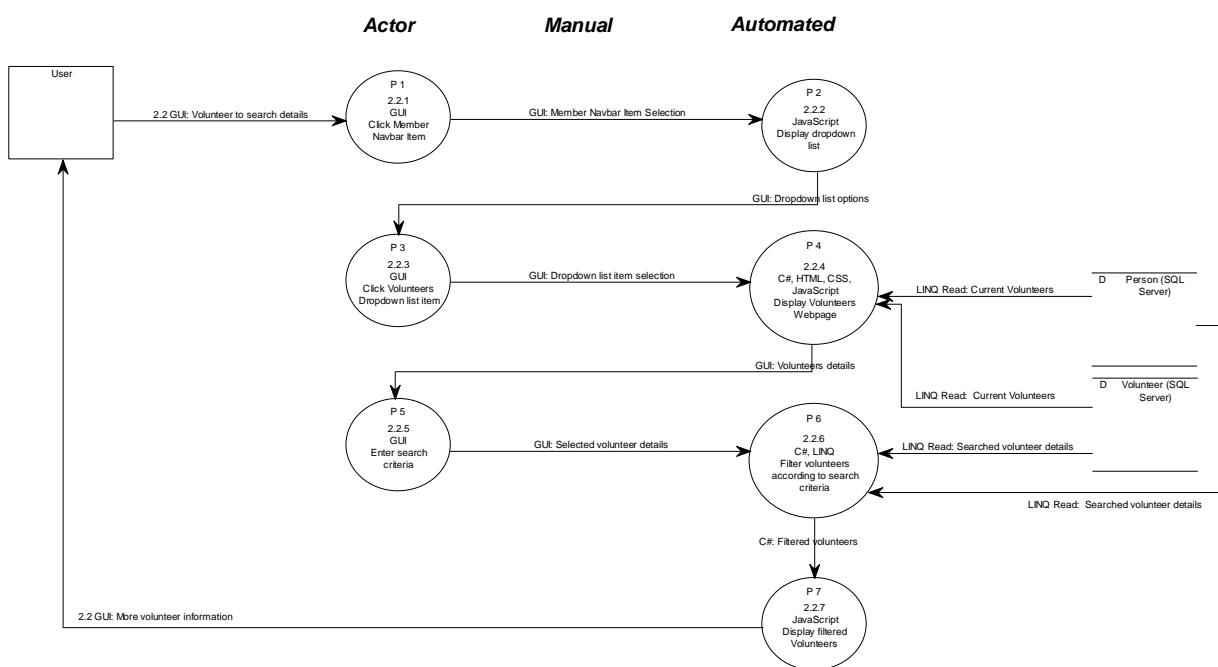


Figure 122: 2.2 Primitive Level Data Flow Diagram

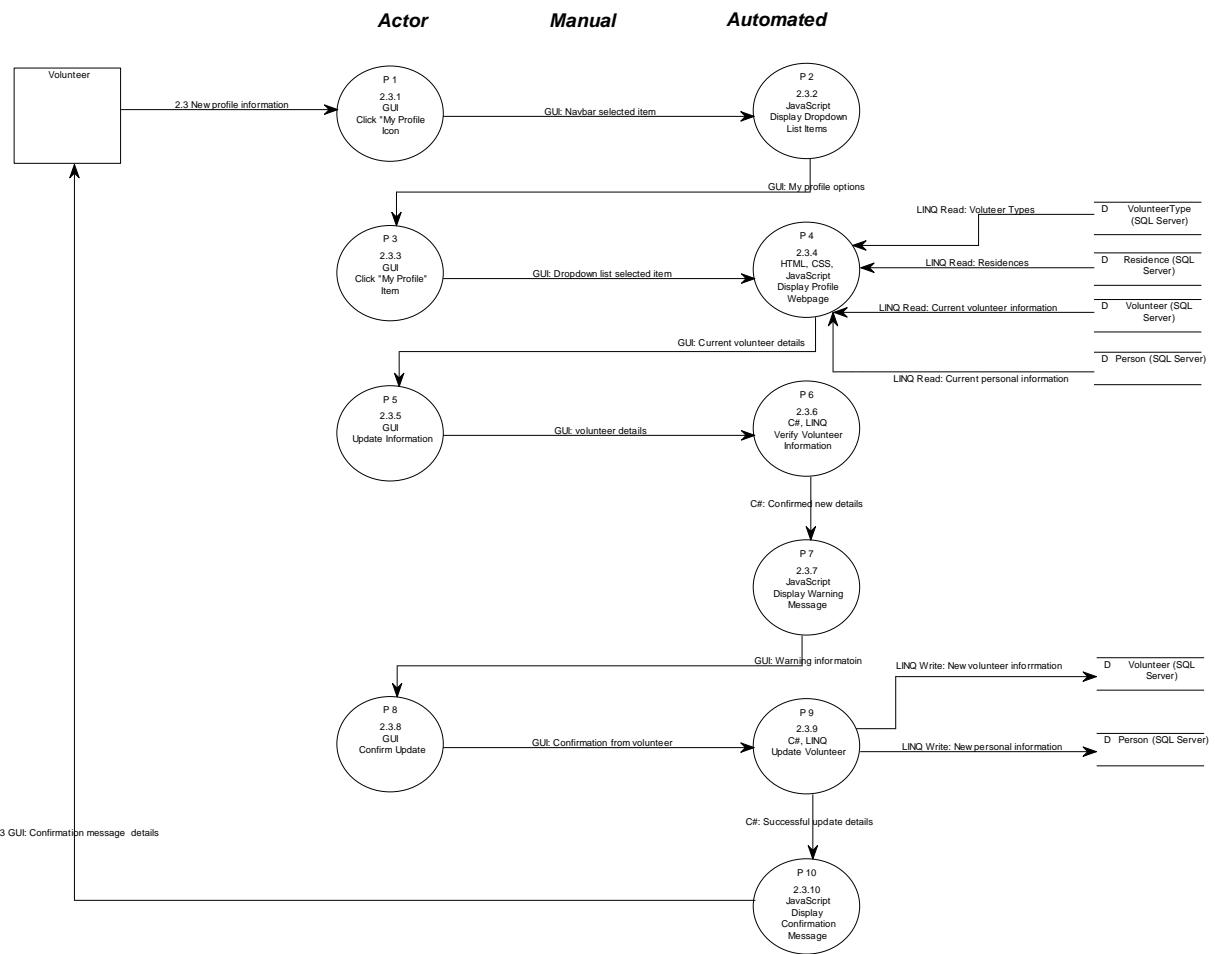


Figure 123: 2.3 Primitive Level Data Flow Diagram

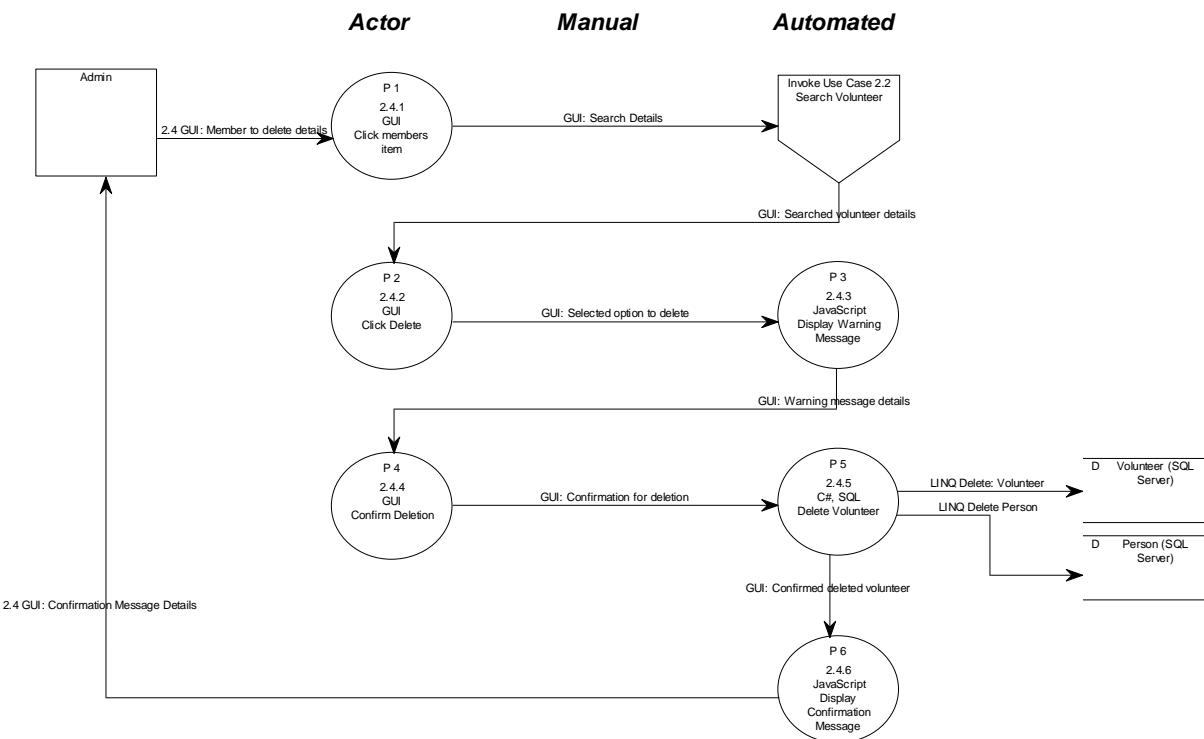


Figure 124: 2.4 Primitive Level Data Flow Diagram

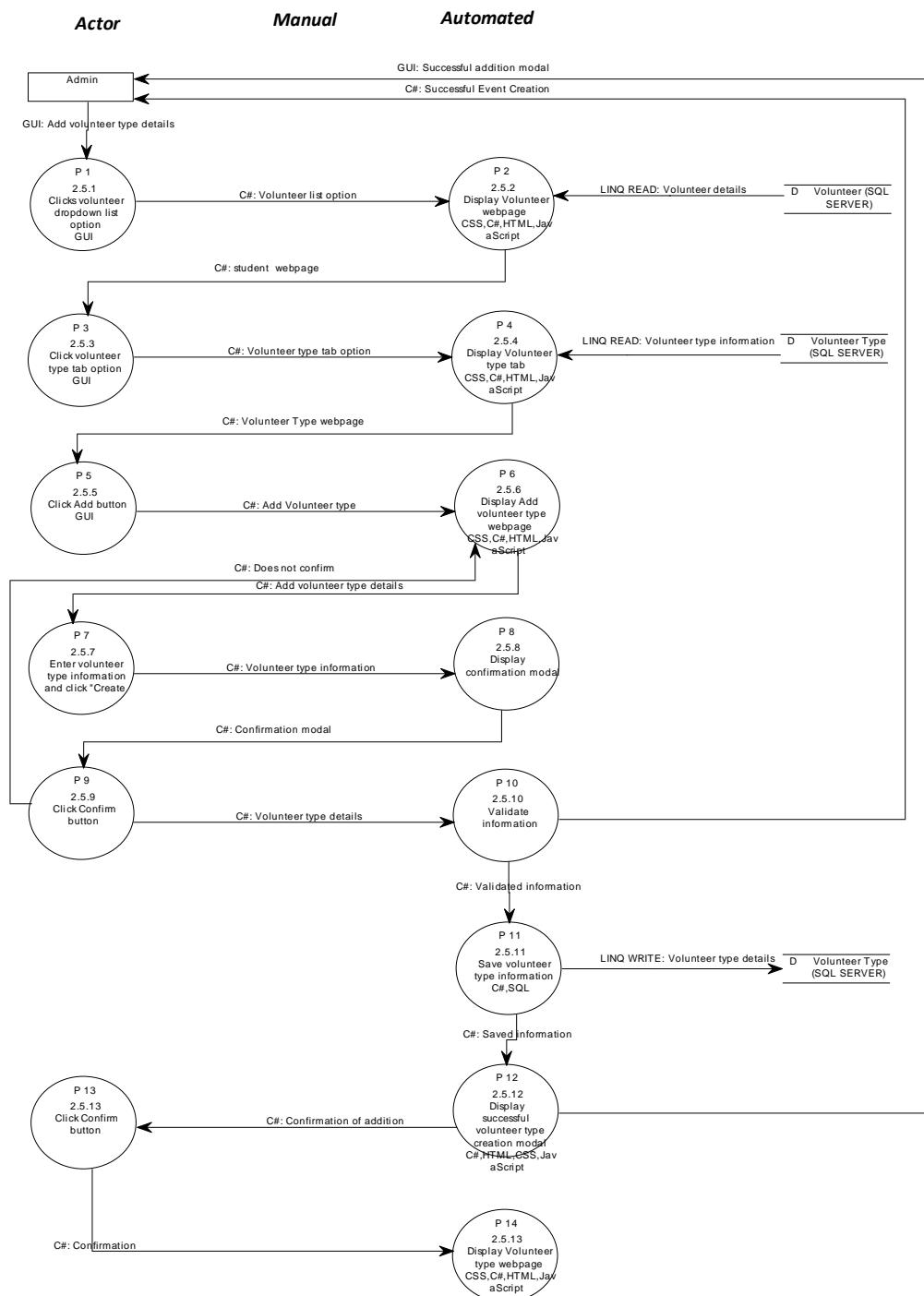


Figure 125: 2.5 Primitive Level Data Flow Diagram

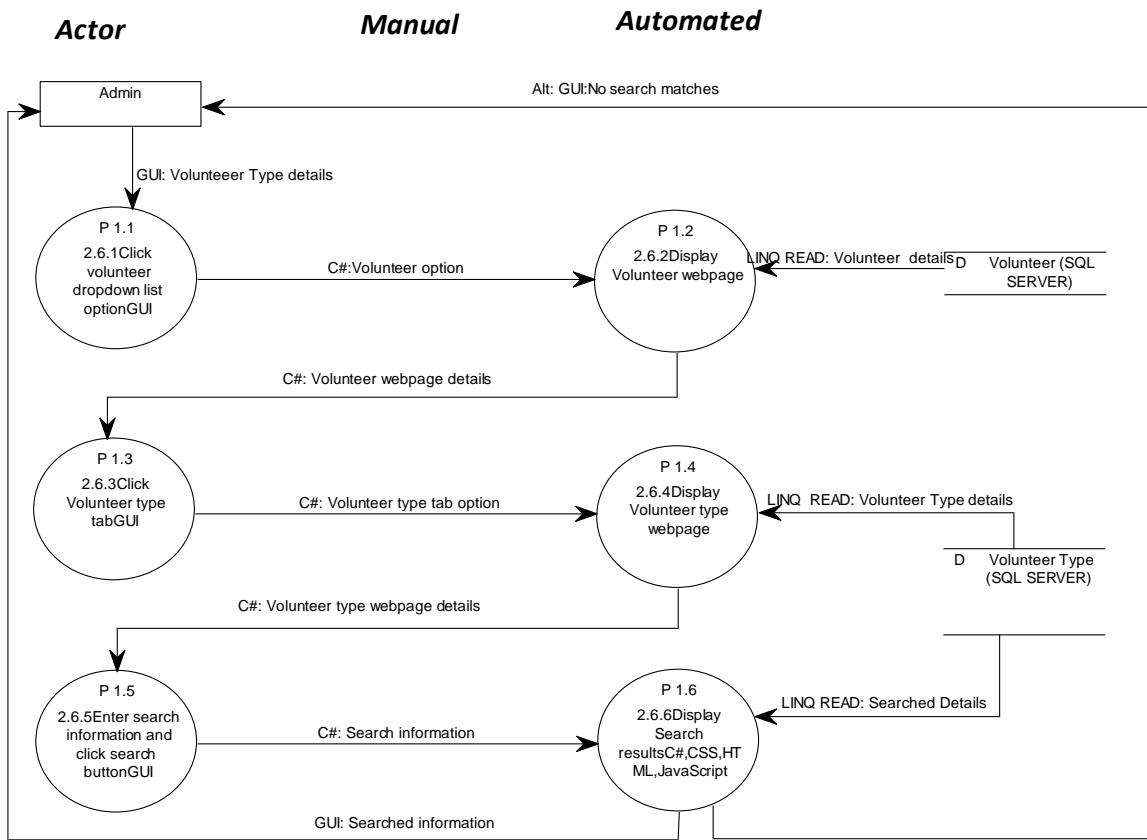


Figure 126: 2.6 Primitive Level Data Flow Diagram

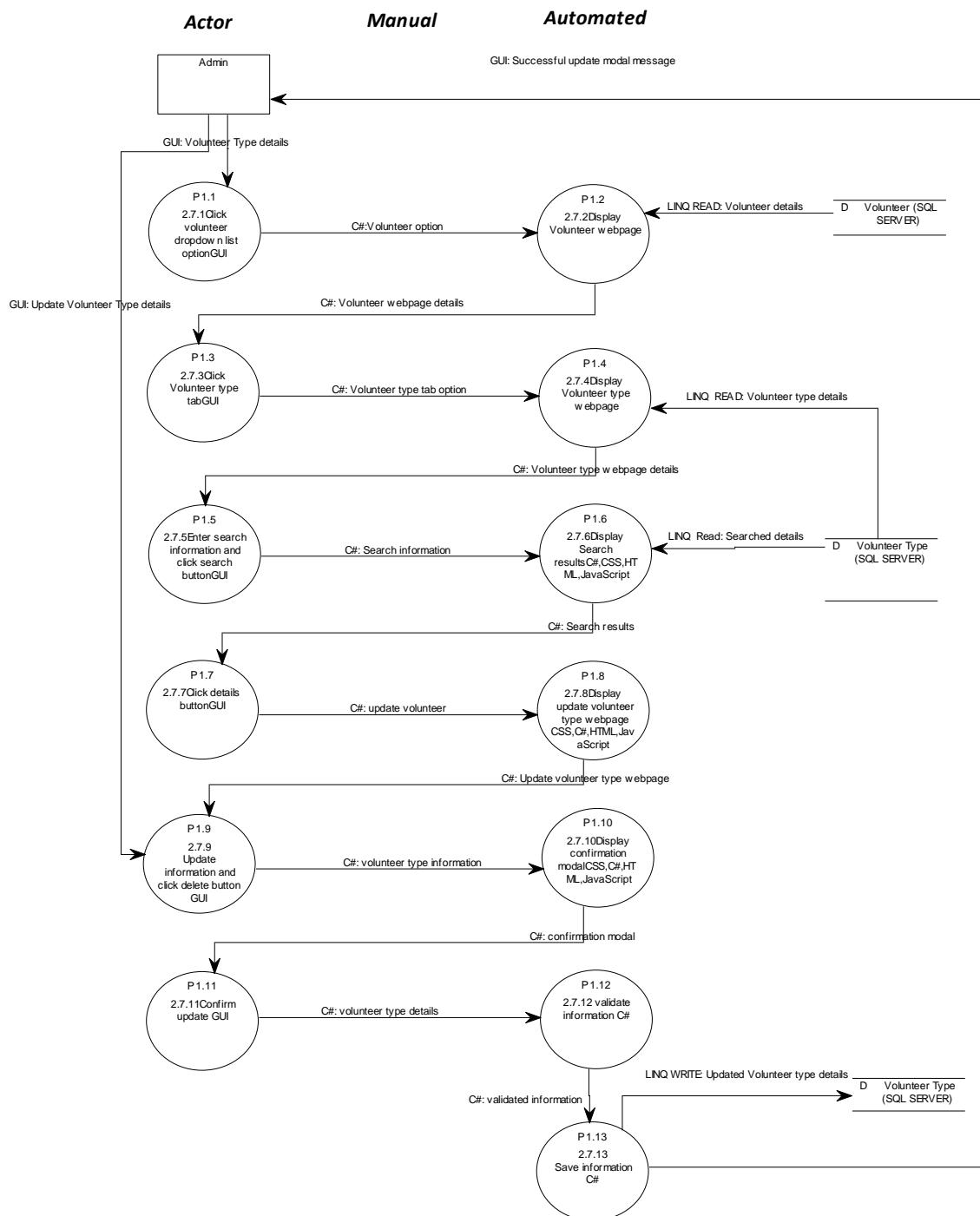


Figure 127: 2.7 Primitive Level Data Flow Diagram

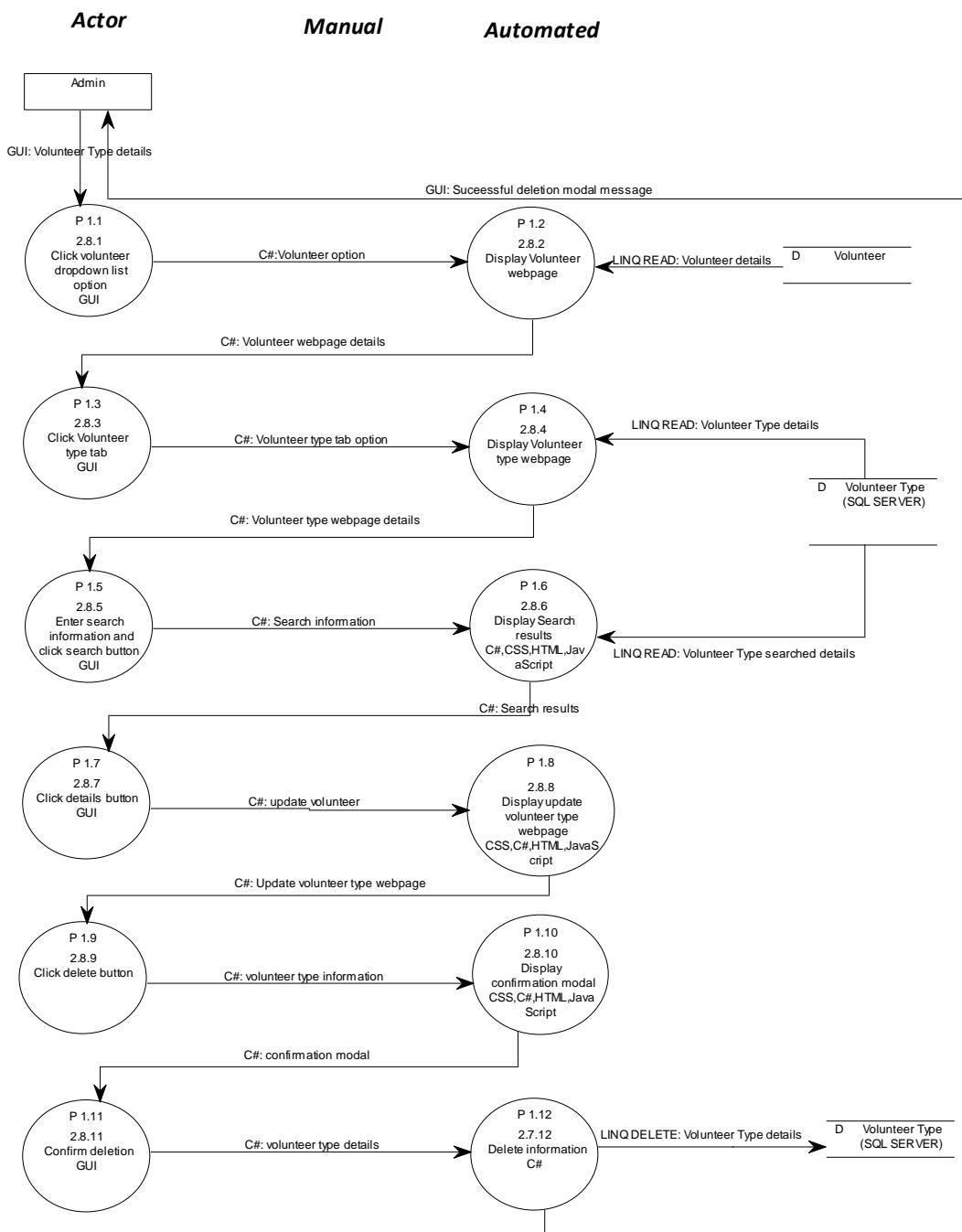


Figure 128 2.8 Primitive Level Data Flow Diagram

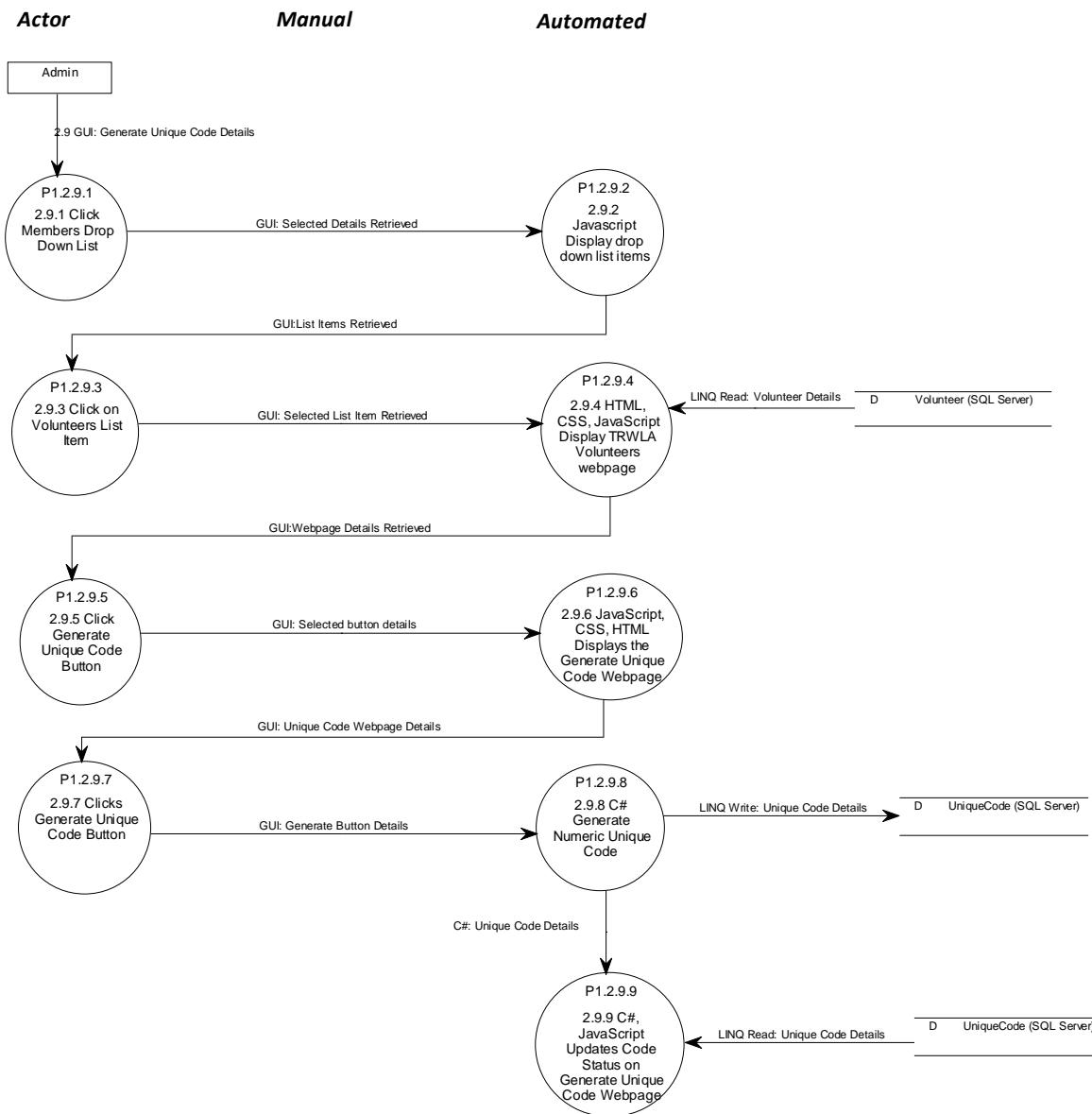


Figure 129: 2.9 Primitive Level Data Flow Diagram

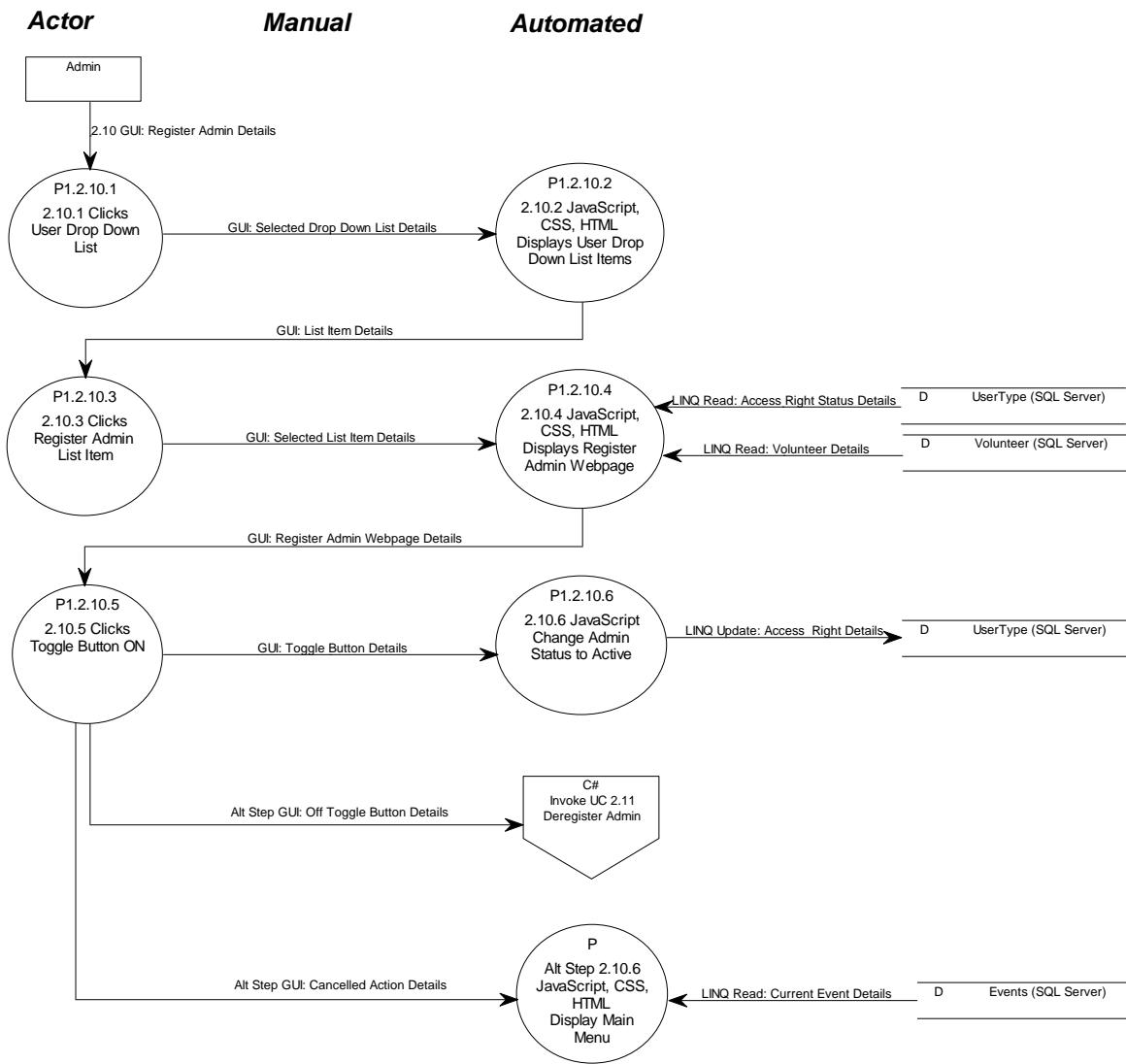


Figure 130: 2.10 Primitive Level Data Flow Diagram

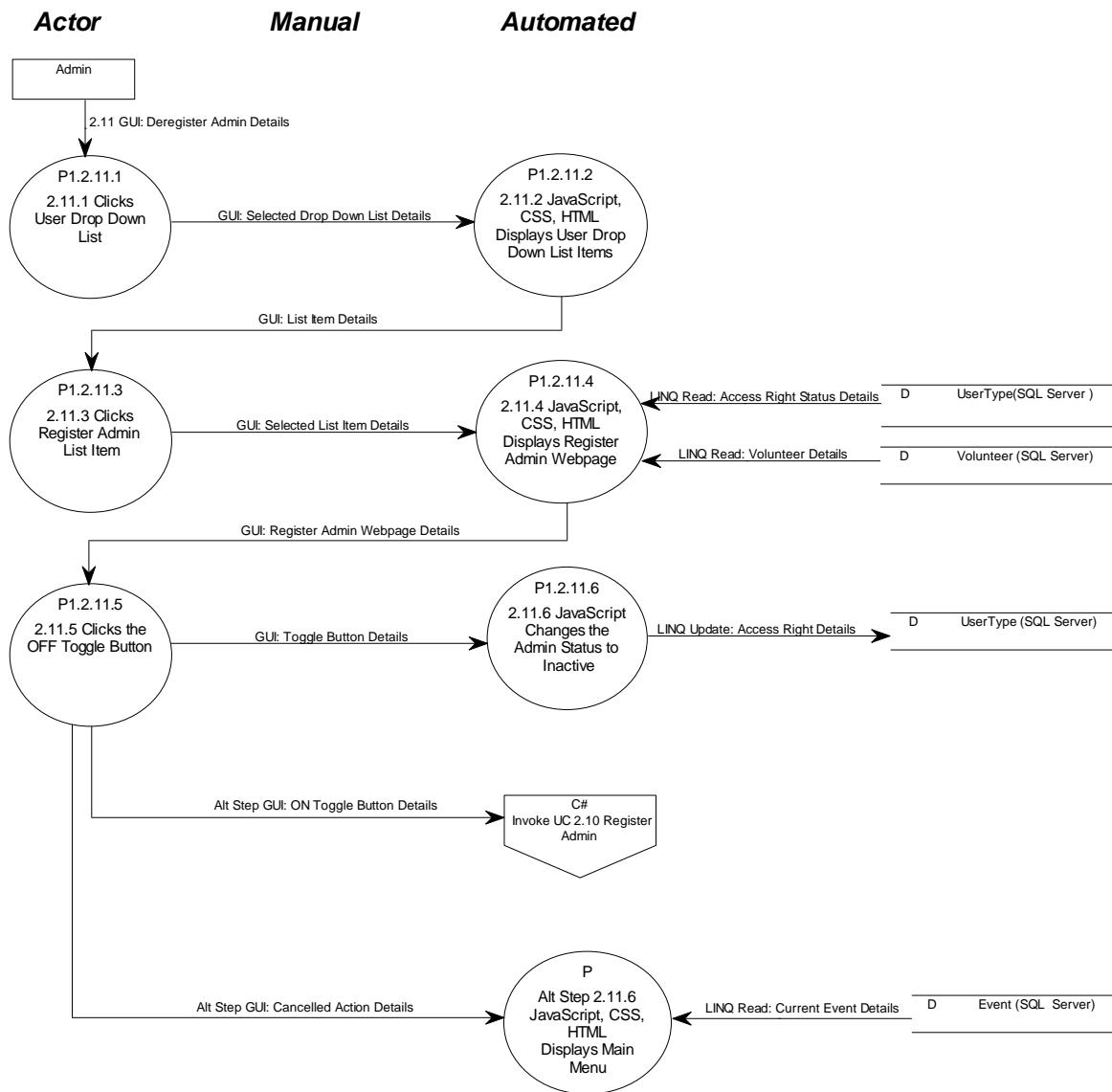


Figure 131: 2.11 Primitive Level Data Flow Diagram

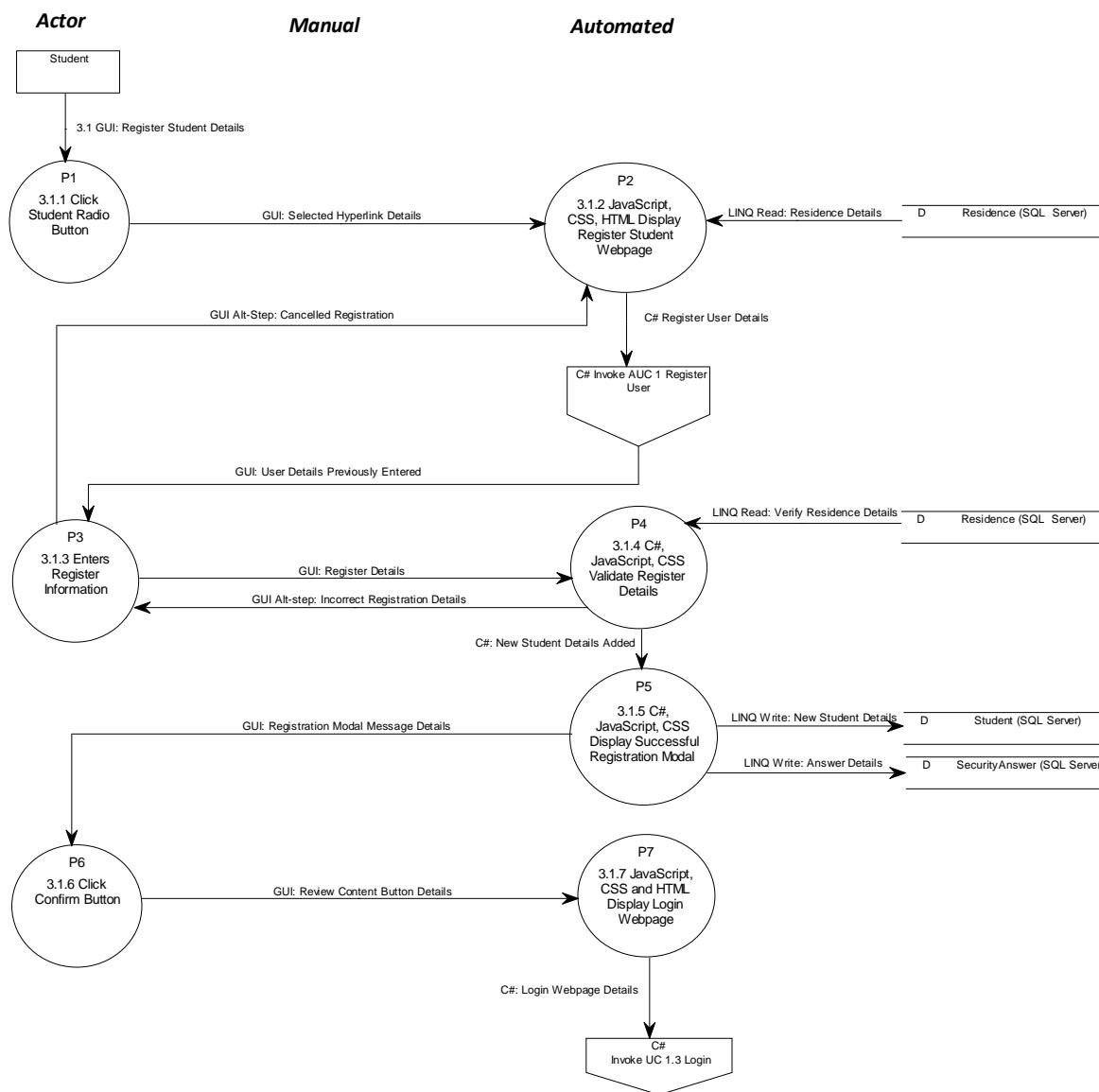


Figure 132: 3.1 Primitive Level Data Flow Diagram

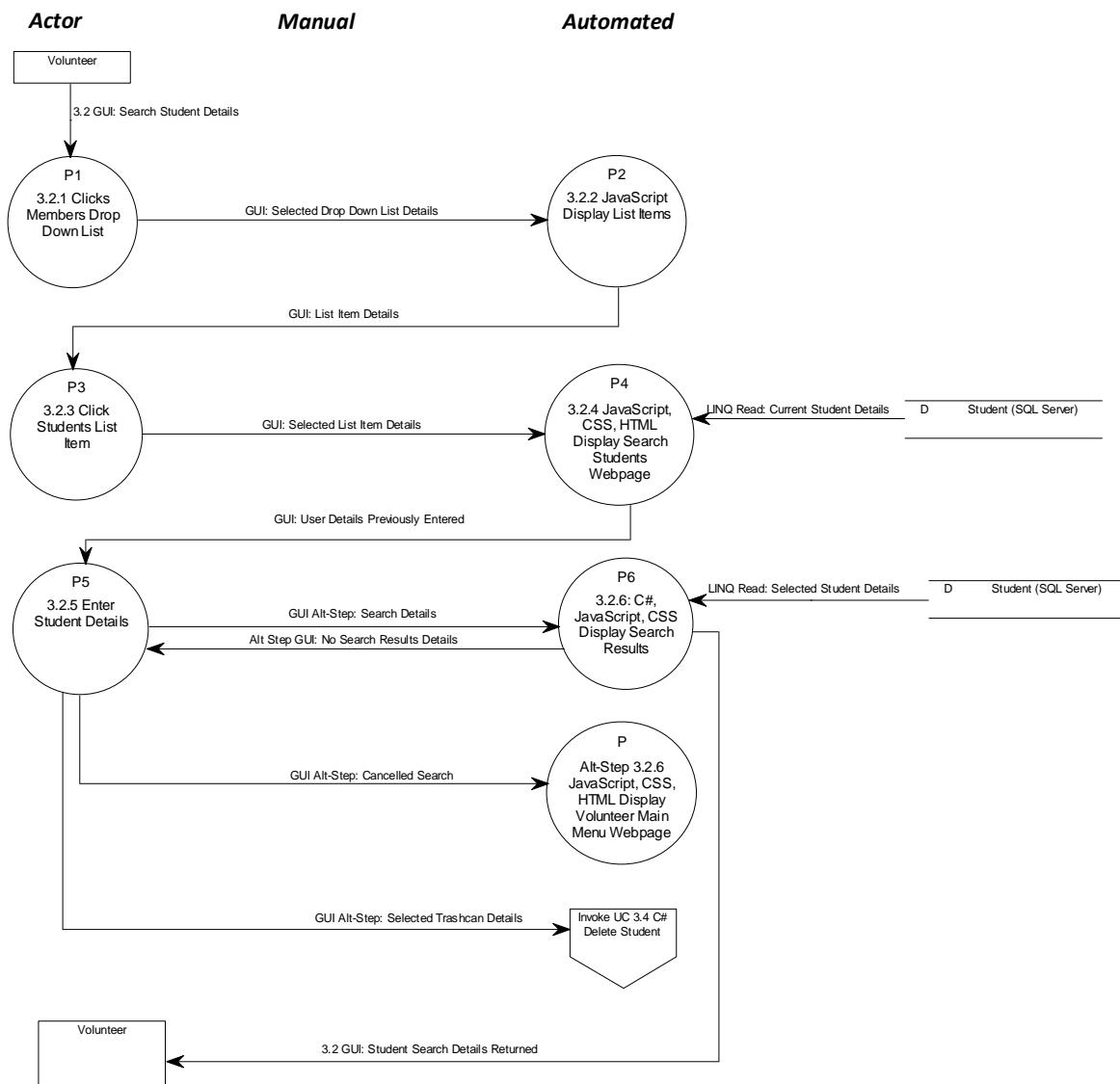


Figure 133: 3.2 Primitive Level Data Flow Diagram

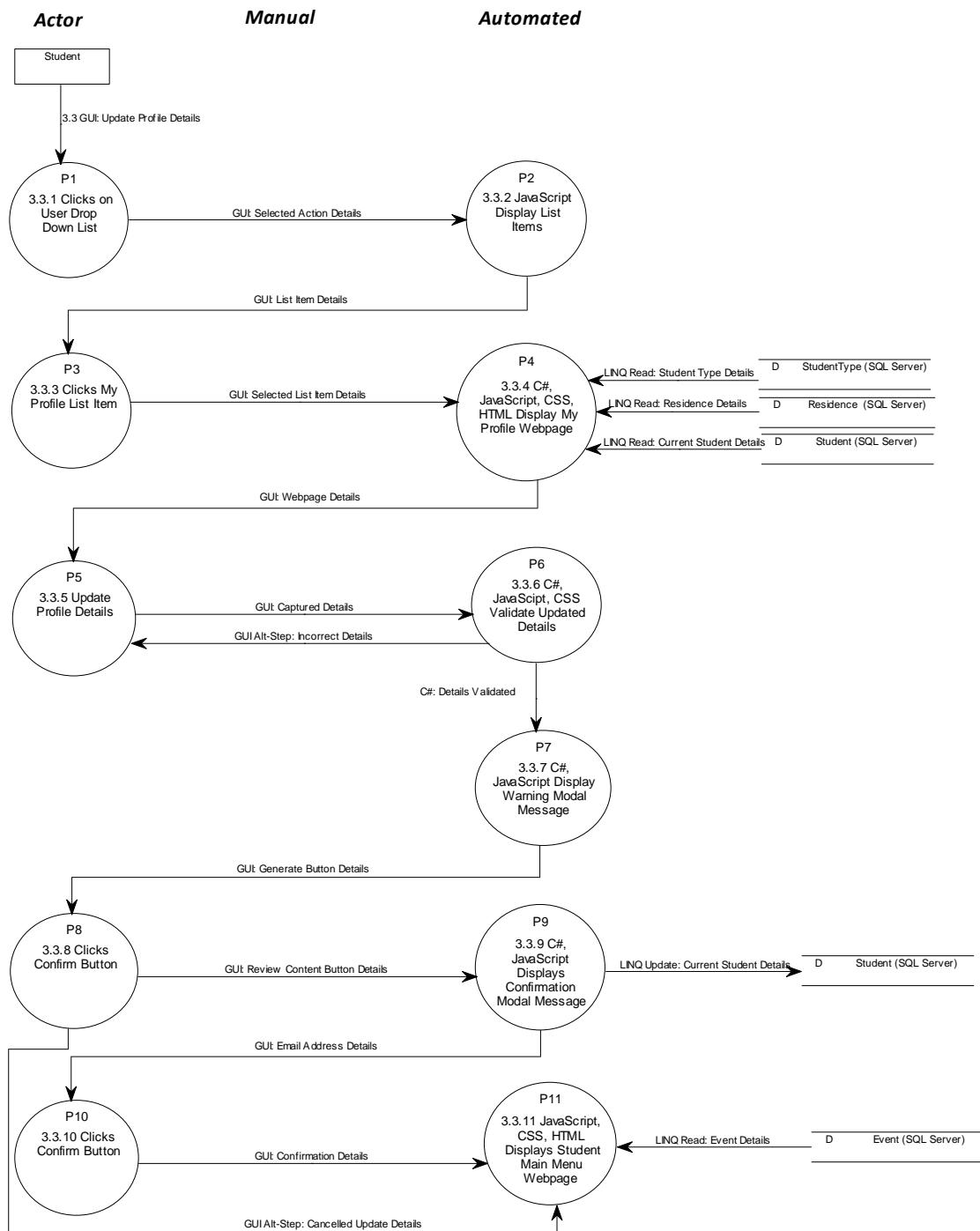


Figure 134: 3.3 Primitive Level Data Flow Diagram

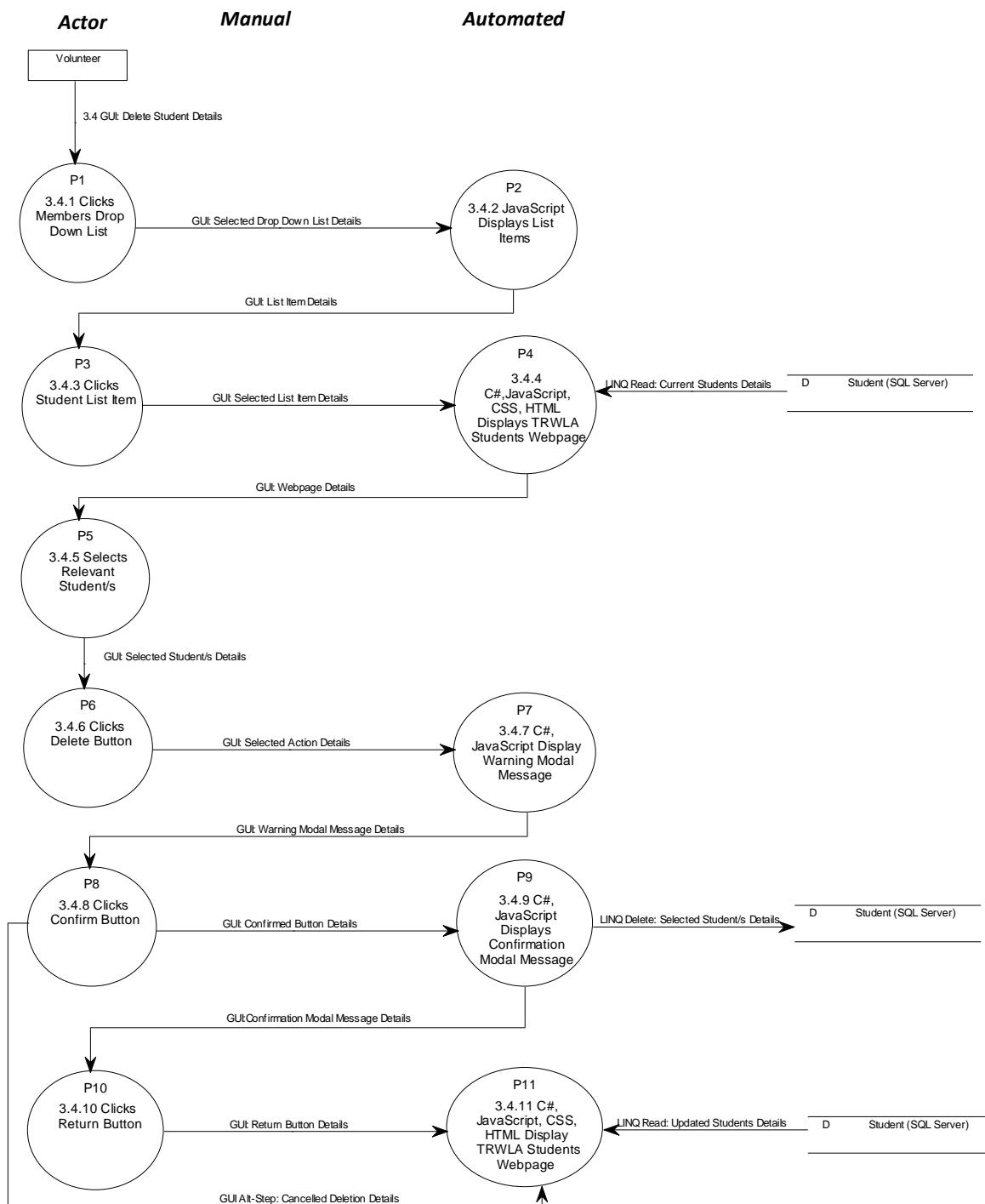


Figure 135: 3.4 Primitive Level Data Flow Diagram

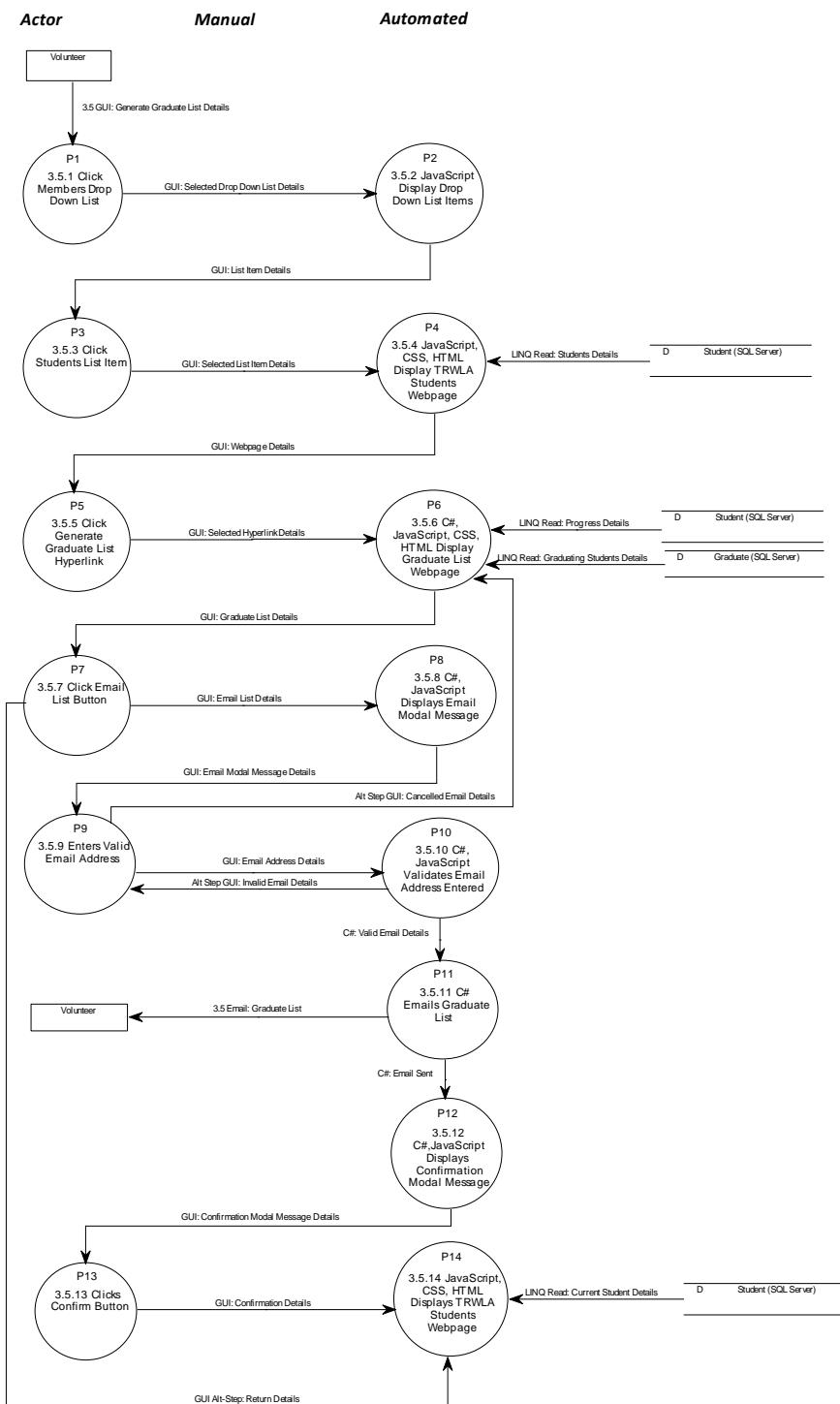


Figure 136: 3.5 Primitive Level Data Flow Diagram

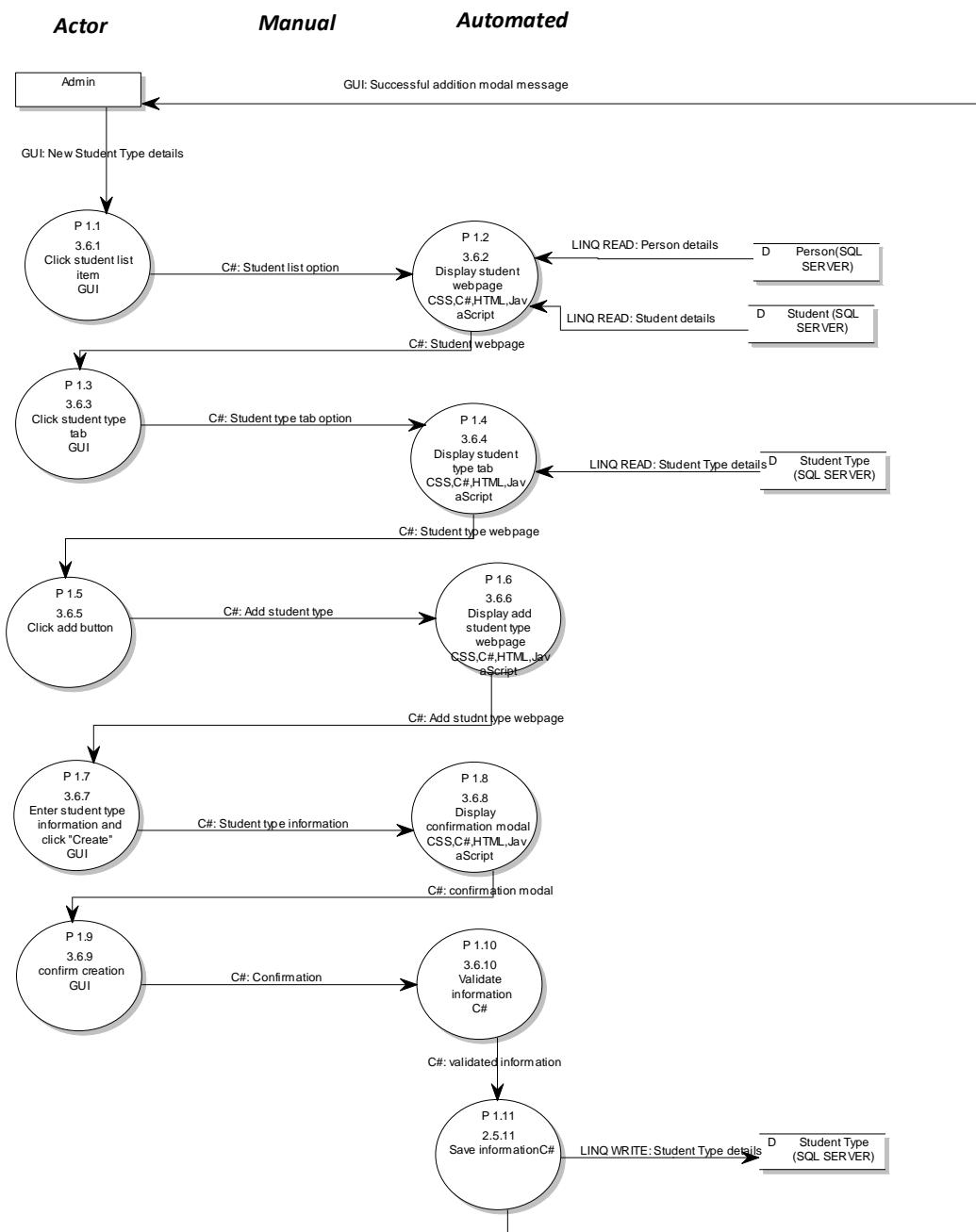


Figure 137: 3.6 Primitive Level Data Flow Diagram

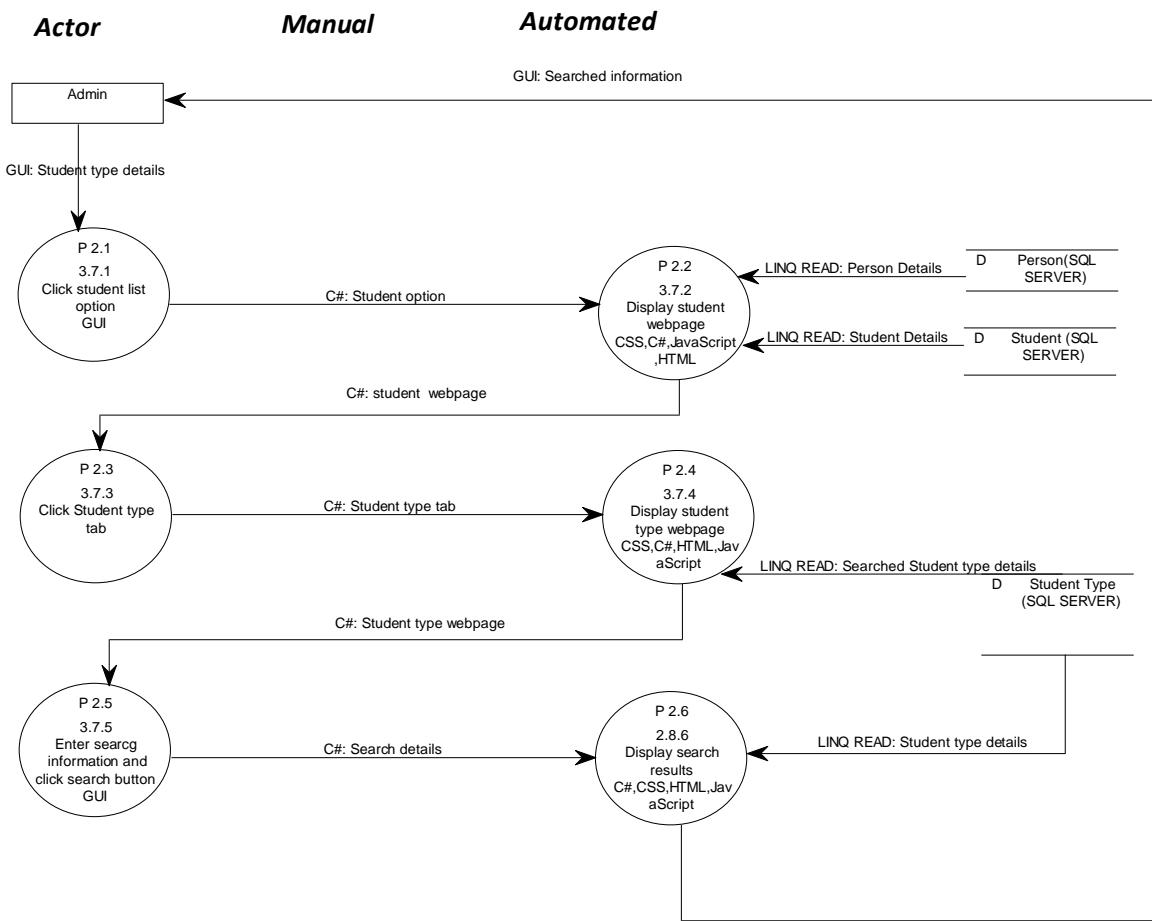


Figure 138: 3.7 Primitive Level Data Flow Diagram

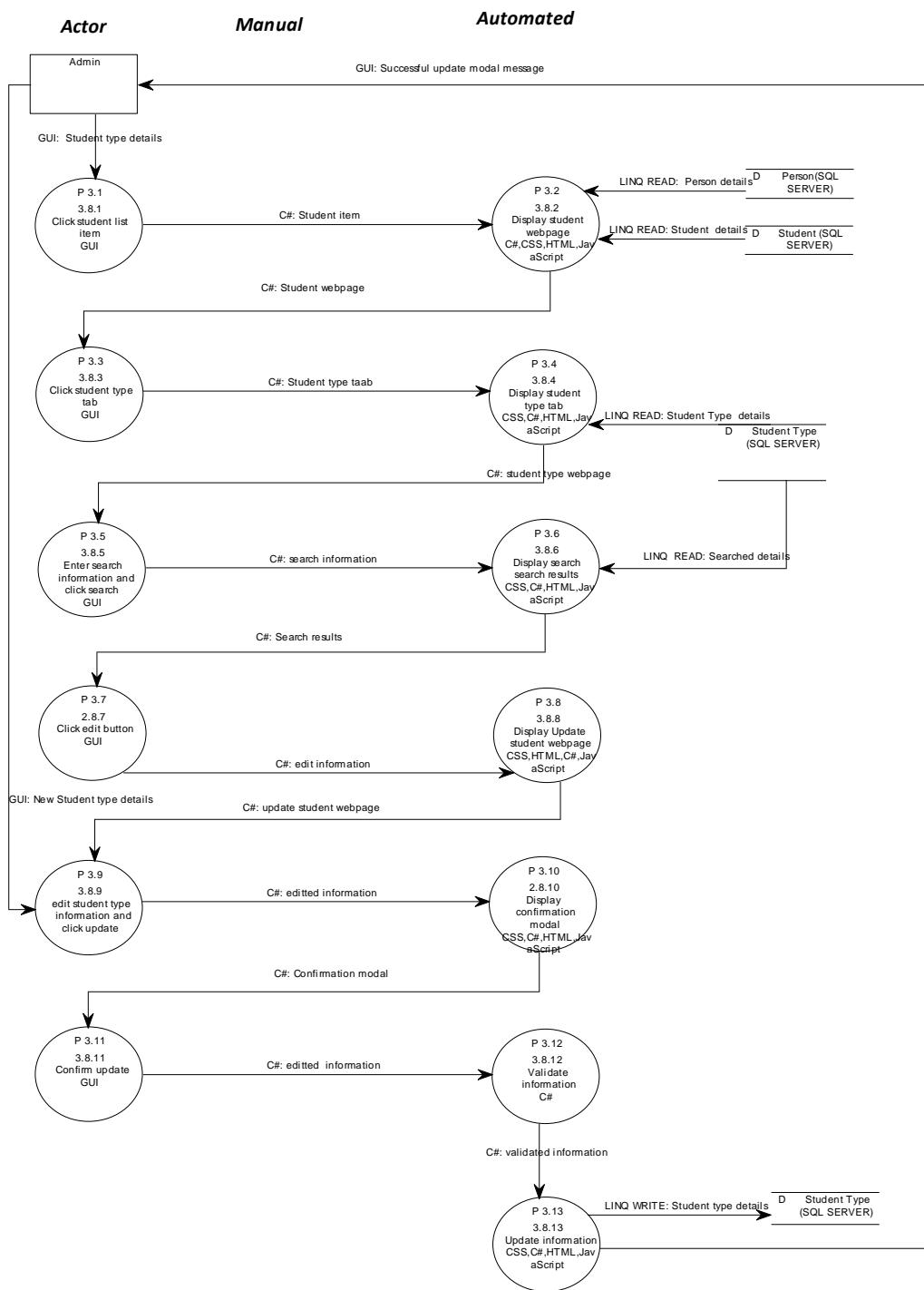


Figure 139: 3.8 Primitive Level Data Flow Diagram

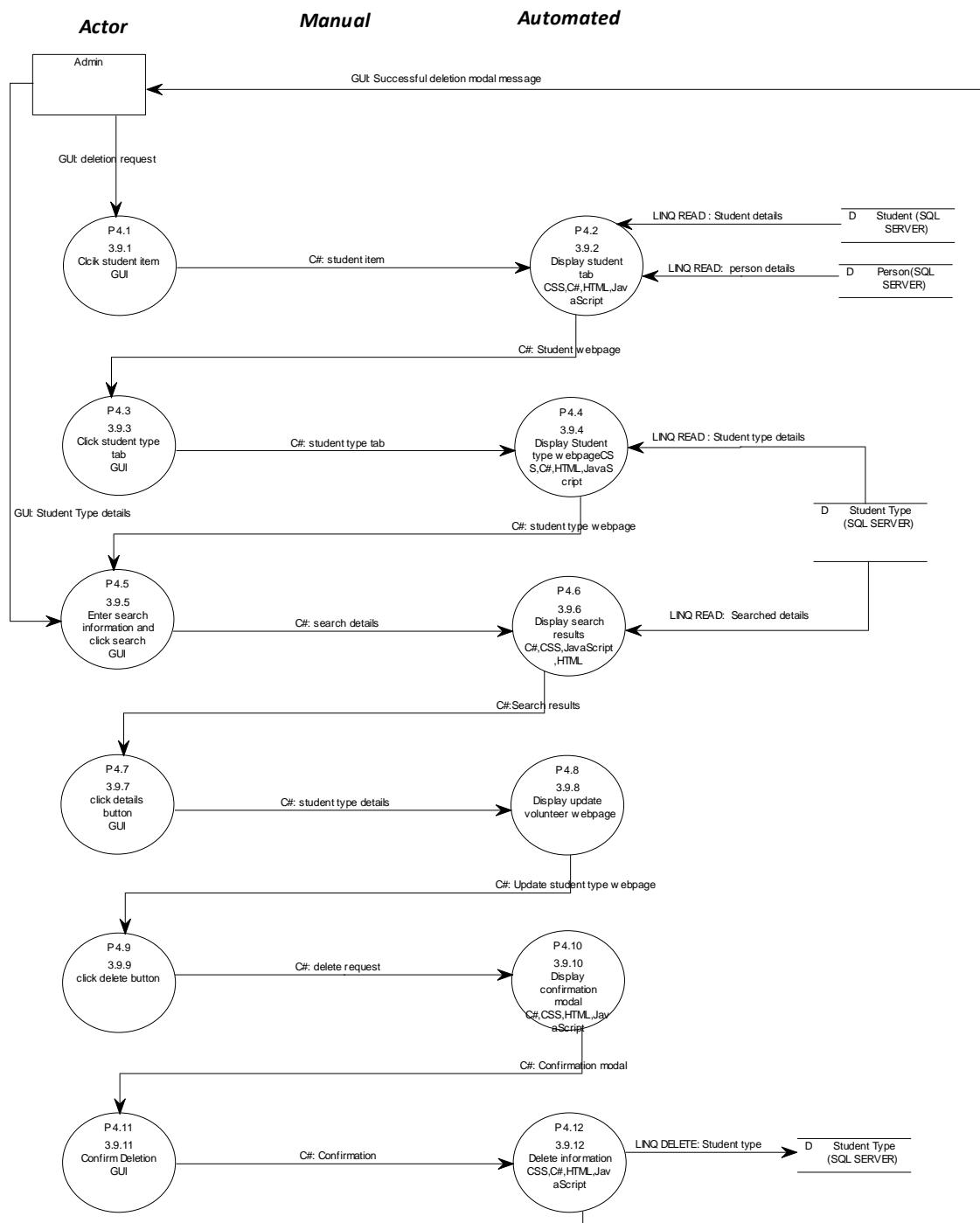


Figure 140: 3.9 Primitive Level Data Flow Diagram

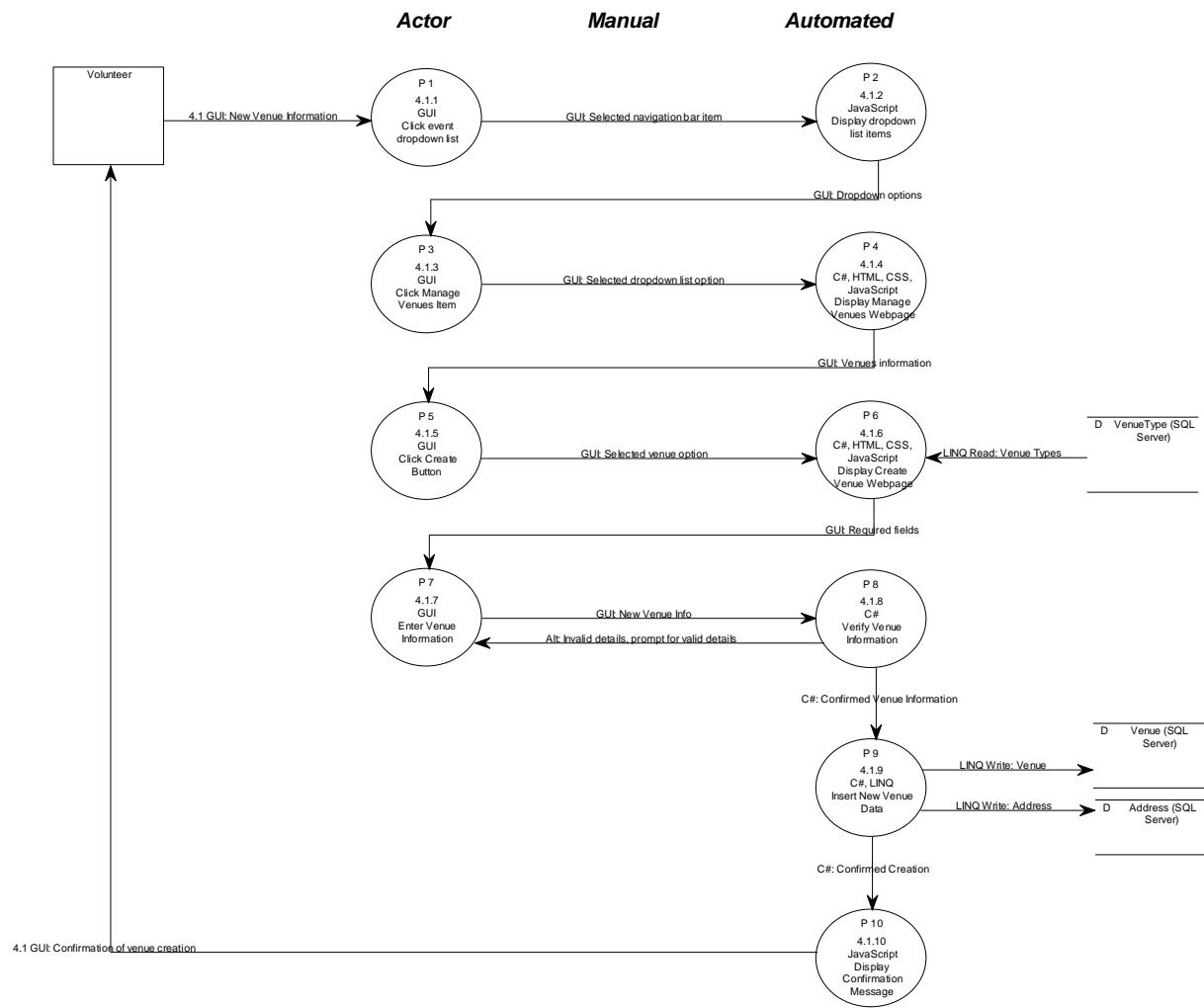


Figure 141: 4.1 Primitive Level Data Flow Diagram

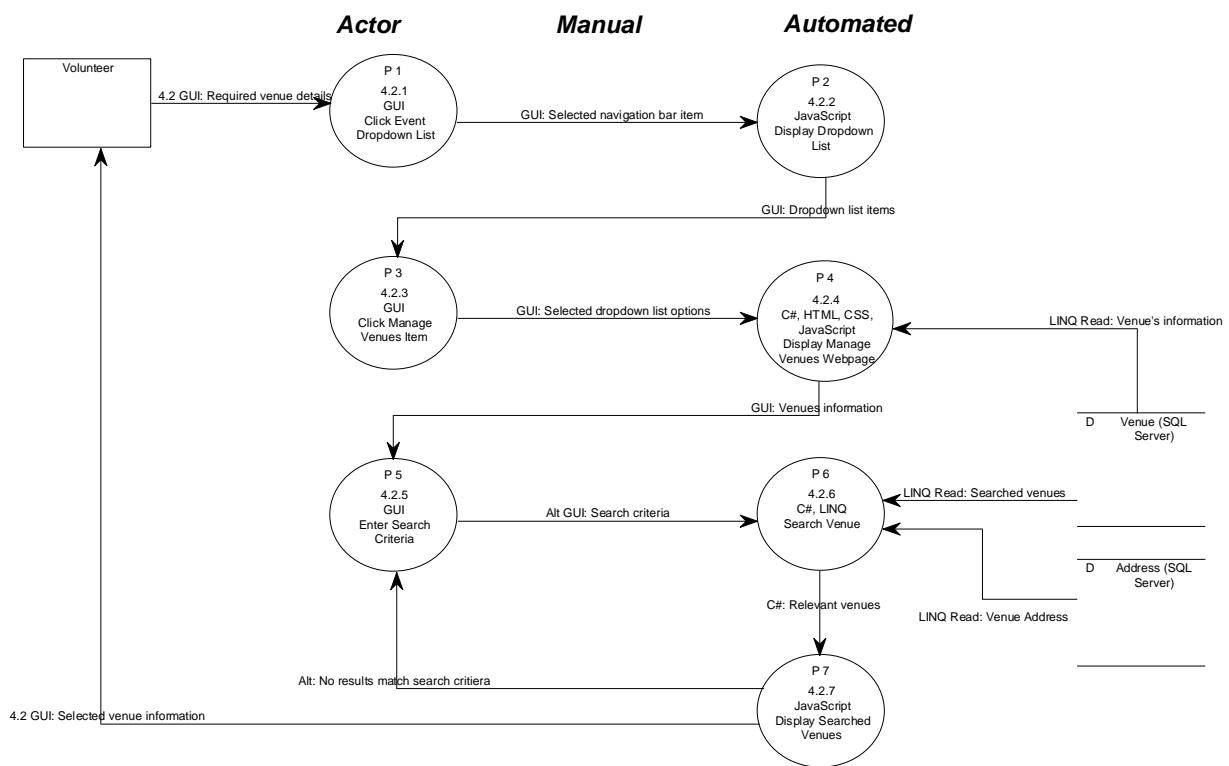


Figure 142: 4.2 Primitive Level Data Flow Diagram

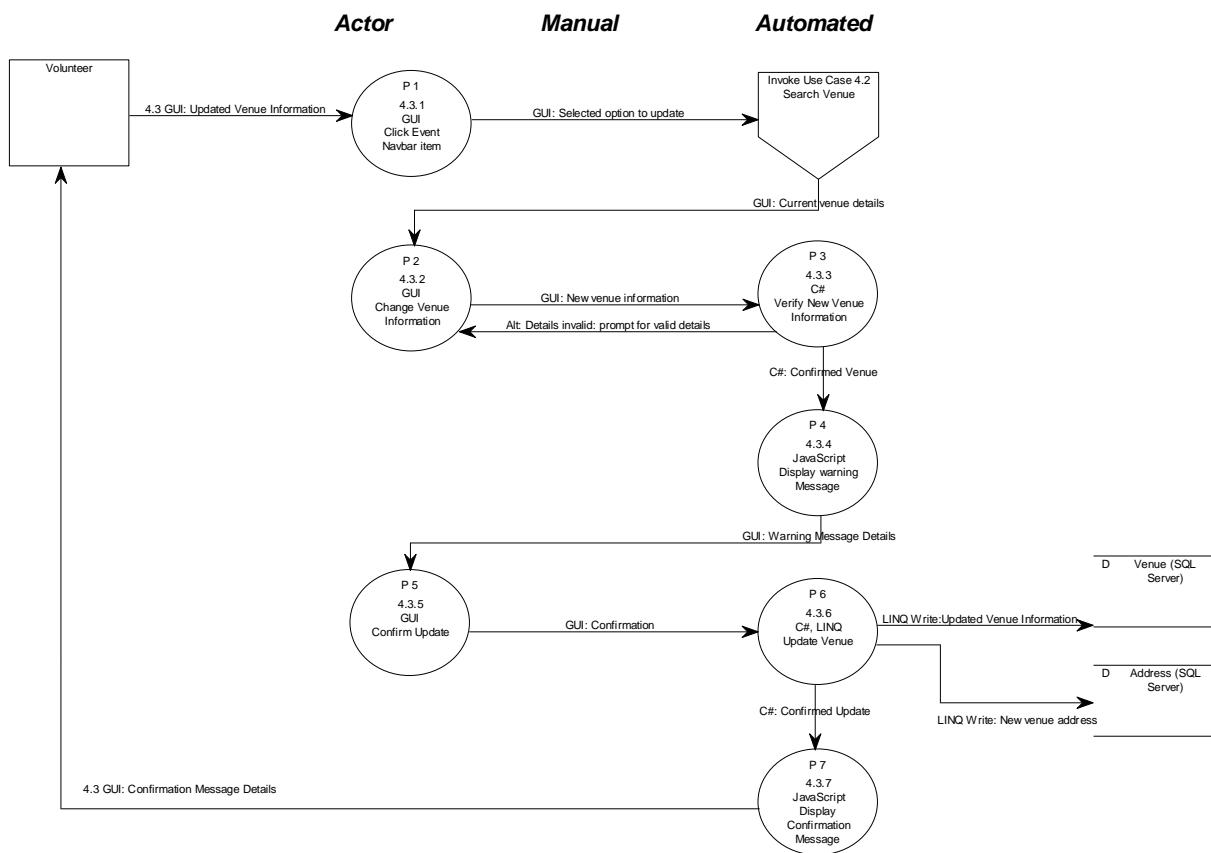


Figure 143: 4.3 Primitive Level Data Flow Diagram

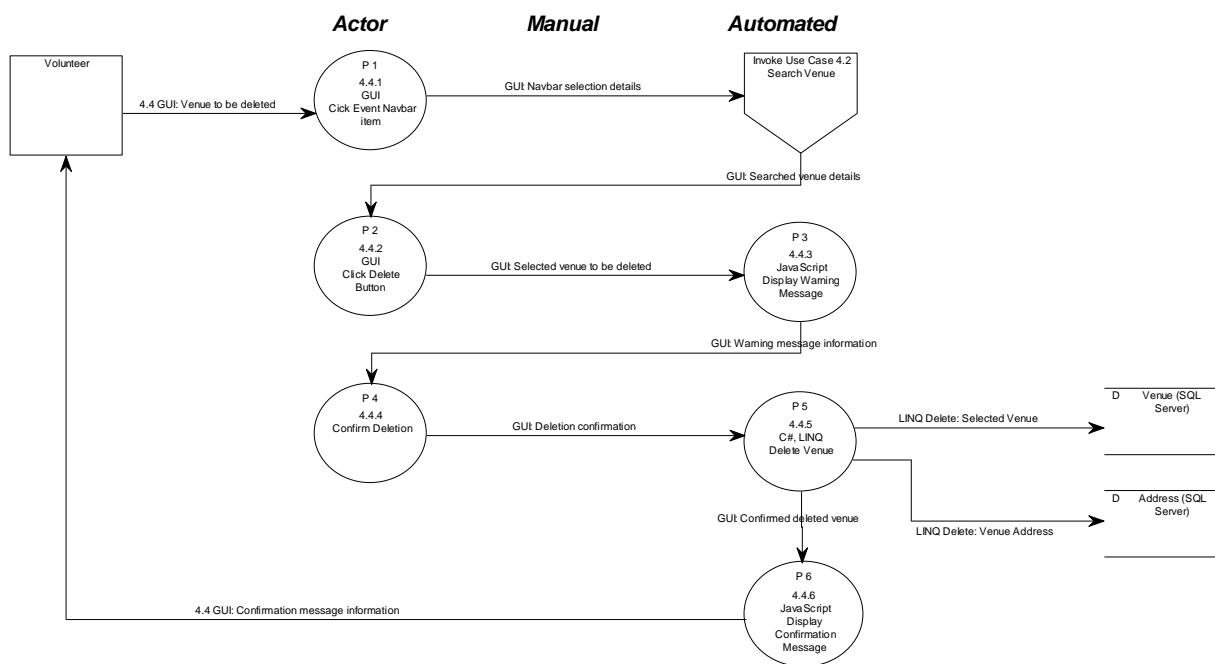


Figure 144: 4.4 Primitive Level Data Flow Diagram

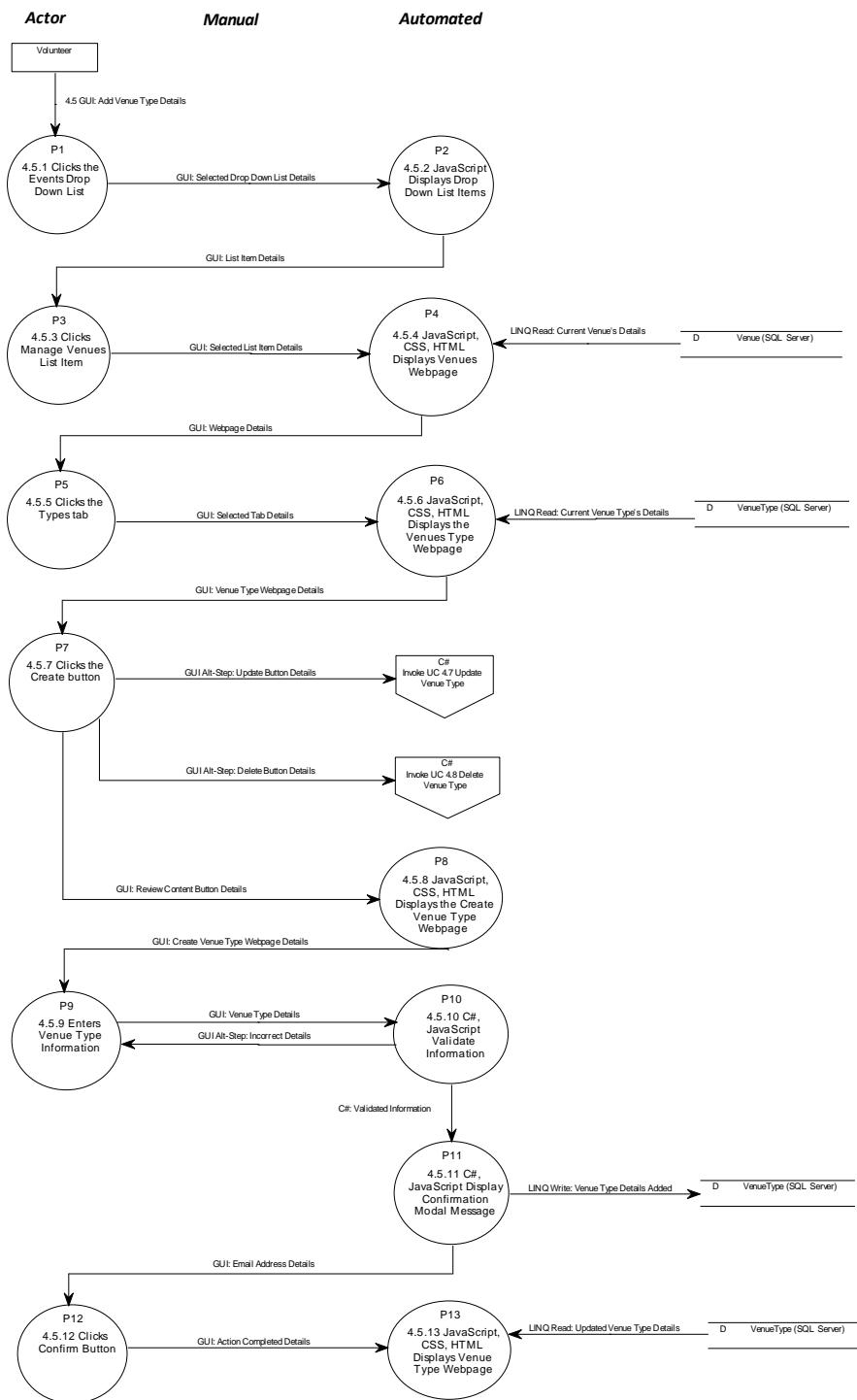


Figure 145: 4.5 Primitive Level Data Flow Diagram

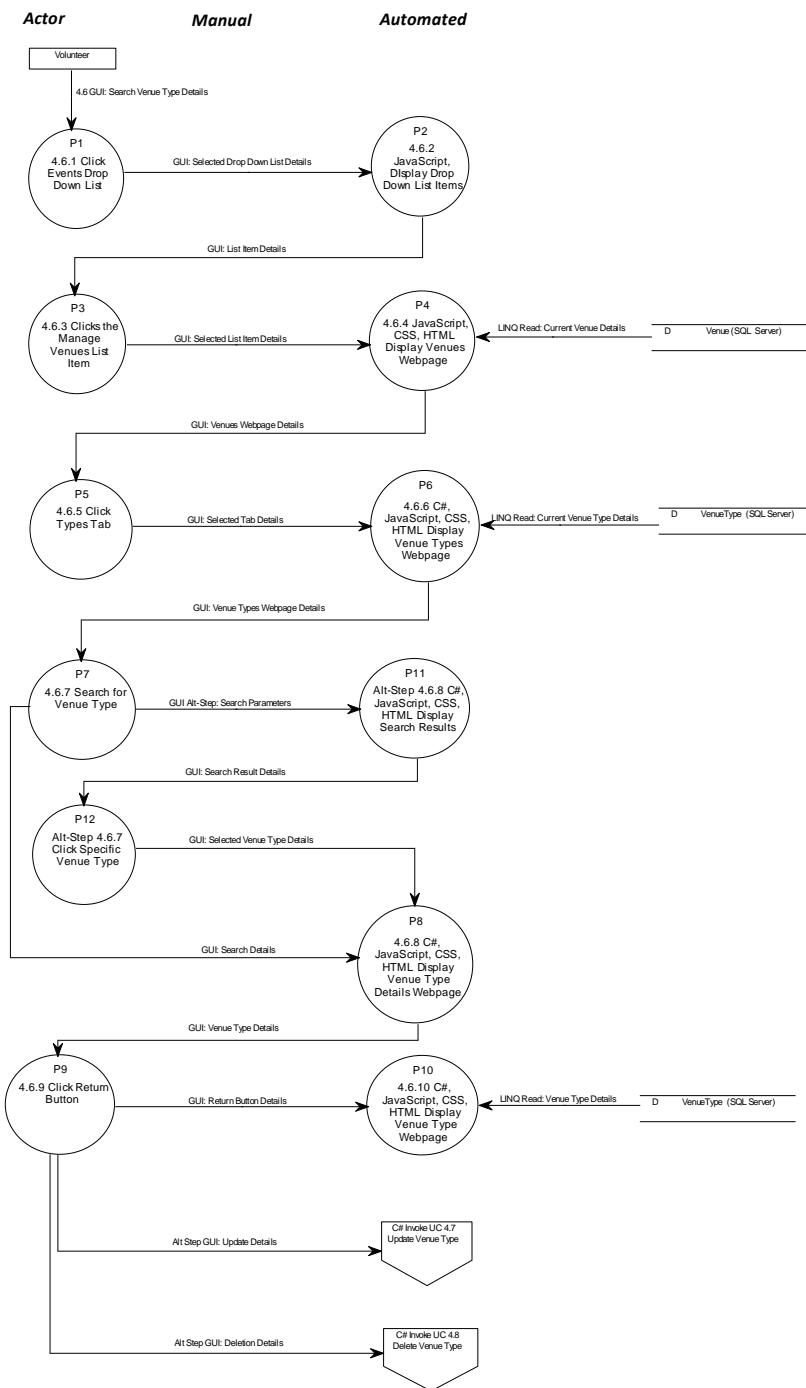


Figure 146: 4.6 Primitive Level Data Flow Diagram

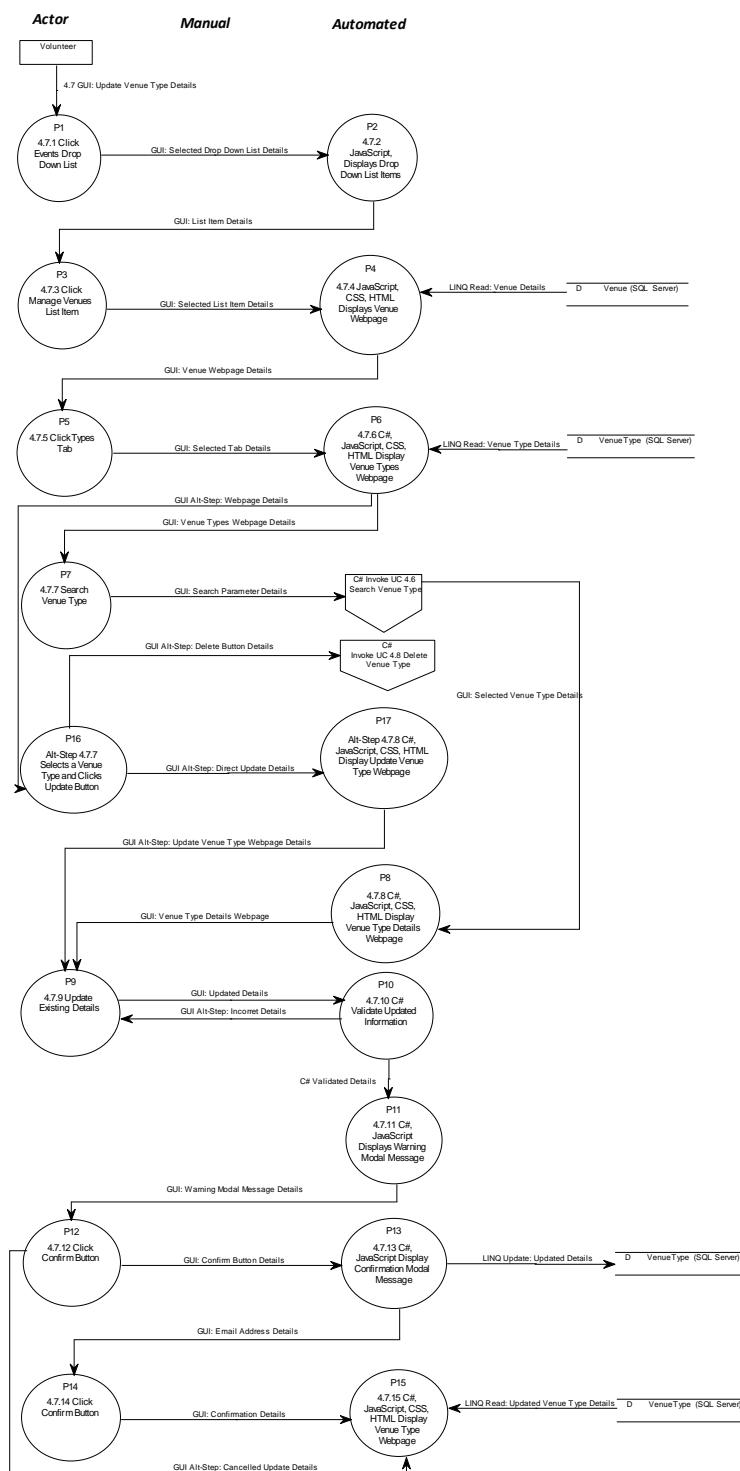


Figure 147: 4.7 Primitive Level Data Flow Diagram

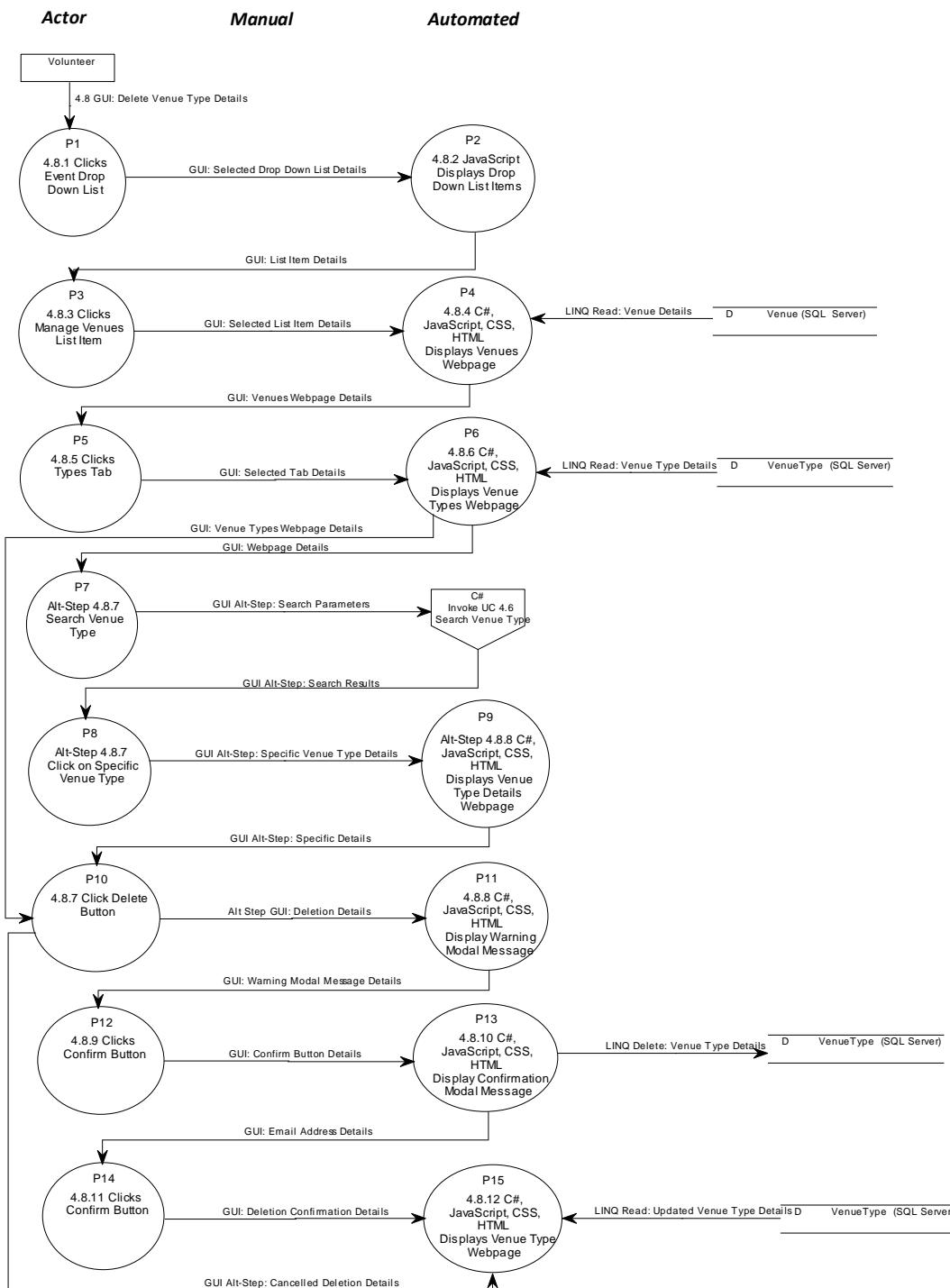


Figure 148: 4.8 Primitive Level Data Flow Diagram

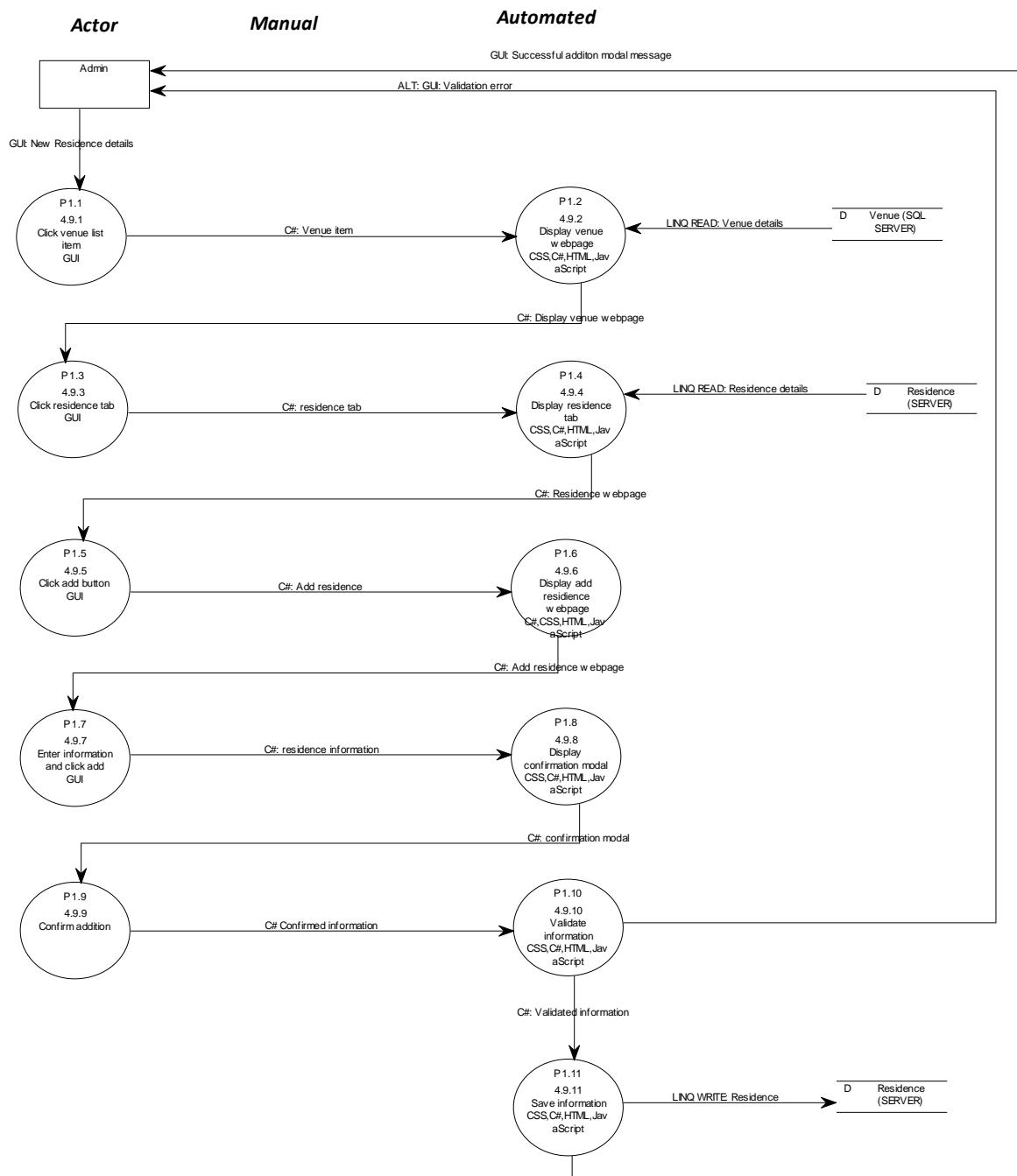


Figure 149: 4.9: Primitive Level Data Flow Diagram

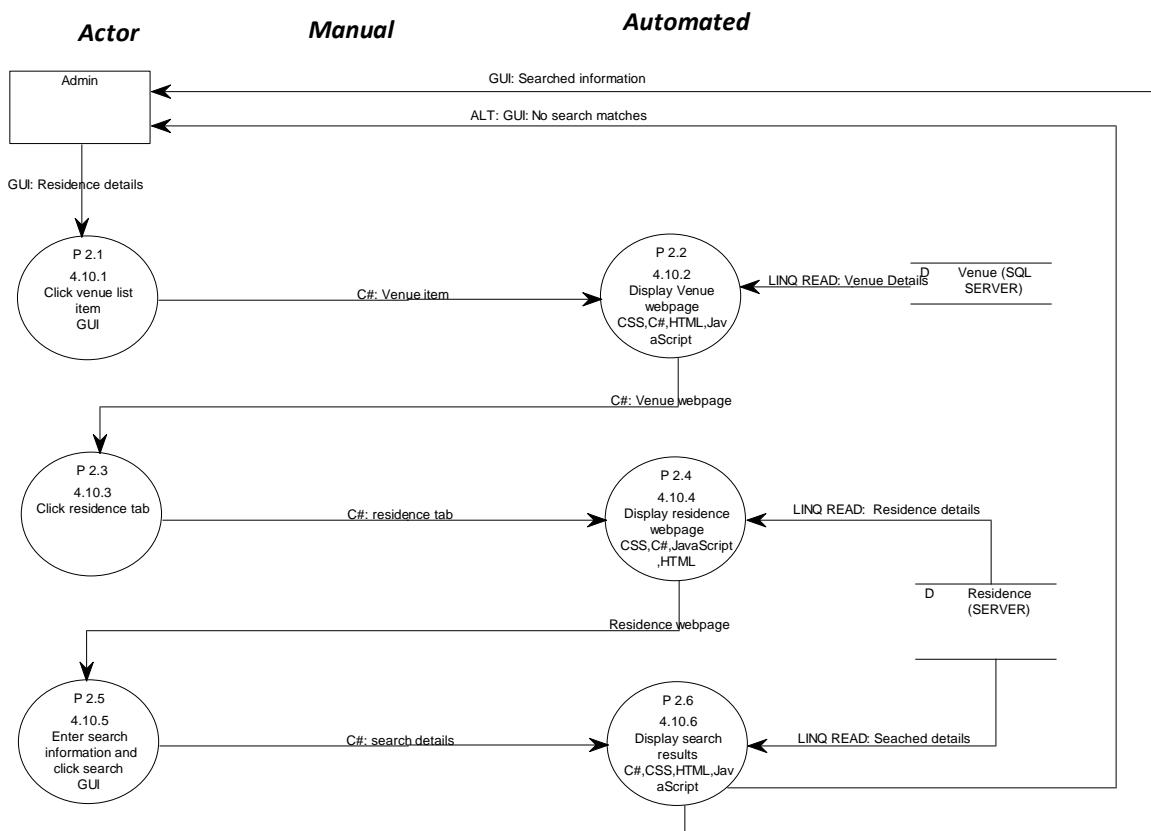


Figure 150: 4.10 Primitive Level Data Flow Diagram

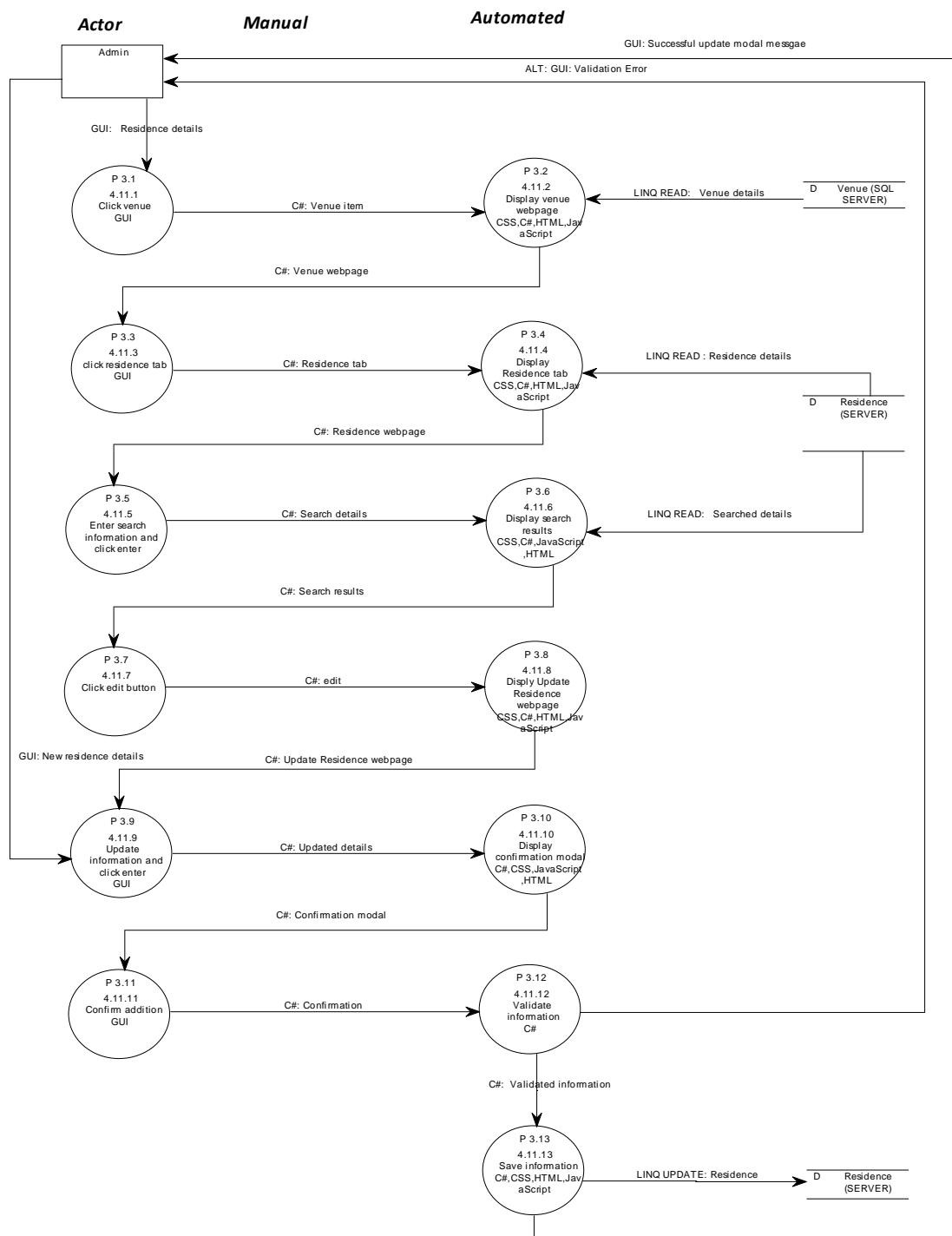


Figure 151: 4.11 Primitive Level Data Flow Diagram

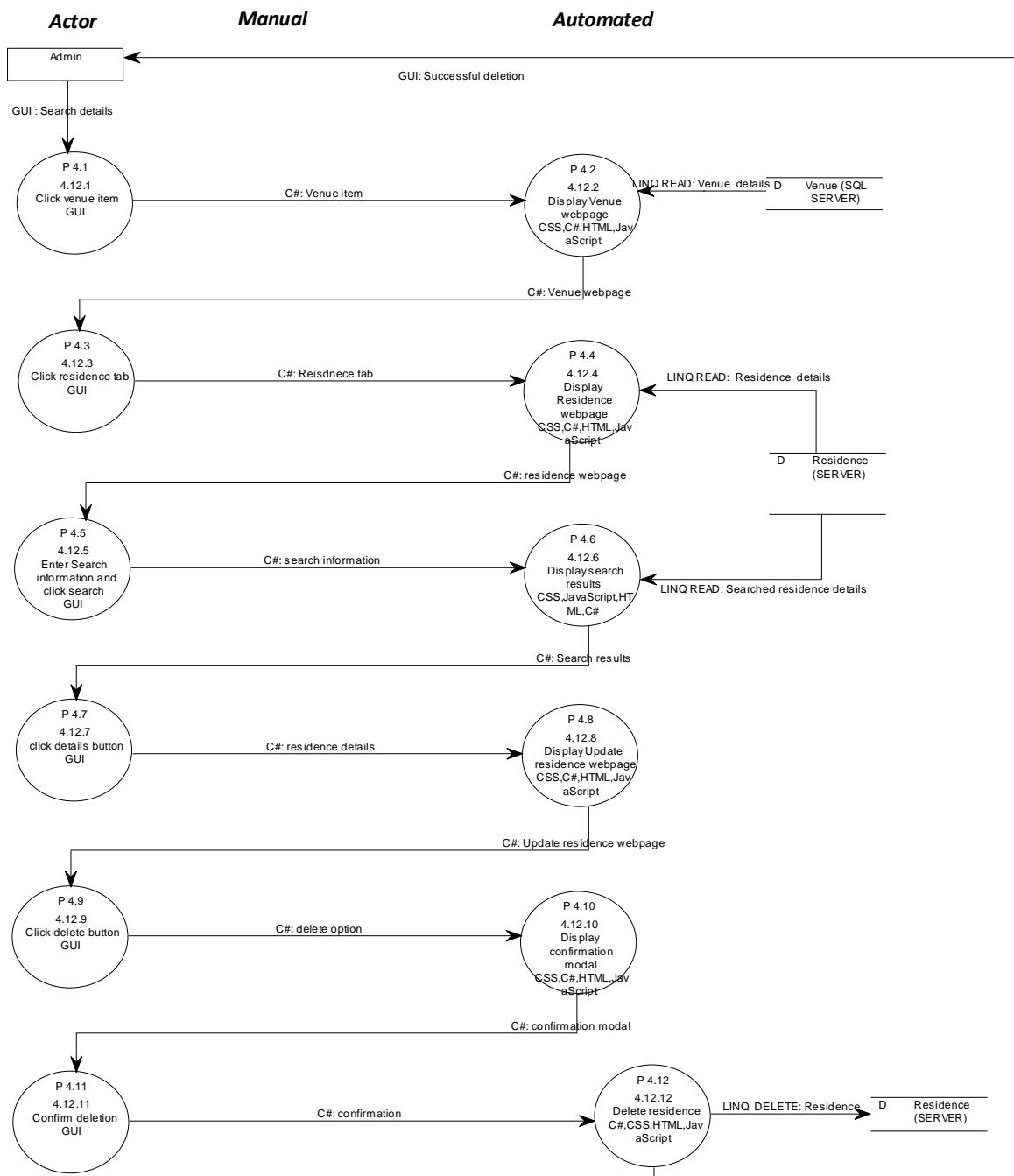


Figure 152: 4.12 Primitive Level Data Flow Diagram

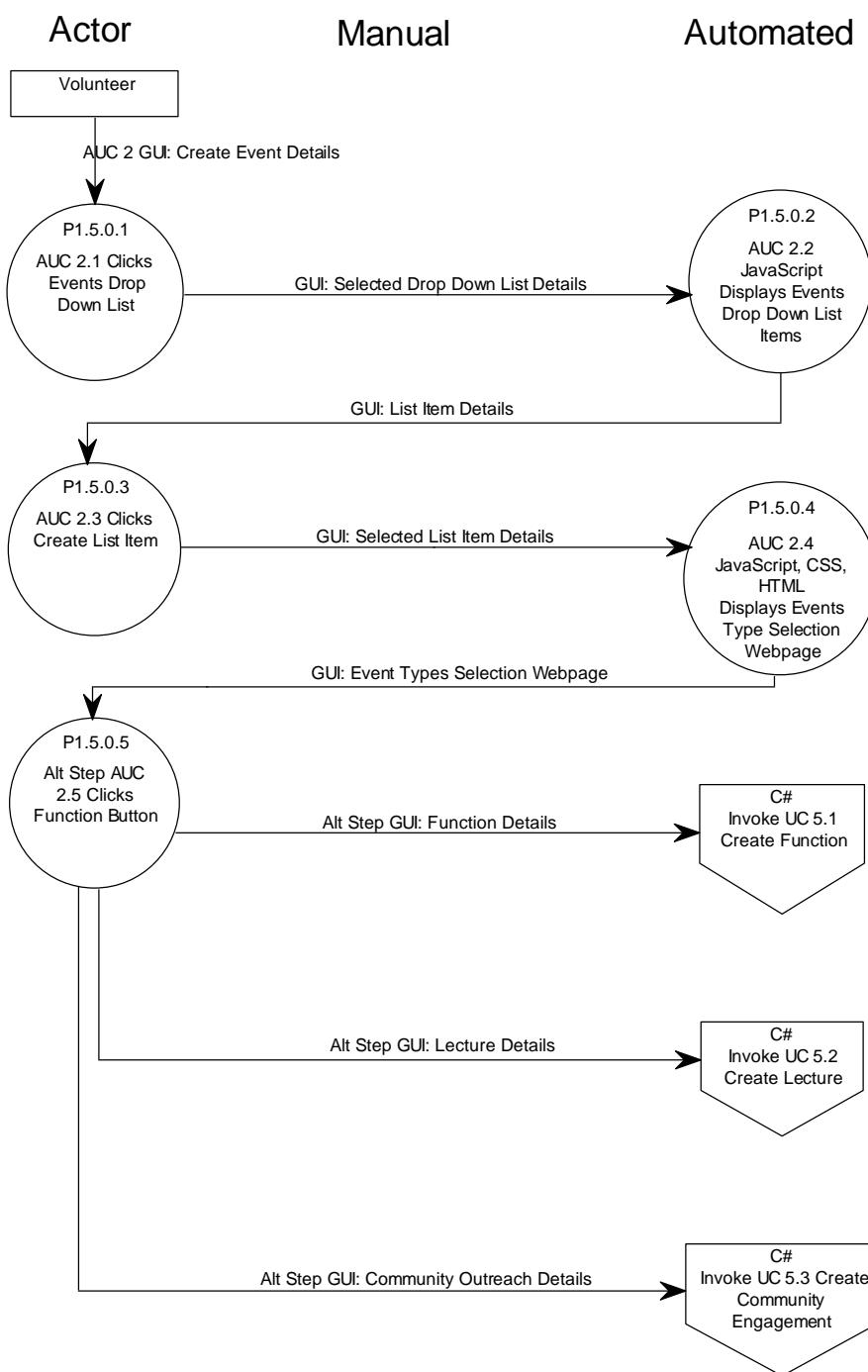


Figure 153: AUC 2 Primitive Level Data Flow Diagram

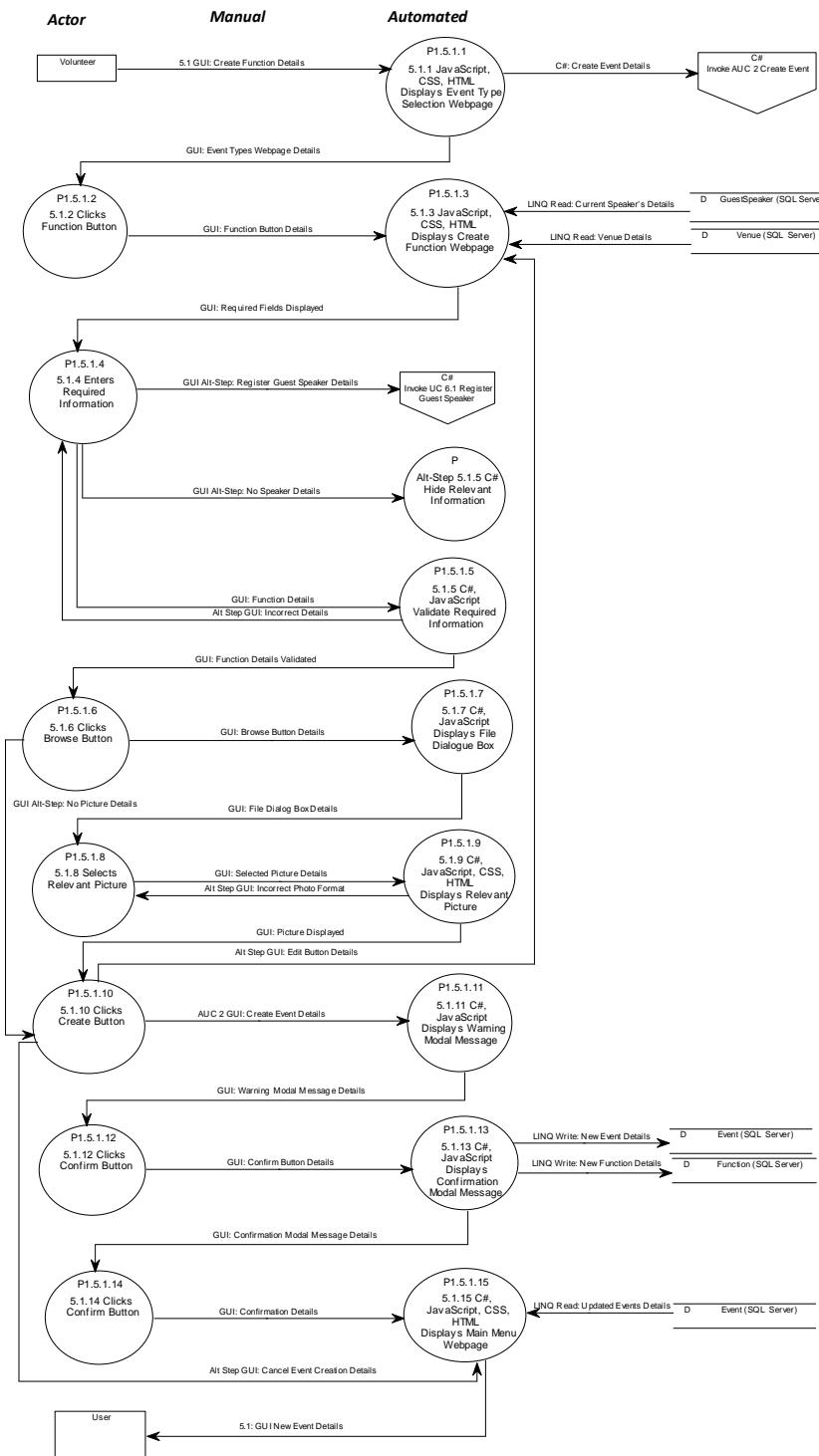


Figure 154: 5.1 Primitive Level Data Flow Diagram

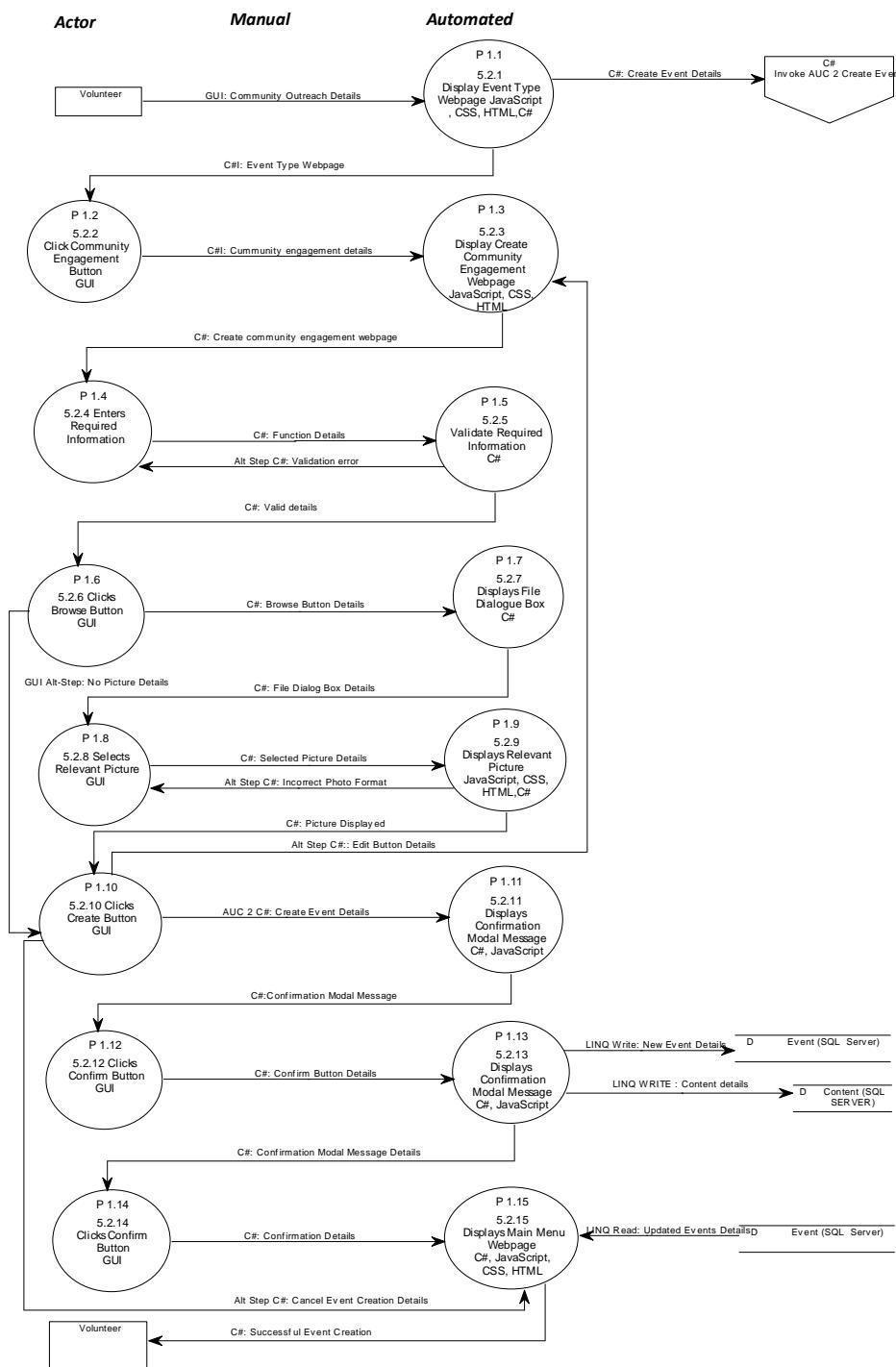


Figure 155: 5.2 Primitive Level Data Flow Diagram

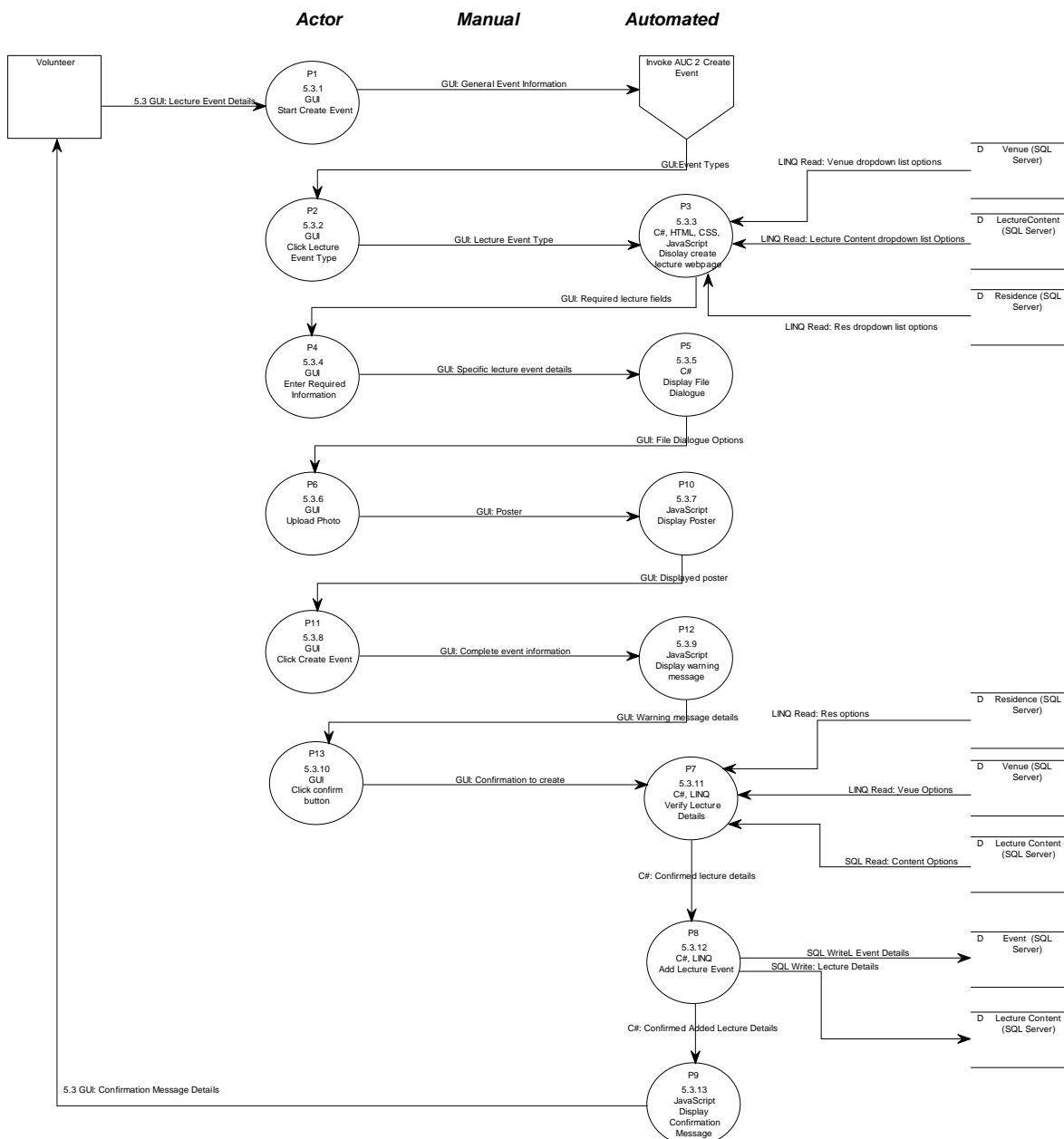


Figure 156- 5.3 Primitive Level Data Flow Diagram

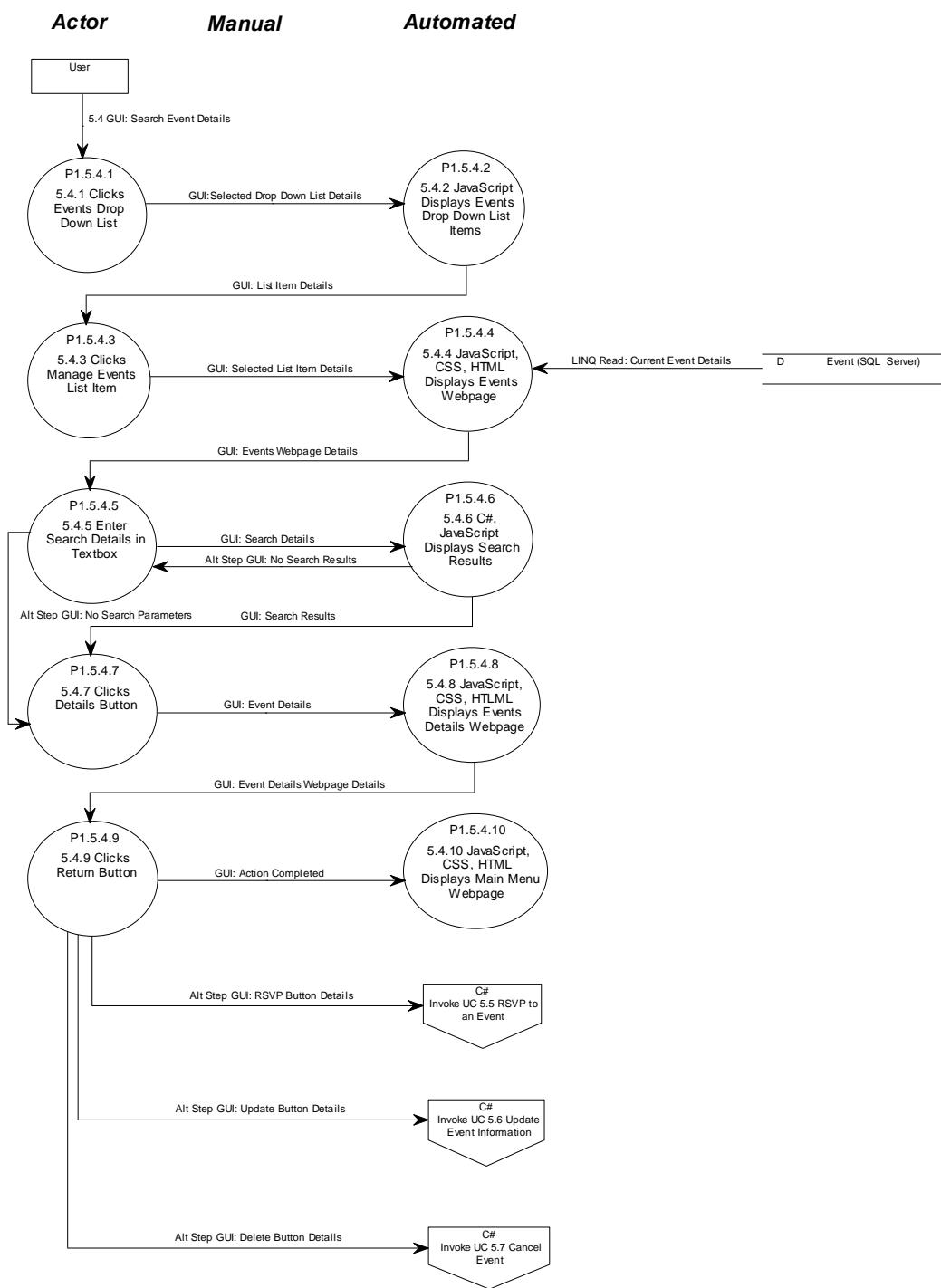


Figure 157: 5.4 Primitive Level Data Flow Diagram

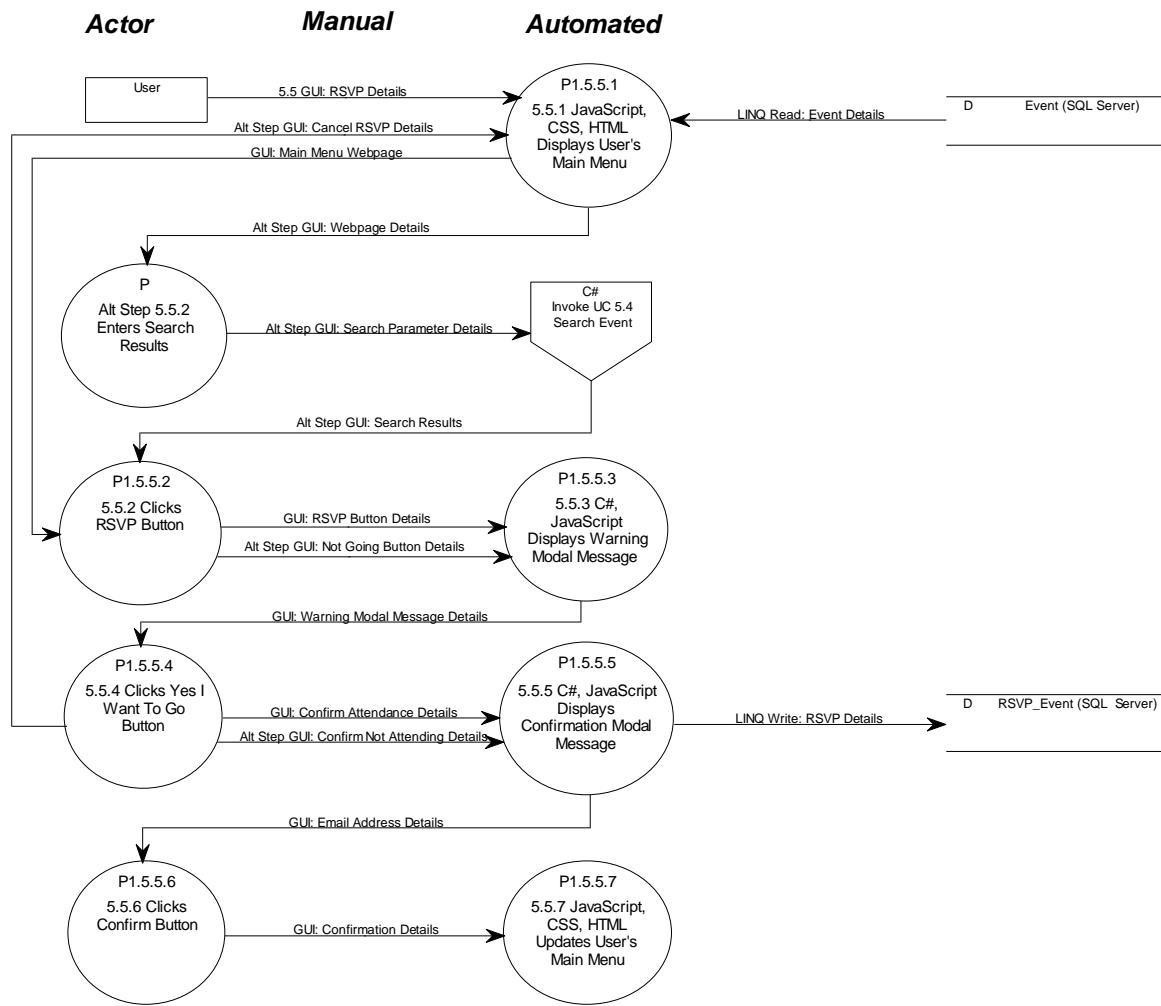


Figure 158: 5.5 Primitive Level Data Flow Diagram

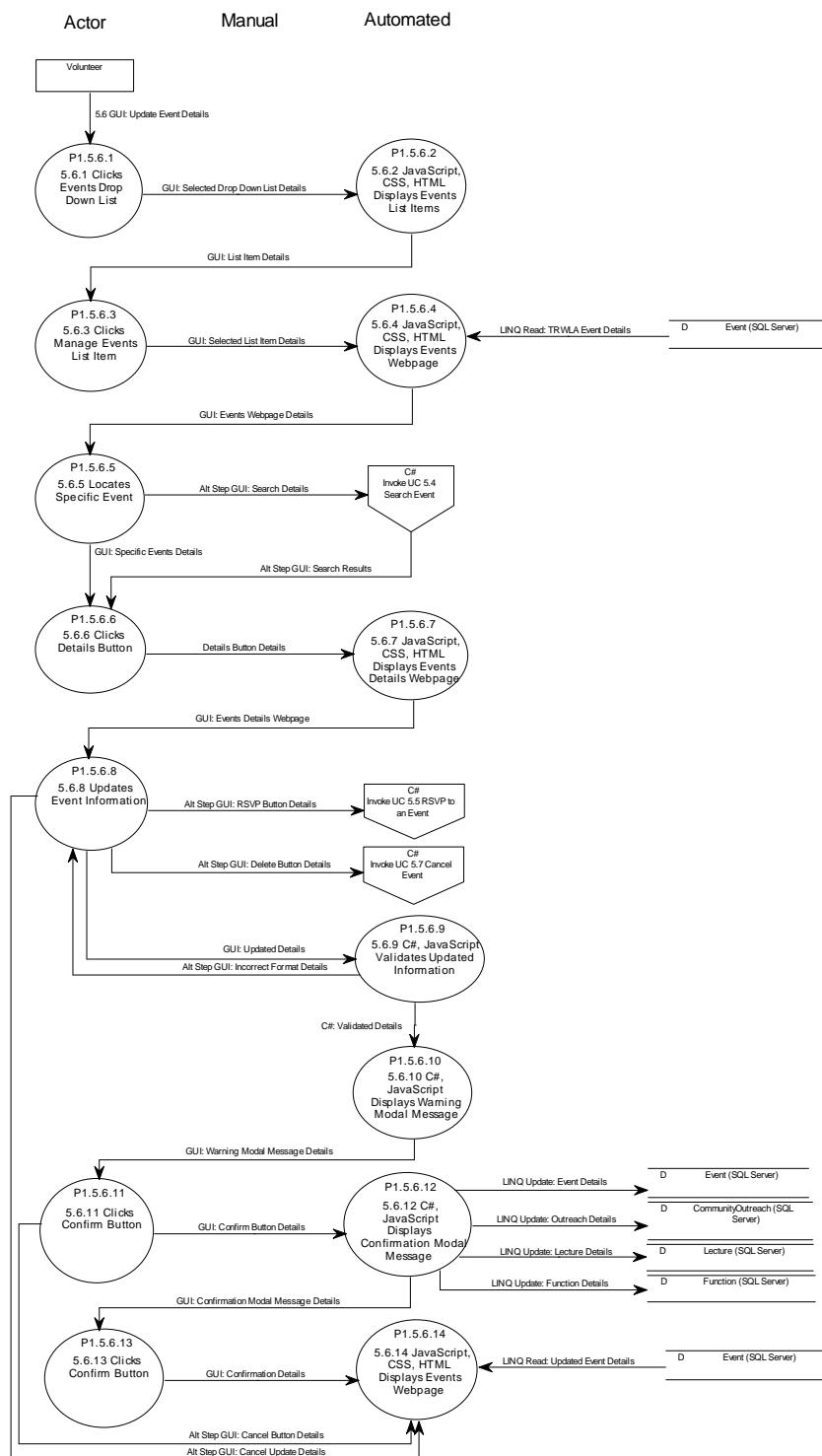


Figure 159: 5.6 Primitive Level Data Flow Diagram

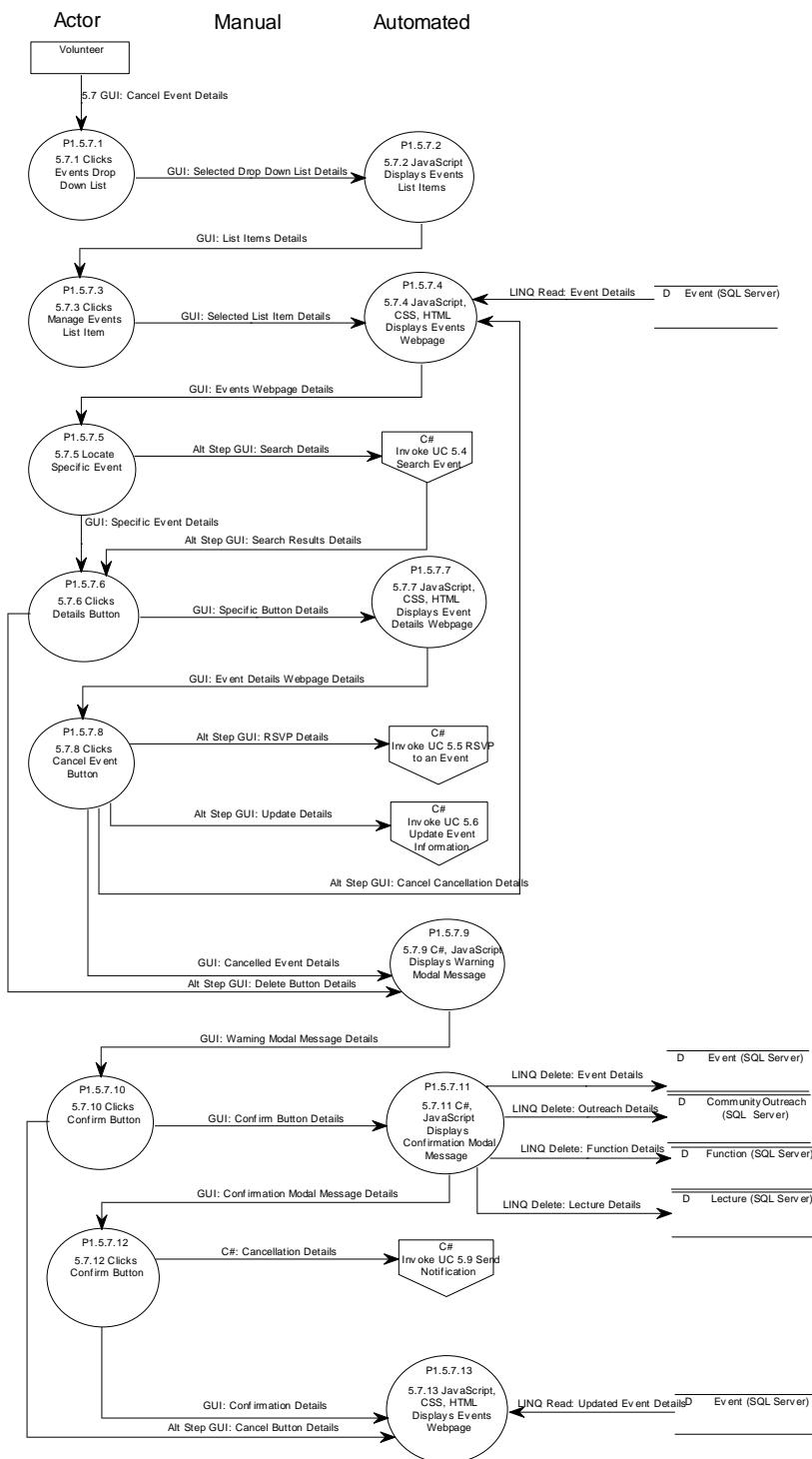


Figure 160: 5.7 Primitive Level Data Flow Diagram

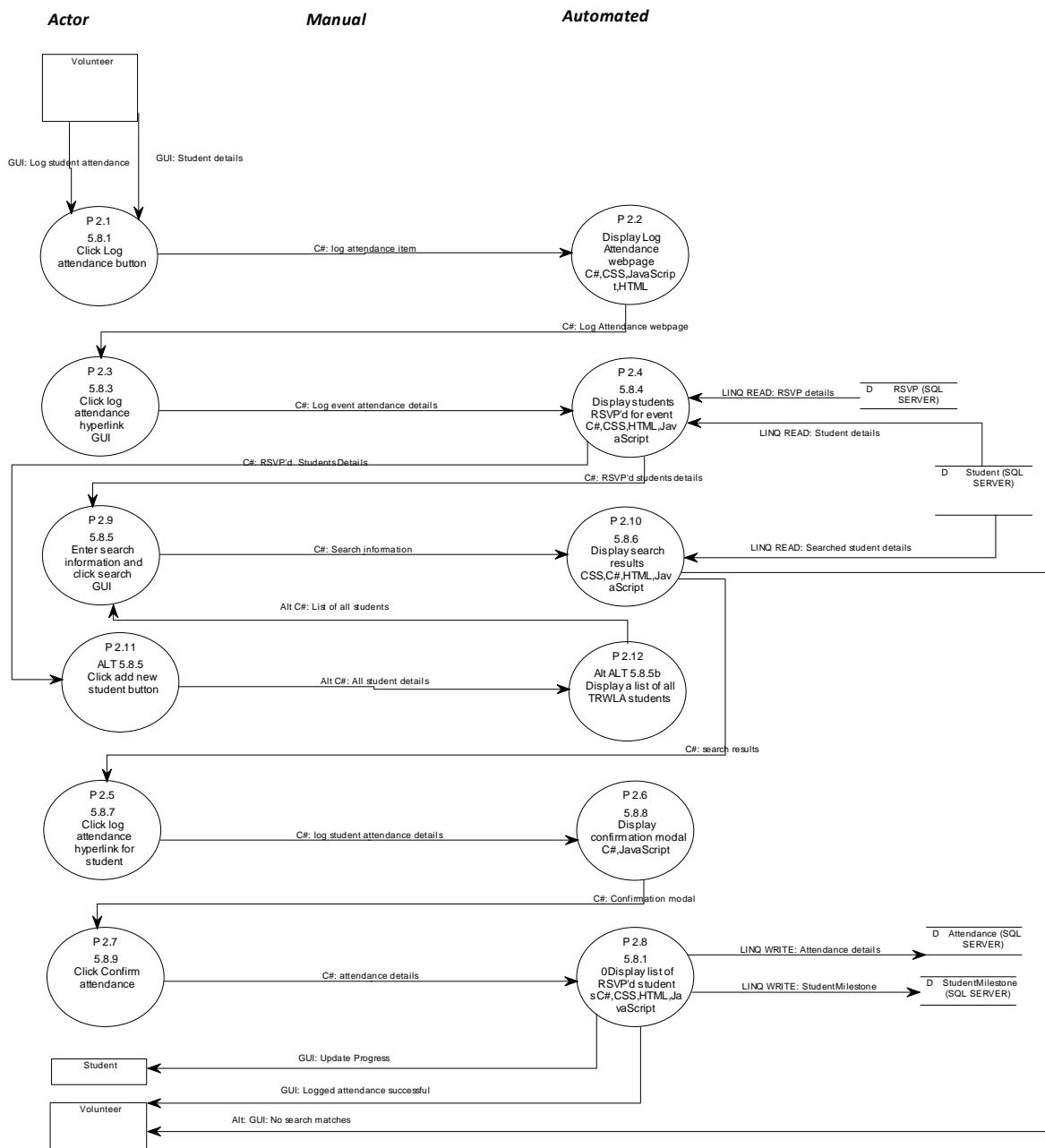


Figure 161: 5.8 Primitive Level Data Flow Diagram

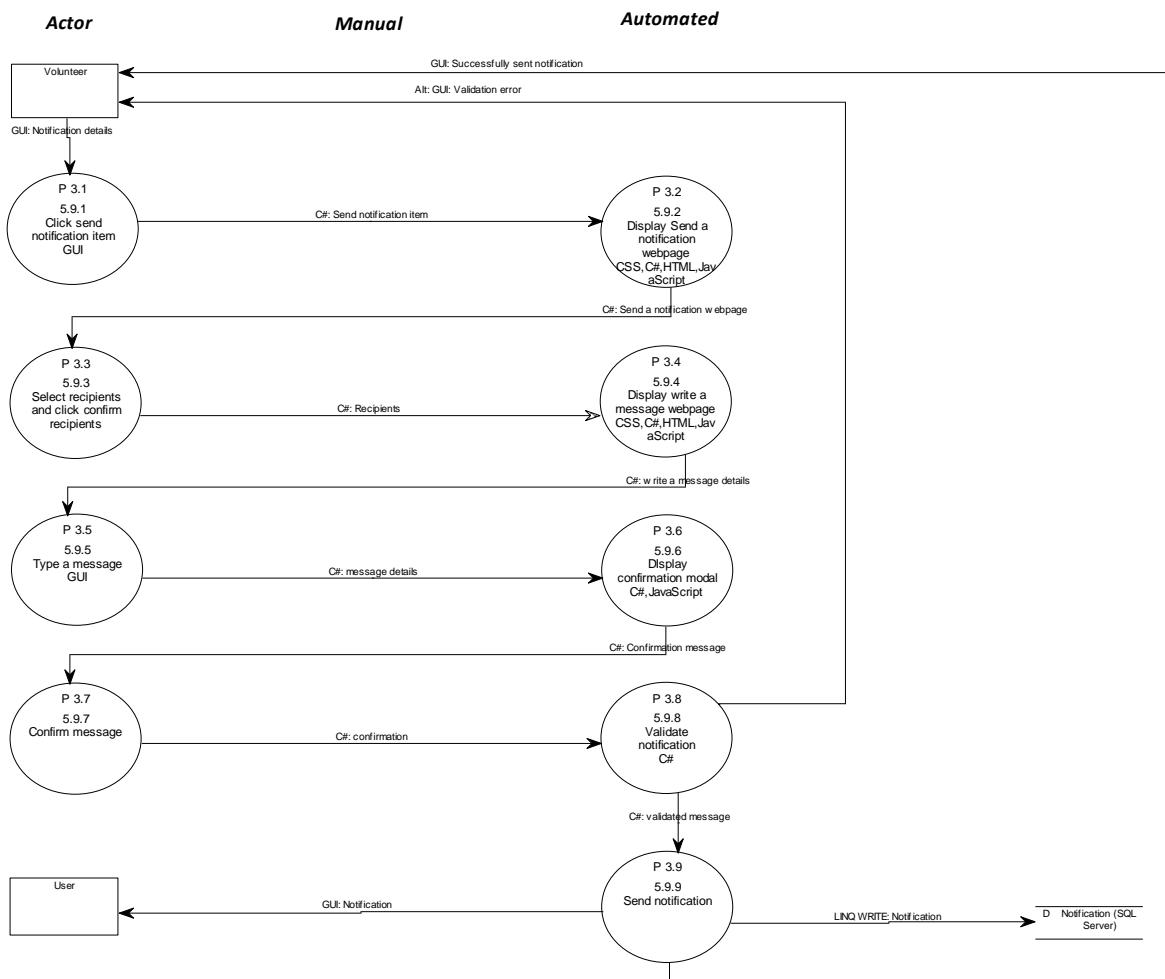


Figure 162: 5.9 Primitive Level Data Flow Diagram

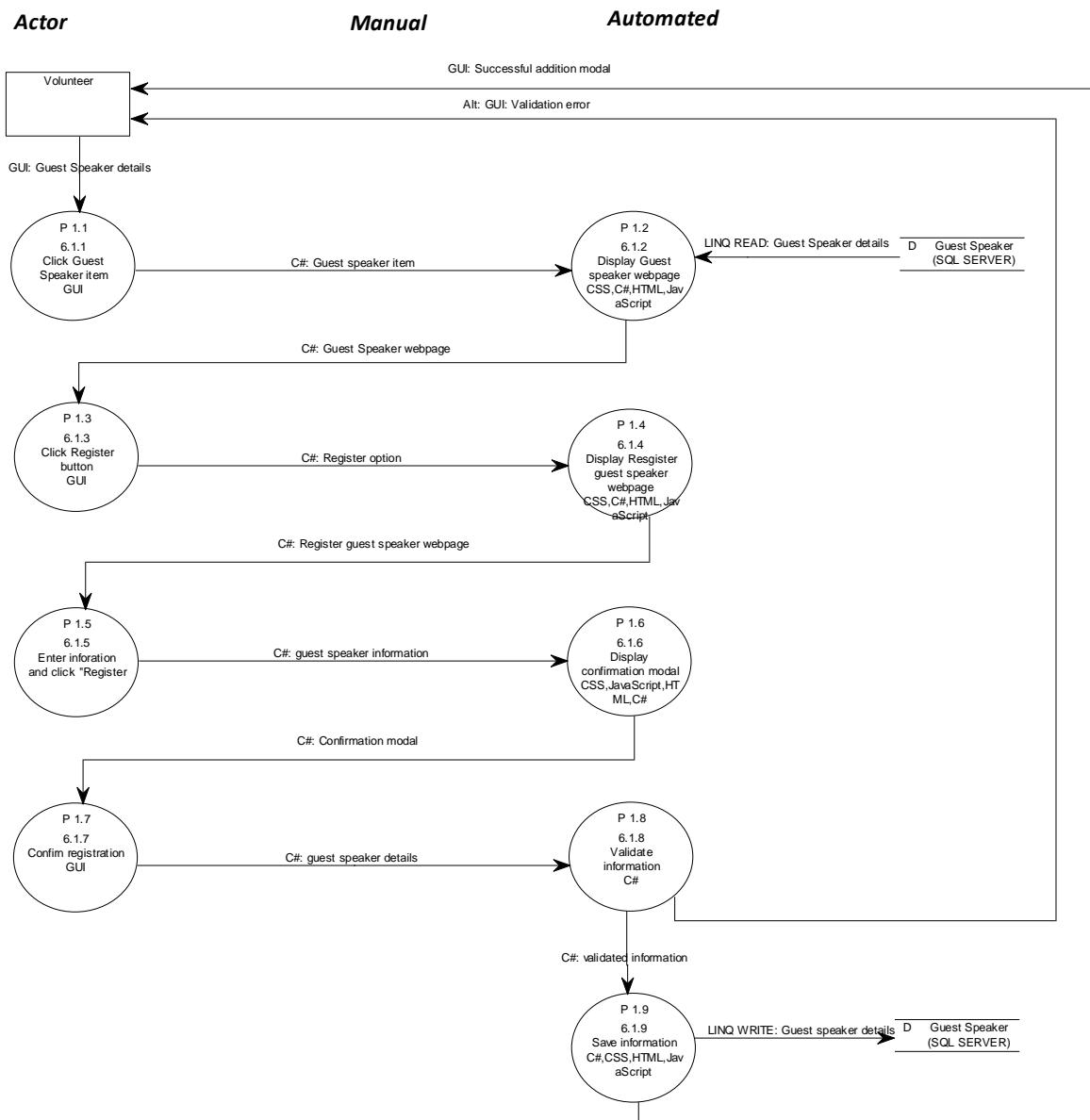


Figure 163: 6.1 Primitive Level Data Flow Diagram

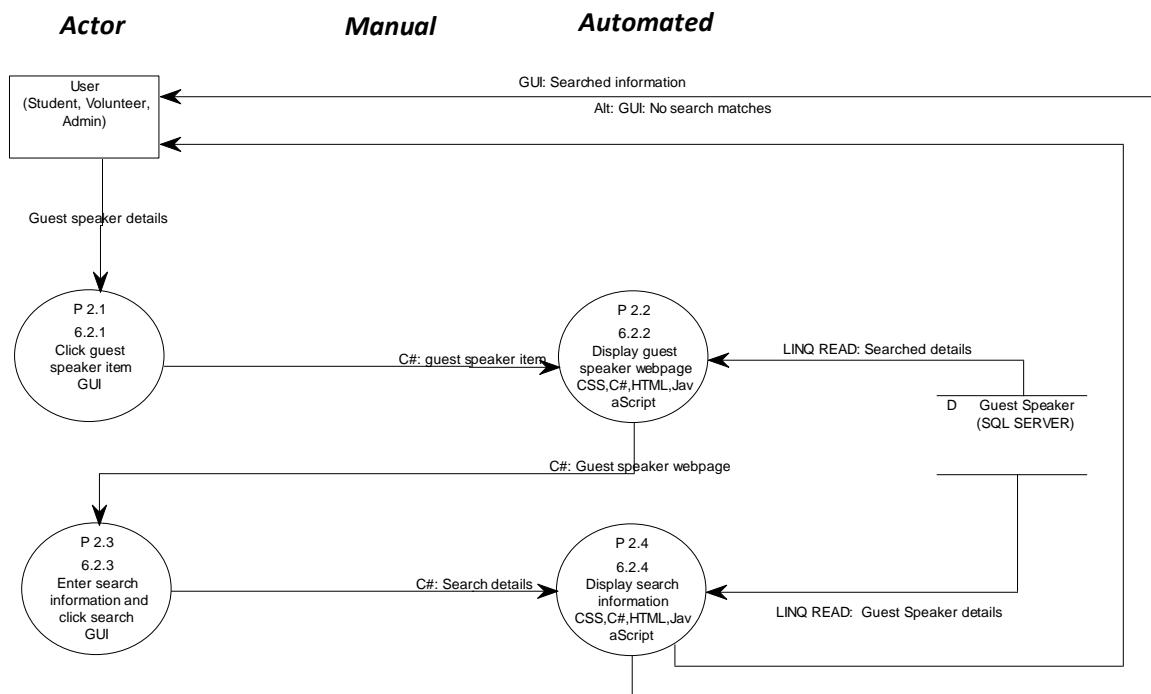


Figure 164: 6.2 Primitive Level Data Flow Diagram

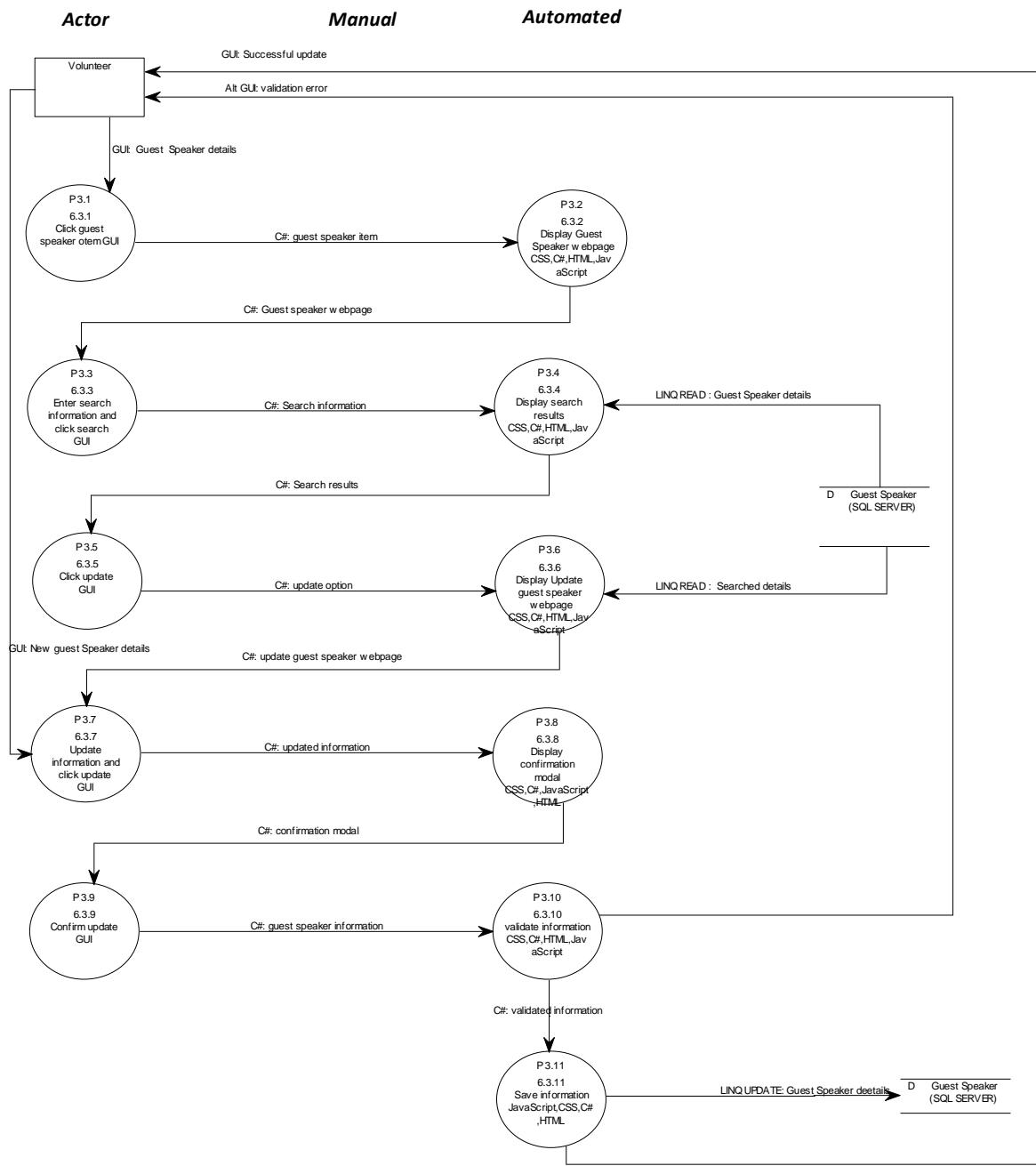


Figure 165: 6.3 Primitive Level Data Flow Diagram

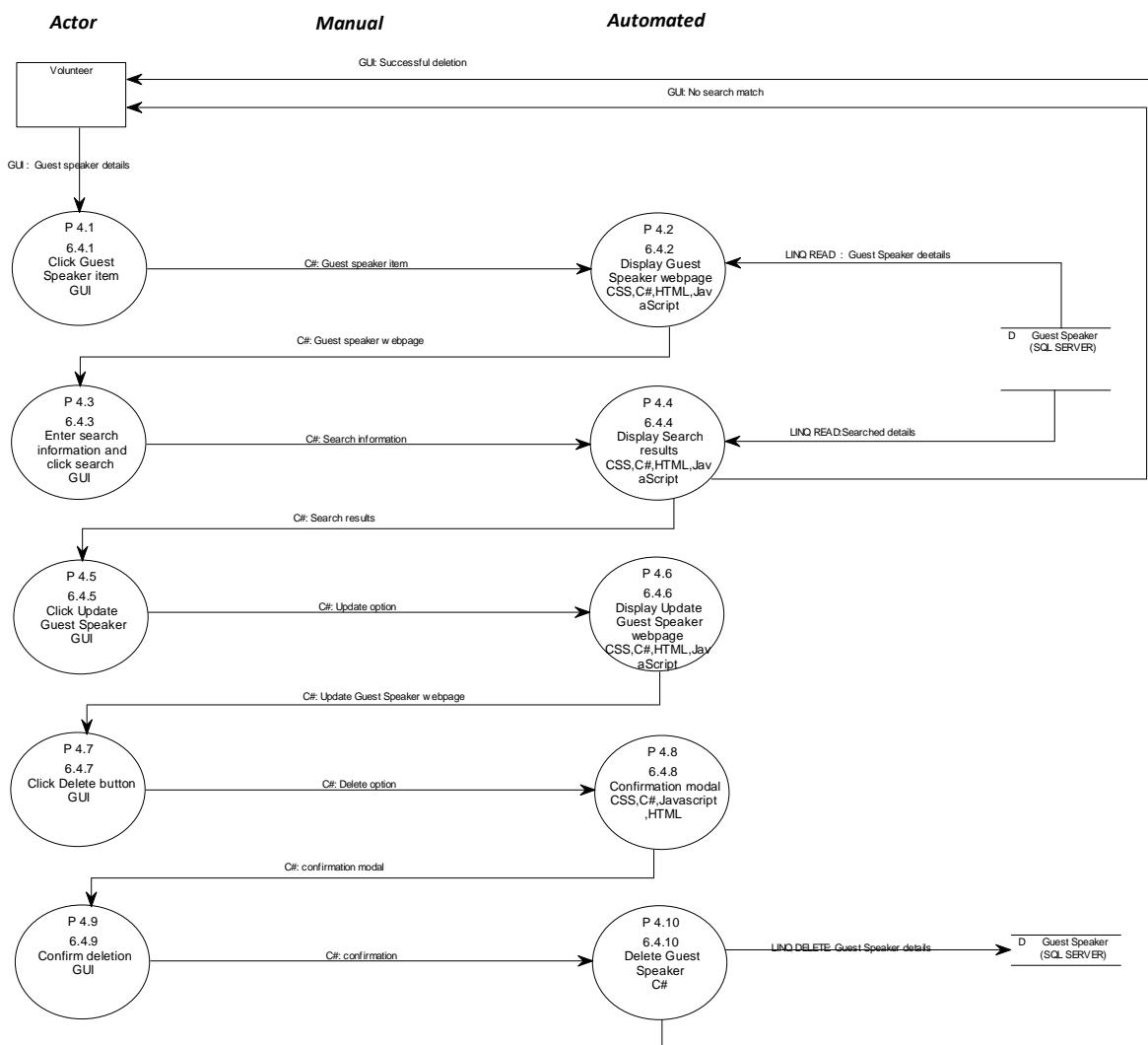


Figure 166: 6.4 Primitive Level Data Flow Diagram

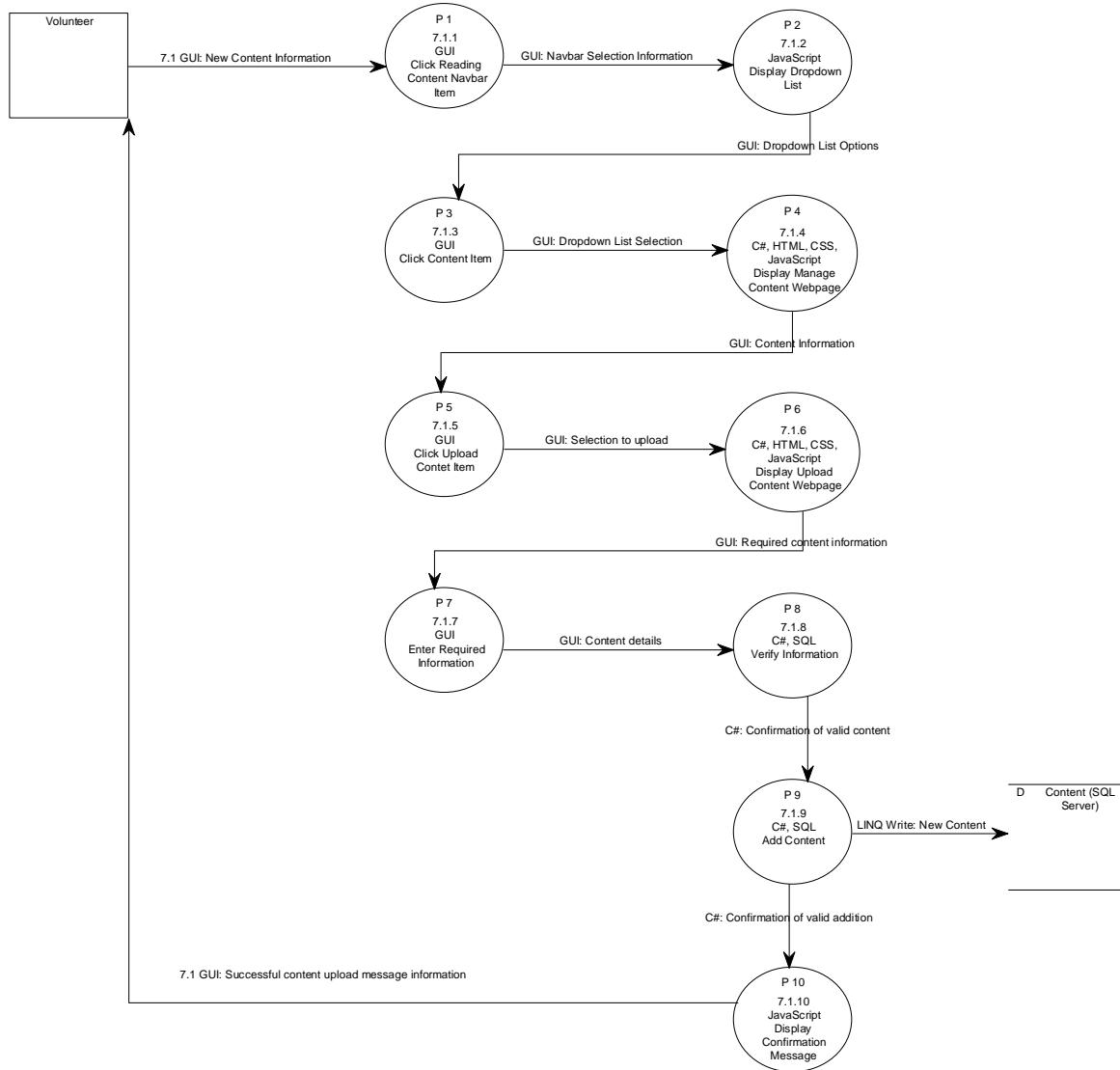


Figure 167: 7.1 Primitive Level Data Flow Diagram

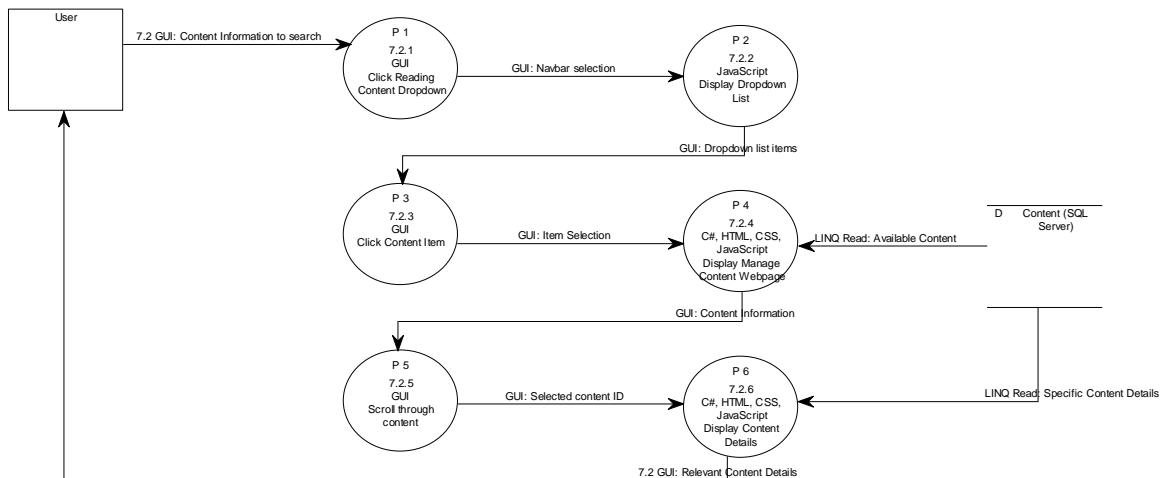


Figure 168: 7.2 Primitive Level Data Flow Diagram

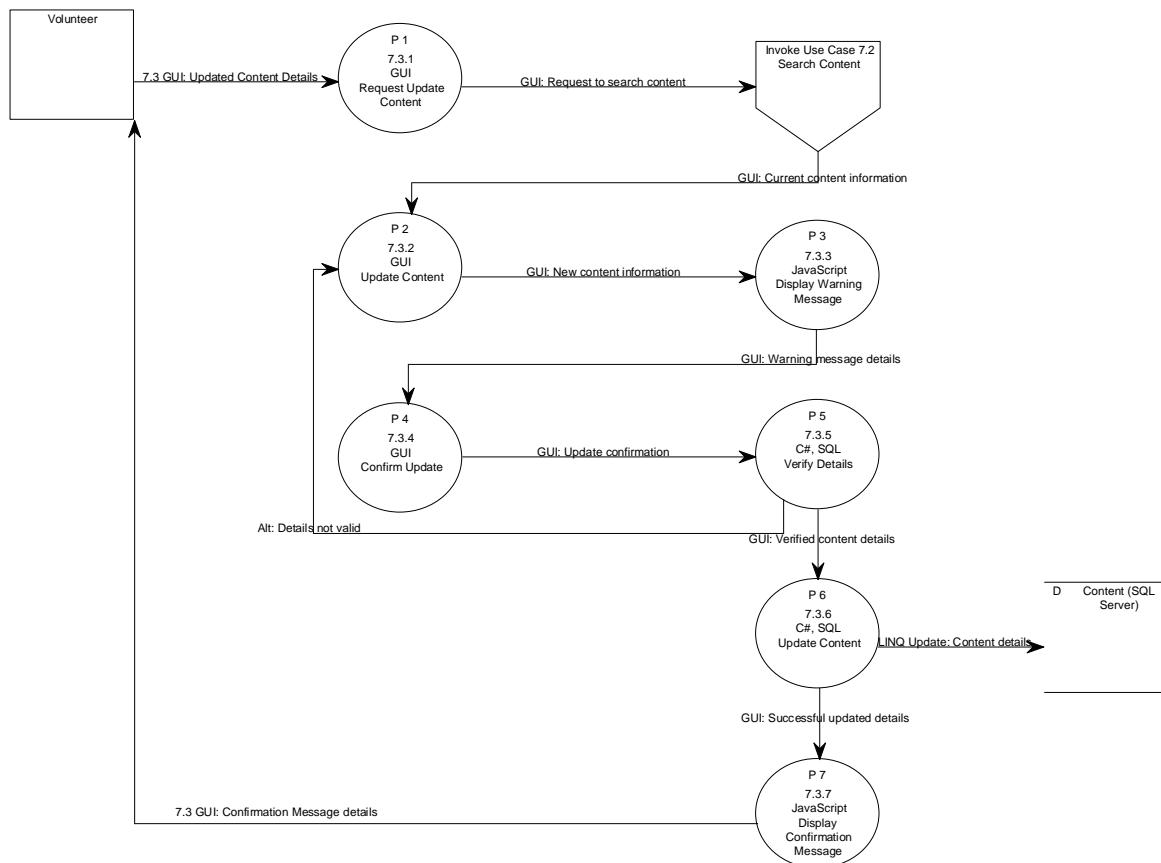


Figure 169: 7.3 Primitive Level Data Flow Diagram

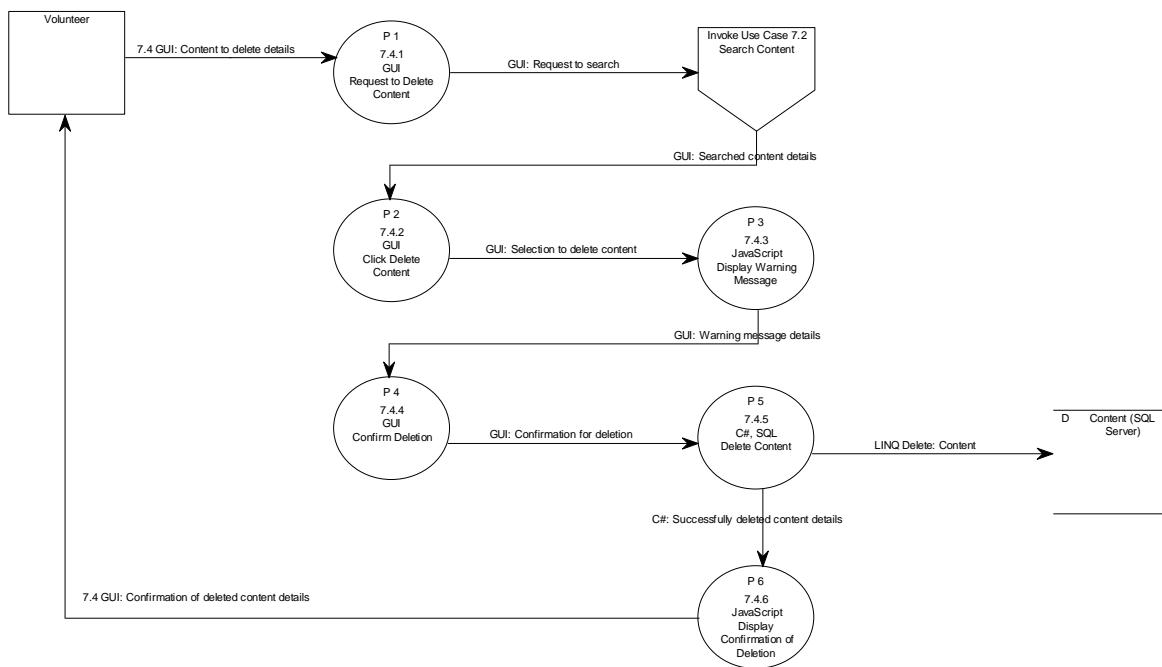


Figure 170: 7.4 Primitive Level Data Flow Diagram

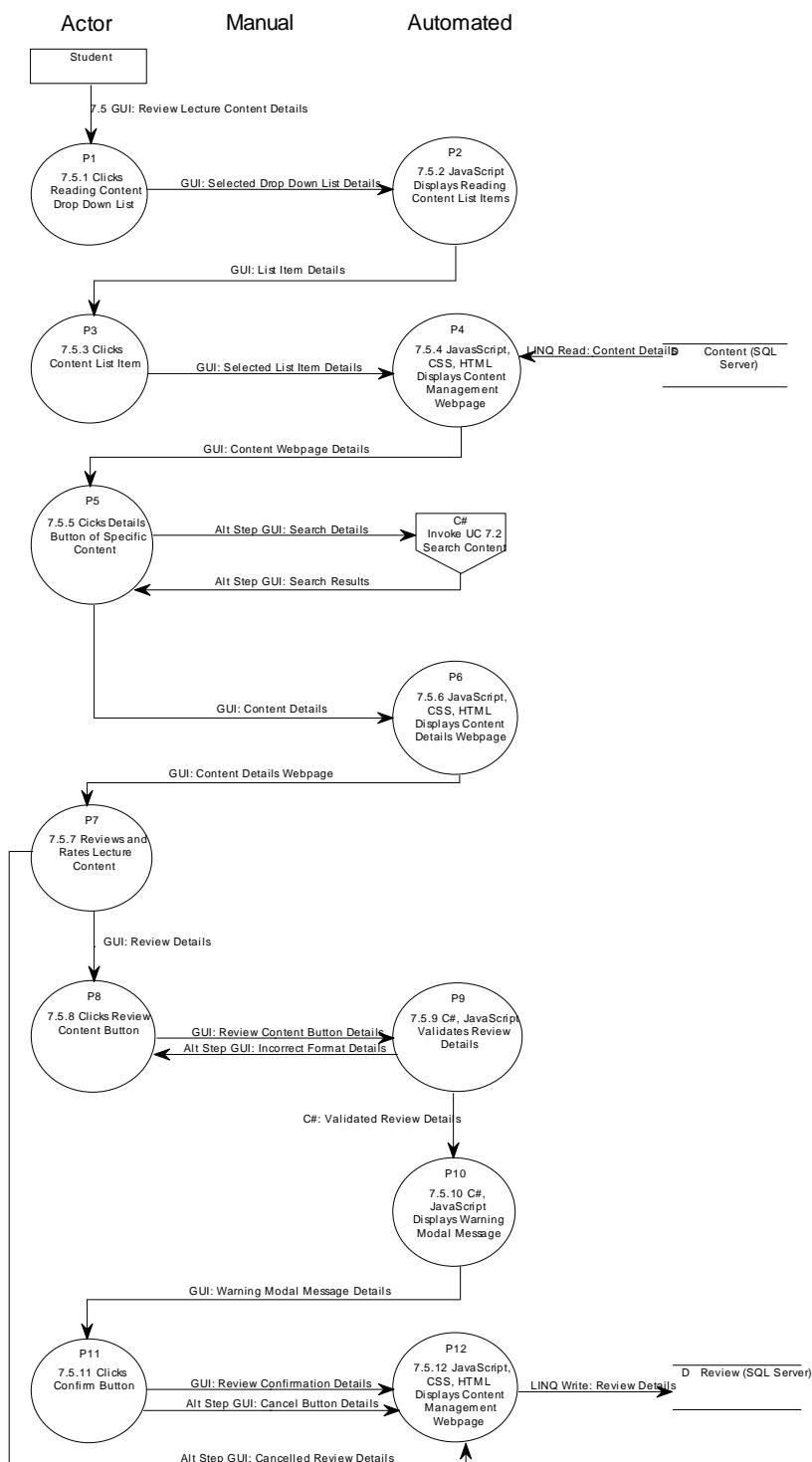


Figure 171- 7.5 Primitive Level Data Flow Diagram

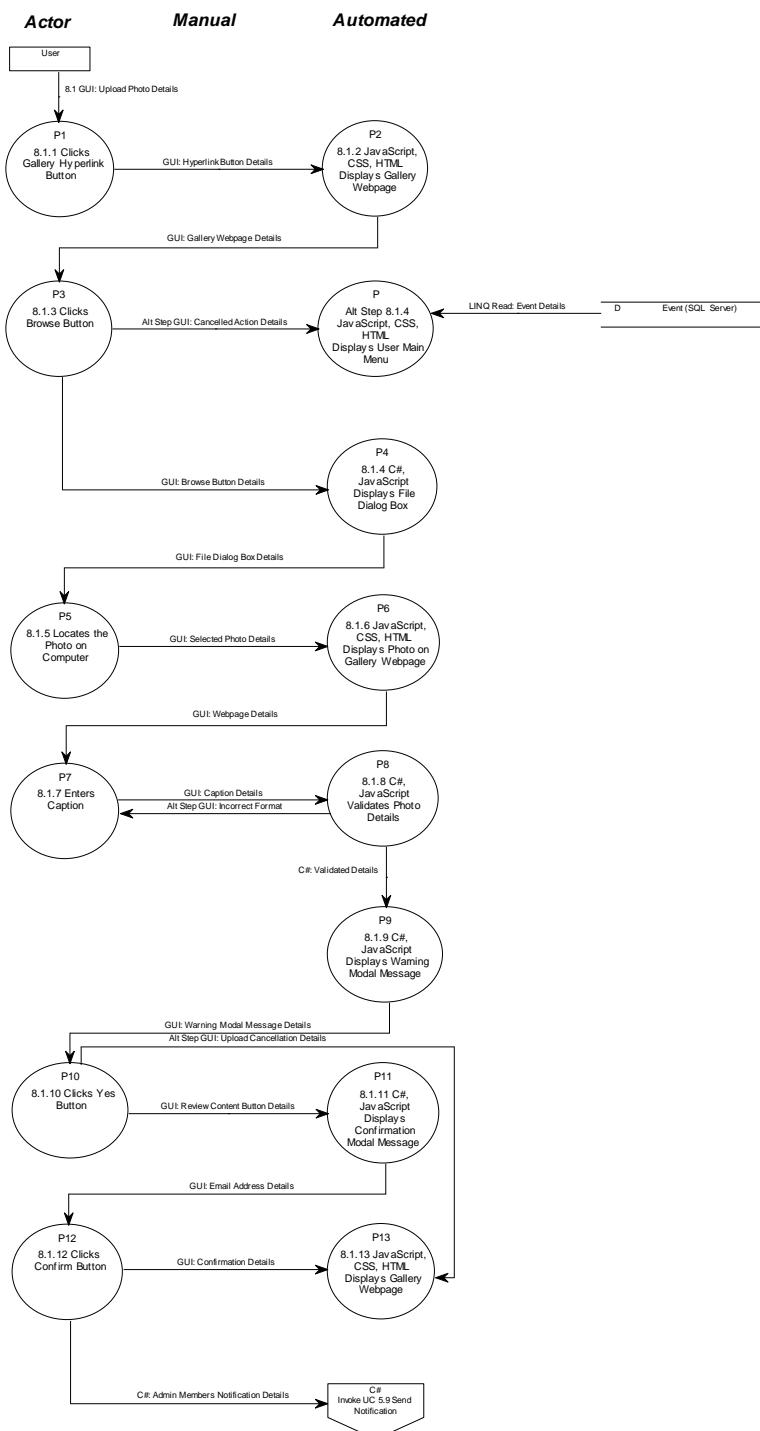


Figure 172: 8.1 Primitive Level Data Flow Diagram

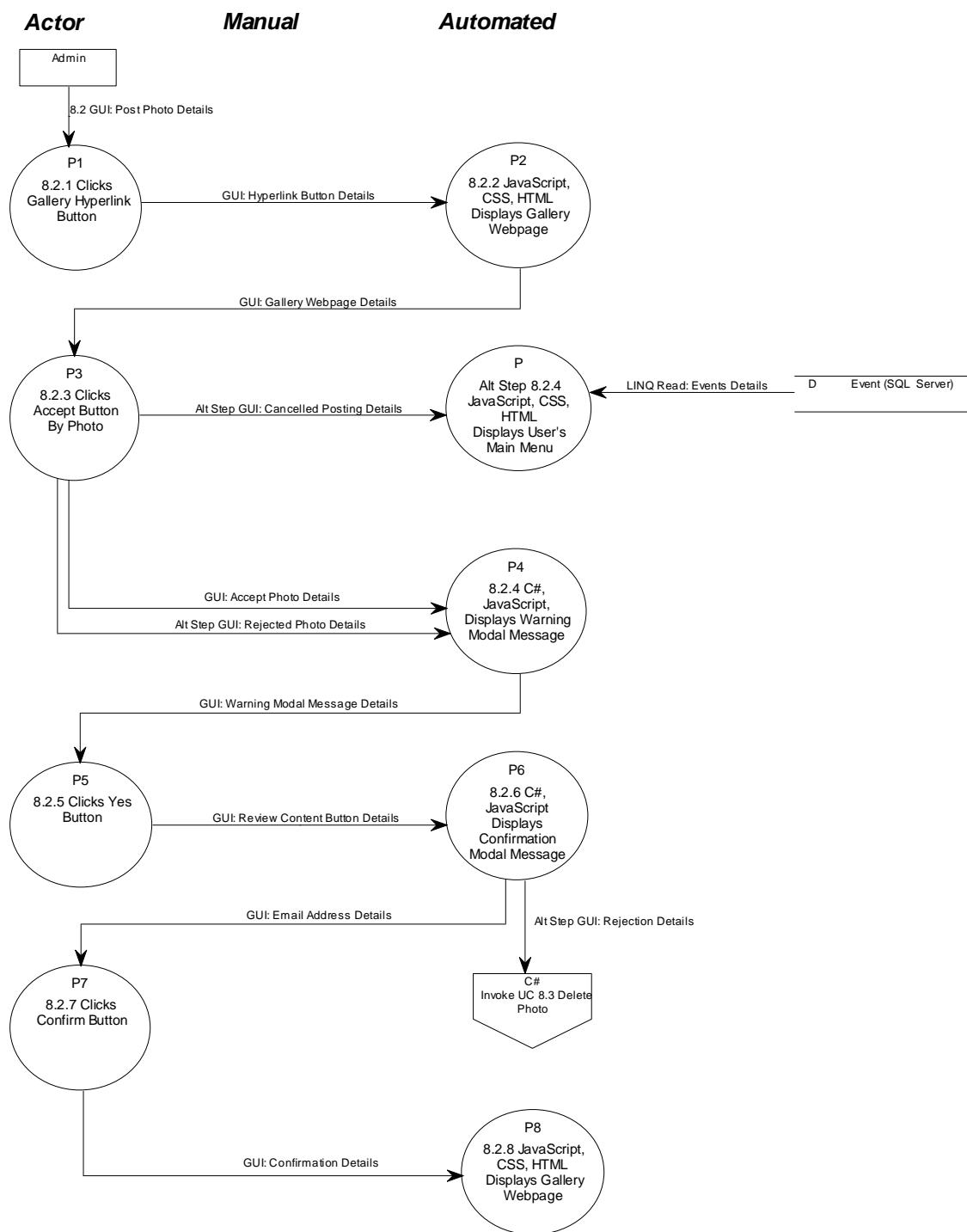


Figure 173: 8.2 Primitive Level Data Flow Diagram

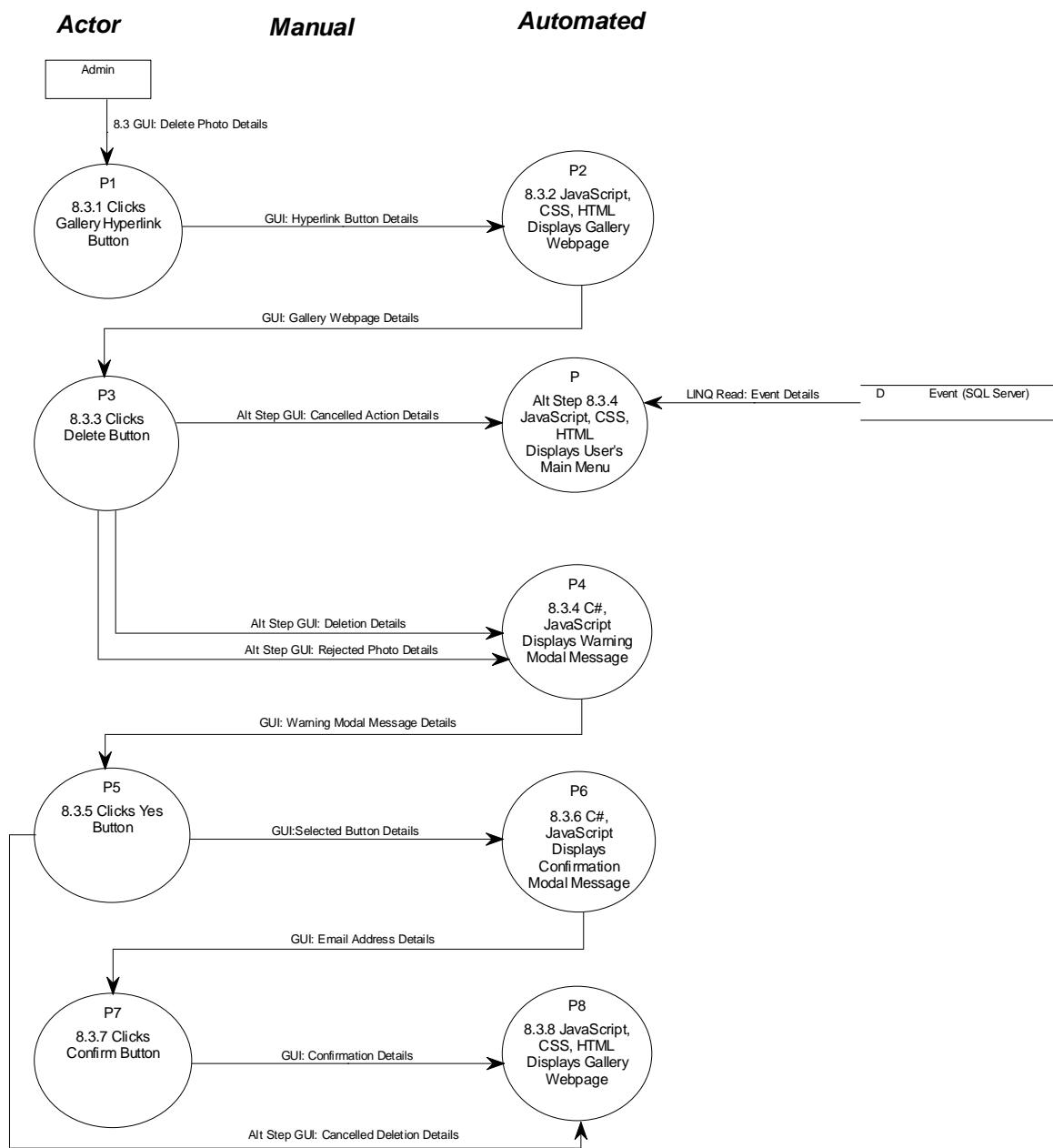


Figure 174: 8.3 Primitive Level Data Flow Diagram

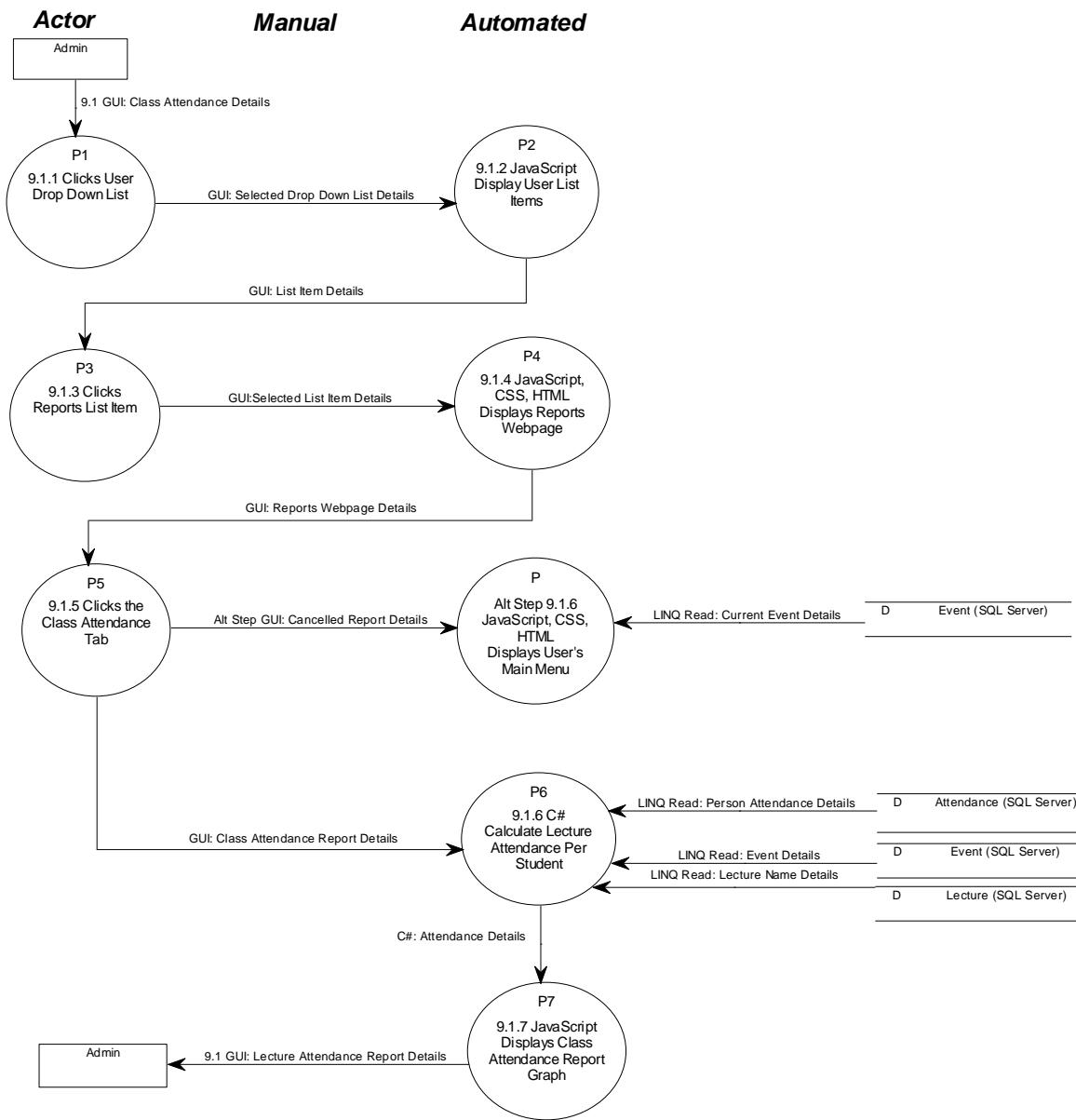


Figure 175: 9.1 Primitive Level Data Flow Diagram

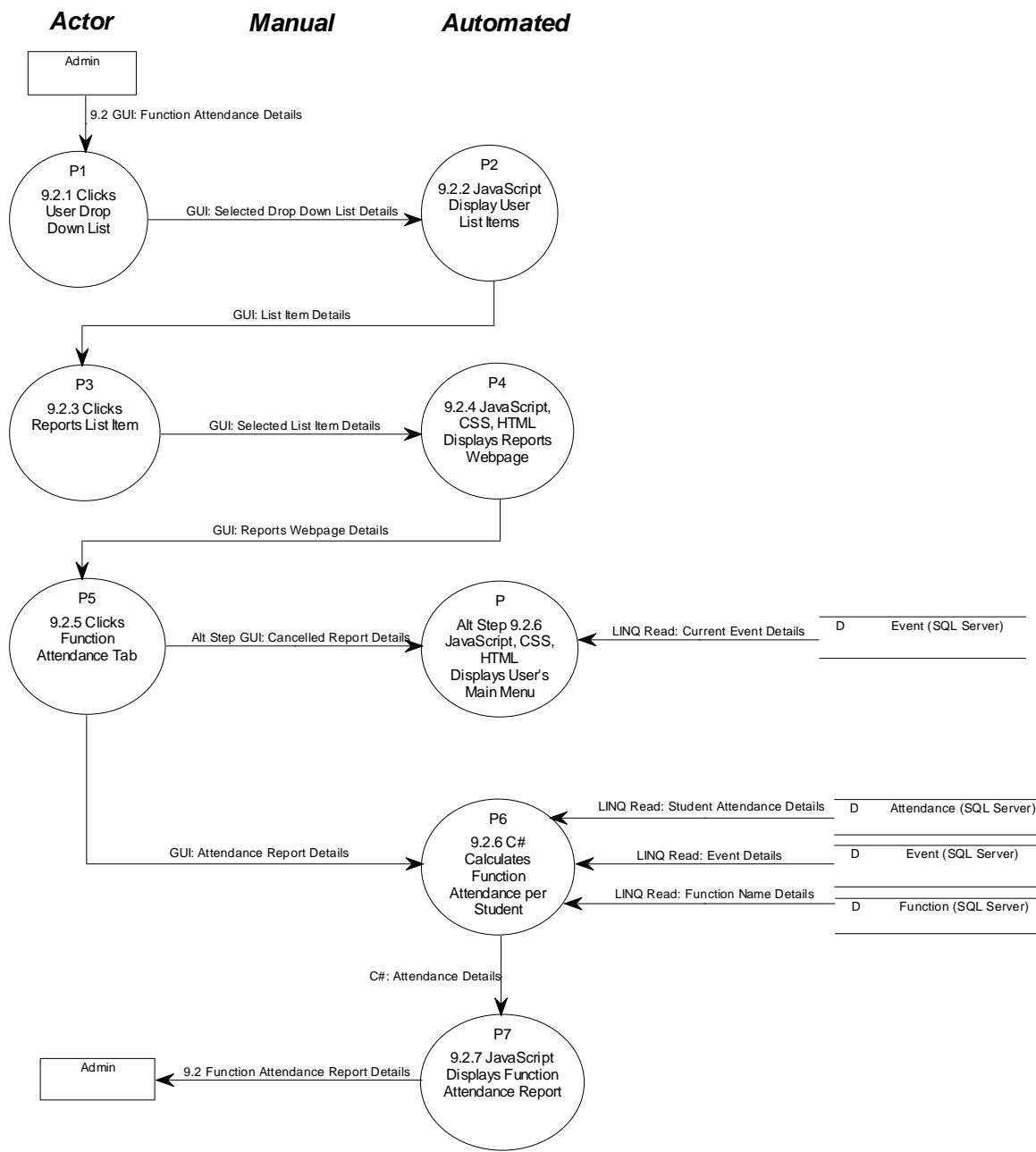


Figure 176: 9.2 Primitive Level Data Flow Diagram

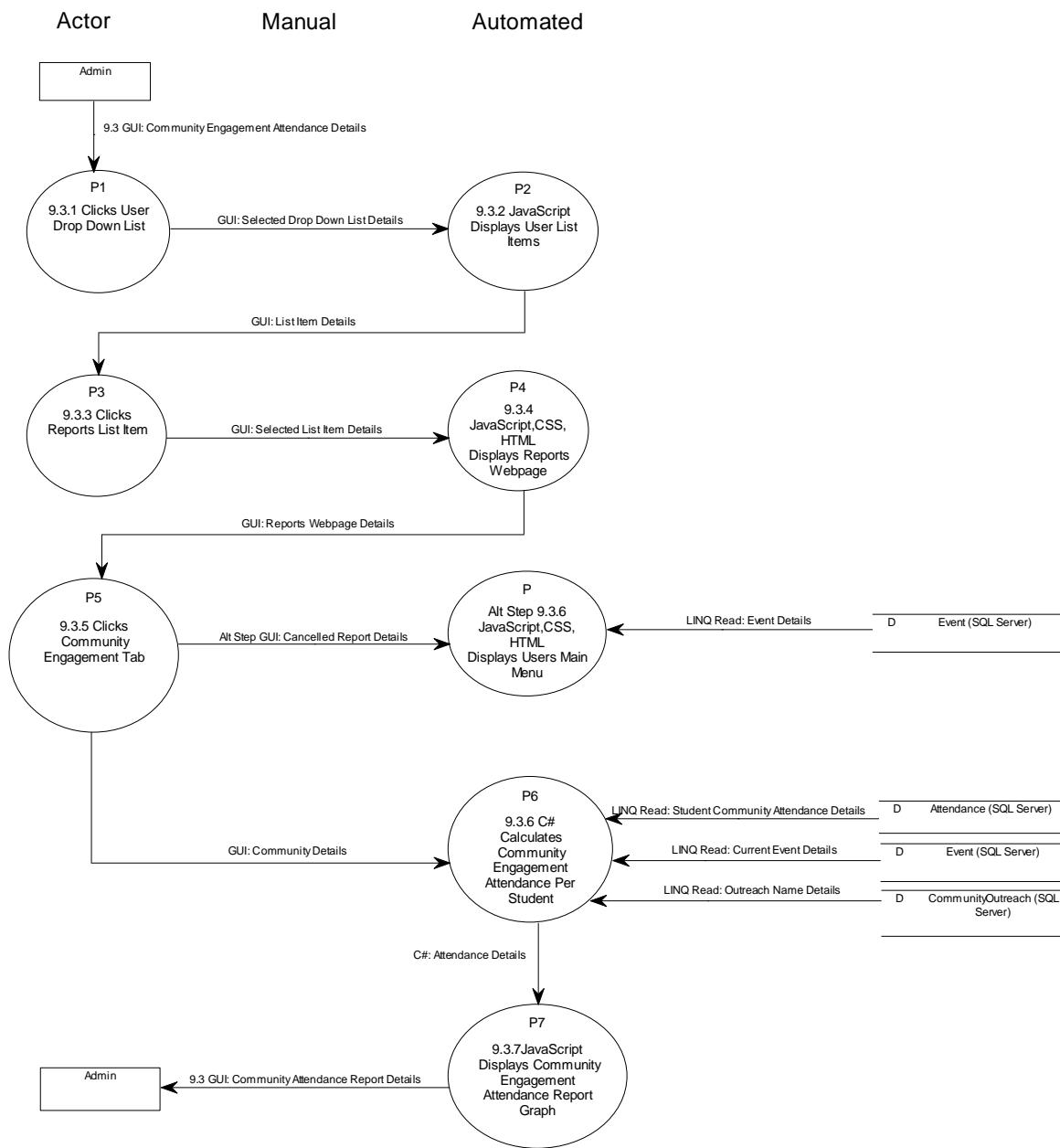


Figure 177: 9.3 Primitive Level Data Flow Diagram

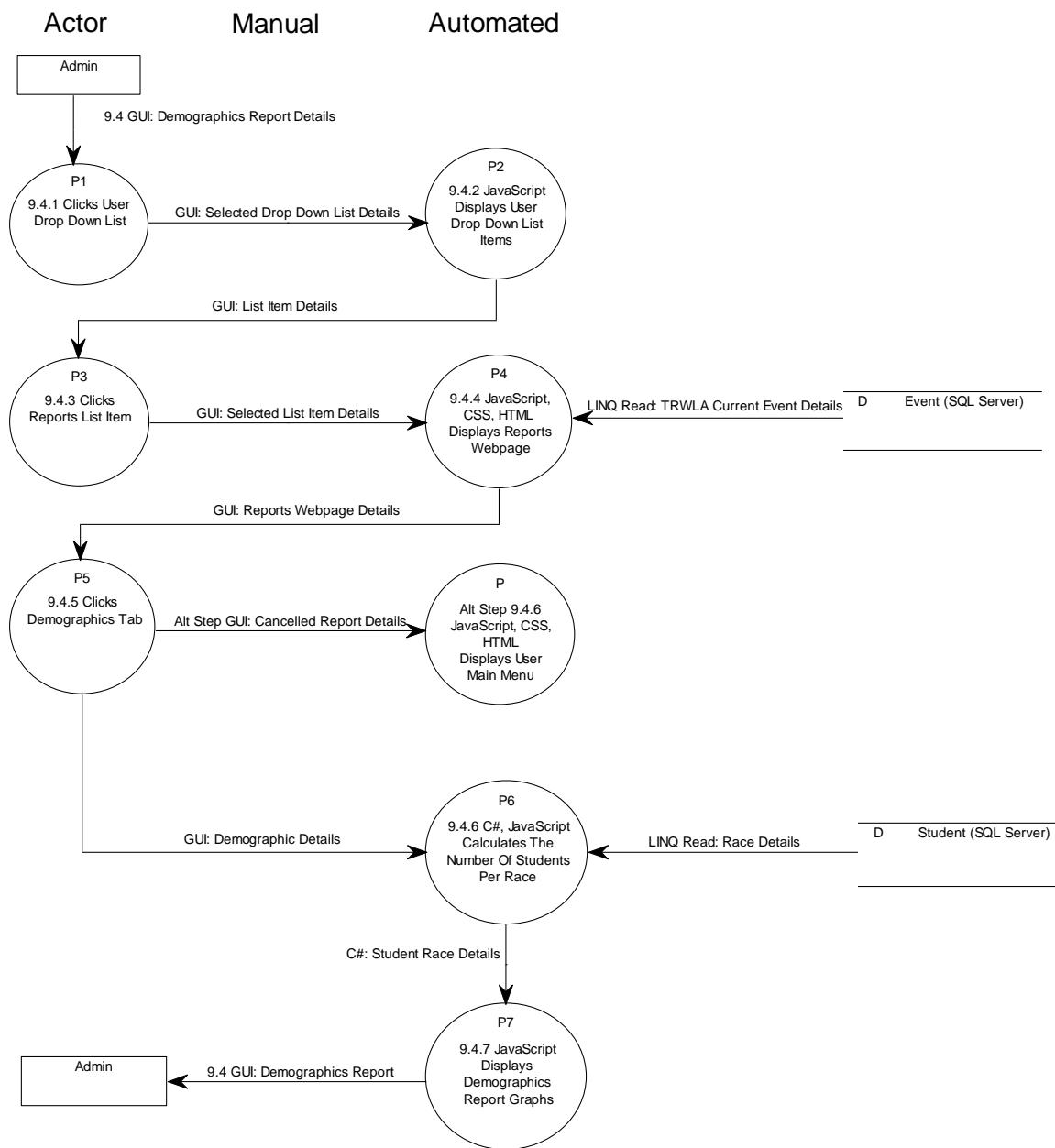


Figure 178: 9.4 Primitive Level Data Flow Diagram

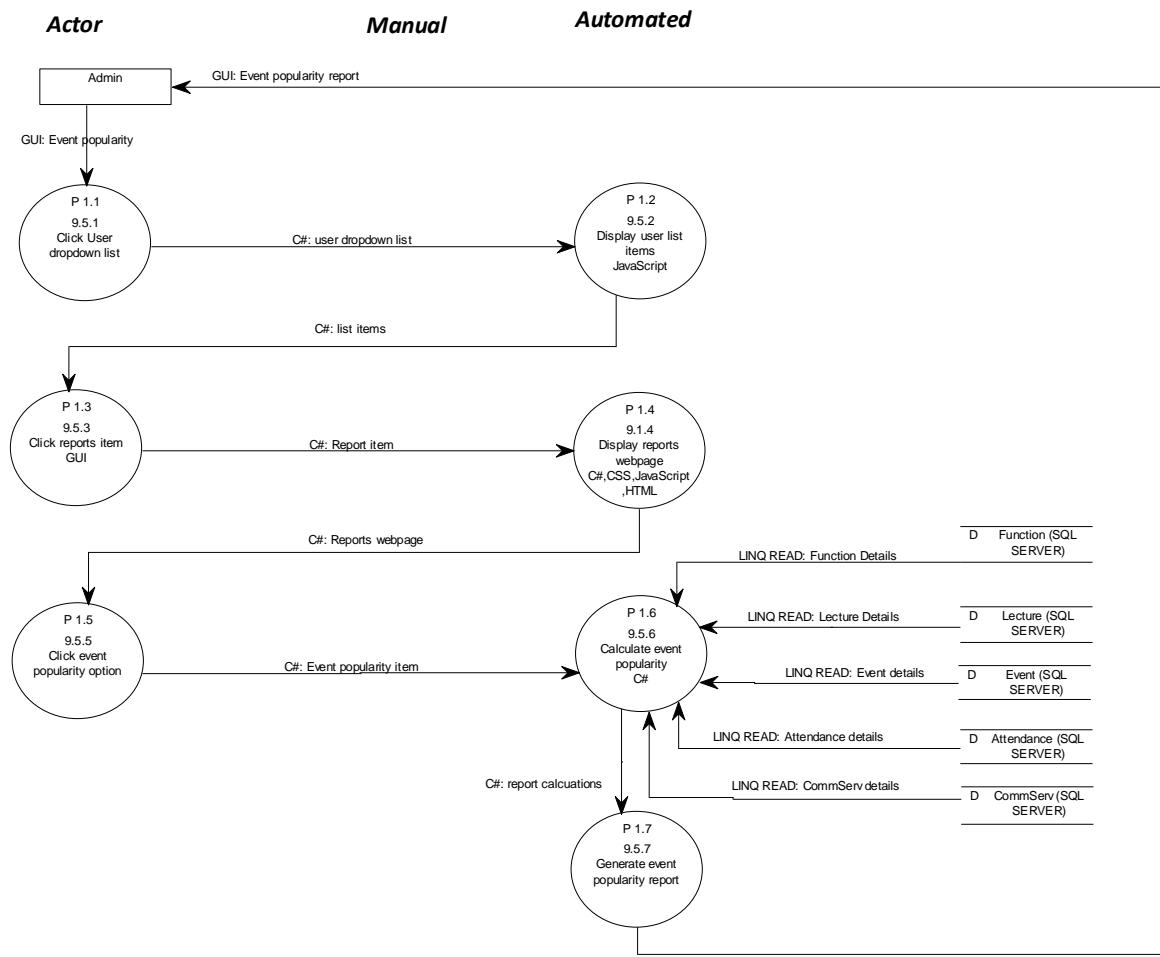


Figure 179: 9.5 Primitive Level Data Flow Diagram

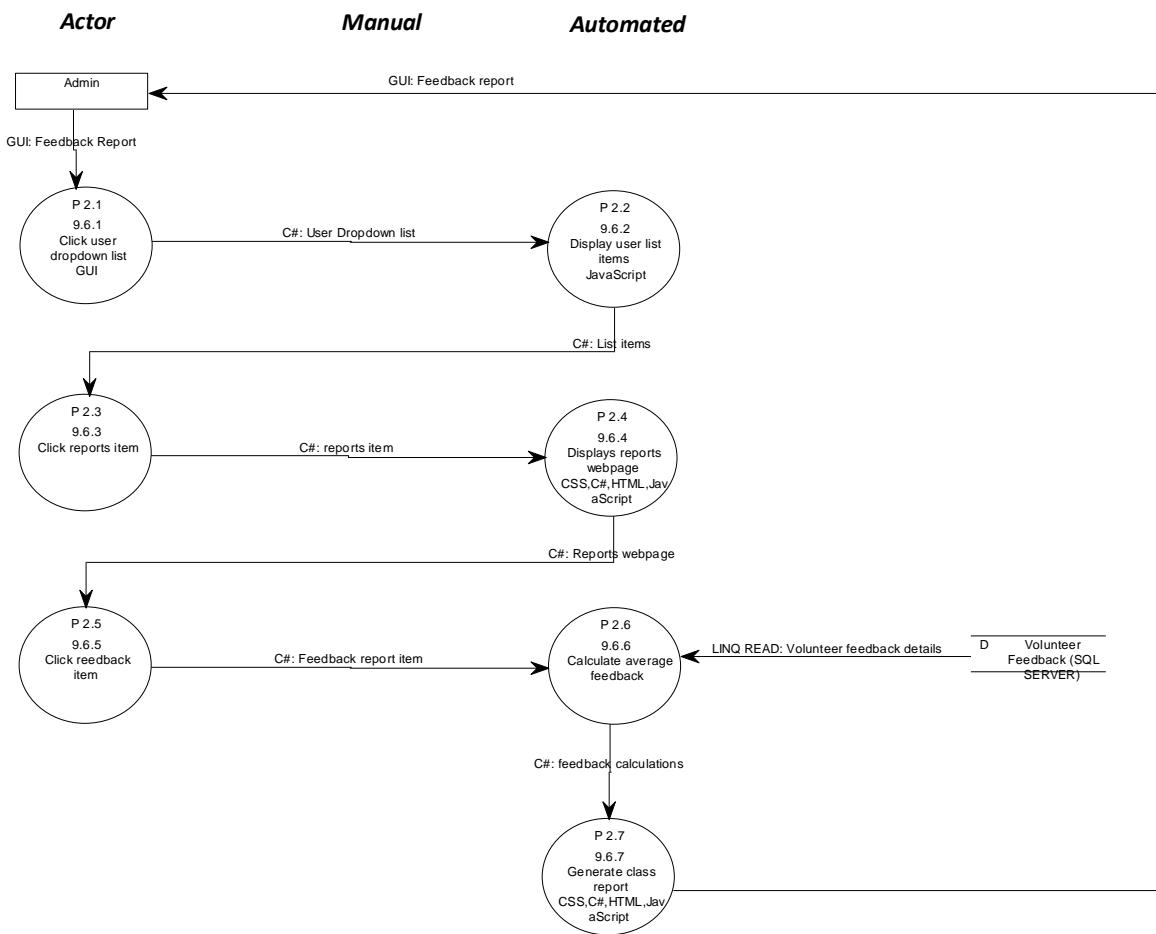


Figure 180: 9.6 Primitive Level Data Flow Diagram

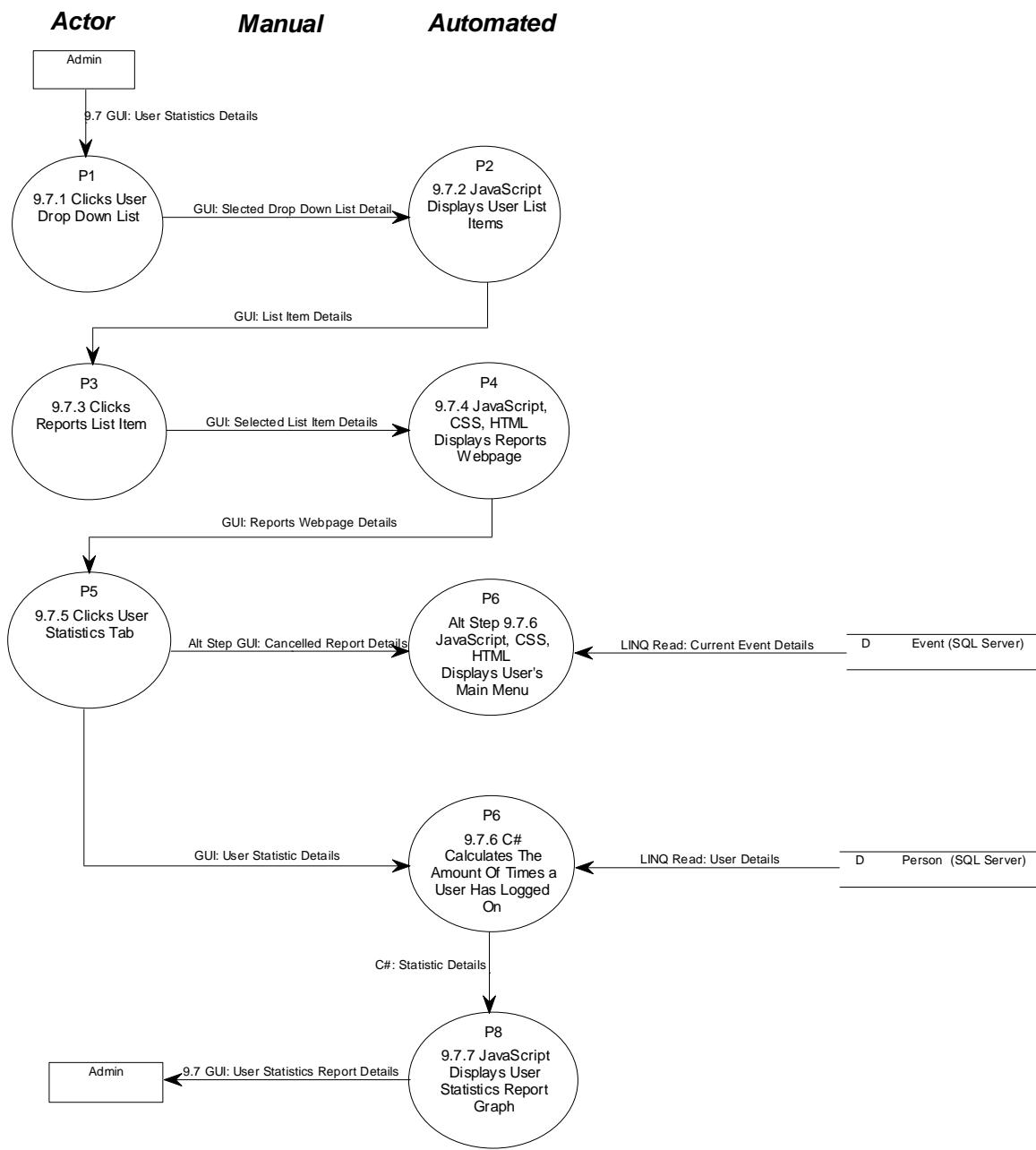


Figure 181: 9.7 Primitive Level Data Flow Diagram

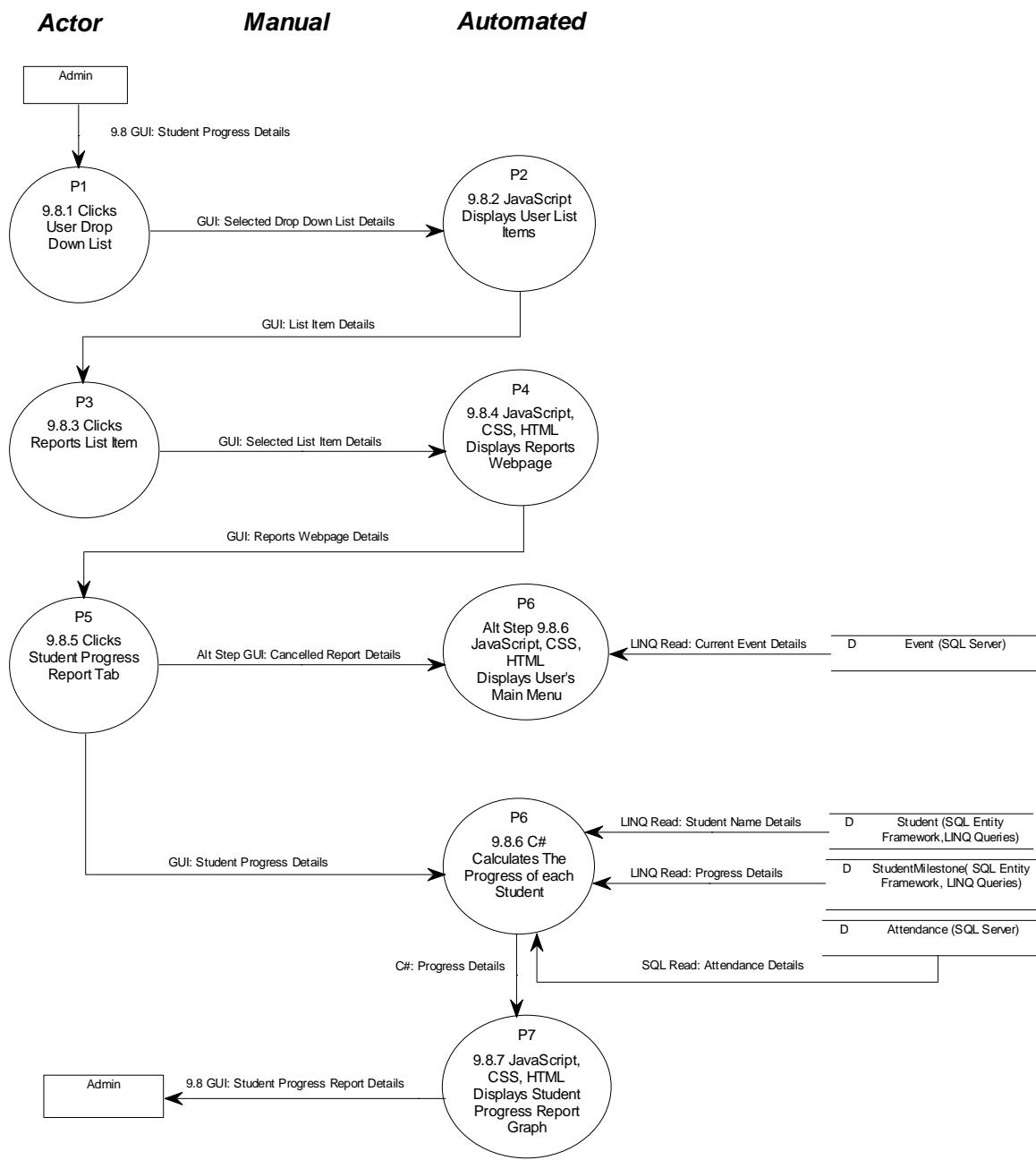


Figure 182: 9.8 Primitive Level Data Flow Diagram

3.5 Physical Description of Each Process (Pseudo Code)

3.5.1 User Subsystem Pseudo Code

Primitive Level Diagram	AUC 1 Register User
Process	AUC 1.2
Pseudo Code	IF (Webpage hyperlink is clicked) THEN SHOW Static Webpage Home ENDIF

Primitive Level Diagram	AUC 1 Register User
Process	AUC 1.4
Pseudo Code	IF (Login Navbar Item is clicked) THEN SHOW Login Webpage ENDIF

Primitive Level Diagram	AUC 1 Register User
Process	AUC 1.6
Pseudo Code	IF (New User Hyperlink is clicked) THEN SHOW Register User Webpage ENDIF

Primitive Level Diagram	AUC 1 Register User
Process	AUC 1.8
Pseudo Code	<p>IF (“Email Address” is greater than 7 characters AND less than 255 characters AND is a full validated email address AND does not exist in person table)</p> <p style="padding-left: 40px;">THEN “Email Address” IS valid</p> <p>ELSE “Email Address” IS NOT valid</p> <p>IF (Password is greater than 8 characters AND less than 35 characters AND has at least 1 integer)</p> <p style="padding-left: 40px;">THEN Password IS valid</p> <p>ELSE Password IS NOT valid</p> <p>IF (“Confirm Password” is equal to Password)</p> <p style="padding-left: 40px;">THEN “Confirm Password” IS valid</p> <p>ELSE “Confirm Password” IS NOT valid</p>

Primitive Level Diagram	AUC 1 Register User
Process	AUC 1.9
Pseudo Code	<p>IF ("Email Address" IS valid AND Password IS valid AND "Confirm Password" IS valid)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 20px;">Registration equals "Success"</p> <p style="padding-left: 20px;">INSERT INTO TABLE Person in attributes (EmailAddress, Password) the values of "Email Address" and "Password"</p> <p>ELSE Registration equals "Fail", Display error message</p>

Primitive Level Diagram	AUC 1 Register User
Process	AUC 1.10
Pseudo Code	<p>IF Registration equals "Success"</p> <p style="padding-left: 20px;">THEN Display Modal "Registration Successful"</p> <p style="padding-left: 20px;">ENDIF</p>

Primitive Level Diagram	1.1 Check Forgotten Password
Process	1.1.2
Pseudo Code	IF (Webpage hyperlink is clicked) THEN SHOW Static Webpage Home ENDIF

Primitive Level Diagram	1.1 Check Forgotten Password
Process	1.1.4
Pseudo Code	IF (Login Navbar Item is clicked) THEN SHOW Login Webpage ENDIF

Primitive Level Diagram	1.1 Check Forgotten Password
Process	1.1.6
Pseudo Code	IF (Login Navbar Item is clicked) THEN SHOW Forgot Password Webpage ENDIF

Primitive Level Diagram	1.1 Check Forgotten Password
Process	1.1.8
Pseudo Code	IF "Email Address" IS NOT null THEN set Mail Message equal to "Email Address" and email hyperlink ENDIF

Primitive Level Diagram	1.1 Check Forgotten Password
Process	1.1.10
Pseudo Code	<p>IF ("Email Address" IS NOT null AND "Email Address" exists as EmailAddress in Person table)</p> <p style="padding-left: 20px;">THEN Display Security Question for Person with EmailAddress equal to "Email Address" with answer field</p> <p style="padding-left: 20px;">ELSE Display error message</p>

Primitive Level Diagram	1.1 Check Forgotten Password
Process	1.1.12
Pseudo Code	<p>IF "Security Answer" entered into textbox equals Answer from SecurityAnswer table in Database</p> <p style="padding-left: 20px;">THEN Answer IS valid</p> <p style="padding-left: 20px;">ELSE Display error message</p>

Primitive Level Diagram	1.2 Change Password
Process	1.2.2
Pseudo Code	IF (Profile Icon is clicked AND UserType equals Student) THEN SHOW Student Profile Webpage ENDIF

Primitive Level Diagram	1.2 Change Password
Process	1.2.4
Pseudo Code	IF (Change Password hyperlink is clicked) THEN SHOW Change Password Webpage ENDIF

Primitive Level Diagram	1.2 Change Password
Process	1.2.6
Pseudo Code	IF ("Email Address" equals EmailAddress in Person table AND "Password" equals Password in Person table AND "New Password" is greater than 8 characters and less than 35 characters and contains at least one numeric value AND "Confirm Password" is equal to "New Password") THEN Verification equals "Success" ELSE Verification equals "Fail"

Primitive Level Diagram	1.2 Change Password
Process	1.2.7
Pseudo Code	IF (Verification equals “Success”) THEN UPDATE Table Person SET Password equal to “New Password” WHERE Email Address equals “Email Address” ENDIF

Primitive Level Diagram	1.2 Change Password
Process	1.2.8
Pseudo Code	IF Update is successful THEN Display Modal “Success Your password has been successfully updated!” ENDIF

Primitive Level Diagram	1.3 Login
Process	1.3.2
Pseudo Code	IF (Login Navbar Item is clicked) THEN SHOW Login Webpage ENDIF

Primitive Level Diagram	1.3 Login
Process	1.3.4
Pseudo Code	IF "Password" equals Password in Person table WHERE EmailAddress equals "Email Address" THEN Login Details ARE valid ELSE Login Details ARE NOT valid

Primitive Level Diagram	1.3 Login
Process	1.3.5
Pseudo Code	SET Access right equal to Current Person User Type ID

Primitive Level Diagram	1.3 Login
Process	1.3.6
Pseudo Code	<p>IF Access right is equal to 1</p> <p style="padding-left: 20px;">THEN SHOW Student Main Menu</p> <p>ELSE IF Access right equal to 2</p> <p style="padding-left: 20px;">THEN SHOW Volunteer Main Menu</p> <p>ELSE IF Access right equal to 3</p> <p style="padding-left: 20px;">THEN SHOW Admin Main Menu</p> <p>ELSE Display error message</p>

Primitive Level Diagram	1.3 Login
Process	1.3.7
Pseudo Code	<p>IF (Login equals True)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 40px;">INSERT INTO Table AuditLog attribues (PersonID, UserID, LoginTime, LoginDate, LoginDuration)</p> <p style="padding-left: 40px;">VALUES Current user PersonID, Current User UserID, Current Time, Current Date and Time between login and logout</p> <p style="padding-left: 20px;">ENDIF</p>

Primitive Level Diagram	1.4 Logout
Process	1.4.1
Pseudo Code	IF (User Dropdown is clicked) THEN SHOW My Profile Dropdown List

Primitive Level Diagram	1.4 Logout
Process	1.4.4
Pseudo Code	IF Logout Requested THEN Display Modal "Are you sure you want to be logged out of your account?"

Primitive Level Diagram	1.4 Logout
Process	1.4.6
Pseudo Code	IF (Confirm Logout button is clicked) THEN SHOW Login Webpage AND clear session user data

Primitive Level Diagram	1.5 Deactivate Account
Process	1.5.2
Pseudo Code	IF (User Dropdown is clicked) THEN SHOW User Dropdown List ENDIF

Primitive Level Diagram	1.5 Deactivate Account
Process	1.5.4
Pseudo Code	IF (My Profile Dropdown List Item is clicked) THEN SHOW My Profile List ENDIF

Primitive Level Diagram	1.5 Deactivate Account
Process	1.5.6
Pseudo Code	IF Deactivate Requested THEN Display Modal “Your account has been successfully deactivated”

Primitive Level Diagram	1.5 Deactivate Account
Process	1.5.8
Pseudo Code	UPDATE Table Person SET Status equal to “Inactive” WHERE PersonID is equal to current user’s PersonID SHOW My Profile Webpage

Primitive Level Diagram	1.6 Create User Type
Process	1.6.2
Pseudo Code	IF (User Dropdown is clicked) THEN SHOW User Dropdown List ENDIF

Primitive Level Diagram	1.6 Create User Type
Process	1.6.4
Pseudo Code	IF (User Type Item is clicked) THEN SHOW User Type Webpage ENDIF

Primitive Level Diagram	1.6 Create User Type
Process	1.6.6
Pseudo Code	IF (Create User Type button is clicked) THEN SHOW Create User Type Webpage ENDIF

Primitive Level Diagram	1.6 Create User Type
Process	1.6.8
Pseudo Code	<p>IF (Description is a string that is less than 20 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN Description IS valid</p> <p>ELSE Description IS NOT valid</p> <p>IF (AccessRight is less than 10 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN AccessRight IS valid</p> <p>ELSE AccessRight IS NOT valid</p> <p>IF (Description IS valid AND AccessRight IS valid)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 40px;">INSERT INTO Table UserType attribues (Description and AccessRight)</p> <p style="padding-left: 40px;">VALUES "Description" and "AccessRight"</p> <p>ELSE Display Error Message</p>

Primitive Level Diagram	1.6 Create User Type
Process	1.6.9
Pseudo Code	<p>IF (User Type Successfully Added)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 20px;">SHOW User Type Webpage</p> <p>ENDIF</p>

Primitive Level Diagram	1.7 Update User Type
Process	1.7.2
Pseudo Code	IF (User Dropdown is clicked) THEN SHOW User Dropdown List ENDIF

Primitive Level Diagram	1.7 Update User Type
Process	1.7.4
Pseudo Code	IF (User Type Item is clicked) THEN SHOW User Type Webpage ENDIF

Primitive Level Diagram	1.7 Update User Type
Process	1.7.6
Pseudo Code	IF (Update button is clicked) THEN SHOW Update User Type Webpage ENDIF

Primitive Level Diagram	1.7 Update User Type
Process	Alt 1.7.6
Pseudo Code	IF (Cancel button is clicked) THEN SHOW Main Menu Webpage ENDIF

Primitive Level Diagram	1.7 Update User Type
Process	1.7.8
Pseudo Code	<p>IF (Description is a string that is less than 20 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN Description IS valid</p> <p>ELSE Description IS NOT valid</p> <p>IF (AccessRight is less than 10 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN AccessRight IS valid</p> <p>ELSE AccessRight IS NOT valid</p> <p>IF (Description IS valid AND AccessRight IS valid)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 40px;">UPDATE table UserType</p> <p style="padding-left: 40px;">SET Description equal to "Description"</p> <p style="padding-left: 40px;">AND AccessRight equal to "Access Right"</p> <p style="padding-left: 40px;">WHERE UserID is equal to current UserID</p> <p>ELSE Display Error Message</p>

Primitive Level Diagram	1.7 Update User Type
Process	1.7.9
Pseudo Code	<p>IF (User Type Successfully Updated)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 20px;">SHOW User Type Webpage</p> <p>ENDIF</p>

Primitive Level Diagram	1.8 Delete User Type
Process	1.8.2

Pseudo Code	IF (User Dropdown is clicked) THEN SHOW User Dropdown List ENDIF
--------------------	---

Primitive Level Diagram	1.8 Delete User Type
Process	1.8.4
Pseudo Code	IF (User Type Item is clicked) THEN SHOW User Type Webpage ENDIF

Primitive Level Diagram	1.8 Delete User Type
Process	1.8.6
Pseudo Code	IF (Update button is clicked) THEN SHOW Delete User Type Webpage ENDIF

Primitive Level Diagram	1.8 Delete User Type
Process	Alt 1.8.8
Pseudo Code	IF (Delete button is clicked) THEN DELETE from table UserType WHERE UserID is equal to selected UserType UserID SHOW Modal with message " ENDIF

Primitive Level Diagram	1.8 Delete User Type
Process	Alt 1.8.10
Pseudo Code	IF (User Type Delete is Successful) THEN

SHOW User Type Webpage

ENDIF

Primitive Level Diagram	1.9 Search Alumni
Process	1.9.2
Pseudo Code	IF (Members Dropdown is clicked) THEN SHOW Members Dropdown List ENDIF

Primitive Level Diagram	1.9 Search Alumni
Process	1.9.4
Pseudo Code	IF (Alumni Dropdown List Item is clicked) THEN SHOW Alumni Webpage ENDIF

Primitive Level Diagram	1.9 Search Alumni
Process	1.9.5.2
Pseudo Code	IF (Search criteria is provided) THEN SELECT all Alumni FROM Alumni table WHERE Name is equal to search criteria in search textbox DISPLAY All alumni that match search in Alumni container ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	1.10.1
Pseudo Code	IF (Webpage hyperlink is clicked) THEN SHOW Static Webpage Home ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	1.10.3
Pseudo Code	IF (About Navbar Item is clicked) THEN SHOW About Webpage ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	Alt 1.10.3a
Pseudo Code	IF (Contact Navbar Item is clicked) THEN SHOW Contact Webpage ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	Alt 1.10.3b
Pseudo Code	IF (Gallery Navbar Item is clicked) THEN SHOW Gallery Webpage ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	Alt 1.10.3c
Pseudo Code	IF (Facebook Icon is clicked) THEN OPEN Facebook ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	Alt 1.10.3d
Pseudo Code	IF (Twitter Icon is clicked) THEN OPEN Twitter ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	Alt 1.10.33e
Pseudo Code	IF (Instagram Icon is clicked) THEN OPEN Instagram ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	Alt 1.10.33f
Pseudo Code	IF (Blog Icon is clicked) THEN OPEN Blog ENDIF

Primitive Level Diagram	1.10 View Static Webpage
Process	Alt 1.10.33f
Pseudo Code	IF (Donation Icon is clicked) THEN SHOW Donation Webpage ENDIF

3.5.2 Volunteer Subsystem Pseudo Code

Primitive Level Diagram	2.1 Register Volunteer
Process	2.1.3
Pseudo Code	IF (Volunteer radio button is clicked) THEN SHOW Modal with message “Enter Unique Code” and required unique code field ENDIF

Primitive Level Diagram	2.1 Register Volunteer
Process	2.1.5
Pseudo Code	IF (“Unique Code” entered in textbox IS NOT null AND “Unique Code” entered in textbox exists in UniqueCode table AND UniqueCode status is ‘Unused’) THEN Unique Code IS valid ELSE Unique Code IS NOT valid

Primitive Level Diagram	2.1 Register Volunteer
Process	2.1.6
Pseudo Code	IF (“Unique Code” is Valid) THEN UPDATE table UniqueCode SET Status equal to “Used” WHERE Code is equal to “Unique Code” entered in textbox Update is Successful ELSE Display Error Message

Primitive Level Diagram	2.1 Register Volunteer
Process	2.1.7
Pseudo Code	IF (Update is Successful) THEN SHOW Register Volunteer Webpage ENDIF
Primitive Level Diagram	2.1 Register Volunteer
Process	2.1.9
Pseudo Code	IF ("Name" is string value less than 35 characters AND IS NOT null) THEN "Name" IS valid ELSE "Name" IS NOT valid IF ("Surname" is string value less than 35 characters AND IS NOT null) THEN "Surname" IS valid ELSE "Surname" IS NOT valid IF ("Phone Number" is less than 15 characters with no spaces AND IS NOT null) THEN "Phone Number" IS valid ELSE "Phone Number" IS NOT valid IF ("Email Address" is greater than 7 characters AND less than 255 characters AND is a full validated email address AND IS NOT null) THEN "Email Address" IS valid ELSE "Email Address" IS NOT valid IF ("Date Of Birth" is less than 10 characters in the format CCYY-MM-DD AND IS NOT null) THEN "Date Of Birth" IS valid

	<p>ELSE "Date Of Birth" IS NOT valid</p> <p>IF ("Security Question" is selected from dropdown list AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Security Question" IS valid</p> <p>ELSE "Security Question" IS NOT valid</p> <p>IF ("Security Answer" is less than 200 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Security Answer" IS valid</p> <p>ELSE "Security Answer" IS NOT valid</p> <p>IF ("Volunteer Type" is selected from dropdown list AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Volunteer Type" IS valid</p> <p>ELSE "Volunteer Type" IS NOT valid</p> <p>IF ("Residence" is selected from dropdown list AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Residence" IS valid</p> <p>ELSE "Residence" IS NOT valid</p>
--	---

Primitive Level Diagram	2.1 Register Volunteer
Process	2.1.10
Pseudo Code	<p>IF (“Name” IS valid AND “Surname” IS valid AND “Phone Number” IS valid AND “Email Address” IS valid AND “Date Of Birth” IS valid “Security Question” IS valid AND “Security Answer” IS valid “Volunteer Type” IS valid AND “Residence” IS valid)</p> <p>THEN</p> <pre>INSERT INTO Table Person attributes (Name, Surname, Phone, DateOfBirth) VALUES “Name”, “Surname”, “Phone”, “DateOfBirth” WHERE PersonID equals Current User’s PersonID</pre> <p>AND</p> <pre>INSERT INTO Table SecurityAnswer attributes (Question, Answer) VALUES “Security Question”, “Security Answer” WHERE PersonID equals Current User’s PersonID</pre> <p>AND</p> <pre>INSERT INTO Table Volunteer attributes (VolunteerTypeID, ResidencelD) VALUES “Volunteer Type”, “Residence” WHERE PersonID equals Current User’s PersonID</pre> <p>ELSE Display Error Message</p>

Primitive Level Diagram	2.1 Register Volunteer
Process	2.1.11

Pseudo Code**IF**(Volunteer successfully Added)**THEN** Display Modal with message
“Congratulations! Your registration was
successful!”**ENDIF**

Primitive Level Diagram	2.2 Search Volunteer
Process	2.2.2
Pseudo Code	IF (Member Navbar Item is clicked) THEN Display Members Dropdown List ENDIF

Primitive Level Diagram	2.2 Search Volunteer
Process	2.2.4
Pseudo Code	IF (Volunteers Dropdown List Item is clicked) THEN SHOW Volunteers Webpage ENDIF

Primitive Level Diagram	2.2 Search Volunteer
Process	2.2.6
Pseudo Code	IF (Search button is clicked) THEN SELECT all Volunteers FROM Volunteer joined with the Person tables WHERE the Name is equal to the text entered into the search by name textbox ENDIF

Primitive Level Diagram	2.2 Search Volunteer
Process	2.2.7
Pseudo Code	IF (Number of selected Volunteers is not equal to null) THEN FOREACH Volunteer retrieved DISPLAY Volunteer details ELSE Display Error Message

Primitive Level Diagram	2.3 Update Volunteer
Process	2.3.2

Pseudo Code	IF (User Navbar Item is clicked) THEN Display User Dropdown List ENDIF
-------------	---

Primitive Level Diagram	2.3 Update Volunteer
Process	2.3.4
Pseudo Code	IF (My Profile dropdown list Item is clicked) THEN Display My Profile Webpage ENDIF

Primitive Level Diagram	2.3 Update Volunteer
Process	2.3.6
Pseudo Code	<p>IF ("Name" is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Name" IS valid</p> <p style="padding-left: 20px;">ELSE "Name" IS NOT valid</p> <p>IF ("Surname" is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Surname" IS valid</p> <p style="padding-left: 20px;">ELSE "Surname" IS NOT valid</p> <p>IF ("Phone Number" is less than 15 characters with no spaces AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Phone Number" IS valid</p> <p style="padding-left: 20px;">ELSE "Phone Number" IS NOT valid</p> <p>IF ("Email Address" is greater than 7 characters AND less than 255 characters AND is a full validated email address AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Email Address" IS valid</p> <p style="padding-left: 20px;">ELSE "Email Address" IS NOT valid</p> <p>IF ("Date Of Birth" is less than 10 characters in the format CCYY-MM-DD AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Date Of Birth" IS valid</p> <p style="padding-left: 20px;">ELSE "Date Of Birth" IS NOT valid</p> <p>IF ("Volunteer Type" is selected from dropdown list AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Volunteer Type" IS valid</p> <p style="padding-left: 20px;">ELSE "Volunteer Type" IS NOT valid</p> <p>IF ("Residence" is selected from dropdown list AND IS NOT null)</p>

	<p>THEN "Residence" IS valid</p> <p>ELSE "Residence" IS NOT valid</p> <p>IF ("Name" IS valid AND "Surname" IS valid AND "Phone Number" IS valid AND "Email Address" IS valid AND "Date Of Birth" IS valid AND "Volunteer Type" IS valid AND "Residence" IS valid)</p> <p>THEN Verification equals "Success"</p> <p>ELSE Verification equals "Fail"</p>
--	---

Primitive Level Diagram	2.3 Update Volunteer
Process	2.3.7
Pseudo Code	<p>IF (Update button is clicked and Verification equals "Success")</p> <p>THEN Display Modal with message "Are you sure you would like to update your account?"</p> <p>ELSE Display Error Message</p>

Primitive Level Diagram	2.3 Update Volunteer
Process	2.3.9
Pseudo Code	<p>IF (Confirm button is clicked)</p> <p>THEN</p> <p> UPDATE Table Person</p> <p> AND</p> <p> Update Table Volunteer</p> <p>ENDIF</p>

Primitive Level Diagram	2.3 Update Volunteer
Process	2.3.10
Pseudo Code	IF (Update is a success) THEN Display Modal with message: "Success! You have successfully updated your profile" ENDIF

Primitive Level Diagram	2.4 Delete Volunteer
Process	2.4.3
Pseudo Code	IF (Delete button is clicked) THEN Display Modal with message: "Are you sure? This volunteer will be deleted forever from the system?" ENDIF

Primitive Level Diagram	2.4 Delete Volunteer
Process	2.4.5
Pseudo Code	IF (Confirm button is clicked) THEN Delete Volunteer from Volunteer Table Where the selected volunteers ID is equal to PersonID ENDIF

Primitive Level Diagram	2.4 Delete Volunteer
Process	2.4.6
Pseudo Code	IF (Delete is successful) THEN Display Modal with message: "You have successfully deleted the volunteer" ENDIF

Primitive Level Diagram	2.5 Create Volunteer Type
Process	2.5.2

Pseudo Code	IF (Volunteers Dropdown List Item is clicked) THEN SHOW Volunteers Webpage ENDIF
--------------------	---

Primitive Level Diagram	2.5 Create Volunteer Type
Process	2.5.4
Pseudo Code	IF (Volunteers Type Tab is clicked) THEN Display Volunteer Type Tab ENDIF

Primitive Level Diagram	2.5 Create Volunteer Type
Process	2.5.6
Pseudo Code	IF (Add button is clicked) THEN SHOW Add Volunteer Type Webpage ENDIF

Primitive Level Diagram	2.5 Create Volunteer Type
Process	2.5.8
Pseudo Code	IF (Create Button is clicked) THEN Display Modal with message: "Are you sure you want to create this volunteer type?" ENDIF

Primitive Level Diagram	2.5 Create Volunteer Type
Process	2.5.10
Pseudo Code	IF ("Description" is a string value less than 25 characters and IS NOT null) THEN "Description" IS valid ELSE "Description" IS NOT valid

Primitive Level Diagram	2.5 Create Volunteer Type
Process	2.5.11
Pseudo Code	IF ("Description" IS valid) THEN INSERT INTO table VolunteerType attribute Description the value of "Description" ELSE Display Error Message

Primitive Level Diagram	2.5 Create Volunteer Type
Process	2.5.12
Pseudo Code	IF (Insert is success) THEN Display Modal with message: "The volunteer type has been successfully updated" ENDIF

Primitive Level Diagram	2.5 Create Volunteer Type
Process	2.5.14
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Volunteers Webpage ENDIF

Primitive Level Diagram	2.6 Search Volunteer Type
--------------------------------	---------------------------

Process	2.6.2
Pseudo Code	IF (Volunteers Dropdown List Item is clicked) THEN SHOW Volunteers Webpage ENDIF

Primitive Level Diagram	2.6 Search Volunteer Type
Process	2.6.4
Pseudo Code	IF (Volunteers Type Tab is clicked) THEN Display Volunteer Type Tab ENDIF

Primitive Level Diagram	2.6 Search Volunteer Type
Process	2.6.6
Pseudo Code	IF (Search Button is clicked) THEN SELECT all volunteer types FROM VolunteerType table WHERE the Description is equal to the search textbox value Display all results in volunteer type container ENDIF

Primitive Level Diagram	2.7 Update Volunteer Type
Process	2.7.2
Pseudo Code	IF (Volunteers Dropdown List Item is clicked) THEN SHOW Volunteers Webpage ENDIF

Primitive Level Diagram	2.7 Update Volunteer Type
Process	2.7.4
Pseudo Code	IF (Volunteers Type Tab is clicked) THEN Display Volunteer Type Tab ENDIF

Primitive Level Diagram	2.7 Update Volunteer Type
Process	2.7.6
Pseudo Code	IF (Search Button is clicked) THEN SELECT all volunteer types FROM VolunteerType table WHERE the Description is equal to the search textbox value Display all results in volunteer type container ENDIF

Primitive Level Diagram	2.7 Update Volunteer Type
Process	2.7.8
Pseudo Code	IF (Update button is clicked) THEN SHOW Update Volunteer Type Webpage ENDIF

Primitive Level Diagram	2.7 Update Volunteer Type
Process	2.7.10

Pseudo Code	IF (Save Button is clicked) THEN Display Modal with message: "Are you sure you want to update this volunteer type?" ENDIF
--------------------	---

Primitive Level Diagram	2.7 Update Volunteer Type
Process	2.7.12
Pseudo Code	IF ("Description" is a string value less than 25 characters and IS NOT null) THEN "Description" IS valid ELSE "Description" IS NOT valid

Primitive Level Diagram	2.7 Update Volunteer Type
Process	2.7.13
Pseudo Code	IF ("Description" IS valid) THEN UPDATE table VolunteerType SET attribute Description equal to the value of "Description" ELSE Display Error Message

Primitive Level Diagram	2.8 Delete Volunteer Type
Process	2.8.2
Pseudo Code	IF (Volunteers Dropdown List Item is clicked) THEN SHOW Volunteers Webpage ENDIF

Primitive Level Diagram	2.8 Delete Volunteer Type
Process	2.8.4
Pseudo Code	IF (Volunteers Type Tab is clicked) THEN Display Volunteer Type Tab ENDIF

Primitive Level Diagram	2.8 Delete Volunteer Type
Process	2.8.6
Pseudo Code	IF (Search Button is clicked) THEN SELECT all volunteer types FROM VolunteerType table WHERE the Description is equal to the search textbox value Display all results in volunteer type container ENDIF

Primitive Level Diagram	2.8 Delete Volunteer Type
Process	2.8.8
Pseudo Code	IF (Details button is clicked) THEN SHOW Update Volunteer Type Webpage ENDIF

Primitive Level Diagram	2.8 Delete Volunteer Type
Process	2.8.10

Pseudo Code	IF (Delete button is clicked) THEN Display modal with message: "Are you sure you want to delete this volunteer type?" ENDIF
-------------	---

Primitive Level Diagram	2.8 Delete Volunteer Type
Process	2.8.12
Pseudo Code	IF (Confirm button is clicked) THEN DELETE Volunteer Type from VolunteerType table ENDIF

Primitive Level Diagram	2.9 Generate Unique Code
Process	2.9.2
Pseudo Code	IF (Members Navbar item is clicked) THEN Display Members dropdown list ENDIF

Primitive Level Diagram	2.9 Generate Unique Code
Process	2.9.4
Pseudo Code	IF (Volunteers item is clicked) THEN SHOW Volunteers Webpage ENDIF

Primitive Level Diagram	2.9 Generate Unique Code
Process	2.9.6
Pseudo Code	IF (Generate Unique Code button is clicked) THEN SHOW Generate Unique Code Webpage ENDIF

Primitive Level Diagram	2.9 Generate Unique Code
Process	2.9.8
Pseudo Code	<pre> IF (Generate Unique Code button is clicked) THEN Generate Random Unique Code IF(New Code does not exist in UniqueCode table) THEN INSERT INTO Table UniqueCode Attributes (Code, Status) Value of New Code and "Unused" ELSE GO TO Generate Random Unique Code ENDIF </pre>

Primitive Level Diagram	2.9 Generate Unique Code
Process	2.9.9
Pseudo Code	IF (Generate Unique Code button is clicked) THEN SHOW Generate Unique Code Webpage ENDIF

Primitive Level Diagram	2.10 Register Admin
Process	2.10.2
Pseudo Code	IF (User Navbar item is clicked) THEN Display User dropdown list ENDIF

Primitive Level Diagram	2.10 Register Admin
Process	2.10.4
Pseudo Code	IF (Register admin dropdown list item is clicked) THEN SHOW Register Admin Webpage ENDIF

Primitive Level Diagram	2.10 Register Admin
Process	2.10.6
Pseudo Code	IF (IF toggle is set to ON) THEN UPDATE UserType Table SET Status equal to "Active" ENDIF

Primitive Level Diagram	2.10 Register Admin
Process	Alt 2.10.6
Pseudo Code	IF (Cancel button is clicked) THEN SHOW Main Menu Webpage ENDIF

Primitive Level Diagram	2.11 Deregister Admin
Process	2.11.2
Pseudo Code	IF (User Navbar item is clicked) THEN Display User dropdown list ENDIF

Primitive Level Diagram	2.11 Deregister Admin
Process	2.11.4
Pseudo Code	IF (Register admin dropdown list item is clicked) THEN SHOW Register Admin Webpage ENDIF

Primitive Level Diagram	2.11 Deregister Admin
Process	2.11.6
Pseudo Code	IF (IF toggle is set to OFF) THEN UPDATE UserType Table SET Status equal to "Inactive" ENDIF

Primitive Level Diagram	2.11 Deregister Admin
Process	Alt 2.11.6
Pseudo Code	IF (Cancel button is clicked) THEN SHOW Main Menu Webpage ENDIF

3.5.3 Student Subsystem Pseudo Code

Primitive Level Diagram	3.1 Register Volunteer
Process	3.1.2
Pseudo Code	IF (Student radio button is clicked) THEN SHOW Register Student Webpage ENDIF

Primitive Level Diagram	3.1 Register Volunteer
Process	3.1.4
Pseudo Code	IF ("Name" is string value less than 35 characters AND IS NOT null) THEN "Name" IS valid ELSE "Name" IS NOT valid IF ("Surname" is string value less than 35 characters AND IS NOT null) THEN "Surname" IS valid ELSE "Surname" IS NOT valid IF ("Student Number" is less than 8 characters AND IS NOT null) THEN "Student Number" IS valid ELSE "Student Number" IS NOT valid IF ("Phone Number" is less than 15 characters with no spaces AND IS NOT null) THEN "Phone Number" IS valid ELSE "Phone Number" IS NOT valid IF ("Date Of Birth" is less than 10 characters in the format CCYY-MM-DD AND IS NOT null) THEN "Date Of Birth" IS valid ELSE "Date Of Birth" IS NOT valid

	<p>IF ("Year Of Study" is less than 2 characters integer value between 1 and 10 AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Year Of Study" IS valid</p> <p style="padding-left: 20px;">ELSE "Year Of Study" IS NOT valid</p> <p>IF ("Security Question" is selected from dropdown list AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Security Question" IS valid</p> <p style="padding-left: 20px;">ELSE "Security Question" IS NOT valid</p> <p>IF ("Security Answer" is less than 200 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Security Answer" IS valid</p> <p style="padding-left: 20px;">ELSE "Security Answer" IS NOT valid</p> <p>IF ("Student Type" IS NOT null)</p> <p style="padding-left: 20px;">THEN "Student Type" IS valid</p> <p style="padding-left: 20px;">ELSE "Student Type" IS NOT valid</p> <p>IF ("Residence" IS NOT null)</p> <p style="padding-left: 20px;">THEN "Residence" IS valid</p> <p style="padding-left: 20px;">ELSE "Residence" IS NOT valid</p>
--	---

Primitive Level Diagram	3.1 Register Volunteer
Process	3.1.6
Pseudo Code	<p>IF ("Name" IS valid AND "Surname" IS valid AND "Student Number" IS valid AND "Phone Number" IS valid AND "Date Of Birth" IS valid AND "Year Of Study" IS valid AND Security Question" IS valid AND "Security Answer" IS valid AND "Student Type" IS valid AND "Residence" IS valid)</p> <p style="padding-left: 20px;">THEN</p> <pre style="padding-left: 40px;">INSERT INTO Table Person attributes (Name, Surname, Phone, DateOfBirth) VALUES "Name", "Surname", "Phone", "DateOfBirth"</pre>

	<pre> WHERE PersonID equals Current User's PersonID AND INSERT INTO Table SecurityAnswer attributes (Question, Answer) VALUES "Security Question", "Security Answer" WHERE PersonID equals Current User's PersonID AND INSERT INTO Table Student attributes (StudentTypeID, ResID) VALUES "StudentTypeID" of selected student type, "ResID" of selected Res Type WHERE PersonID equals Current User's PersonID DISPLAY Modal with message: "Registration was successful" ENDIF </pre>
--	---

Primitive Level Diagram	3.1 Register Volunteer
Process	3.1.7
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Login Webpage ENDIF

Primitive Level Diagram	3.2 Search Student
Process	3.2.2
Pseudo Code	IF (Member Navbar Item is clicked) THEN Display Members Dropdown List ENDIF

Primitive Level Diagram	3.2 Search Student
Process	3.2.4
Pseudo Code	IF (Students Dropdown List Item is clicked) THEN SHOW Students Webpage ENDIF

Primitive Level Diagram	3.2 Search Student
Process	Alt 3.2.5
Pseudo Code	IF (Search button is clicked) THEN SELECT all Students FROM Student joined with the Person tables WHERE the Name is equal to the text entered into the search by name textbox FOREACH Student retrieved DISPLAY Student details in container ENDIF

Primitive Level Diagram	3.2 Search Student
Process	3.2.6
Pseudo Code	IF (Details button Item is clicked) THEN SHOW Student Details Webpage ENDIF

Primitive Level Diagram	3.2 Search Student
Process	Alt 3.2.6
Pseudo Code	IF (Cancel button Item is clicked) THEN SHOW Main Menu Webpage ENDIF

Primitive Level Diagram	3.3 Update Student
Process	3.3.2
Pseudo Code	IF (User Navbar Item is clicked) THEN Display User Dropdown List ENDIF

Primitive Level Diagram	3.3 Update Student
Process	3.3.4
Pseudo Code	IF (My Profile Dropdown list Item is clicked) THEN SHOW My Profile Webpage ENDIF

Primitive Level Diagram	3.3 Update Student
Process	3.3.6
Pseudo Code	IF ("Name" is string value less than 35 characters AND IS NOT null) THEN "Name" IS valid ELSE "Name" IS NOT valid IF ("Surname" is string value less than 35 characters AND IS NOT null) THEN "Surname" IS valid ELSE "Surname" IS NOT valid IF ("Student Number" is less than 8 characters AND IS NOT null) THEN "Student Number" IS valid ELSE "Student Number" IS NOT valid IF ("Phone Number" is less than 15 characters with no spaces AND IS NOT null) THEN "Phone Number" IS valid ELSE "Phone Number" IS NOT valid

	<p>IF ("Date Of Birth" is less than 10 characters in the format CCYY-MM-DD AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Date Of Birth" IS valid</p> <p style="padding-left: 20px;">ELSE "Date Of Birth" IS NOT valid</p> <p>IF ("Year Of Study" is less than 2 characters integer value between 1 and 10 AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Year Of Study" IS valid</p> <p style="padding-left: 20px;">ELSE "Year Of Study" IS NOT valid</p> <p>IF ("Student Type" IS NOT null)</p> <p style="padding-left: 20px;">THEN "Student Type" IS valid</p> <p style="padding-left: 20px;">ELSE "Student Type" IS NOT valid</p> <p>IF ("Residence" IS NOT null)</p> <p style="padding-left: 20px;">THEN "Residence" IS valid</p> <p style="padding-left: 20px;">ELSE "Residence" IS NOT valid</p>
--	--

Primitive Level Diagram	3.3 Update Student
Process	3.3.7
Pseudo Code	<p>IF (Save button is clicked)</p> <p style="padding-left: 20px;">THEN Display modal with message: "Save Changes, Are you sure you want to update your account?"</p> <p style="padding-left: 20px;">ENDIF</p>

Primitive Level Diagram	3.3 Update Student
Process	3.3.9
Pseudo Code	<p>IF ("Name" IS valid AND "Surname" IS valid AND "Student Number" IS valid AND "Phone Number" IS valid AND "Date Of Birth" IS valid AND "Year Of Study" IS valid AND "Student Type" IS valid AND "Residence" IS valid)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 20px;">UPDATE Table Person</p>

	<pre> SET attributes (Name, Surname, Phone, DateOfBirth) Equal to VALUES "Name", "Surname", "Phone", "DateOfBirth" WHERE PersonID equals Current User's PersonID AND UPDATE Table Student SET attributes (StudentTypeID, ResID) Equal to VALUES "StudentTypeID" of selected student type, "ResID" of selected Res Type WHERE PersonID equals Current User's PersonID DISPLAY Modal with message: "You have successfully updated your profile" ENDIF </pre>
--	--

Primitive Level Diagram	3.3 Update Student
Process	3.3.11
Pseudo Code	<pre> IF (Confirm button is clicked) THEN SHOW Main Menu Webpage ENDIF </pre>

Primitive Level Diagram	3.4 Delete Student
Process	3.4.2
Pseudo Code	IF (Members Navbar Item is clicked) THEN Display Members Dropdown List ENDIF

Primitive Level Diagram	3.4 Delete Student
Process	3.4.4
Pseudo Code	IF (Students Dropdown list Item is clicked) THEN SHOW Students Webpage ENDIF

Primitive Level Diagram	3.4 Delete Student
Process	3.4.7
Pseudo Code	IF (Delete button is clicked) THEN Display modal with message: "Are you sure you want to delete this student?" ENDIF

Primitive Level Diagram	3.4 Delete Student
Process	3.4.9
Pseudo Code	IF (Confirm button is clicked) THEN DELETE Student FROM Student table AND Display Modal with message: "You have successfully deleted this student" ENDIF

Primitive Level Diagram	3.4 Delete Student
Process	3.4.11
Pseudo Code	IF (Return button is clicked)

	THEN SHOW Students Webpage ENDIF
--	---

Primitive Level Diagram	3.5 Generate Graduate List
Process	3.5.2
Pseudo Code	IF (Members dropdown list item is clicked) THEN Display Members dropdown list ENDIF

Primitive Level Diagram	3.5 Generate Graduate List
Process	3.5.4
Pseudo Code	IF (Students item is clicked) THEN SHOW Students Webpage ENDIF

Primitive Level Diagram	3.5 Generate Graduate List
Process	3.5.6
Pseudo Code	IF (Graduate List button is clicked) THEN SHOW Graduate List Webpage ENDIF

Primitive Level Diagram	3.5 Generate Graduate List
Process	3.5.8
Pseudo Code	IF (Email button is clicked) THEN Display Modal with message : “Please enter an email address:” with an “Email” textbox, “Email” button and “Cancel” button ENDIF

Primitive Level Diagram	3.5 Generate Graduate List
Process	3.5.10
Pseudo Code	IF (Email button is clicked) THEN IF (Email Address is greater than 7 characters AND less Maximum 255 characters AND is a full validated email address) THEN "Email Address" IS valid ELSE "Email Address" IS NOT valid ENDIF

Primitive Level Diagram	3.5 Generate Graduate List
Process	3.5.11
Pseudo Code	IF ("Email Address" IS valid) THEN Convert Graduate List to pdf format and email pdf document to "Email Address" ELSE Display error message

Primitive Level Diagram	3.5 Generate Graduate List
Process	3.5.12
Pseudo Code	IF (Email is sent successfully) THEN Display Modal with message: "Graduate list has been successfully emailed!" ENDIF

Primitive Level Diagram	3.5 Generate Graduate List
Process	3.5.14
Pseudo Code	IF ("Confirm" button is clicked) THEN SHOW Student Webpage ENDIF

Primitive Level Diagram	3.6 Add Student Type
Process	3.6.2
Pseudo Code	IF (Students item is clicked) THEN SHOW Students Webpage ENDIF

Primitive Level Diagram	3.6 Add Student Type
Process	3.6.4
Pseudo Code	IF (Student Type tab is clicked) THEN SHOW Student Type Webpage ENDIF

Primitive Level Diagram	3.6 Add Student Type
Process	3.6.6
Pseudo Code	IF (Add button is clicked) THEN SHOW Create Student Type Webpage ENDIF

Primitive Level Diagram	3.6 Add Student Type
Process	3.6.8
Pseudo Code	IF (Create button is clicked) THEN Display modal with message "Are you sure you want to add this student type?" ENDIF

Primitive Level Diagram	3.6 Add Student Type
Process	3.6.10
Pseudo Code	IF (Confirm button is clicked) THEN IF (Student type is less than 25 characters

	long AND is a string AND is not null) THEN "Student Type" IS valid ELSE "Student Type" IS NOT valid ENDIF
--	---

Primitive Level Diagram	3.6 Add Student Type
Process	3.6.11
Pseudo Code	IF ("Student Type" IS valid) THEN INSERT INTO Table StudentType attributes (Description) VALUES "Student Type" ELSE Display Error Message

Primitive Level Diagram	3.7 Search Student Type
Process	3.7.2
Pseudo Code	IF (Students item is clicked) THEN SHOW Students Webpage ENDIF

Primitive Level Diagram	3.6 Add Student Type
Process	3.7.4
Pseudo Code	IF (Student Type tab is clicked) THEN SHOW Student Type Webpage ENDIF

Primitive Level Diagram	3.6 Add Student Type
Process	3.7.6
Pseudo Code	IF (Search textbox contains text and search button is clicked) THEN SELECT all StudentTypes FROM StudentType table WHERE 'Description' is equal to search textbox text FOREACH StudentType returned Display Student Type details in table ENDIF

Primitive Level Diagram	3.8 Update Student Type
Process	3.8.2
Pseudo Code	IF (Students item is clicked) THEN SHOW Students Webpage ENDIF

Primitive Level Diagram	3. 8 Update Student Type
Process	3.8.4
Pseudo Code	IF (Student Type tab is clicked) THEN SHOW Student Type Webpage ENDIF

Primitive Level Diagram	3. 8 Update Student Type
Process	3.8.6
Pseudo Code	IF (Search textbox contains text and search button is clicked) THEN SELECT all StudentTypes FROM StudentType table WHERE 'Description' is equal to search textbox text FOREACH StudentType returned Display Student Type details in table ENDIF

Primitive Level Diagram	3. 8 Update Student Type
Process	3.8.8
Pseudo Code	IF (Update button is clicked) THEN SHOW Update Student Type Webpage ENDIF

Primitive Level Diagram	3. 8 Update Student Type
Process	3.8.10
Pseudo Code	IF (Update button is clicked) THEN Display modal with message “Are you sure you want to update this student type?” ENDIF

Primitive Level Diagram	3. 8 Update Student Type
Process	3.8.12
Pseudo Code	IF (Confirm button is clicked) THEN IF (Student type is less than 25 characters long AND is a string AND is not null) THEN “Student Type” IS valid ELSE “Student Type” IS NOT valid ENDIF

Primitive Level Diagram	3. 8 Update Student Type
Process	3.8.13
Pseudo Code	IF (“Student Type” IS valid) THEN UPDATE Table StudentType SET attributes (Description) Equal to “Student Type” ELSE Display Error Message

Primitive Level Diagram	3.9 Delete Student Type
Process	3.9.2
Pseudo Code	IF (Students item is clicked) THEN SHOW Students Webpage ENDIF

Primitive Level Diagram	3.9 Delete Student Type
Process	3.9.4
Pseudo Code	IF (Student Type tab is clicked) THEN SHOW Student Type Webpage ENDIF

Primitive Level Diagram	3.9 Delete Student Type
Process	3.9.6
Pseudo Code	IF (Search textbox contains text and search button is clicked) THEN SELECT all StudentTypes FROM StudentType table WHERE 'Description' is equal to search textbox text FOREACH StudentType returned Display Student Type details in table ENDIF

Primitive Level Diagram	3.9 Delete Student Type
Process	3.9.8
Pseudo Code	IF (Details button is clicked) THEN SHOW Update Student Type Webpage ENDIF

Primitive Level Diagram	3.9 Delete Student Type
Process	3.9.10
Pseudo Code	IF (Delete button is clicked) THEN Display modal with message "Are you sure you want to delete this student type?" ENDIF

Primitive Level Diagram	3.9 Delete Student Type
Process	3.8.12
Pseudo Code	IF (Confirm button is clicked) THEN DELETE Record FROM Table StudentType StudentTypeID is equal to selected StudentTypeID ELSE Display Error Message

3.5.4 Venues Subsystem Pseudo Code

Primitive Level Diagram	4.1 Add New Venue
Process	4.1.2
Pseudo Code	IF (Events Navbar Item is clicked) THEN Display Events Dropdown List ENDIF

Primitive Level Diagram	4.1 Add New Venue
Process	4.1.4
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.1 Add New Venue
Process	4.1.6
Pseudo Code	IF (Create new venue button is clicked) THEN SHOW Create Venue Webpage ENDIF

Primitive Level Diagram	4.1 Add New Venue
Process	4.1.8
Pseudo Code	<p>IF ("Name" is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Name" IS valid</p> <p style="padding-left: 20px;">ELSE "Name" IS NOT valid</p> <p>IF ("Capacity" is integer value less than 10 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Capacity" IS valid</p> <p style="padding-left: 20px;">ELSE "Capacity" IS NOT valid</p> <p>IF ("Accessibility" is string value less than 20 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Accessibility" IS valid</p> <p style="padding-left: 20px;">ELSE "Accessibility" IS NOT valid</p> <p>IF ("Street Number" is integer value less than 10 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Street Number" IS valid</p> <p style="padding-left: 20px;">ELSE "Street Number" IS NOT valid</p> <p>IF ("Street Name" is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Street Name" IS valid</p> <p style="padding-left: 20px;">ELSE "Street Name" IS NOT valid</p> <p>IF ("Suburb" is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Suburb" IS valid</p> <p style="padding-left: 20px;">ELSE "Suburb" IS NOT valid</p> <p>IF ("Province" is NOT null)</p> <p style="padding-left: 20px;">THEN "Province" IS valid</p> <p style="padding-left: 20px;">ELSE "Province" IS NOT valid</p>

	IF ("Venue Type" is NOT null) THEN "Venue Type" IS valid ELSE "Venue Type" IS NOT valid
--	--

Primitive Level Diagram	4.1 Add New Venue
Process	4.1.9
Pseudo Code	<p>IF ("Name" IS valid AND "Capacity" IS valid AND "Accessibility" IS valid AND "Street Number" IS valid AND "Street Name" IS valid AND "Suburb" IS valid AND "Province" IS valid AND "Venue Type" IS valid)</p> <p>THEN</p> <pre>INSERT INTO Table Venue attributes (Name, Capacity, Accessibility, VenueTypeID) VALUES "Name", "Capacity", "Accessibility", VenueTypeID where VenueType equals "VenueType"</pre> <p>AND</p> <pre>INSERT INTO Table Address attributes (StreetNumber, StreetName, Suburb, Province) VALUES "Street Number", "Street Name", "Suburb", "Province"</pre> <p>ELSE Display Error Message</p>

Primitive Level Diagram	4.1 Add New Venue
Process	4.1.10
Pseudo Code	<p>IF (Insert is successful)</p> <p>THEN Display Modal with message: "Congratulations! You have successfully added a new venue!"</p> <p>ENDIF</p>

Primitive Level Diagram	4.2 Search Venue
Process	4.2.2
Pseudo Code	IF (Events Navbar Item is clicked) THEN Display Events Dropdown List ENDIF

Primitive Level Diagram	4.2 Search Venue
Process	4.2.4
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.2 Search Venue
Process	4.2.6
Pseudo Code	IF (Search by name textbox is not empty and Search button is clicked) THEN SELECT all venues from Venue table joined with Address table WHERE Venue Name is equal to text in the textbox ENDIF

Primitive Level Diagram	4.2 Search Venue
Process	4.2.7
Pseudo Code	IF (Select statement returns values) THEN Display all venues that match search criteria ENDIF

Primitive Level Diagram	4.3 Update Venue
Process	4.3.3
Pseudo Code	<p>IF ("Name" is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Name" IS valid</p> <p style="padding-left: 20px;">ELSE "Name" IS NOT valid</p> <p>IF ("Capacity" is integer value less than 10 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Capacity" IS valid</p> <p style="padding-left: 20px;">ELSE "Capacity" IS NOT valid</p> <p>IF ("Accessibility" is string value less than 20 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Accessibility" IS valid</p> <p style="padding-left: 20px;">ELSE "Accessibility" IS NOT valid</p> <p>IF ("Street Number" is integer value less than 10 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Street Number" IS valid</p> <p style="padding-left: 20px;">ELSE "Street Number" IS NOT valid</p> <p>IF ("Street Name" is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Street Name" IS valid</p> <p style="padding-left: 20px;">ELSE "Street Name" IS NOT valid</p> <p>IF ("Suburb" is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Suburb" IS valid</p> <p style="padding-left: 20px;">ELSE "Suburb" IS NOT valid</p> <p>IF ("Province" is NOT null)</p> <p style="padding-left: 20px;">THEN "Province" IS valid</p> <p style="padding-left: 20px;">ELSE "Province" IS NOT valid</p>

	IF ("Venue Type" is NOT null) THEN "Venue Type" IS valid ELSE "Venue Type" IS NOT valid
--	--

Primitive Level Diagram	4.3 Update Venue
Process	4.3.4
Pseudo Code	IF (Update button is clicked) THEN Display Modal with message: "Are you sure you want to update this venue?" ENDIF

Primitive Level Diagram	4.3 Update Venue
Process	4.3.6
Pseudo Code	<pre> IF ("Name" IS valid AND "Capacity" IS valid AND "Accessibility" IS valid AND "Street Number" IS valid AND "Street Name" IS valid AND "Suburb" IS valid AND "Province" IS valid AND "Venue Type" IS valid) THEN UPDATE Table Venue SET attributes (Name, Capacity, Accessibility, VenueTypeID) TO VALUES "Name", "Capacity", "Accessibility", VenueTypeID where VenueType equals "VenueType" AND UPDATE Table Address SET attributes (StreetNumber, StreetName, Suburb, Province) TO VALUES "Street Number", "Street Name", "Suburb", "Province" ELSE Display Error Message </pre>

Primitive Level Diagram	4.3 Update Venue
Process	4.3.7
Pseudo Code	<pre> IF (Update is successful) THEN Display Modal with message: "Congratulations! You successfully updated this venue" ENDIF </pre>

Primitive Level Diagram	4.4 Delete Venue
Process	4.4.3
Pseudo Code	IF (Delete button is clicked) THEN Display Modal with message: "Are you sure? Once this venue is deleted it will be deleted forever" ENDIF

Primitive Level Diagram	4.4 Delete Venue
Process	4.4.5
Pseudo Code	IF (Confirm button is clicked) THEN DELETE Record FROM Venue Table WHERE VenueID is equal to selected venue VenueID ENDIF

Primitive Level Diagram	4.4 Delete Venue
Process	4.4.6
Pseudo Code	IF (Delete is successful) THEN Display Modal with message: "This venue has been successfully deleted" ENDIF

Primitive Level Diagram	4.5 Add Venue Type
Process	4.5.2
Pseudo Code	IF (Events Navbar Item is clicked) THEN Display Events Dropdown List ENDIF

Primitive Level Diagram	4.5 Add Venue Type
Process	4.5.4
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.5 Add Venue Type
Process	4.5.6
Pseudo Code	IF (Venue Type Tab is clicked) THEN SHOW Venue Type Webpage ENDIF

Primitive Level Diagram	4.5 Add Venue Type
Process	4.5.8
Pseudo Code	IF (Create button is clicked) THEN SHOW Create Venue Type Webpage ENDIF

Primitive Level Diagram	4.5 Add Venue Type
Process	4.5.10
Pseudo Code	<pre> IF (Create button is clicked) THEN IF ("Name" is a string value less than 50 characters AND IS NOT null) THEN "Name" IS valid ELSE "Name" IS NOT valid IF ("Description" is less than 300 characters AND IS NOT null) THEN "Description" IS valid ELSE "Description" IS NOT valid ENDIF </pre>

Primitive Level Diagram	4.5 Add Venue Type
Process	4.5.11
Pseudo Code	<pre> IF ("Name" IS valid and "Description" IS valid) THEN INSERT INTO Table VenueType attributes (Name, Description) VALUES "Name", "Description" ELSE Display error message </pre>

Primitive Level Diagram	4.6 Search Venue Type
Process	4.6.2
Pseudo Code	IF (Events Navbar Item is clicked) THEN Display Events Dropdown List ENDIF

Primitive Level Diagram	4.6 Search Venue Type
Process	4.6.4
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.6 Search Venue Type
Process	4.6.6
Pseudo Code	IF (Venue Type Tab is clicked) THEN SHOW Venue Type Webpage ENDIF

Primitive Level Diagram	4.6 Search Venue Type
Process	4.6.8
Pseudo Code	IF (Details hyperlink is clicked) THEN SHOW Venue Type Details Webpage ENDIF

Primitive Level Diagram	4.6 Search Venue Type
Process	Alt 4.6.8
Pseudo Code	IF (Search button is clicked) THEN SELECT all VenueTypes FROM VenueType table

	WHERE 'Description' is equal to search textbox text FOREACH VenueType returned Display Student Type details in table ENDIF
--	--

Primitive Level Diagram	4.6 Search Venue Type
Process	4.6.10
Pseudo Code	IF (Update button is clicked) THEN IF ("Name" is a string value less than 50 characters AND IS NOT null) THEN "Name" IS valid ELSE "Name" IS NOT valid IF ("Description" is less than 300 characters AND IS NOT null) THEN "Description" IS valid ELSE "Description" IS NOT valid ENDIF

Primitive Level Diagram	4.7 Update Venue Type
Process	4.7.2
Pseudo Code	IF (Events Navbar Item is clicked) THEN Display Events Dropdown List ENDIF

Primitive Level Diagram	4.7 Update Venue Type
Process	4.7.4
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.7 Update Venue Type
Process	4.7.6
Pseudo Code	IF (Venue Type Tab is clicked) THEN SHOW Venue Type Webpage ENDIF

Primitive Level Diagram	4.7 Update Venue Type
Process	4.7.8
Pseudo Code	IF (Details hyperlink is clicked) THEN SHOW Venue Type Details Webpage ENDIF

Primitive Level Diagram	4.7 Update Venue Type
Process	Alt 4.7.8
Pseudo Code	IF (Update button is clicked) THEN SHOW Update Venue Type Webpage ENDIF

Primitive Level Diagram	4.7 Update Venue Type
Process	4.7.10
Pseudo Code	IF (Update button is clicked) THEN IF (Venue type is less than 25 characters long AND is a string AND is not null) THEN "Venue Type" IS valid ELSE "Venue Type" IS NOT valid ENDIF

Primitive Level Diagram	4.7 Update Venue Type
Process	4.7.11
Pseudo Code	IF (Venue Type Tab is clicked) THEN Display modal with message "Are you sure you want to update this venue type?" ENDIF

Primitive Level Diagram	4.7 Update Venue Type
Process	4.7.13
Pseudo Code	IF ("Name" IS valid and "Description" IS valid) THEN INSERT INTO Table VenueType attributes (Name, Description) VALUES "Name", "Description" ELSE Display error message

Primitive Level Diagram	4.7 Update Venue Type
Process	4.7.15
Pseudo Code	IF (Venue Type Tab is clicked) THEN SHOW Venue Type Webpage ENDIF

Primitive Level Diagram	4.8 Delete Venue Type
Process	4.8.2
Pseudo Code	IF (Events Navbar Item is clicked) THEN Display Events Dropdown List ENDIF

Primitive Level Diagram	4.8 Delete Venue Type
Process	4.8.4
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.8 Delete Venue Type
Process	4.8.6
Pseudo Code	IF (Venue Type Tab is clicked) THEN SHOW Venue Type Webpage ENDIF

Primitive Level Diagram	4.8 Delete Venue Type
Process	4.8.8
Pseudo Code	IF (Delete button is clicked) THEN Display modal with message "Are you sure you want to delete this venue type?" ENDIF

Primitive Level Diagram	4.8 Delete Venue Type
Process	Alt 4.8.8
Pseudo Code	IF (Details hyperlink is clicked) THEN SHOW Venue Type Details Webpage ENDIF

Primitive Level Diagram	4.8 Delete Venue Type
Process	Alt 4.8.10
Pseudo Code	IF (Confirm button is clicked) THEN DELETE Record FROM VenueType Table WHERE VenueTypeID is equal to selected venue VenueTypeID ENDIF

Primitive Level Diagram	4.8 Delete Venue Type
Process	4.8.12
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Venue Type Webpage ENDIF

Primitive Level Diagram	4.9 Add Residence
Process	4.9.2
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.9 Add Residence
Process	4.9.4
Pseudo Code	IF (Residence tab is clicked) THEN Display residence tab ENDIF

Primitive Level Diagram	4.9 Add Residence
Process	4.9.6
Pseudo Code	IF (Add residence button is clicked) THEN SHOW Add Residences Webpage ENDIF

Primitive Level Diagram	4.9 Add Residence
Process	4.9.8
Pseudo Code	IF (Add button is clicked) THEN Display modal with message: "Are you sure you want to add this residence?" ENDIF

Primitive Level Diagram	4.9 Add Residence
Process	4.9.10
Pseudo Code	IF (Confirm button is clicked) THEN IF("Residence Name" is less than 35 characters and IS NOT null) THEN Residence IS valid ENDIF

Primitive Level Diagram	4.9 Add Residence
Process	4.9.11
Pseudo Code	IF (Residence is valid) THEN INSERT INTO Table Residence attributes (Name) VALUES "Residence Name" ENDIF

Primitive Level Diagram	4.10 Search Residence
Process	4.10.2
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.10 Search Residence
Process	4.10.2
Pseudo Code	IF (Residence tab is clicked) THEN Display residence tab ENDIF

Primitive Level Diagram	4.10 Search Residence
Process	4.10.4
Pseudo Code	IF (Search button is clicked) THEN SELECT all records FROM Residence Table Where Name equals textbox text FOREACH Record Selected DISPLAY Details ENDIF

Primitive Level Diagram	4.11 Update Residence
Process	4.11.2
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.11 Update Residence
Process	4.11.4
Pseudo Code	IF (Residence tab is clicked) THEN Display residence tab ENDIF

Primitive Level Diagram	4.11 Update Residence
Process	4.11.6
Pseudo Code	IF (Search button is clicked) THEN SELECT all records FROM Residence Table Where Name equals textbox text FOREACH Record Selected DISPLAY Details ENDIF

Primitive Level Diagram	4.11 Update Residence
Process	4.11.8
Pseudo Code	IF (Details button is clicked) THEN SHOW Update residence Webpage ENDIF

Primitive Level Diagram	4.11 Update Residence
Process	4.11.10
Pseudo Code	IF (Update button is clicked) THEN THEN Display modal with message: "Are you sure you want to update this residence?" ENDIF

Primitive Level Diagram	4.11 Update Residence
Process	4.11.12
Pseudo Code	IF (Confirm button is clicked) THEN IF ("Residence Name" is less than 35 characters and IS NOT null) THEN Residence IS valid ENDIF

Primitive Level Diagram	4.11 Update Residence
Process	4.11.13
Pseudo Code	IF (Residence is valid) THEN UPDATE Table Residence SET attributes (Name) Equal to VALUES "Residence Name" ENDIF

Primitive Level Diagram	4.12 Delete Residence
Process	4.12.2
Pseudo Code	IF (Manage Venues Dropdown List Item is clicked) THEN SHOW Manage Venues Webpage ENDIF

Primitive Level Diagram	4.12 Delete Residence
Process	4.12.2
Pseudo Code	IF (Residence tab is clicked) THEN Display residence tab ENDIF

Primitive Level Diagram	4.12 Delete Residence
Process	4.12.2
Pseudo Code	IF (Search button is clicked) THEN SELECT all records FROM Residence Table Where Name equals textbox text FOREACH Record Selected DISPLAY Details ENDIF

Primitive Level Diagram	4.12 Delete Residence
Process	4.12.2
Pseudo Code	IF (Details button is clicked) THEN SHOW Update residence Webpage ENDIF

Primitive Level Diagram	4.12 Delete Residence
Process	4.12.2
Pseudo Code	IF (Delete button is clicked) THEN THEN Display modal with message: "Are you sure you want to delete this residence?" ENDIF

Primitive Level Diagram	4.11 Update Residence
Process	4.11.12
Pseudo Code	IF (Confirm button is clicked) THEN DELETE Record FROM Residence Table WHERE ResID equal to selected Residence ResID ENDIF

3.5.5 Events Subsystem Pseudo Code

Primitive Level Diagram	AUC 2 Create Event
Process	AUC 2.2
Pseudo Code	IF (Events dropdown list is clicked) THEN SHOW Dropdown List ENDIF

Primitive Level Diagram	AUC 2 Create Event
Process	AUC 2.4
Pseudo Code	IF (Create list item is clicked) THEN SHOW Event Type Selection webpage ENDIF

Primitive Level Diagram	5.1 Create Function
Process	5.1.1
Pseudo Code	IF (Create list item is clicked) THEN SHOW Event Type Selection webpage ENDIF

Primitive Level Diagram	5.1 Create Function
Process	5.1.3
Pseudo Code	IF (Function is clicked) THEN SHOW Create Function Webpage ENDIF

Primitive Level Diagram	5.1 Create Function
Process	5.1.5
Pseudo Code	<p>IF Name equals string value less than 35 characters</p> <p style="padding-left: 20px;">THEN Name IS Valid</p> <p style="padding-left: 20px;">ELSE Name IS NOT Valid</p> <p> </p> <p>IF Guest Speaker selected from the Guest Speaker dropdown list AND either the “Pending invitation” or “Accepted invitation” radio button must have been clicked</p> <p style="padding-left: 20px;">THEN Guest Speaker IS Valid</p> <p style="padding-left: 20px;">ELSE Guest Speaker IS NOT Valid</p> <p> </p> <p>IF Time in format HH:MM as clicked from the time picker</p> <p style="padding-left: 20px;">THEN Time IS Valid</p> <p style="padding-left: 20px;">ELSE Time IS NOT Valid</p> <p> </p> <p>IF Date in format CCYY-MM-DD as clicked from the date picker</p> <p style="padding-left: 20px;">THEN Date IS Valid</p> <p style="padding-left: 20px;">ELSE Date IS NOT Valid</p> <p> </p> <p>IF Venue NOT NULL</p> <p style="padding-left: 20px;">THEN Venue IS Valid</p> <p style="padding-left: 20px;">ELSE Venue IS NOT Valid</p> <p> </p> <p>IF Summary less than 100 characters</p> <p style="padding-left: 20px;">THEN Summary IS Valid</p> <p style="padding-left: 20px;">ELSE Summary IS NOT Valid</p> <p> </p> <p>IF Description is a string value less than 300 characters</p> <p style="padding-left: 20px;">THEN Description IS Valid</p> <p style="padding-left: 20px;">ELSE Description IS NOT Valid</p>

	IF (Name IS Valid AND Guest Speaker IS Valid AND Time IS Valid AND Date IS Valid AND Venue IS Valid AND Summary IS Valid AND Description IS Valid) THEN Validation equals success ELSE Validation equals Fail
--	---

Primitive Level Diagram	5.1 Create Function
Process	Alt 5.1.5
Pseudo Code	IF (No Guest Speaker Selected) THEN SHOW Hide guest speaker fields ENDIF

Primitive Level Diagram	5.1 Create Function
Process	5.1.7
Pseudo Code	IF (Browse button is clicked) THEN SHOW File dialogue box ENDIF

Primitive Level Diagram	5.1 Create Function
Process	5.1.9
Pseudo Code	IF (Picture uploaded from file dialogue box AND Picture is in SVG or PNG format) THEN Display uploaded picture in picturebox ENDIF

Primitive Level Diagram	5.1 Create Function
Process	5.1.11
Pseudo Code	IF (Create button is clicked) THEN Display modal with message "Are you sure you want to create this event?" ENDIF

Primitive Level Diagram	5.1 Create Function
Process	5.1.13
Pseudo Code	IF (Confirm button is clicked AND Validation equals Success) THEN Insert New Function into Event and Function tables Display modal with message: "Congratulations Your Event has been created" ENDIF

Primitive Level Diagram	5.1 Create Function
Process	5.1.15
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Main Menu Webpage ENDIF

Primitive Level Diagram	5.2 Create Community Outreach
Process	5.2.1
Pseudo Code	IF (Create list item is clicked) THEN SHOW Event Type Selection webpage ENDIF

Primitive Level Diagram	5.2 Create Community Outreach
Process	5.2.3
Pseudo Code	IF (Community Engagement button is clicked) THEN SHOW Create Community Engagement Webpage ENDIF

Primitive Level Diagram	5.2 Create Community Outreach
Process	5.2.5
Pseudo Code	IF (Name equals string value less than 35 characters THEN Name IS Valid ELSE Name IS NOT Valid IF Time in format HH:MM as clicked from the time picker THEN Time IS Valid ELSE Time IS NOT Valid IF Date in format CCYY-MM-DD as clicked from the date picker THEN Date IS Valid ELSE Date IS NOT Valid IF Venue NOT NULL THEN Venue IS Valid ELSE Venue IS NOT Valid

	<p>IF Summary less than 100 characters THEN Summary IS Valid ELSE Summary IS NOT Valid</p> <p>IF Description is a string value less than 300 characters THEN Description IS Valid ELSE Description IS NOT Valid</p> <p>IF Event Type IS NOT null THEN Event Type IS Valid ELSE Event Type IS NOT Valid</p> <p>Reading content IS NOT null THEN Reading content IS Valid ELSE Reading content IS NOT Valid</p> <p>IF (Name IS Valid AND Time IS Valid AND Date IS Valid AND Venue IS Valid AND Summary IS Valid AND Description IS Valid AND Reading content IS Valid AND Event Type IS Valid) THEN Validation equals success ELSE Validation equals Fail</p>
--	---

Primitive Level Diagram	5.2 Create Community Outreach
Process	5.2.7
Pseudo Code	IF (Browse button is clicked) THEN SHOW File dialogue box ENDIF

Primitive Level Diagram	5.2 Create Community Outreach
Process	5.2.9
Pseudo Code	IF (Picture uploaded from file dialogue box AND Picture is in SVG or PNG format) THEN Display uploaded picture in picturebox ENDIF

Primitive Level Diagram	5.2 Create Community Outreach
Process	5.2.11
Pseudo Code	IF (Create button is clicked) THEN Display modal with message "Are you sure you want to create this event?" ENDIF

Primitive Level Diagram	5.2 Create Community Outreach
Process	5.2.13
Pseudo Code	IF (Confirm button is clicked AND Validation equals Success) THEN Insert New Community Outreach into Event, CommunityOutreach and Content tables Display modal with message: "Congratulations Your Event has been created" ENDIF

Primitive Level Diagram	5.2 Create Community Outreach
Process	5.2.15
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Main Menu Webpage ENDIF

Primitive Level Diagram	5.3 Create Lecture
Process	5.3.3
Pseudo Code	IF (Lecture button is clicked) THEN SHOW Create Lecture Webpage ENDIF

Primitive Level Diagram	5.3 Create Lecture
Process	5.3.5
Pseudo Code	IF (Browse button is clicked) THEN Display file dialogue box ENDIF

Primitive Level Diagram	5.3 Create Lecture
Process	5.3.7
Pseudo Code	IF (Picture uploaded from file dialogue box AND Picture is in SVG or PNG format) THEN Display uploaded picture in picturebox ENDIF

Primitive Level Diagram	5.3 Create Lecture
Process	5.3.9
Pseudo Code	IF (Create button is clicked) THEN Display modal with message "Are you sure you want to create this event?" ENDIF

Primitive Level Diagram	5.3 Create Lecture
Process	5.3.11
Pseudo Code	IF (Name equals string value less than 35 characters THEN Name IS Valid ELSE Name IS NOT Valid IF Time in format HH:MM as clicked from the time picker THEN Time IS Valid ELSE Time IS NOT Valid IF Date in format CCYY-MM-DD as clicked from the date picker THEN Date IS Valid ELSE Date IS NOT Valid IF Venue NOT NULL THEN Venue IS Valid ELSE Venue IS NOT Valid IF Summary less than 100 characters THEN Summary IS Valid ELSE Summary IS NOT Valid IF Description is a string value less than 300 characters THEN Description IS Valid ELSE Description IS NOT Valid

	IF Residence IS NOT null THEN Residence IS Valid ELSE Residence IS NOT Valid Reading content IS NOT null THEN Reading content IS Valid ELSE Reading content IS NOT Valid IF (Name IS Valid AND Time IS Valid AND Date IS Valid AND Venue IS Valid AND Summary IS Valid AND Description IS Valid AND Reading content IS Valid AND Residence IS Valid) THEN Validation equals success ELSE Validation equals Fail
--	--

Primitive Level Diagram	5.3 Create Lecture
Process	5.3.13
Pseudo Code	IF (Confirm button is clicked AND Validation equals Success) THEN Insert New Lecture into Event and Lecture tables ENDIF

Primitive Level Diagram	5.3 Create Lecture
Process	5.3.15
Pseudo Code	IF (New Lecture is added) THEN Display modal with message: ““Congratulations Your Event has been created”” ENDIF

Primitive Level Diagram	5.3 Create Lecture
Process	5.3.15
Pseudo Code	IF (New Lecture is added) THEN Display modal with message: ““Congratulations Your Event has been created”” ENDIF

Primitive Level Diagram	5.4 Search Event
Process	5.4.2
Pseudo Code	IF (Events dropdown list is clicked) THEN SHOW Dropdown List ENDIF

Primitive Level Diagram	5.4 Search Event
Process	5.4.4
Pseudo Code	IF (Manage Events dropdown list item is clicked) THEN SHOW Manage Event Webpage ENDIF

Primitive Level Diagram	5.4 Search Event
Process	5.4.6
Pseudo Code	IF (Search button is clicked) THEN SELECT all Records FROM Event Table WHERE Event Name equals textbox text ENDIF

Primitive Level Diagram	5.4 Search Event
Process	5.4.8
Pseudo Code	IF (Details hyperlink is clicked) THEN SHOW Event Details Webpage ENDIF

Primitive Level Diagram	5.4 Search Event
Process	5.4.10
Pseudo Code	IF (Return button is clicked)

	THEN SHOW Main menu Webpage ENDIF
--	--

Primitive Level Diagram	5.5 RSVP to Event
Process	5.5.1
Pseudo Code	IF (User has logged in) THEN SHOW Main menu Webpage ENDIF

Primitive Level Diagram	5.5 RSVP to Event
Process	5.5.3
Pseudo Code	IF (RSVP button is clicked) THEN Display modal with message: "Are you sure you want to RSVP to this event?" ENDIF

Primitive Level Diagram	5.5 RSVP to Event
Process	5.5.5
Pseudo Code	IF (Confirm button is clicked) THEN //Insert RSVP INSERT INTO Table RSVP_Event Attributes (PersonID, EventID) VALUES Current user's PersonID and Selected event EventID //Display Confirmation Display modal with message: "You have successfully RSVP'd for the following event: xxxxxxxx" ENDIF

Primitive Level Diagram	5.5 RSVP to Event
Process	5.5.7
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Main menu Webpage

Primitive Level Diagram	5.6 Update Event Information
Process	5.6.2
Pseudo Code	IF (Events dropdown list is clicked) THEN SHOW Dropdown List ENDIF

Primitive Level Diagram	5.6 Update Event Information
Process	5.6.4
Pseudo Code	IF (Manage Events dropdown list item is clicked) THEN SHOW Events Webpage ENDIF

Primitive Level Diagram	5.6 Update Event Information
Process	5.6.7
Pseudo Code	IF (Details button is clicked) THEN SHOW Events Details Webpage ENDIF

Primitive Level Diagram	5.6 Update Event Information
Process	5.6.9
Pseudo Code	IF (Name equals string value less than 35 characters THEN Name IS Valid ELSE Name IS NOT Valid IF Time in format HH:MM as clicked from the time picker THEN Time IS Valid ELSE Time IS NOT Valid

	<p>IF Date in format CCYY-MM-DD as clicked from the date picker</p> <p style="padding-left: 20px;">THEN Date IS Valid</p> <p style="padding-left: 20px;">ELSE Date IS NOT Valid</p> <p>IF Venue NOT NULL</p> <p style="padding-left: 20px;">THEN Venue IS Valid</p> <p style="padding-left: 20px;">ELSE Venue IS NOT Valid</p> <p>IF Summary less than 100 characters</p> <p style="padding-left: 20px;">THEN Summary IS Valid</p> <p style="padding-left: 20px;">ELSE Summary IS NOT Valid</p> <p>IF Description is a string value less than 300 characters</p> <p style="padding-left: 20px;">THEN Description IS Valid</p> <p style="padding-left: 20px;">ELSE Description IS NOT Valid</p> <p>IF (Name IS Valid AND Time IS Valid AND Date IS Valid AND Venue IS Valid AND Summary IS Valid AND Description IS Valid)</p> <p style="padding-left: 20px;">THEN Validation equals success</p> <p style="padding-left: 20px;">ELSE Validation equals Fail</p>
--	---

Primitive Level Diagram	5.6 Update Event Information
Process	5.6.12
Pseudo Code	<p>IF (Validation equals success)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 40px;">Update Event into Event Table and Lecture, CommunityOutreach or Function tables according to EventID</p> <p style="padding-left: 20px;">ENDIF</p>

Primitive Level Diagram	5.6 Update Event Information
Process	5.6.14
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Events Webpage ENDIF

Primitive Level Diagram	5.7 Cancel Event
Process	5.7.2
Pseudo Code	IF (Manage Events dropdown list item is clicked) THEN SHOW Manage Event Webpage ENDIF

Primitive Level Diagram	5.7 Cancel Event
Process	5.7.2
Pseudo Code	IF (Events dropdown list is clicked) THEN SHOW Dropdown List ENDIF

Primitive Level Diagram	5.7 Cancel Event
Process	5.7.4
Pseudo Code	IF (Manage Events dropdown list item is clicked) THEN SHOW Manage Event Webpage ENDIF

Primitive Level Diagram	5.7 Cancel Event
Process	5.7.7
Pseudo Code	IF (Details button is clicked) THEN SHOW Event details Webpage ENDIF

Primitive Level Diagram	5.7 Cancel Event
Process	5.7.9
Pseudo Code	IF (Cancel Event button is clicked) THEN Display modal with message: "Are you sure you want to cancel this event?" ENDIF

Primitive Level Diagram	5.7 Cancel Event
Process	5.7.11
Pseudo Code	IF (Confirm button is clicked) THEN //Delete Event DELETE Record FROM Event Table WHERE EventID equals selected events EventID //Display confirmation message Display modal with message: "The event was successfully cancelled" ENDIF

Primitive Level Diagram	5.7 Cancel Event
Process	5.7.13
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Event Webpage ENDIF

Primitive Level Diagram	5.8 Log Event Attendance
Process	5.8.2
Pseudo Code	IF (Log attendance button is clicked) THEN SHOW Log Attendance Webpage ENDIF

Primitive Level Diagram	5.8 Log Event Attendance
Process	5.8.4
Pseudo Code	IF (Log attendance hyperlink is clicked) THEN SELECT all Records FROM RSVP_Event Table JOINED with the Student Table AND Display a list of all students who RSVP'd WHERE EventID is equal to current EventID ENDIF

Primitive Level Diagram	5.8 Log Event Attendance
Process	5.8.6
Pseudo Code	IF (Search button is clicked) THEN SELECT all Students FROM Student Table WHERE Name equals textbox text AND WHERE StudentID exists in RSVP_Event Table ENDIF

Primitive Level Diagram	5.8 Log Event Attendance
Process	Alt 5.8.5b
Pseudo Code	IF (Add new student button is clicked) THEN SELECT all Students FROM Student Table WHERE Name equals textbox text ENDIF

Primitive Level Diagram	5.8 Log Event Attendance
Process	5.8.8
Pseudo Code	IF (Log Attendance button is clicked) THEN Display Modal with message: "Are you sure that you want to log the attendance of this student?" ENDIF

Primitive Level Diagram	5.8 Log Event Attendance
Process	5.8.10
Pseudo Code	<p>IF (Confirm button is clicked)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 40px;">INSERT INTO Table Attendance</p> <p style="padding-left: 40px;">Attributes(EventID, PersonID)</p> <p style="padding-left: 40px;">The values of students PersonID and current events EventID</p> <p style="padding-left: 20px;">AND</p> <p style="padding-left: 40px;">INSERT INTO Table Milestone</p> <p style="padding-left: 40px;">Attributes(EventID, MilestoneTypeID)</p> <p style="padding-left: 40px;">Current events EventID and Event Type Milestone</p> <p>ENDIF</p>

Primitive Level Diagram	5.9 Send Notification
Process	5.9.2
Pseudo Code	IF (Send notification button is clicked) THEN SHOW Send Notification Webpage ENDIF

Primitive Level Diagram	5.9 Send Notification
Process	5.9.4
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Write A Message Webpage ENDIF

Primitive Level Diagram	5.9 Send Notification
Process	5.9.6
Pseudo Code	IF (Send notification button is clicked) THEN Display modal with message: "Are you sure that you want to send this message?" ENDIF

Primitive Level Diagram	5.9 Send Notification
Process	5.9.8
Pseudo Code	IF (Confirm button is clicked) IF Message is less than 255 characters AND IS NOT null THEN Message IS Valid ELSE Message IS NOT Valid ENDIF

Primitive Level Diagram	5.9 Send Notification
Process	5.9.9
Pseudo Code	IF (Message IS Valid) INSERT INTO Table Notification Attributes (Notification) VALUES (Message) ENDIF

3.5.6 Guest Speaker Subsystem Pseudo Code

Primitive Level Diagram	6.1 Register Guest Speaker
Process	6.1.2
Pseudo Code	IF (Manage Guest Speakers Dropdown List Item is clicked) THEN SHOW Manage Guest Speaker Webpage ENDIF

Primitive Level Diagram	6.1 Register Guest Speaker
Process	6.1.4
Pseudo Code	IF (Register Guest Speaker button is clicked) THEN SHOW Register Guest Speaker Webpage ENDIF

Primitive Level Diagram	6.1 Register Guest Speaker
Process	6.1.6
Pseudo Code	IF (Create Button is clicked) THEN SHOW Register Guest Speaker Webpage ENDIF

Primitive Level Diagram	6.1 Register Guest Speaker
Process	6.1.6
Pseudo Code	IF ("Speaker_Name" is string value less than 35 characters AND IS NOT null) THEN "Speaker_Name" IS valid ELSE "Speaker_Name" IS NOT valid IF ("Speaker_Surname" is string value less than 35 characters AND IS NOT null) THEN "Speaker_Surname" IS valid ELSE "Speaker_Surname" IS NOT valid

	IF (“Phone Number” is integer AND IS NOT null) THEN “Phone Number” IS valid ELSE “Phone Number” IS NOT valid IF (“Email” is string value less than 255 characters AND IS NOT null) THEN “Email” IS valid ELSE “Email” IS NOT valid IF (“PictureLink” is string value less than 255 characters AND IS NOT null) THEN “PictureLink” IS valid ELSE “PictureLink” IS NOT valid
--	---

Primitive Level Diagram	6.1 Register Guest Speaker
Process	6.1.7
Pseudo Code	IF (“Speaker_Name” IS valid AND “Speaker_Surname” IS valid AND “Phone Number” IS valid AND “Email” IS valid AND “PictureLink” IS valid) THEN INSERT INTO Table Guest Speaker attributes(Speaker_Name, Speaker_Surname, Phone Number, PictureLink, SpeakerID) VALUES “Speaker_Name”, “Speaker_Surname”, “Phone Number”, “ PictureLink” SpeakerID where SpeakerID equals “SpeakerID” ELSE Display Error Message

Primitive Level Diagram	6.1 Register Guest Speaker
Process	6.1.8
Pseudo Code	IF (Insert is successful) THEN Display Modal with message: "Congratulations! You have successfully created a new guest speaker!" ENDIF

Primitive Level Diagram	6.1 Register Guest Speaker
Process	6.1.10
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Guest Speaker Webpage ENDIF

Primitive Level Diagram	6.2 Search Guest Speaker
Process	6.2.2
Pseudo Code	IF ("Guest Speakers" dropdown list item is clicked) THEN SHOW Guest Speaker Webpage ENDIF

Primitive Level Diagram	6.2 Search Guest Speaker
Process	6.2.4
Pseudo Code	IF (Search by name textbox is not empty and Search button is clicked) THEN SELECT all Guest Speakers from Guest Speaker table. WHERE Speaker_Name is equal to text in the textbox THEN Display all Guest Speakers that match search criteria

	ENDIF
--	--------------

Primitive Level Diagram	6.2 Search Guest Speaker
Process	6.2.6
Pseudo Code	IF ("Details" hyperlink is clicked) THEN Display all the associated attributes : "Speaker_Name", "Speaker_Surname", "Phone Number", "PictureLink" from Guest Speaker Table ENDIF

Primitive Level Diagram	6.2 Search Guest Speaker
Process	6.2.8
Pseudo Code	IF ("Return" Button is clicked) THEN SHOW Guest Speaker Webpage ENDIF

Primitive Level Diagram	6.3 Update Guest Speaker
Process	6.3.2
Pseudo Code	IF ("Guest Speakers" dropdown list item is clicked) THEN SHOW Guest Speaker Webpage ENDIF

Primitive Level Diagram	6.3 Update Guest Speaker
Process	6.3.4
Pseudo Code	IF (Search by name textbox is not empty and Search button is clicked) THEN SELECT all Guest Speakers from Guest

	<p>Speaker table.</p> <p>WHERE Speaker_Name is equal to text in the textbox</p> <p>THEN Display all Guest Speakers that match search criteria</p> <p>ENDIF</p>
--	--

Primitive Level Diagram	6.3 Search Guest Speaker
Process	6.3.6
Pseudo Code	<p>IF ("Update" button is clicked)</p> <p>THEN Display all the associated attributes : "Speaker_Name", "Speaker_Surname", "Phone Number", "PictureLink" from Guest Speaker Table</p> <p>ENDIF</p>

Primitive Level Diagram	6.3 Update Guest Speaker
Process	6.3.8
Pseudo Code	<p>IF (Update button is clicked)</p> <p>THEN Display Modal with message: "Are you sure you want to update Speaker_Name's information?"</p> <p>ENDIF</p>

Primitive Level Diagram	6.3 Update Guest Speaker
Process	6.3.10
Pseudo Code	<p>IF ("Speaker_Name" is string value less than 35 characters AND IS NOT null)</p> <p>THEN "Speaker_Name" IS valid</p> <p>ELSE "Speaker_Name" IS NOT valid</p> <p>IF ("Speaker_Surname" is string value less than 35 characters AND IS NOT null)</p> <p>THEN "Speaker_Surname" IS valid</p>

	<p>ELSE “Speaker_Surname” IS NOT valid</p> <p>IF (“Phone Number” is integer AND IS NOT null)</p> <p style="padding-left: 20px;">THEN “Phone Number” IS valid</p> <p>ELSE “Phone Number” IS NOT valid</p> <p>IF (“Email” is string value less than 255 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN “Email” IS valid</p> <p>ELSE “Email” IS NOT valid</p> <p>IF (“PictureLink” is string value less than 255 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN “PictureLink” IS valid</p> <p>ELSE “PictureLink” IS NOT valid</p>
--	--

Primitive Level Diagram	6.3 Update Guest Speaker
Process	6.3.11
Pseudo Code	<p>IF (“Speaker_Name” IS valid AND “Speaker_Surname” IS valid AND “Phone Number” IS valid AND “Email” IS valid AND “PictureLink” IS valid)</p> <p style="padding-left: 20px;">THEN</p> <pre>INSERT INTO Table Guest Speaker attributes(Speaker_Name, Speaker_Surname, Phone Number, PictureLink, SpeakerID) VALUES "Speaker_Name", "Speaker_Surname", "Phone Number", "PictureLink" SpeakerID where SpeakerID equals "SpeakerID"</pre> <p>ELSE Display Error Message</p>

Primitive Level Diagram	6.3 Update Guest Speaker
Process	6.3.12
Pseudo Code	<p>IF (Update is successful)</p> <p style="padding-left: 20px;">THEN Display Modal with message: “Congratulations! You have successfully created a new guest speaker!”</p>

ENDIF

Primitive Level Diagram	6.4 Delete Guest Speaker
Process	6.4.2
Pseudo Code	IF ("Guest Speakers" dropdown list item is clicked) THEN SHOW Guest Speaker Webpage ENDIF

Primitive Level Diagram	6.4 Delete Guest Speaker
Process	6.4.4
Pseudo Code	IF (Search by name textbox is not empty and Search button is clicked) THEN SELECT all Guest Speakers from Guest Speaker table. WHERE Speaker_Name is equal to text in the textbox THEN Display all Guest Speakers that match search criteria ENDIF

Primitive Level Diagram	6.4 Delete Guest Speaker
Process	6.4.6
Pseudo Code	IF ("Details" hyperlink is clicked) THEN Display all the associated attributes : "Speaker_Name", "Speaker_Surname", "Phone Number", "PictureLink" from Guest Speaker Table ENDIF

Primitive Level Diagram	6.4 Delete Guest Speaker
Process	6.4.8
Pseudo Code	IF (Delete button is clicked) THEN Display Modal with message: "Are you sure you want to update Speaker_Name's information?" ENDIF

Primitive Level Diagram	6.4 Delete Guest Speaker
Process	6.4.10
Pseudo Code	IF ("Confirm" button is clicked) THEN Delete from Table Guest Speaker attributes(Speaker_Name, Speaker_Surname, Phone Number, PictureLink, SpeakerID) VALUES "Speaker_Name", "Speaker_Surname", "Phone Number", "PictureLink" SpeakerID where SpeakerID equals "SpeakerID" ELSE Display Error Message

Primitive Level Diagram	6.4 Delete Guest Speaker
Process	6.4.11
Pseudo Code	IF (Update is successful) THEN Display Guest Speaker Page ENDIF

3.5.7 Content Subsystem Pseudo Code

Primitive Level Diagram	7.1 Upload Content
Process	7.1.2
Pseudo Code	IF (Reading Content Dropdown list item is clicked) THEN Display Dropdown List Items ENDIF

Primitive Level Diagram	7.1 Upload Content
Process	7.1.4
Pseudo Code	IF (Content Dropdown list item is clicked) THEN Display Content Management Page ENDIF

Primitive Level Diagram	7.1 Upload Content
Process	7.1.6
Pseudo Code	IF ("Upload Content" Button is clicked) THEN Create Display Content Page ENDIF

Primitive Level Diagram	7.1 Upload Content
Process	7.1.8
Pseudo Code	IF ("Content Type" is string value less than 35 characters AND IS NOT null) THEN "Content Type" IS valid ELSE "Content Type" IS NOT valid IF ("Name" is string value less than 35 characters AND IS NOT null) THEN "Name" IS valid ELSE "Name" IS NOT valid IF ("Author" is string value less than 70 characters AND IS NOT null) THEN "Author" IS valid

	<p>ELSE “ Author ” IS NOT valid</p> <p>IF (“ Date Published ” is valid date format - 10 characters in CCYY-MM-DD format AND IS NOT null)</p> <p style="padding-left: 20px;">THEN “ Date Published ” IS valid</p> <p>ELSE “ Date Published ” IS NOT valid</p> <p>IF (“ Content Link ” is string value less than 100 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN “ Content Link ” IS valid</p> <p>ELSE “ Content Link ” IS NOT valid</p> <p>IF (“Theme” is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN “Theme” IS valid</p> <p>ELSE “Theme” IS NOT valid</p> <p>IF (“Status” is binary AND IS NOT null)</p> <p style="padding-left: 20px;">THEN “Status” IS valid</p> <p>ELSE “Status” IS NOT valid</p> <p>IF (“Description” is string value less than 35 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN “Description” IS valid</p> <p>ELSE “Description” IS NOT valid</p>
--	---

Primitive Level Diagram	7.1 Upload Content
Process	7.1.8
Pseudo Code	<p>IF (“ Content Type ” IS valid AND “ Name ” IS valid AND “ Author ” IS valid AND “ Date Published ” IS valid AND “ Content Link ” IS valid AND “ Theme ” IS valid AND “ Status ” IS valid AND “ Description ” IS valid)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 40px;">INSERT INTO Table Content attributes(Content Type , Name , Author , Date Published , Content Link, Theme, Status)</p>

	<pre> , Description) VALUES "ContentType", "Name", "Author"," Date Published ","Content Link" , "Theme" , " Status " , " Description " ContentType where ContentType equals " ContentType " ELSE Display Error Message </pre>
--	---

Primitive Level Diagram	7.1 Upload Content
Process	7.1.10
Pseudo Code	IF (Insert is successful) THEN Display Modal with message: "Congratulations! You have uploaded Content!" ENDIF

Primitive Level Diagram	7.2 Search Content
Process	7.2.2
Pseudo Code	IF (Reading Content Dropdown list item is clicked) THEN Display Dropdown List Items ENDIF

Primitive Level Diagram	7.2 Search Content
Process	7.2.4
Pseudo Code	IF (Content Dropdown list item is clicked) THEN Display Content Management Page ENDIF

Primitive Level Diagram	7.2 Search Content
Process	7.2.6
Pseudo Code	IF (Content is Selected) THEN Display Relevant Content Page ENDIF

Primitive Level Diagram	7.3 Update Content
Process	7.3.2
Pseudo Code	IF (Reading Content Item clicked) THEN Display Relevant Content Page ENDIF
Primitive Level Diagram	7.3 Update Content
Process	7.3.4
Pseudo Code	IF (Update Button Clicked) THEN Display Confirmation Modal "Are you sure you want to update 'Content Name'. ENDIF

Primitive Level Diagram	7.3 Update Content
Process	7.3.6
Pseudo Code	IF ("Content Type" is string value less than 35 characters AND IS NOT null) THEN "Content Type" IS valid ELSE "Content Type" IS NOT valid IF ("Name" is string value less than 35 characters AND IS NOT null) THEN "Name" IS valid ELSE "Name" IS NOT valid IF ("Author" is string value less than 70 characters AND IS NOT null) THEN "Author" IS valid ELSE "Author" IS NOT valid <ul style="list-style-type: none"> • IF ("Date Published" is valid date format - 10 characters in CCYY-MM-DD format AND IS NOT null) THEN "Date Published" IS valid ELSE "Date Published" IS NOT valid IF ("Content Link" is string value less than 100 characters

	<p>AND IS NOT null)</p> <p>THEN “Content Link” IS valid</p> <p>ELSE “Content Link” IS NOT valid</p> <p>IF (“Theme” is string value less than 35 characters AND IS NOT null)</p> <p>THEN “Theme” IS valid</p> <p>ELSE “Theme” IS NOT valid</p> <p>IF (“Status” is binary AND IS NOT null)</p> <p>THEN “Status” IS valid</p> <p>ELSE “Status” IS NOT valid</p> <p>IF (“Description” is string value less than 35 characters AND IS NOT null)</p> <p>THEN “Description” IS valid</p> <p>ELSE “Description” IS NOT valid</p>
--	---

Primitive Level Diagram	7.3 Update Content
Process	7.3.7
Pseudo Code	<p>IF (“Content Type” IS valid AND “Name” IS valid AND “Author” IS valid AND “Date Published” IS valid AND “Content Link” IS valid AND “Theme” IS valid AND “Status” IS valid AND “Description” IS valid)</p> <p>THEN</p> <pre>INSERT INTO Table Content attributes(Content Type , Name , Author , Date Published , Content Link, Theme, Status , Description) VALUES “ContentType”, “Name”, “Author”, “Date Published ”,”Content Link” ,”Theme” ,” Status ”,” Description “ ContentType where ContentType equals “ContentType”</pre> <p>ELSE Display Error Message</p>

Primitive Level Diagram	7.3 Update Content
Process	7.3.8
Pseudo Code	IF (Update is successful) THEN Display Modal with message: "Congratulations! You have update Content!" ENDIF

Primitive Level Diagram	7.4 Delete Content
Process	7.4.2
Pseudo Code	IF (Reading Content Item clicked) THEN Display Relevant Content Page ENDIF

Primitive Level Diagram	7.4 Delete Content
Process	7.4.4
Pseudo Code	IF (Update Button Clicked) THEN Display Confirmation Modal "Are you sure you want to delete 'Content Name'?" ENDIF

Primitive Level Diagram	7.4 Delete Content
Process	7.4.6
Pseudo Code	IF (" Confirm" Button is Clicked) THEN Delete from Table Content attributes(Content Type , Name , Author , Date Published , Content Link, Theme, Status , Description) VALUES "ContentType", "Name", "Author", "Date Published ", "Content Link", "Theme", "Status", "Description" "ContentType where ContentType equals "ContentType "

	ELSE Display Error Message
--	----------------------------

Primitive Level Diagram	7.4 Delete Content
Process	7.4.7
Pseudo Code	IF (Delete is successful) THEN Display Modal with message: "You have Successfully delete Content!" ENDIF

Primitive Level Diagram	7.5 Review Lecture Content
Process	7.5.2
Pseudo Code	IF (Reading Content Dropdown list item is clicked) THEN Display Dropdown List Items ENDIF

Primitive Level Diagram	7.5 Review Lecture Content
Process	7.5.4
Pseudo Code	IF (Content Dropdown list item is clicked) THEN Display Content Management Page ENDIF

Primitive Level Diagram	7.5 Review Lecture Content
Process	7.5.6
Pseudo Code	IF ("Details" Button is clicked) THEN Display the selected Content Page ENDIF

Primitive Level Diagram	7.5 Review Lecture Content
Process	7.5.11
Pseudo Code	IF (Validation is successful) THEN Display Modal with message: "Are you sure you want to review this lecture content"

ENDIF

Primitive Level Diagram	7.5 Review Lecture Content
Process	7.1.13
Pseudo Code	<p>IF ("Review" IS valid AND "Rating" IS valid)</p> <p style="padding-left: 20px;">THEN</p> <p style="padding-left: 40px;">INSERT INTO Table Content attributes(RatingID , Rating)</p> <p style="padding-left: 40px;">VALUES "Review", "Rating" Rating where Rating equals "Rating"</p> <p>ELSE Display Error Message</p>

Primitive Level Diagram	7.5 Review Lecture Content
Process	7.5.10
Pseudo Code	<p>IF ("Review" is string value less than 300 characters AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Review" IS valid</p> <p style="padding-left: 20px;">ELSE "Review" IS NOT valid</p> <p>IF ("Rating" is integer AND IS NOT null)</p> <p style="padding-left: 20px;">THEN "Rating" IS valid</p> <p style="padding-left: 20px;">ELSE "Rating" IS NOT valid</p>

Primitive Level Diagram	7.5 Review Lecture Content
Process	7.5.14
Pseudo Code	<p>IF (Review is successful)</p> <p style="padding-left: 20px;">THEN Display Content Management Page</p> <p>ENDIF</p>

3.5.8 Marketing Subsystem Pseudo Code

Primitive Level Diagram	8.1 Upload Photo
Process	8.1.2
Pseudo Code	IF (Gallery Navigation Bar Item is clicked) THEN SHOW Gallery Webpage END

Primitive Level Diagram	8.1 Upload Photo
Process	8.1.4
Pseudo Code	IF (Browse Button is clicked) THEN Display File Dialog box END

Primitive Level Diagram	8.1 Upload Photo
Process	8.1.6
Pseudo Code	IF (Photo is selected) THEN Display Photo in PictureBox END

Primitive Level Diagram	8.1 Upload Photo
Process	8.1.8
Pseudo Code	IF (Upload button is clicked) THEN IF (Photo is in PNG OR SVG format) THEN Photo IS valid ELSE Photo IS INVALID THEN Display Modal with message: "This photo is not in a valid format" END

Primitive Level Diagram	8.1 Upload Photo
Process	8.1.9
Pseudo Code	IF (Photo is VALID) THEN Display Modal with message: "Are you sure you want to Upload this photo?" END

Primitive Level Diagram	8.1 Upload Photo
Process	8.1.10
Pseudo Code	IF (Confirm button is clicked) THEN Upload Photo to SERVER END

Primitive Level Diagram	8.1
Process	8.1.11
Pseudo Code	IF (Upload is successful) THEN Display Modal with message: "Your Photo has been successfully uploaded" END

Primitive Level Diagram	8.1 Upload Photo
Process	8.1.13
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Gallery Webpage END

Primitive Level Diagram	8.2 Post Photo
Process	8.2.2
Pseudo Code	IF (Gallery Navigation Bar Item is clicked) THEN SHOW Gallery Webpage END

Primitive Level Diagram	8.2 Post Photo
Process	8.2.4
Pseudo Code	THEN IF (Accept button is clicked) THEN Display Modal with message: "Are you sure you want to Post this photo?" END

Primitive Level Diagram	8.2 Post Photo
Process	8.2.5
Pseudo Code	THEN IF (Yes button is clicked) THEN ADD photo to gallery ELSE SHOW Display Gallery Webpage END

Primitive Level Diagram	8.2 Post Photo
Process	8.2.6
Pseudo Code	THEN IF (Photo ADD successful) THEN Display Modal with message: "You have successfully posted the photo to the gallery" END

Primitive Level Diagram	8.2 Post Photo
Process	8.2.8
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Gallery Webpage END

Primitive Level Diagram	8.3 Delete Photo
Process	8.3.2
Pseudo Code	IF (Gallery Navigation Bar Item is clicked) THEN SHOW Gallery Webpage

	END
--	------------

Primitive Level Diagram	8.3 Delete Photo
Process	8.3.4
Pseudo Code	IF (Delete button is clicked) THEN Display Modal with message: "Are you sure you want to delete this Photo?" END

Primitive Level Diagram	8.3 Delete Photo
Process	8.3.5
Pseudo Code	IF (Yes button is clicked) THEN DELETE photo from gallery and SERVER ELSE SHOW Gallery webpage END

Primitive Level Diagram	8.3 Delete Photo
Process	8.3.6
Pseudo Code	THEN IF (Photo ADD successful) THEN Display Modal with message: "You have successfully posted the photo to the gallery" END

Primitive Level Diagram	8.3 Delete Photo
Process	8.3.8
Pseudo Code	IF (Confirm button is clicked) THEN SHOW Gallery Webpage END

3.5.9 Report Subsystem Pseudo Code

Primitive Level Diagram	9.1 Generate Class Attendance Report
Process	9.1.2
Pseudo Code	IF (User Navigation Bar item is clicked) THEN Display User dropdown list options END

Primitive Level Diagram	9.1 Generate Class Attendance Report
Process	9.1.4
Pseudo Code	IF (Reports dropdown list item is clicked) THEN SHOW Reports webpage END

Primitive Level Diagram	9.1 Generate Class Attendance Report
Process	9.1.6
Pseudo Code	IF (Class Attendance tab is clicked) THEN READ Attendance table, Event table and Lecture. Calculate Attendance per Lecture. Generate Formatted Report. END

Primitive Level Diagram	9.1 Generate Class Attendance Report
Process	9.1.7
Pseudo Code	IF (Generate Formatted Report is complete) THEN SHOW Class Attendance Report END

Primitive Level Diagram	9.2 Generate Function Attendance Report
Process	9.2.2
Pseudo Code	IF (User Navigation Bar item is clicked) THEN Display User dropdown list options END

Primitive Level Diagram	9.2 Generate Function Attendance Report
Process	9.2.4
Pseudo Code	IF (Reports dropdown list item is clicked) THEN SHOW Reports webpage END

Primitive Level Diagram	9.2 Generate Function Attendance Report
Process	9.2.6
Pseudo Code	IF (Function Attendance tab is clicked) THEN READ Attendance table, Event table and Function. Calculate Attendance per Function. Generate Formatted Report. END

Primitive Level Diagram	9.2 Generate Function Attendance Report
Process	9.2.7
Pseudo Code	IF (Generate Formatted Report is complete) THEN SHOW Function Attendance Report END

Primitive Level Diagram	9.3 Generate Community Engagement Attendance Report
Process	9.3.2
Pseudo Code	IF (User Navigation Bar item is clicked) THEN Display User dropdown list options END

Primitive Level Diagram	9.3 Generate Community Engagement Attendance Report
Process	9.3.4
Pseudo Code	IF (Reports dropdown list item is clicked) THEN SHOW Reports webpage END

Primitive Level Diagram	9.3 Generate Community Engagement Attendance Report
Process	9.3.6
Pseudo Code	IF (Community Engagement tab is clicked) THEN READ Attendance table, Event table and CommunityOutreach. Calculate Attendance per Community Engagement. Generate Formatted Report. END

Primitive Level Diagram	9.3 Generate Community Engagement Attendance Report
Process	9.3.7
Pseudo Code	IF (Generate Formatted Report is complete) THEN SHOW Community Engagement Report END

Primitive Level Diagram	9.4 Generate Demographics Report
Process	9.4.2
Pseudo Code	IF (User Navigation Bar item is clicked) THEN Display User dropdown list options END

Primitive Level Diagram	9.4 Generate Demographics report
Process	9.4.4
Pseudo Code	IF (Reports dropdown list item is clicked) THEN SHOW Reports webpage END

Primitive Level Diagram	9.4 Generate Demographics Report
Process	9.4.6
Pseudo Code	IF (Demographics tab is clicked) THEN READ Person table. Calculate Number of Students per Race. Generate Formatted Report. END

Primitive Level Diagram	9.4 Generate Demographics Report
Process	9.4.7
Pseudo Code	IF (Generate Formatted Report is complete) THEN SHOW Demographics Report END

Primitive Level Diagram	9.5 Generate Event Popularity Report
Process	9.5.2
Pseudo Code	IF (User Navigation Bar item is clicked) THEN Display User dropdown list options END

Primitive Level Diagram	9.5 Generate Event Popularity Report
Process	9.5.4
Pseudo Code	IF (Reports dropdown list item is clicked) THEN SHOW Reports webpage END

Primitive Level Diagram	9.5 Generate Event Popularity Report
Process	9.5.6
Pseudo Code	IF (Event Popularity tab is clicked) THEN READ Attendance table, Event table, Function table, Lecture table and CommunityOutreach. Calculate Attendance per Community Engagement. Generate Formatted Report. END

Primitive Level Diagram	9.5 Generate Event Popularity Report
Process	9.5.7
Pseudo Code	IF (Generate Formatted Report is complete) THEN SHOW Event Popularity Report END

Primitive Level Diagram	9.6 Generate Feedback Report
Process	9.6.2
Pseudo Code	IF (User Navigation Bar item is clicked) THEN Display User dropdown list options END

Primitive Level Diagram	9.6 Generate Feedback Report
Process	9.6.4
Pseudo Code	IF (Reports dropdown list item is clicked) THEN SHOW Reports webpage END

Primitive Level Diagram	9.6 Generate Feedback Report
Process	9.6.6
Pseudo Code	IF (Feedback tab is clicked) THEN READ VolunteerFeedback table. Average Feedback Number per Assessment. Generate Formatted Report. END

Primitive Level Diagram	9.6 Generate Feedback Report
Process	9.6.7
Pseudo Code	IF (Generate Formatted Report is complete) THEN SHOW Feedback Report END

Primitive Level Diagram	9.7 Generate User Statistics Report
Process	9.7.2
Pseudo Code	IF (User Navigation Bar item is clicked) THEN Display User dropdown list options END

Primitive Level Diagram	9.7 Generate User Statistics Report
Process	9.7.4
Pseudo Code	IF (Reports dropdown list item is clicked) THEN SHOW Reports webpage END

Primitive Level Diagram	9.7 Generate User Statistics Report
Process	9.7.6
Pseudo Code	IF (User Statistics tab is clicked) THEN READ Person table. Calculate Number of Times a user logged on. Generate Formatted Report. END

Primitive Level Diagram	9.7 Generate User Statistics Report
Process	9.7.7
Pseudo Code	IF (Generate Formatted Report is complete) THEN SHOW User Statistics Report END

Primitive Level Diagram	9.8 Generate Student Progress Report
Process	9.8.2
Pseudo Code	IF (User Navigation Bar item is clicked) THEN Display User dropdown list options END

Primitive Level Diagram	9.8 Generate Student Progress Report
Process	9.8.4
Pseudo Code	IF (Reports dropdown list item is clicked) THEN SHOW Reports webpage END

Primitive Level Diagram	9.8 Generate Student Progress Report
Process	9.8.6
Pseudo Code	IF (User Statistics tab is clicked) THEN READ Student table, StudentMilestone table and Attendance table Calculate Progress for each student. Generate Formatted Report. END

Primitive Level Diagram	9.8 Generate Student Progress Report
Process	9.8.7
Pseudo Code	IF (Generate Formatted Report is complete) THEN SHOW Student Progress Report END

3.6 Conclusion

3.6.1 The process models of the TRWLA system were illustrated above. This included the context diagram which described how the external actors interacted directly with the TRWLA system. The decomposition diagram describes the hierarchical order that flows throughout the system from the TRWLA system level, function level, activity level to task level. The data flow diagrams were then illustrated at the end which described every process within the TRWLA system as well as their corresponding pseudo code.

4. Entity Relationship Model

4.1 Introduction

4.1.1 This section encompasses the database design and data modelling of a technical data model. The group designed a normalized entity relationship diagram to represent the database that would need to be implemented to create the proposed system. The group also applied referential integrity to the data model to apply certain restrictions with regards to updating and deletion.

4.2 Entity Relationship Diagram

[See Next Page]

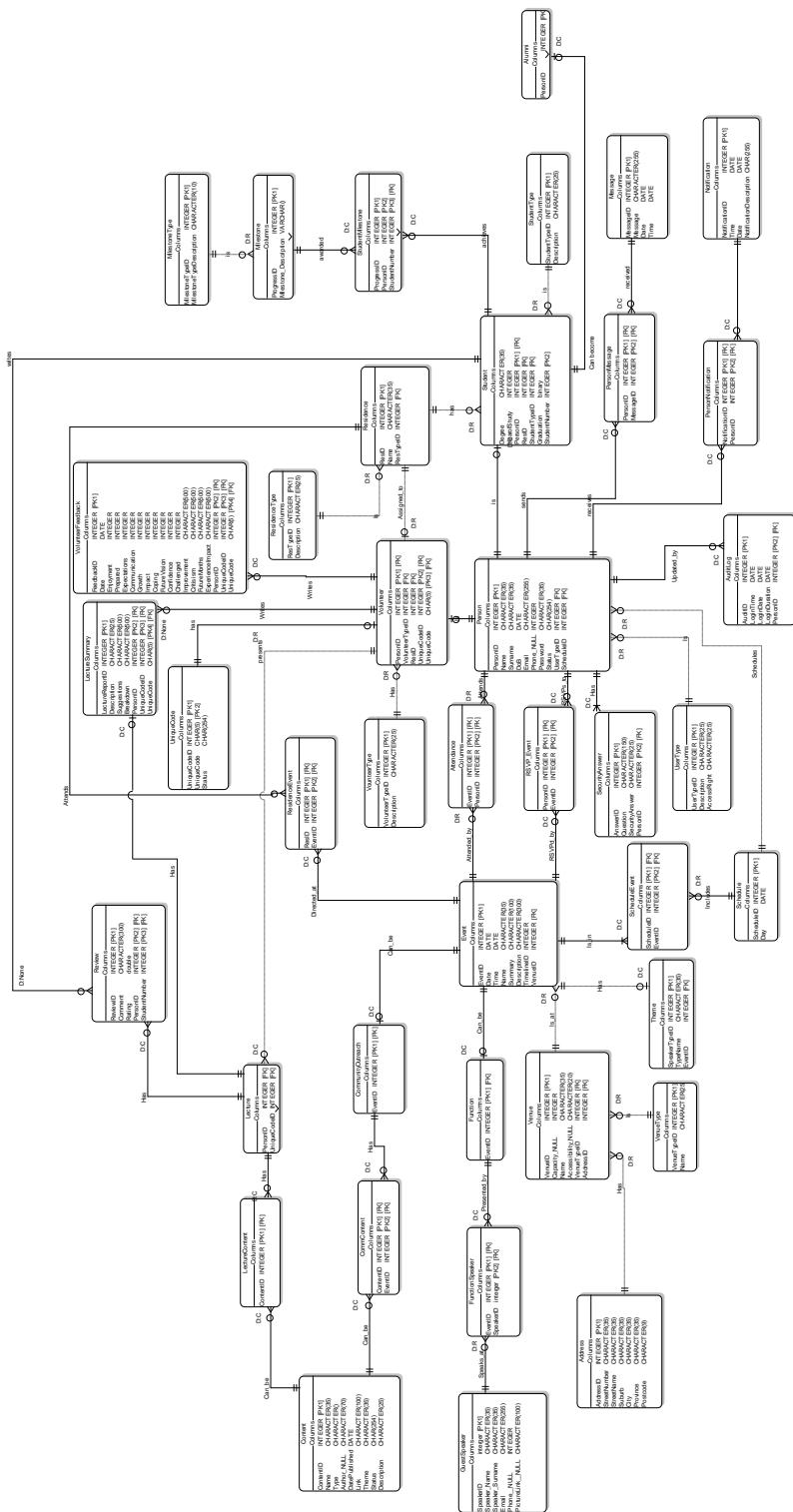


Figure 183: Physical ERD

4.3 Conclusion

4.3.1 This section encompassed the database design and data modelling of a technical data model. The group designed a normalized entity relationship diagram with referential to represent the database that would need to be implemented to create the proposed system.

5. Size Estimation of Proposed Database

5.1 Introduction

5.1.1 This section encompasses the estimation of the size of the database as well as the five year projection. This enables the group to produce a rough estimation as to how much space is needed for the organisation's database before it is actually developed. The five year projection also helps the group determine how much the organisation is going to grow and if the database can grow as the organisation does.

5.2 Size Estimation

Size Estimation of Proposed Database				
Entity Number	Entity Name	Expected Rows	Entity Size (In bytes)	Total Size (In bytes)
1	Guest Speaker	20	852	17040
2	Content	24	1540	36960
3	Address	10	369	3690
4	FunctionSpeaker	200	2	400
5	LectureContent	16	2	32
6	CommContent	8	2	16
7	VenueType	10	51	510
8	Venue	10	114	1140
9	Function	10	1	10
10	CommunityOutreach	10	1	10
11	Lecture	75	1	75
12	Theme	1125	70	78750
13	Schedule	591	1	591
14	SchedleEvent	56145	2	112290
15	Event	95	1089	103455
16	Review	41250	611	25203750
17	UserType	3	101	303
18	SecurityAnswer	1770	352	623040
19	RSVP_Event	52250	2	104500

20	Attendance	4583	2	9166
21	VolunteerType	5	51	255
22	ResidenceEvent	1125	2	2250
23	AuditLog	88000	26	2288000
24	Person	880	832	732160
24	Volunteer	40	3	120
26	UniqueCode	40	12	480
27	LectureSummary	75	2052	153900
28	Alumni	290	1	290
29	PersonNotification	472000	2	944000
30	PersonMessage	800	2	1600
31	Student	550	75	41250
32	Residence	15	72	1080
33	ResidenceType	2	51	102
34	VolunteerFeedback	40	4020	160800
35	Notification	472000	17	8024000
36	Message	800	527	421600
37	StudentType	4	51	204
38	StudentMilestone	2750	2	5500
39	Milestone	5	71	355
40	MilestoneType	3	51	153
			Total:	39073827

5.3 Five Year Projection

Growth Estimation of the TRWLA Database		
Year	Expected Growth (In %) x100	Expected Size (In Bytes)
Current (2017)	N/A	39073827
2018	0,1	42981209,7
2019	0,1	47279330,67
2020	0,1	52007263,74
2021	3	208029054,9
2022	0,1	228831960,4

Not all tables will increase, only a few tables will such as events, messages, functions, lectures, community engagements and so on and thus the table will only see a 10% increase. Students who graduate will not be removed but will be placed in the Alumni table thus the Alumni table will grow as well. Therefore the ERD in total will increase only by 10%.

TRWLA stipulated that they would like to expand in the future and thus in the year 2021 this is accounted for with a 300% increase in the size of the database to accommodate the students from across multiple universities.

5.4 Conclusion

5.4.1 This section encompassed the estimated size of the database as well as the five year projection which refers to how much the organisation is going to grow within the next five years and to ensure that the database grows with the organisation.

6. Input Design

6.1 Introduction

6.1.1 The following section gives a complete description of the interface and other inputs for each requirement of the system based on the high-fidelity prototypes with a description of every possible action the user might have with the screens of the system.

6.2 Prototype Screens



Figure 184: Navigation Bar for Students

Number	Component Name	Functionality
1	Home navigation bar item	When clicked this displays the students home webpage
2	Volunteers navigation bar item	When clicked this displays the Volunteer s webpage
3	Reading Content navigation bar item	This displays a dropdown list with the following items: Lecture, Community Outreach and E-library
4	Gallery navigation bar item	This displays the Gallery page.



Figure 185: Navigation Bar for Volunteers: Members Dropdown List

Number	Component Name	Functionality
1	Members dropdown list	This contains four items, Students, Volunteers, Alumni and Send notification. The students item will display Screen 3.2.1 Search Student. The students item will display Screen 3.2.1 Search Student. The volunteers item will display Screen 2.2.1 Search Volunteer. The Alumni item will display Screen 1.9.1 Search Alumni. The send notification item will display Screen 5.9.1 Send Notification

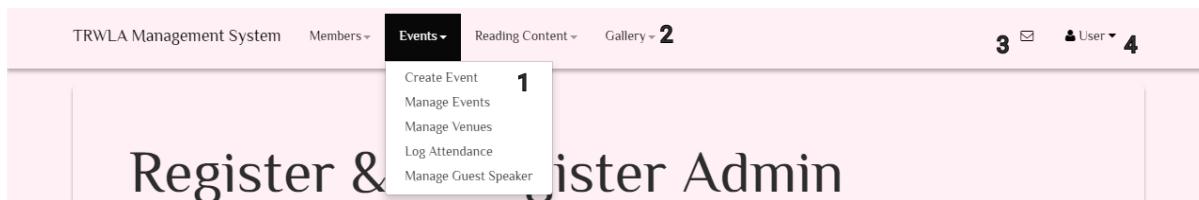


Figure 186: Navigation Bar for Volunteers: Events Dropdown List

Number	Component Name	Functionality
1	Events dropdown list	This contains five items, Create Event, Manage Events, Manage Venues, Log Attendance and Manage Guest Speaker. The create event item will display Screen 5.0.2 Events Home VA. The manage events item will display Screen 5.0.1 Events Home VA. The manage venues item will display Screen Venue. The log attendance item will display Screen 5.8.1 Log Attendance. The manage guest speaker item will display Screen Guest Speaker Home.
2	Gallery dropdown list	Displays the gallery page.
3	Notification icon	Displays user's notifications
4	User dropdown list item	Displays user dropdown list items

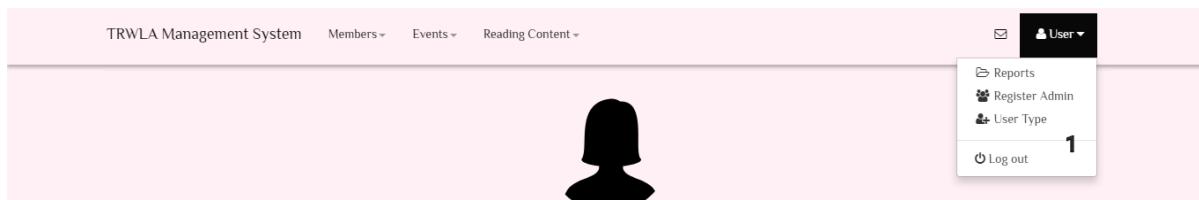
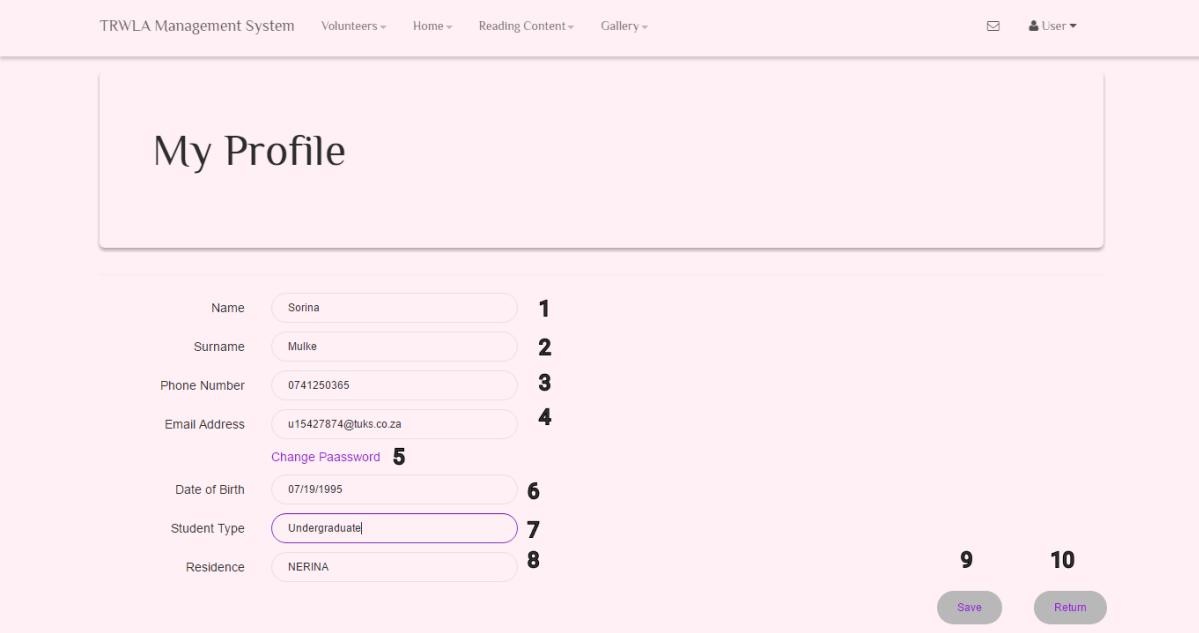


Figure 187: Navigation Bar for Volunteers: User Dropdown List

Number	Component Name	Functionality
1	User dropdown list	When clicked on the items display the following: Reports displays Screen Reports Home. Register Admin displays Screen Register Admin. User Type displays Search User Type Logout displays Screen Logout



The screenshot shows the 'My Profile' page of the TRWLA Management System. At the top, there is a navigation bar with links for Volunteers, Home, Reading Content, and Gallery. On the far right, there are user-related icons. The main content area has a title 'My Profile'. Below the title is a form with the following fields and their corresponding numbers:

- Name: Sorina (1)
- Surname: Mulke (2)
- Phone Number: 0741250365 (3)
- Email Address: u15427874@tuks.co.za (4)
- Change Paassword (5)
- Date of Birth: 07/19/1995 (6)
- Student Type: Undergraduate (7)
- Residence: NERINA (8)
- Save (9)
- Return (10)

Figure 188: General Profile page

Number	Component Name	Functionality
1	Name textbox	The system displays the user's name in an editable textbox
2	Surname textbox	The system displays the user's surname in an editable textbox
3	Phone Number textbox	The system displays the user's phone number in an editable textbox
4	Email address textbox	The system displays the user's email address in an editable textbox

Number	Component Name	Functionality
5	Change password hyperlink	The user clicks on the hyperlink if they want to change their password and will be directed to Screen 1.2.2 change password.
6	Date of birth picker	The system displays the user's date of birth
7	Student type dropdown list	The dropdown list contains a list of student types for the user to choose from.
8	Residence textbox	The system displays the user's residence in an editable textbox
9	Save button	The user clicks on the save button when they want to save any changes made to their profile, this system will display a warning modal.
10	Return Button	The user clicks on this to return to their home page.

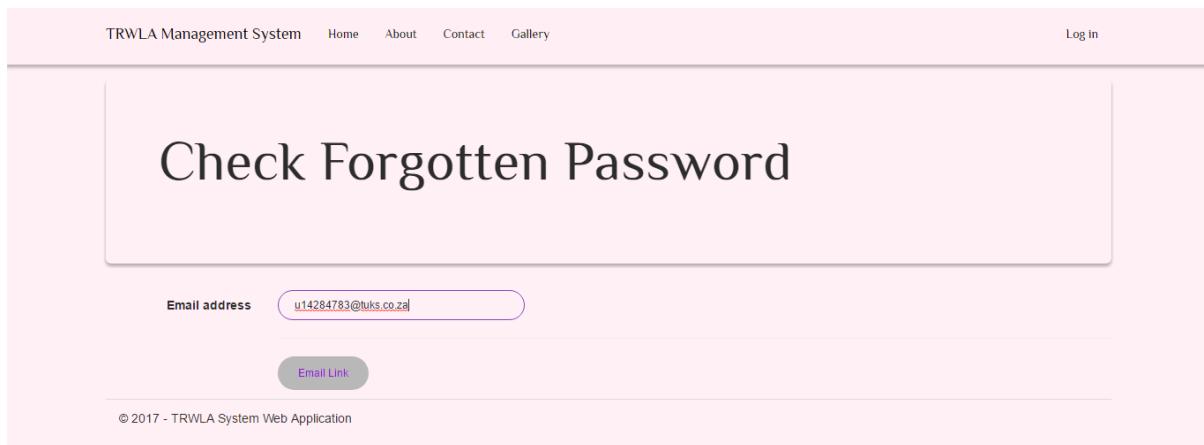


Figure 189: 1.1.1 Check Forgotten Password

TRWLA Management System Home About Contact Gallery Log in

Check Forgotten Password

Email address 1

2

Please answer the following security question.

What was the name of your third grade teacher?
 3

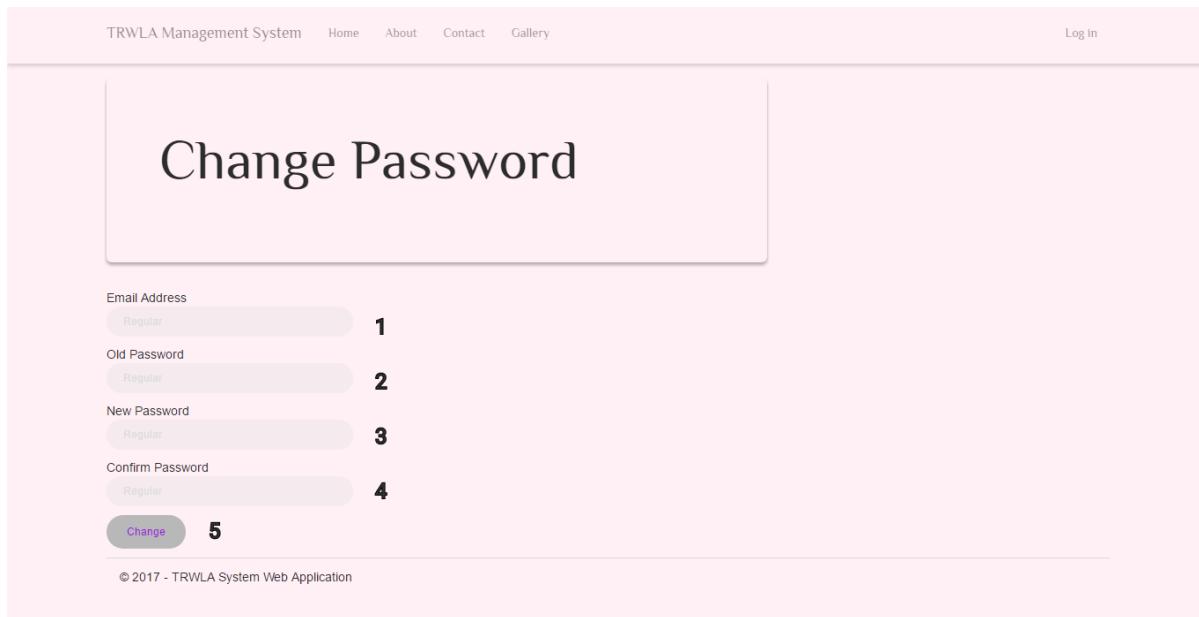
4

© 2017 - TRWLA System Web Application

Figure 190: 1.1.2 Check Forgotten Password

Number	Component Name	Functionality
1	Email Textbox	The user enters their email address that they wish the email link should be sent to.
2	Email link Button	Once clicked, the system will email the to the entered to get back into the system. As well as verify that the user's email address is in the system.
3	Security Question Textbox	The user enters the answer to their security question from the one they used to sign up.
4	Change password button	Once clicked, the user's the answer will be validated against the one in the

		database. If the validation is successful it will load the Change password screen.
--	--	--



The screenshot shows the 'Change Password' page of the TRWLA Management System. The page has a pink header with the system name and navigation links. The main content area is white and contains a title 'Change Password' and five input fields. Each field has a placeholder 'Regular' and is labeled with a number from 1 to 4. A 'Change' button is located at the bottom left. The footer contains a copyright notice.

Figure 191: 1.2.2 Change Password

Number	Component Name	Functionality
1	Email textbox	The user will enter their email address linked to their account.
2	Old password textbox	The user will enter their current password.
3	New password textbox	The user will enter a new password
4	Confirm Password textbox	The user will re-enter their new password.
5	Change button	When this button is clicked the system will display a warning modal to confirm the change.

TRWLA Management System Home About Contact Gallery Log in

Log in

Use a local account to log in.

Email 1

Password 2

Remember me? 3

4



TUKSRES WOMEN IN
LEADERSHIP ACADEMY

© 2017 - TRWLA System Web Application

Figure 192: 1.3.1 Login

Number	Component Name	Functionality
1	Email textbox	The user will enter their email address
2	Password textbox	The user will enter their password
3	Remember me? checkbox	The user will check this checkbox if they want to stay logged into the system after they have closed their web browser.
4	Log in button	The user will click this button to login to the system, the system will then validate their information and display their homepage or display a viewbag error messages.
5	Register as a new user hyperlink	A user will click this to register on the system for the first time.
6	Forgot password hyperlink	A user will click this if they have forgotten their password and the system will display Screen 1.1.2 Check Forgotten Password.

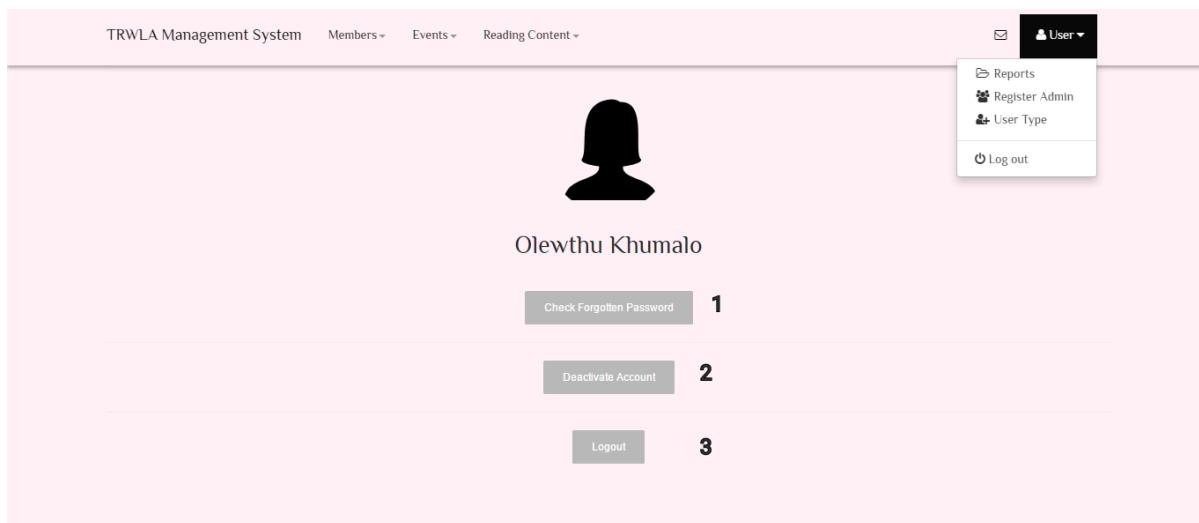
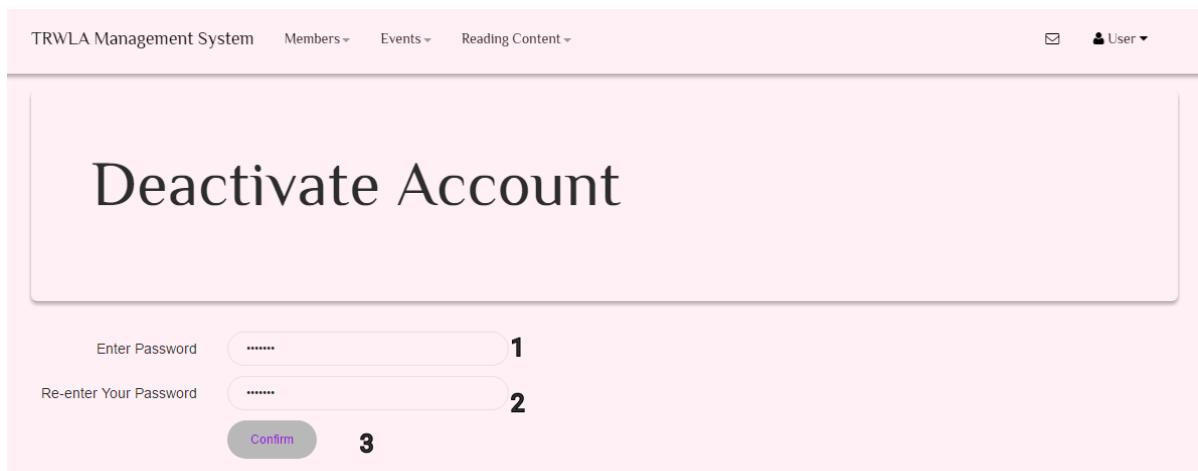


Figure 193: 1.4.1 Logout

Number	Component Name	Functionality
1	Check Forgotten Password button	The user clicks this button to change their password, the system will display Screen 1.2.2 Change Password.
2	Deactivate Account button	The user clicks this button to deactivate their account, the system will display Screen 1.5.2 Deactivate Account.
3	Logout Button	The user clicks this to logout of the system, the system will display Screen 1.3.1 Login



TRWLA Management System Members ▾ Events ▾ Reading Content ▾

User

Deactivate Account

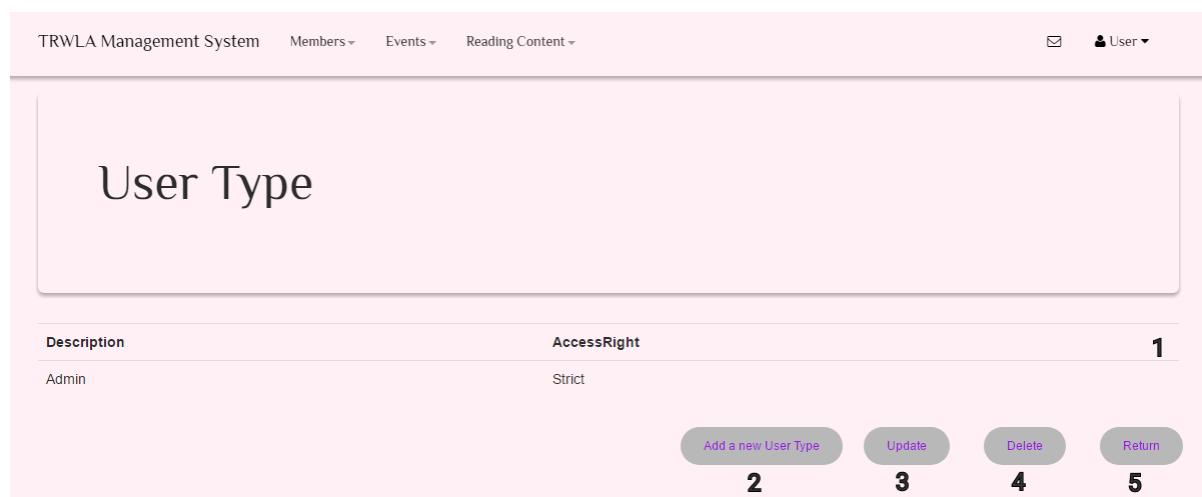
Enter Password 1

Re-enter Your Password 2

Confirm 3

Figure 194: 1.5.2 Deactivate Account

Number	Component Name	Functionality
1	Enter Password textbox	The user will enter their password.
2	Re-enter Your Password textbox	The user will re-enter their password.
3	Confirm Button	The user clicks this button to deregister, the system will display a warning modal to confirm this action.



Description	AccessRight	1
Admin	Strict	2

Add a new User Type
Update
Delete
Return

Figure 195: 1.6.1 Create User type

Number	Component Name	Functionality
1	Table	This table contains the <u>UserType</u> attributes
2	Add a new User Type button	Admin clicks this button to add a new user type, the system displays Screen 1.6.2 Create User Type
3	Update button	Admin clicks this to update a specific user type, the system will display Screen 1.7.1 Update User Type
4	Delete button	Admin clicks this to delete a specific user type, the system will display Screen 1.8.1 Delete User Type
5	Return button	Admin clicks this button to return to their Home Page.

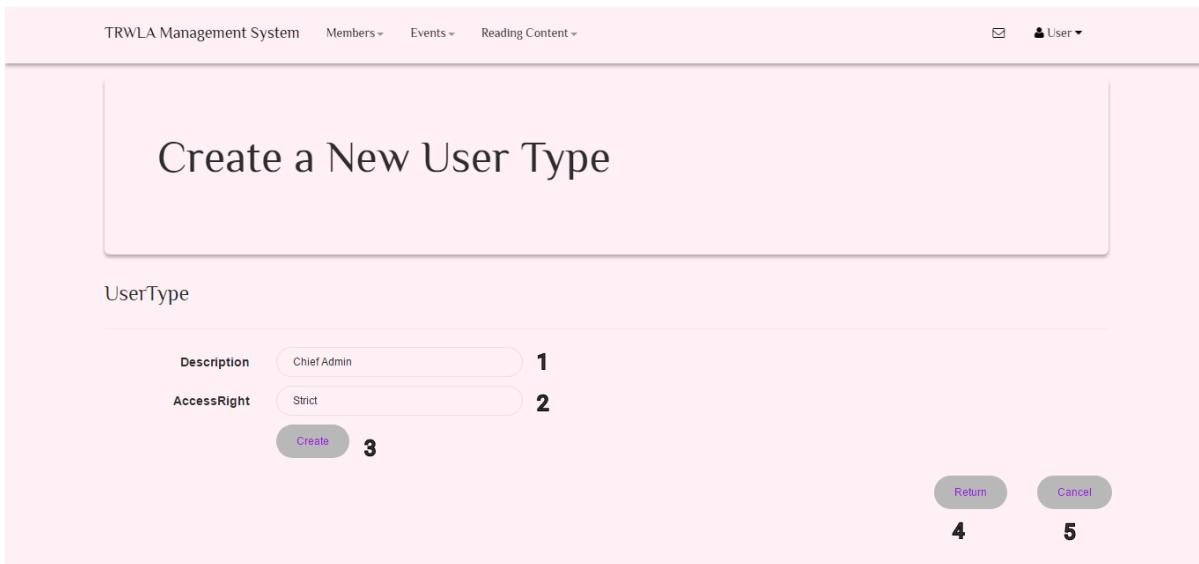


Figure 196: 1.6.2 Create User Type

Number	Component Name	Functionality
1	Description textbox	Admin will enter a user type description
2	Access Right textbox	Admin will enter an access right
3	Create button	Admin clicks this to create a user type, the system will display a warning modal.
4	Return Button	Admin clicks this to return to Screen 1.6.1 Create User Type
5	Cancel Button	Admin clicks this to cancel the creation process and return to Screen 1.6.1 Create User Type

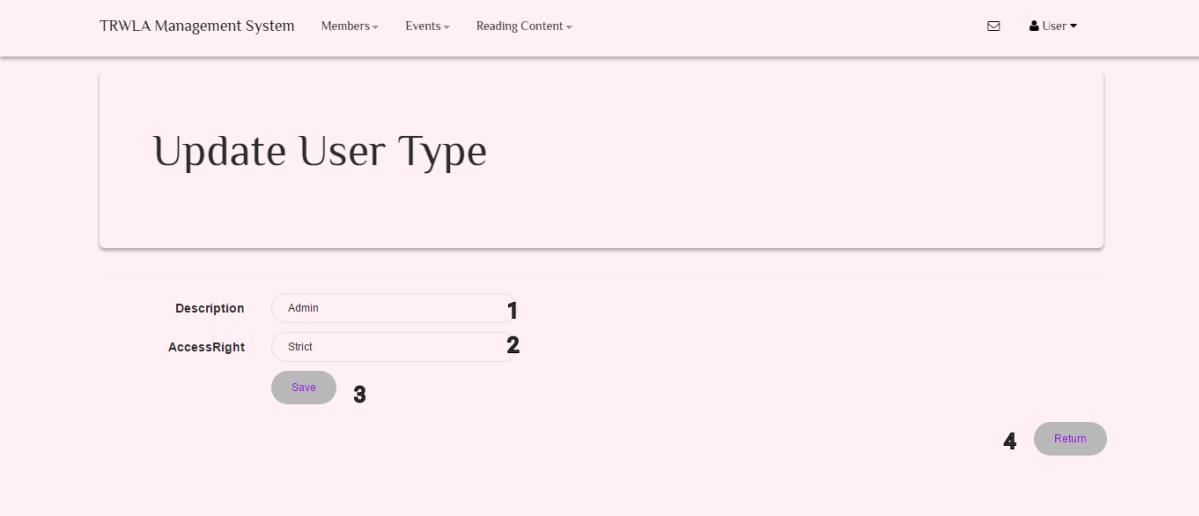


Figure 197: 1.7.1 Update User Type

Number	Component Name	Functionality
1	Description textbox	The system will display the user type description in an editable textbox
2	Access Right textbox	The system will display the access right in an editable textbox
3	Save button	Admin clicks this button to update the information, the system will display a warning modal to confirm.
4	Return button	Admin clicks this to return to Screen 1.6.1 Create User Type

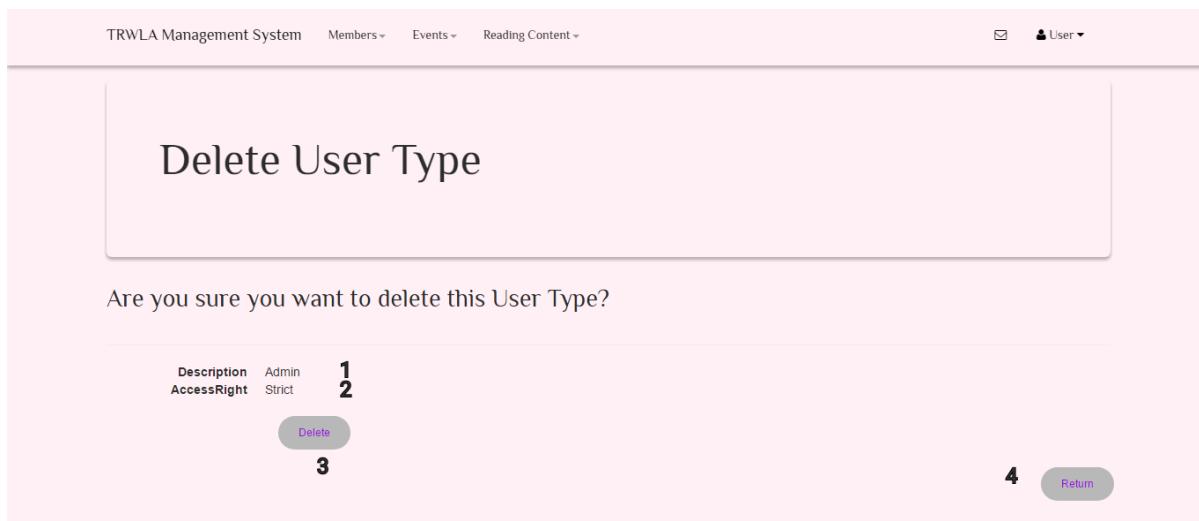


Figure 198: 1.8.1 Delete User Type

Number	Component Name	Functionality
1	Description textbox	The system displays the user type description
2	Access Right textbox	The system displays the Access Right Description
3	Delete button	Admin clicks this button to delete a user type, the system displays a warning modal.
4	Return Button	Admin clicks this to return to Screen 1.6.1 Create User Type

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ [✉](#) [User ▾](#)

Alumni

You can search for students who have graduated from the Tuks Women In Res Leadership Academy

1	Find by name: <input type="text"/>	2	Search	3
FirstName	Surname	Phone Number	Email Address	
Ciara	Mash	0147258786	u14284795@tuks.co.za	
Una	De Burger	07895847	u14254786@tuks.co.za	
Sinqo	Renwille	0174536985	u18789654@tuks.co.za	
Martina	Khiosi	0214785963	u103658@tuks.co.za	
Ellie	Martuns	0612587963	u10285478@tuks.co.za	

Figure 199: 1.9.1 Search Alumni

Number	Component Name	Functionality
1	Find by name textbox	Used to enter search information
2	Search button	This is clicked to search through the table below
3	Table	The table is populated by the <u>Alumni</u> and <u>Person</u> tables

TRWLA Management System Home About Contact Gallery Log in

TuksRes Women in Leadership Academy

Background

The TuksRes Woman in Leadership Academy is a non-profit organisation based on campus of the University of Pretoria which aims to equip young women with the personal, professional and leadership skills necessary to thrive in their respective aspirations. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc quis condimentum purus, nec commodo nisi. Proin consectetur, dui id facilisis vehicula, lorem ante convallis arcu, quis consequat eros ipsum ac ex. Donec volutpat ut augue a molestie. Nam porttitor dictum accumsan.

3 Core Pillars

Service
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc quis condimentum purus, nec commodo nisi. Proin consectetur, dui id facilisis vehicula, lorem ante convallis arcu, quis consequat eros ipsum ac ex. Donec volutpat ut augue a molestie. Nam porttitor dictum accumsan.

Skills
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc quis condimentum purus, nec commodo nisi. Proin consectetur, dui id facilisis vehicula, lorem ante convallis arcu, quis consequat eros ipsum ac ex. Donec volutpat ut augue a molestie. Nam porttitor dictum accumsan.

Networking
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc quis condimentum purus, nec commodo nisi. Proin consectetur, dui id facilisis vehicula, lorem ante convallis arcu, quis consequat eros ipsum ac ex. Donec volutpat ut augue a molestie. Nam porttitor dictum accumsan.

[!\[\]\(d5a74c101f1c40edd4f59f57ed7f4b77_img.jpg\)](#) [!\[\]\(52f6bca1c7d280a1883c92f820f21a95_img.jpg\)](#) [!\[\]\(2b2112dbc057aba51d5174ab91d9e9e7_img.jpg\)](#) [!\[\]\(5bd4ef0954b0d0dd3a0e1a04a8aa0372_img.jpg\)](#) [!\[\]\(dc3f4b3361d6f996732a5b580b725a8f_img.jpg\)](#)

© 2017 - TRWLA System Web Application



TUKSRES WOMEN IN LEADERSHIP ACADEMY

Figure 200: Home page

[TRWLA Management System](#) [Home](#) [About](#) [Contact](#) [Gallery](#)[Log in](#)

About

Service

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Skills

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Networking

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

© 2017 - TRWLA System Web Application

Figure 201: About Page

[TRWLA Management System](#) [Home](#) [About](#) [Contact](#) [Gallery](#)[Log in](#)

Contact.

Duxbury Palace
Elandspoort 357-Jr, Pretoria, 0002

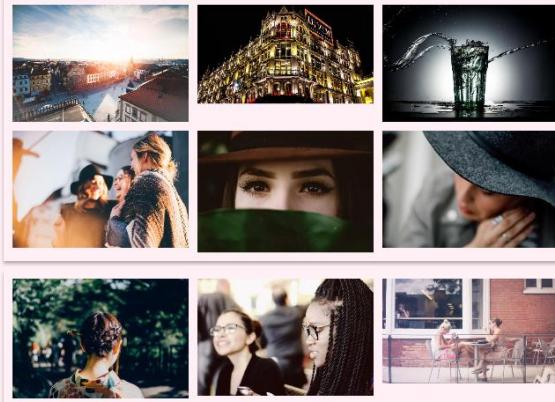
Email: office@trwla.co.za

© 2017 - TRWLA System Web Application

Figure 202: Contact Page

TRWLA Management System [Home](#) [About](#) [Contact](#) [Gallery](#)[Log in](#)

Gallery



© 2017 - TRWLA System Web Application

Figure 203: Gallery Page

TRWLA Management System Home About Contact Gallery Log In

Register.

Create a new account

Email 1
Password 2
Confirm password 3
Register 4

© 2017 - TRWLA System Web Application

Figure 204: 2.1 Register

Number	Component Name	Functionality
1	Email textbox	The user enters their email address
2	Password textbox	The user enters a password
3	Confirm password textbox	The user will re-enter their password
4	Register Button	The user clicks on this button to register as a user, the system will validate the information, if the information is valid a modal will be displayed for the user to choose if they are student or a volunteer.

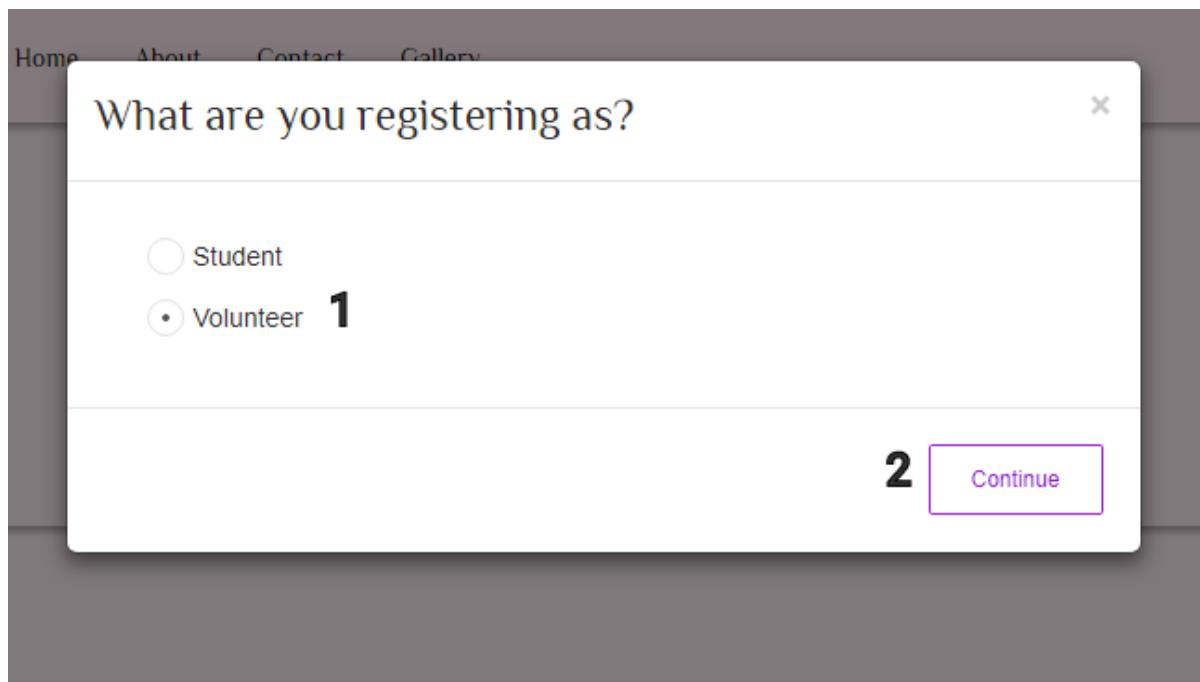


Figure 205: 2.1.1 Register Volunteer

Number	Component Name	Functionality
1	Student and Volunteer Radio buttons	The user registering will select one of the radio buttons.
2	Continue Button	When this button is clicked the system will display the enter unique code modal or Add Student webpage according to the radiobutton clicked.

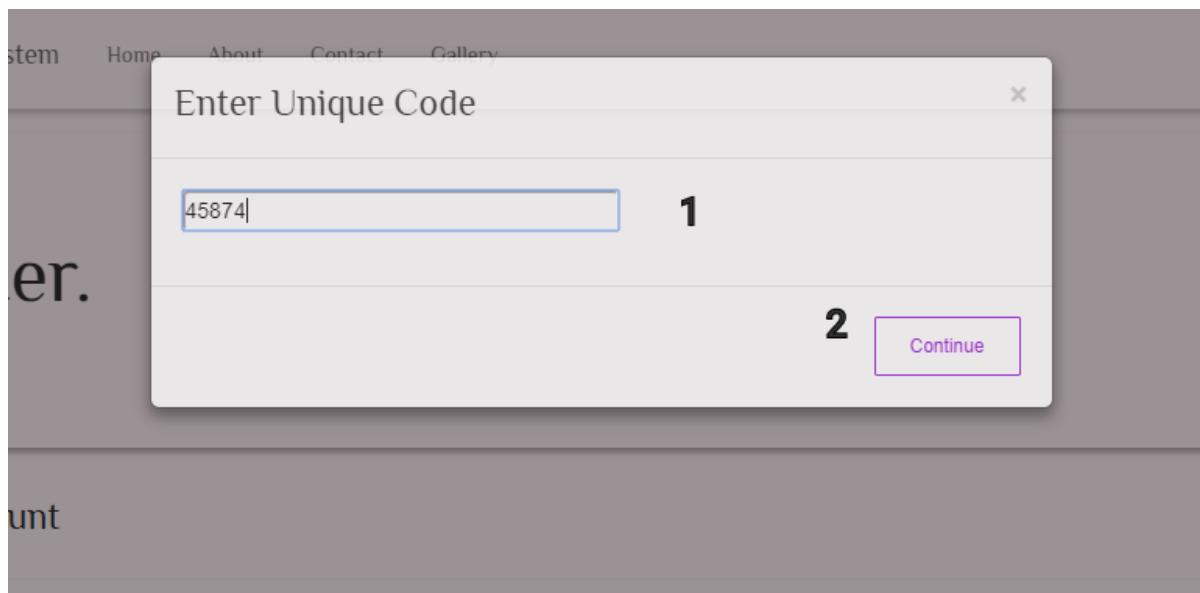


Figure 206: 2.1.2 Register Volunteer

Number	Component Name	Functionality
1	Unique code textbox	The volunteer enters the unique code assigned to them.
2	Continue button	When this button is clicked the system validates the unique code with the UniqueCode table and displays the Add Volunteer webpage.

TRWLA Management System Home About Contact Gallery Log In

Register As A Volunteer

Register

Name	<input type="text" value="Michelle"/>	1
Surname	<input type="text" value="Alexander"/>	2
Phone Number	<input type="text" value="098765432"/>	3
Email Address	<input type="text" value="u1398746@tuks.co.za"/>	4
Date of Birth	<input type="text" value="10/05/2016"/>	5
Security Question	<input type="text" value="First Church?"/>	6
Security Question Answer	<input type="text" value="Elim"/>	7
Volunteer Type	<input type="text" value="Admin"/>	8
Residence	<input type="text" value="Nerina"/>	9
		10 11

[Create](#) [Return](#)

© 2017 - TRWLA System Web Application

Figure 207: 2.1.3 Register Volunteer

Number	Component Name	Functionality
1	Name Textbox	The user will enter their name.
2	Surname Textbox	The user will enter their surname.
3	Phone Number Textbox	The user will enter their phone number.
4	Email Address Textbox:	The user will enter their email address.
5	Date of Birth Date picker:	The user will enter their date of birth.
6	Security Question Textbox	The user will enter their security question.

Number	Component Name	Functionality
7	Security Question Answer Textbox	The user will enter their security question answer.
8	Volunteer Type dropdown	The user will select their volunteer type.
9	Residence dropdown	The User will select their residence.
10	Register Button	The system will validate that the details are unique and will then insert the data from the textboxes in the Person and Volunteer table. While also the person table being updated.
11	Cancel Button	Will cancel the registration for the Student.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ [✉](#) [User ▾](#)

Volunteers

1 Find by name: **2**

Name	Surnmae	Phone Number	Email Address	Date of Birth	Volunteer Type	Residence	3
Cheryl	Wyngardd	078475321	cheryl@twrla.co.za	10/09/1998	Admin	Nerina	
Sarah	Lawler	07894258963	u1428963@twrla.co.za	10/09/1994	Facilitator	Asterhof	
Mavius	Khekna	08745896	u1489632@twrla.co.za	10/10/1996	Facilitator	Tuks Naledi	
Sultte	Drawbridge	0258747965	u14578749@twrla.co.za	10/09/1992	Facilitator	Hatfield Studios	
Suzane	Helderblom	0796874521	u14287931@twrla.co.za	11/10/1998	Admin	Lilium	

4 **5** **6**

Figure 208: 2.2.1 Search Volunteer

Number	Component Name	Functionality
1	Search textbox	A textbox whereby the user can enter their search parameters.
2	Search Button	Once the button is clicked, a LINQ query will execute that will match the search string with an records found inside the person and volunteer table.
3	Table	The columns of the table inherit attributes from the <u>Person</u> and <u>Volunteer Table</u>
4	Update button	A button where a user can update a volunteer's information.
5	Delete button	A button whereby the user can delete a volunteer.
6	Return Button	The system will display the user's home page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ [✉](#) [User ▾](#)

My Profile

Name	<input type="text" value="Cheryl"/>	1
Surname	<input type="text" value="Wyngaard"/>	2
Phone Number	<input type="text" value="0741258963"/>	3
Email Address	<input type="text" value="u14284789@tuks.co.za"/>	4
Date of Birth	<input type="text" value="10/05/2016"/>	5
Volunteer Type	<input type="text" value="Facilitator"/>	6
Residence	<input type="text" value="NERINA"/>	7

8 **9** **10**

Figure 209: 2.3.1 Update Volunteer

Number	Component Name	Functionality
1	Name textbox	The system displays the user's name in an editable textbox
2	Surname textbox	The system displays the user's surname in an editable textbox
3	Phone Number textbox	The system displays the user's phone number in an editable textbox
4	Email address textbox	The system displays the user's email address in an editable textbox
5	Date picker	The system displays the user's date of birth
6	Volunteer type dropdown	The dropdown list contains a list of student types for the user to choose from.
7	Residence dropdown	The system displays the user's residence in an editable textbox

Number	Component Name	Functionality
8	Save button	The user clicks on the save button when they want to save any changes made to their profile, this system will display a warning modal.
9	Return Button	The user clicks on this to return to their home page.
10	Delete Button	The user clicks on this button to delete a volunteer

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ [✉](#) User ▾

Delete Volunteer

Name	Cheryl	1
Surname	Wyngaard	2
Phone Number	0741258639	3
Email Address	u14875978@tuks.co.za	4
Date of Birth	10/17/1991	5
Volunteer Type	Facilitator	6
Residence	NERINA	7

[Return](#) [Delete](#)

Figure 210: 2.4.1 Delete Volunteer

Number	Component Name	Functionality
1	Name textbox	The system displays the user's name in an editable textbox
2	Surname textbox	The system displays the user's surname in an editable textbox
3	Phone Number textbox	The system displays the user's phone number in an editable textbox
4	Email address textbox	The system displays the user's email address in an editable textbox
5	Date picker	The system displays the user's date of birth
6	Volunteer type dropdown	The dropdown list contains a list of student types for the user to choose from.
7	Residence dropdown	The system displays the user's residence in an editable textbox

Number	Component Name	Functionality
8	Return Button	The user clicks on this to return to their home page.
9	Delete Button	The user clicks on this button to delete a volunteer

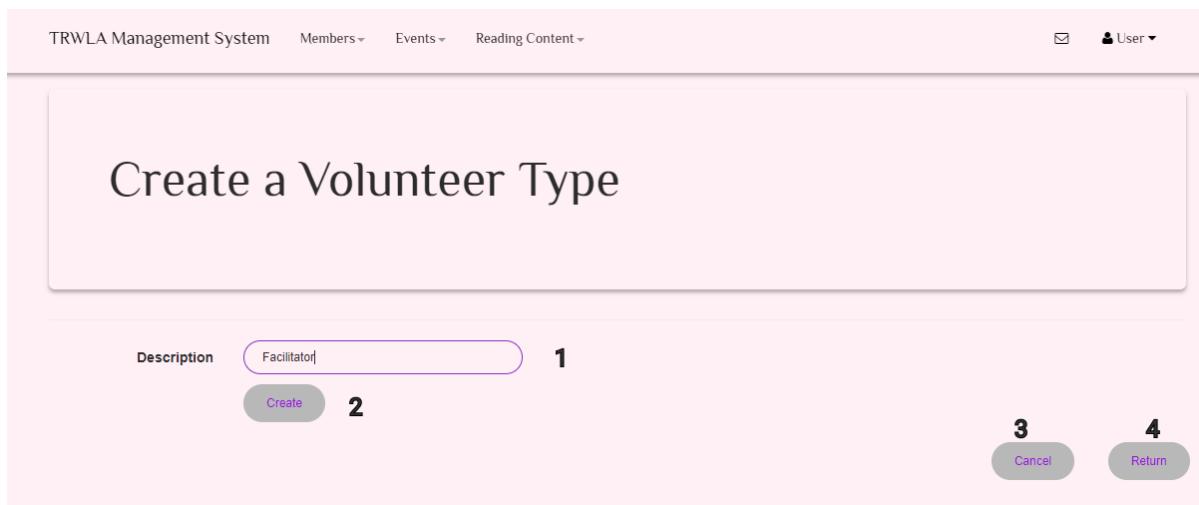
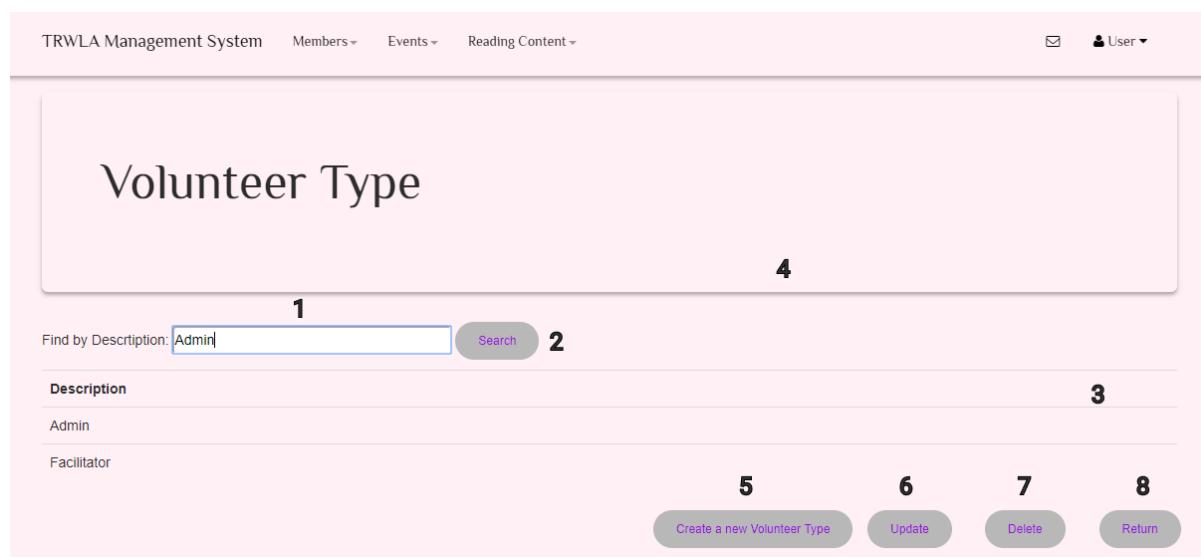


Figure 211: 2.5.2 Create Volunteer Type

Number	Component Name	Functionality
1	Description textbox	Admin will enter a volunteer type description
2	Create button	Admin clicks this to create a volunteer type, the system will display a warning modal.
3	Cancel Button	Admin clicks this to cancel the creation process and return to Screen Search Volunteer Type
4	Return Button	Admin clicks this to return to Screen Search Volunteer Type



TRWLA Management System Members ▾ Events ▾ Reading Content ▾

User ▾

Volunteer Type

Find by Description: Admin

Description	3
Admin	
Facilitator	

4

1 **2**

5 **6** **7** **8**

5 **6** **7** **8**

Figure 212: 2.6.1 Search Volunteer Type

Number	Component Name	Functionality
1	Search textbox	A textbox whereby the user can enter their search parameters.
2	Search Button	once the button is clicked, a LINQ query will execute that will match the search string with any records found inside the volunteer type table.
3	Table	The columns inherit the attributes of the Volunteer Type table
4	Container	Container containing a header entitled “Volunteer Type”
5	Create a new volunteer type button	Will redirect the user to add volunteer type page.
6	Update button	will allow user to update volunteer type by redirecting them to update volunteer type page.
7	Delete button	will allow user to update volunteer type by redirecting them to delete volunteer type page.
8	Return button	Will take the user back to the main menu page.

1 Description Admin

2 Save

3 Cancel

4 Return

Figure 213: 2.7.1 Update Volunteer Type

Number	Component Name	Functionality
1	Description textbox	The system will display the volunteer type description in an editable textbox
2	Save button	Admin clicks this button to update the information, the system will display a warning modal to confirm.
3	Cancel button	Admin clicks this to cancel the update and the system displays to Screen Search Volunteer Type
4	Return Button	Admin clicks this to return to Screen Search Volunteer Type

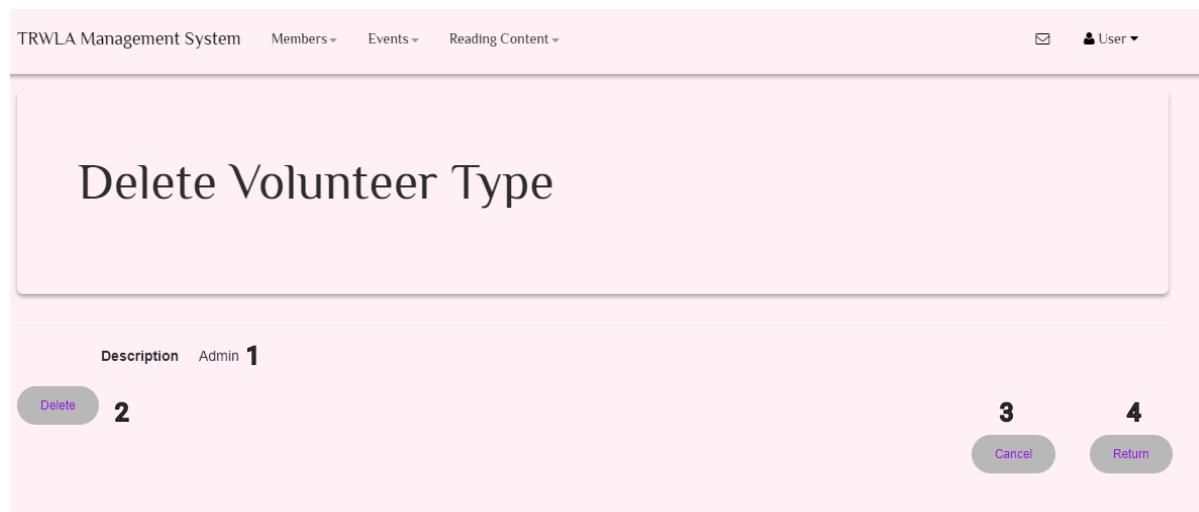
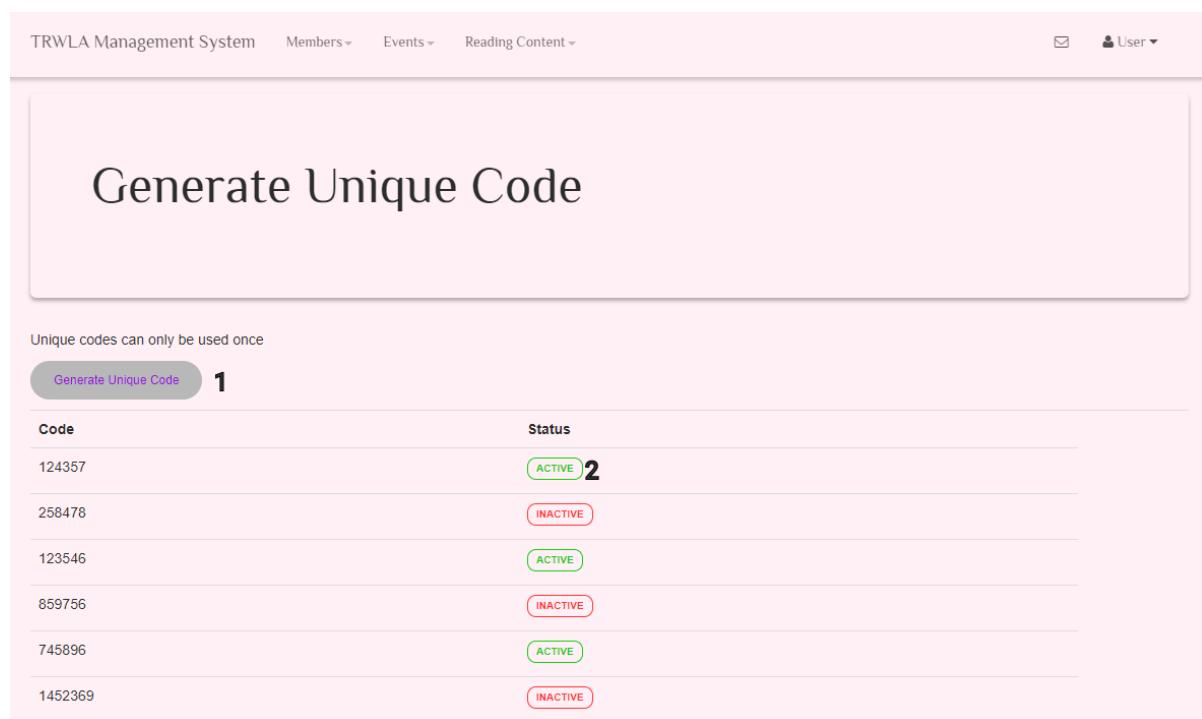


Figure 214: 2.8.1 Delete Volunteer Type

Number	Component Name	Functionality
1	Description textbox	The system will retrieve the description from the volunteer type table.
2	Delete Button	Once clicked the system will delete the existing record of the description from the volunteer type table.
3	Cancel button	Will cancel the updating of a volunteer type.
4	Return button	Will take the user back to volunteer Type page.

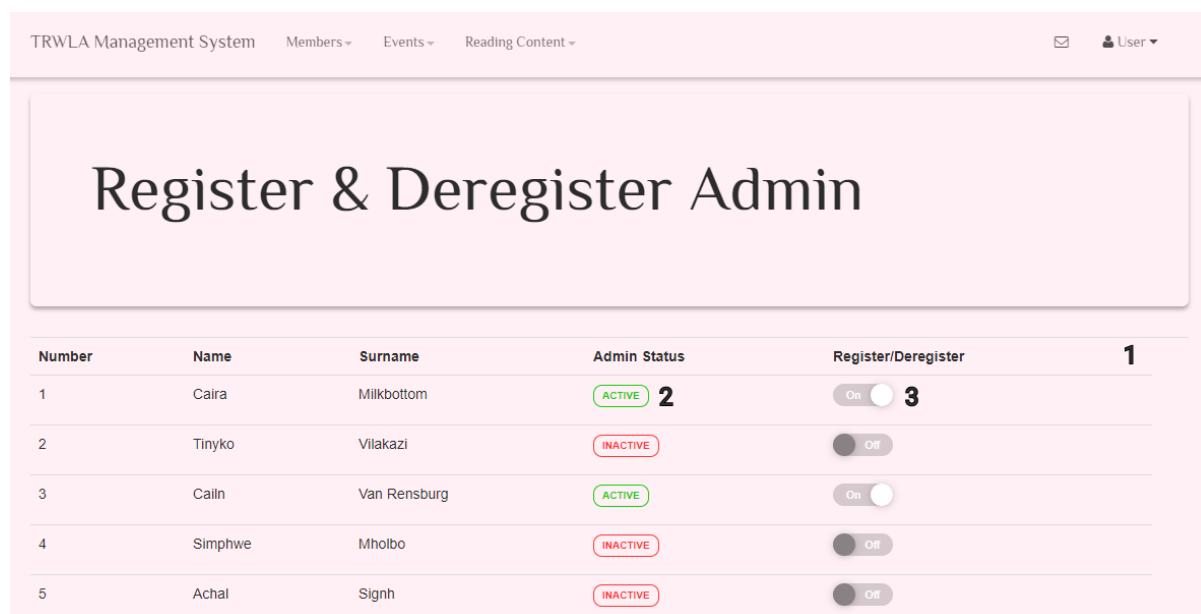


Unique codes can only be used once

Code	Status
124357	ACTIVE 2
258478	INACTIVE
123546	ACTIVE
859756	INACTIVE
745896	ACTIVE
1452369	INACTIVE

Figure 215: 2.9 Generate Unique Code

Number	Component Name	Functionality
1	Generate Unique Code Button	Once clicked the system will generate a unique five digit code that can only be used once.
2	Table	Column with the generate code and another for the status of the code if it has been used or expired. Active reflects that it has been issued and not yet user. Inactive reflects that the code has expired or been used.



Number	Name	Surname	Admin Status	Register/Deregister	1
1	Caira	Milkbottom	ACTIVE 2	<input checked="" type="checkbox"/> On <input type="checkbox"/> Off	
2	Tinyko	Vilakazi	INACTIVE	<input type="checkbox"/> Off	
3	Cailn	Van Rensburg	ACTIVE	<input checked="" type="checkbox"/> On <input type="checkbox"/> Off	
4	Simpewe	Mholbo	INACTIVE	<input type="checkbox"/> Off	
5	Achal	Signh	INACTIVE	<input type="checkbox"/> Off	

Figure 216: 2.10 and 2.11 Register and Deregister Admin

Number	Component Name	Functionality
1	Table	The table includes columns: Number, the name and surnames of volunteers from the volunteer table, admin status which indicates whether the volunteer is an administrator and deregister/deregister
2	Status Label	Once the toggle switch state changes, the label will either change to active or inactive.
3	Register/Deregister toggle switch	On is a binary value that provides the registered volunteers access to admin rights. While off is also a binary value that revokes admin rights from the volunteer.

TRWLA Management System Home About Contact Gallery Log In

Register.

Create a new account

Email 1
 Password 2
 Confirm password 3
 4

© 2017 - TRWLA System Web Application

Figure 217: 3.1.1 Register Student

Number	Component Name	Functionality
1	Email Textbox	The user will enter their email address here.
2	Password Textbox	The user will enter their password here.
3	Confirm Password Textbox	The user will re-enter their password here.
4	Register Button	Once clicked the system will temporarily store the data from the textboxes in its current state. And then open a modal.

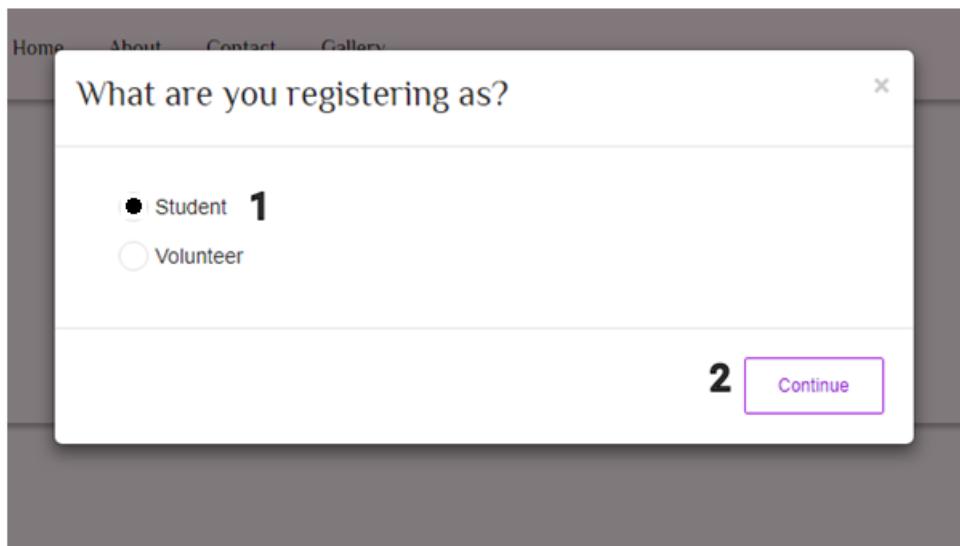


Figure 218: 3.1.2 Register Student

Number	Component Name	Functionality
1	Student Radio Button	Once selected will let the system identify what the user is registering as.
2	Continue Button	Once clicked this will load another modal based on the selected radio button.

TRWLA Management System Home About Contact Gallery Log in

Register as a New Student

Register

Name	Sinaqo	1
Surname	Maquezo	2
Phone Number	0713456786	3
Email Address	u12345678@tuks.co.za	4
Date of Birth	10/05/2016	5
Security Question	What is your favorite colour?	6
Security Question Answer	Pink	7
Student Type	Undergraduate	8
Residence	Nerina	9
		10
		11

10 **11**

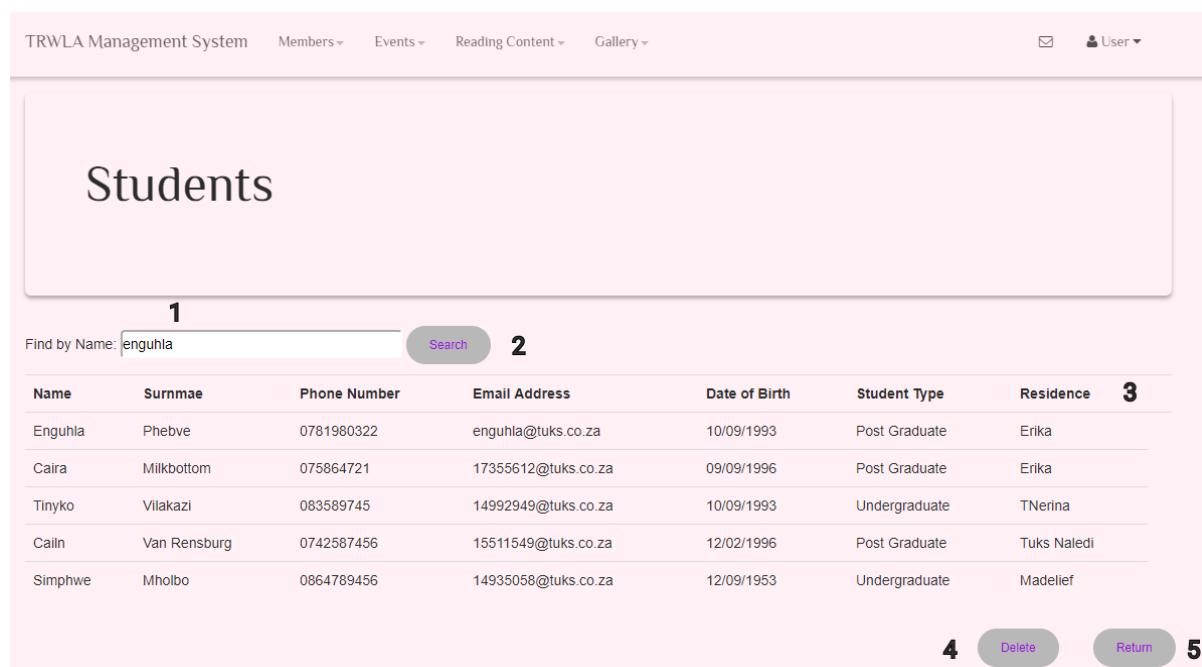
Register **Return**

© 2017 - TRWLA System Web Application

Figure 219: 3.1.3 Register Student

Number	Component Name	Functionality
1	Name Textbox	The user will enter their name.
2	Surname Textbox	The user will enter their surname.
3	Phone Number Textbox	The user will enter their phone number.
4	Email Address Textbox:	The user will enter their email address.
5	Date of Birth Date picker:	The user will enter their date of birth.
6	Security Question Textbox	The user will enter their security question.
7	Security Question Answer Textbox	The user will enter their security question answer.

8	Student Type dropdown	The user will select their volunteer type.
9	Residence dropdown	The User will select their residence.
10	Register Button	The system will validate that the details are unique and will then insert the data from the textboxes in the Student table. While also the person table being updated.
11	Cancel Button	Will cancel the registration for the student.



Find by Name: **1**

2

Name	Surname	Phone Number	Email Address	Date of Birth	Student Type	Residence	3
Enguhla	Phebve	0781980322	enguhla@tuks.co.za	10/09/1993	Post Graduate	Erika	
Caira	Milkbottom	075864721	17355612@tuks.co.za	09/09/1996	Post Graduate	Erika	
Tinyko	Vilakazi	083589745	14992949@tuks.co.za	10/09/1993	Undergraduate	TNerina	
Cailn	Van Rensburg	0742587456	15511549@tuks.co.za	12/02/1996	Post Graduate	Tuks Naledi	
Simpahwe	Mholbo	0864789456	14935058@tuks.co.za	12/09/1953	Undergraduate	Madelief	

4 **Delete** **5** **Return**

Figure 220: 3.2.1 Search Student

Number	Component Name	Functionality
1	Search textbox	A textbox whereby the user can enter their search parameters.
2	Search Button	once the button is clicked, a linq query will execute that will match the search string with an records found inside the student table.
3	Table	The columns of the table inherit attributes from the <u>Student Table</u>
4	Delete button	A textbox whereby the user can delete a student.
5	Return Button	The system will display the user's home page.

TRWLA Management System Home ▾ Volunteer ▾ Reading Content ▾ Gallery

User ▾

My Profile

Name	Rachel	1
Surname	Adams	2
Phone Number	0748963214	3
Email Address	u14789834@tuks.co.za	4
Date of Birth	10/05/2016	5
Student Type	Post Graduate	6
Residence	NERINA	7

8 **9** **10**

Save Delete Return

Figure 221: 3.3.1 Update Student

Number	Component Name	Functionality
1	Name Textbox	The user's name attribute retrieved from the <u>Student</u> table.
2	Surname Textbox	The user's surname attribute retrieved from the <u>Student</u> table.
3	Phone Number Textbox	The user's phone number attribute retrieved from the <u>Student</u> table.
4	Email Address Textbox	The user's email address attribute retrieved from the <u>Student</u> table.
5	Date of Birth Date picker	The user's date of birth attribute retrieved from the <u>Student</u> table.
6	Student Type dropdown	The user's Student type attribute retrieved from the <u>Student</u> table.
7	Residence dropdown	The user's residence attribute retrieved from the <u>Student</u> table.
8	Save Button	Will allow the student to save their Details.
9	Delete Button	Once clicked the system will Then will delete the existing record of the in the Student table.

10

Return Button

Once clicked it will take the user back to their previous screen.

The screenshot shows the 'My Profile' section of the TRWLA Management System. It contains the following fields:

- Name: Rachel (callout 1)
- Surname: Adams (callout 2)
- Phone Number: 0748963214 (callout 3)
- Email Address: u14789834@tuks.co.za (callout 4)
- Date of Birth: 10/05/2016 (callout 5)
- Student Type: Post Graduate (callout 6)
- Residence: NERINA (callout 7)

At the bottom right are three buttons:

- Save (callout 8)
- Delete (callout 9)
- Return (callout 10)

Figure 222: 3.4.1 Delete Student

Number	Component Name	Functionality
1	Name Textbox	The user's name attribute retrieved from the <u>Student</u> table.
2	Surname Textbox	The user's surname attribute retrieved from the <u>Student</u> table.
3	Phone Number Textbox	The user's phone number attribute retrieved from the <u>Student</u> table.
4	Email Address Textbox	The user's email address attribute retrieved from the <u>Student</u> table.
5	Date of Birth Date picker	The user's date of birth attribute retrieved from the <u>Student</u> table.
6	Student Type dropdown	The user's Student type attribute retrieved from the <u>Student</u> table.
7	Residence dropdown	The user's residence attribute retrieved from the <u>Student</u> table.
8	Save Button	Will allow the student to save their Details.
9	Delete Button	Once clicked the system will Then will delete the existing record of the in the Student table.

10

Return Button

Once clicked it will take the user back to their previous screen.

You can generate a list of graduates in the academy that will qualify based on the requirements you have set.

FirstName	Surname	Phone Number	Email Address	2
Clara	Mash	0147258786	u14284795@tuks.co.za	
Una	De Burger	07895847	u14254786@tuks.co.za	
Sinquo	Renwillie	0174536985	u18789654@tuks.co.za	
Martina	Khosi	0214785963	u103658@tuks.co.za	
Ellie	Martuns	0612587963	u10285478@tuks.co.za	

Figure 223: 3.5 Generate Graduate List

Number	Component Name	Functionality
1	Generate Graduate List button	Once clicked the system will execute a linq query from the database to gather students who have passed in the student table and then put them in the graduate table.
2	Table	The table will be filled with a for loop that parse data from the graduate table and displays its attributes as columns: First Name, Surname, Phone Number and Email Address.

Figure 224: 3.6.2 Add Student Type

Number	Component Name	Functionality
1	Description textbox	The user will provide a description of the new student type they are creating.
2	Create Button	Once clicked the system will check that it does not exist, will validate the length of the description according to 25 characters. Then will insert the description into the student type table.
3	Return button	Will take the user back to create student Type page.
4	Cancel button	Will cancel the creation of a student type.

The screenshot shows the 'Student Type' search interface. At the top, there is a navigation bar with links for 'TRWLA Management System', 'Members', 'Events', 'Reading Content', and 'Gallery'. On the right, there are icons for email and user profile, with a dropdown menu for 'User'. Below the navigation is a search bar with a placeholder 'Find by name:' and a 'Search' button. To the right of the search bar is a table with two rows of data. The first row contains 'Undergraduate' and the second row contains 'Post Graduate'. At the bottom of the table are seven numbered buttons: 1 (Search textbox), 2 (Search button), 3 (Table), 4 (Add new student type button), 5 (Update button), 6 (Delete button), and 7 (Return button).

Figure 225: 3.7 Search Student Type

Number	Component Name	Functionality
1	Search textbox	A textbox whereby the user can enter their search parameters.
2	Search Button	once the button is clicked, a linq query will execute that will match the search string with an records found inside the student type table.
3	Table	The columns inherit the attributes of the Student Type table .
4	Add new student type button	Will redirect the user to add student type type page.
5	Update button	will allow user to update student type by redirecting them to update student type page.
6	Delete button	will allow user to update student type by redirecting them to delete student type page.
7	Return button	Will take the user back to the main menu page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery ▾

User ▾

Update a Student Type

Student Type

Description	Undergraduate	1
Save	2	3
		4
	Cancel	Return

Figure 226: 3.8.1 Update Student Type

Number	Component Name	Functionality
1	Description textbox	The system will retrieve the description from the Student type table. And display this data in the textbox field.
2	Delete Button	Once clicked the system will delete the existing record of the description from the Student type table.
3	Return button	Will take the user back to Student Type page.
4	Cancel button	Will cancel the updating of a volunteer type.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery ▾ [✉](#) [User ▾](#)

Delete Student Type

StudentType

Description Undergraduate **1**

2

3 **4**

[Delete](#) [Cancel](#) [Return](#)

Figure 227: 3.9.1 Delete Student Type

Number	Component Name	Functionality
1	Description textbox	The system will retrieve the description from the Student type table. And display this data in the textbox field.
2	Delete Button	Once clicked the system will delete the existing record of the description from the Student type table.
3	Return button	Will take the user back to Student Type page.
4	Cancel button	Will cancel the updating of a volunteer type.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery ▾ [✉](#) [User ▾](#)

Add a new Venue

Name	<input type="text" value="Aula"/>	1
Capacity	<input type="text" value="350"/>	2
Accessibility	<input type="text" value="Wheelchair Access"/>	3
Street Number	<input type="text" value="350 Prospect Street"/>	4
Suburb	<input type="text" value="Hartfield"/>	5
Province	<input type="text" value="Gauteng"/>	6
Venue Type	<input type="text" value="Indoor"/>	7

8 **9** **10**

Number	Component Name	Functionality
1	Name Textbox	The user will enter the venue name
2	Capacity Textbox	The user will enter the venue capacity.
3	Accessibility Textbox	The user will enter the accessibility of the venue.
4	Street Number Textbox:	The user will enter the venues street number.
5	Suburb Textbox	The user will enter the suburb textbox
6	Province Dropdown	The user will choose the province of the venue
7	Venue Type dropdown	The user will select the venue type .
8	Create Button	Will execute a LINQ Query to insert the record into the <u>Venue</u> Table
9	Return Button	Will redirect the user to the Venue page.
10	Cancel Button	Will cancel the addition of a venue.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery ▾ [✉](#) [User ▾](#)

Venues

1 Find by name: **2**

Name	Capacity	Accessibility	Street Number	Suburb	Province	Venue Type	3
Roos Hall	150	None	254 Lunnon Road	Hillcrest	Gauteng	Indoor	
Tuks Monate	200	Highly accessible	350 Lunnon Road	Hillcrest	Gauteng	Indoor	
Ampitheatre	500	None	350 Lunnon Road	Hatfield	Gauteng	Outdoor	
LC De Villers	1000	Wheelchair access	254 Lunnon Road	Hillcrest	Gauteng	Multipurpose	
Erica Rec Hall	200	Minimal	254 Lunnon Road	Hatfield	Gauteng	Indoor	

4 [Add a New Venue](#) **5** [Update](#) **6** [Delete](#) **7** [Return](#)

Number	Component Name	Functionality
1	Search textbox	A textbox whereby the user can enter their search parameters.
2	Search Button	once the button is clicked, a linq query will execute that will match the search string with a records found inside the venue table.
3	Table	The tables columns inherit the attributes of the Venue Table.
4	Add new venue button	Will redirect the user to add venue page.
5	Update button	will allow user to update the venue by redirecting them to the update venue page.
6	Delete button	will allow user to update venue by redirecting them to delete venue page.
7	Return button	Will take the user back to the main menu page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery ▾

User ▾

Update Venue

Name	Erica Rec Hall	1
Capacity	200	2
Accessibility	Minimal	3
Street Number	254 Lunnon Road	4
Suburb	Hatfield	5
Province	Gauteng	6
Venue Type	Indoor	7

8

9 Return **10** Cancel

Figure 228: 4.3.1 Update venue

Number	Component Name	Functionality
1	Name Textbox	The venue's name attribute is retrieved from the venue table and displayed in the textbox.
2	Capacity Textbox	The venue's Capacity attribute is retrieved from the venue table and displayed in the textbox.
3	Accessibility Textbox	The venue's Accessibility attribute is retrieved from the venue table and displayed in the textbox.
4	Street Number Textbox	The venue's Street Number attribute is retrieved from the venue table and displayed in the textbox.
5	Suburb Textbox	The venue's Suburb attribute is retrieved from the venue table and displayed in the textbox.

6	Province Dropdown	The venue's Province attribute is retrieved from the venue table and displayed in the textbox.
7	Venue Type dropdown:	The venue's Venue Type attribute is retrieved from the venue table and displayed in the textbox.
8	Save Button	The system will validate all the textboxes for changes and will update the record in the venue table.
9	Cancel Button	Will cancel the addition of a venue.
10	Return Button	Will redirect the user to the Venue page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery ▾ [✉](#) [User ▾](#)

Delete Venue

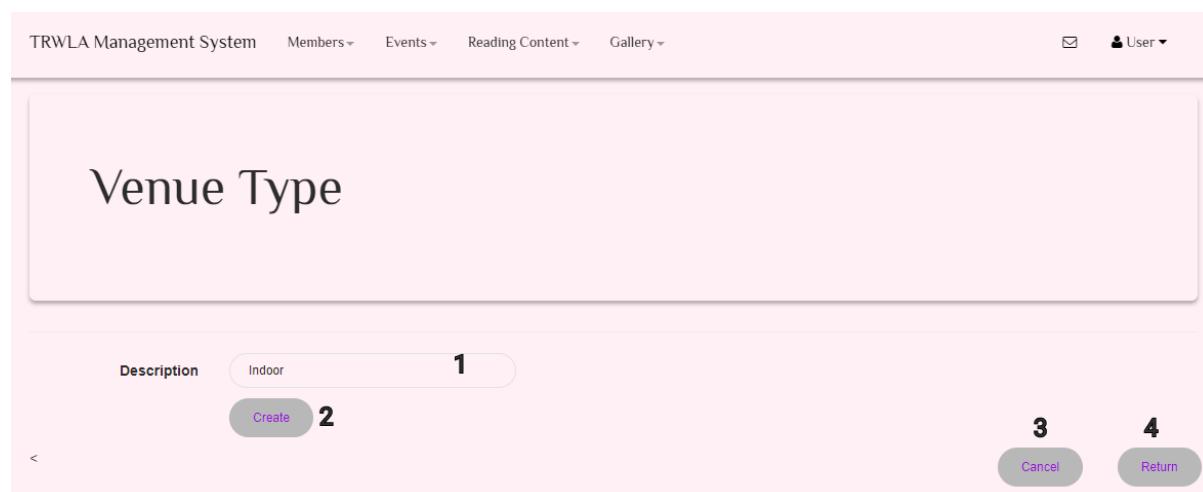
Name	Erica Rec Hall	1
Capacity	200	2
Accessibility	Minimal	3
Street Number	254 Lunnon Road	4
Suburb	Hatfield	5
Province	Gauteng	6
Venue Type	Indoor	7

[Delete](#) **8** [Return](#) **9** [Cancel](#) **10**

Figure 229: 4.4.1 Delete Venue

Number	Component Name	Functionality
1	Name Textbox	The venue's name attribute is retrieved from the venue table and displayed in the textbox.
2	Capacity Textbox	The venue's Capacity attribute is retrieved from the venue table and displayed in the textbox.
3	Accessibility Textbox	The venue's Accessibility attribute is retrieved from the venue table and displayed in the textbox.
4	Street Number Textbox	The venue's Street Number attribute is retrieved from the venue table and displayed in the textbox.
5	Suburb Textbox	The venue's Suburb attribute is retrieved from the venue table and displayed in the textbox.

6	Province Dropdown	The venue's Province attribute is retrieved from the venue table and displayed in the textbox.
7	Venue Type dropdown:	The venue's Venue Type attribute is retrieved from the venue table and displayed in the textbox.
8	Delete Button	The system will delete the record in the venue table.
9	Cancel Button	Will cancel the deletion of a venue.
10	Return Button	Will redirect the user to the Venue page.



Number	Component Name	Functionality
1	Description textbox	The user will provide a description of the new venue type they are creating.
2	Create Button	Once clicked the system will check that it does not exist, will validate the length of the description according to 25 characters. Then will insert the description into the venue type table.
3	Cancel button	Will cancel the creation of a venue type.
4	Return button	Will take the user back to Create venue Type page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery ▾ [✉](#) [User ▾](#)

Add a Venue Type

1
Find by name: **2**

Description	3
Indoor	4 Add a new Venue Type 5 Update 6 Delete 7 Return

Number	Component Name	Functionality
1	Search textbox	A textbox whereby the user can enter their search parameters.
2	Search Button	once the button is clicked, a linq query will execute that will match the search string with an records found inside the venue type table.
3	Table	The columns of the table inherit the attributes of the venue type table.
4	Add new venue type button:	Will redirect the user to add venue type page.
5	Update button:	will allow user to update venue type by redirecting them to update venue type page.
6	Delete button:	will allow user to update venue type by redirecting them to delete venue type page.
7	Return Button	Will take the user back to the main menu page.



Venue Type

Description 1

2 Save **3**

4 Cancel Return

Figure 230: 4.7.1 Update Venue Type

Number	Component Name	Functionality
1	Description textbox:	The system will retrieve the description from the Venue type table.
2	Save Button	Once clicked the system will check that it does not exist, will validate the length of the description according to 25 characters. Then will update existing record of the description into the volunteer Venue table.
3	Cancel button	Will cancel the updating of a Venue type.
4	Return button	Will take the user back to Venue Type page.

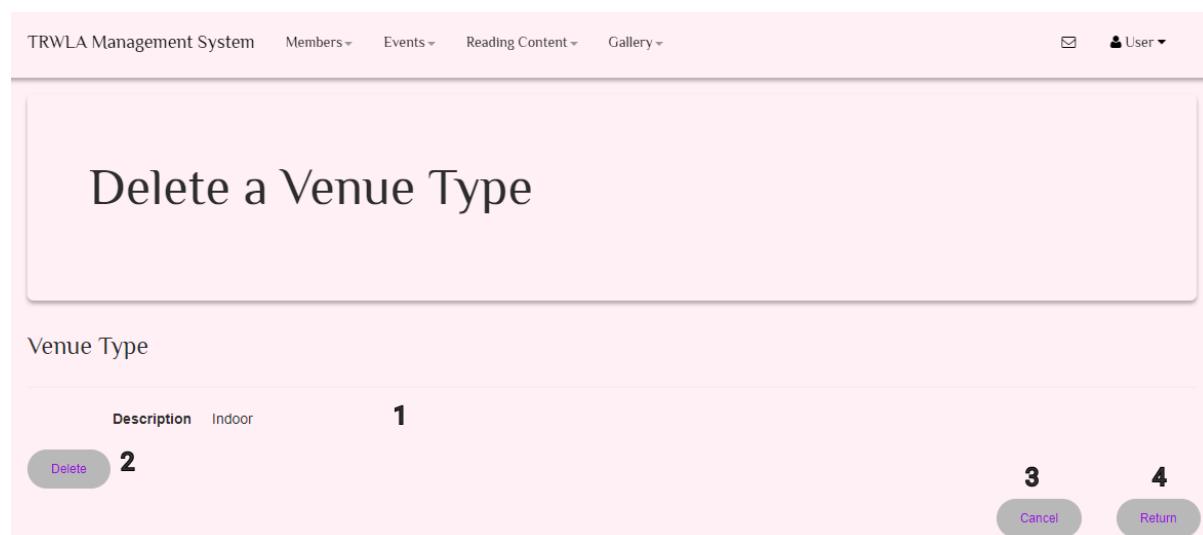


Figure 231: 4.8.1 Delete Venue Type

Number	Component Name	Functionality
1	Description textbox	The system will retrieve the description from the Venue type table.
2	Delete Button	Once clicked the system will delete the existing record of the description from the Venue type table.
3	Cancel button	Will cancel the updating of a Venue type.
4	Return button	Will take the user back to Venue Type page.



The screenshot shows the 'Add a Residence' page of the TRWLA Management System. At the top, there is a navigation bar with links for 'Members', 'Events', 'Reading Content', and 'Gallery'. On the right, there are icons for email and user profile. The main content area has a pink header with the title 'Add a Residence'. Below the header, the word 'Residence' is displayed. The form consists of several input fields and buttons. A text input field for 'ResName' contains the value 'Nerinal'. A dropdown menu for 'ResType' is open. A large grey button labeled 'Create' is positioned below the dropdown. To the right of the 'Create' button are two smaller buttons: 'Cancel' and 'Return'. Numbered callouts (1 through 5) are overlaid on the screen to identify specific components: 1 points to the 'ResName' input field; 2 points to the 'ResType' dropdown; 3 points to the 'Create' button; 4 points to the 'Cancel' button; and 5 points to the 'Return' button.

Figure 232: 4.9.2 Add Residence

Number	Component Name	Functionality
1	Res name Textbox	The user will enter the name of the residence they wish to create.
2	Res Type dropdown	The user will select a residence type from the drop down items.
3	Create Button	The system will validate that the details are unique and will then insert the data from the textboxes in the venue table.
4	Cancel Button	Will cancel the addition of a residence for the volunteer.
5	Return Button	Will cancel the addition of a residence for the volunteer.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery ▾ [✉](#) [User ▾](#)

Residences

1 Find by name: 2 [Search](#)

ResName	Description	3
Nerina		
Madelif		
Lilium		
Tuks Naledi		

4 [Add a new Residence](#) 5 [Update](#) 6 [Delete](#) 7 [Return](#)

Number	Component Name	Functionality
1	Search Textbox	Will accept string search parameters.
2	Search Button	Will execute LINQ query to retrieve search results.
3	Table	The columns of the table inherit the attributes of the <u>Residence</u> Table: Resname and Description
4	Add Residence Button	Will direct the user to the add residence page
5	Update Button	Will direct the user to the update Residence page.
6	Delete Button	Will direct the user to delete Residence page or if a record is selected it will delete that record.
7	Return Button	Will return the user back the previous page they were on.

Figure 233: 4.11.1 Update Residence

Number	Component Name	Functionality
1	Res name Textbox	The system will retrieve the Res name attribute from the residence table through a linq query.
2	Res Type dropdown	The system will retrieve the Res Type attribute from the residence table through a linq query.
3	Save Button	The system will validate that the details are unique and will then update the data from the textboxes in the Residence table.
4	Cancel Button	Will cancel the updating of a residence for the volunteer.
5	Return Button	Will cancel the updating of a residence for the volunteer.



Figure 234: 4.12.1 Delete a Residence

Number	Component Name	Functionality
1	Res name Textbox	The system will retrieve the Res name attribute from the residence table through a linq query.
2	Description Textbox	The system will retrieve the Res Type attribute from the residence table through a linq query.
3	Delete Button	The system will delete the select record from the residence table using a linq query
4	Cancel Button	Will cancel the deleting of a residence for the volunteer.
5	Return Button	Will cancel the deleting of a residence for the volunteer.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Welcome Cailin

1 SORT WITH THE FOLLOWING:

2 Search upcoming events by Name **3** Event Name **4** Date

5

Upcoming TRWLA Events

Finding Yourself	Living the Dream
Summary: This is where you will	Summary: Lorem ipsum dolor sit amet
Date of the Event: 08/09/2017	Date of the Event: 20/09/2017
Time of Event: 20:00	Time of Event: 18:00
Venue Name: Duxbury Palace	Venue Name: Rautenbach
Details RSVP	Details RSVP

6 My Upcoming Events

Finding Yourself	Living the Dream
Summary: This is where you will	Summary: Lorem ipsum dolor sit amet
Date of the Event: 08/09/2017	Date of the Event: 20/09/2017
20:00	18:00
Venue Name: Duxbury Palace	Venue Name: Rautenbach
Details Not Going	Details Not Going

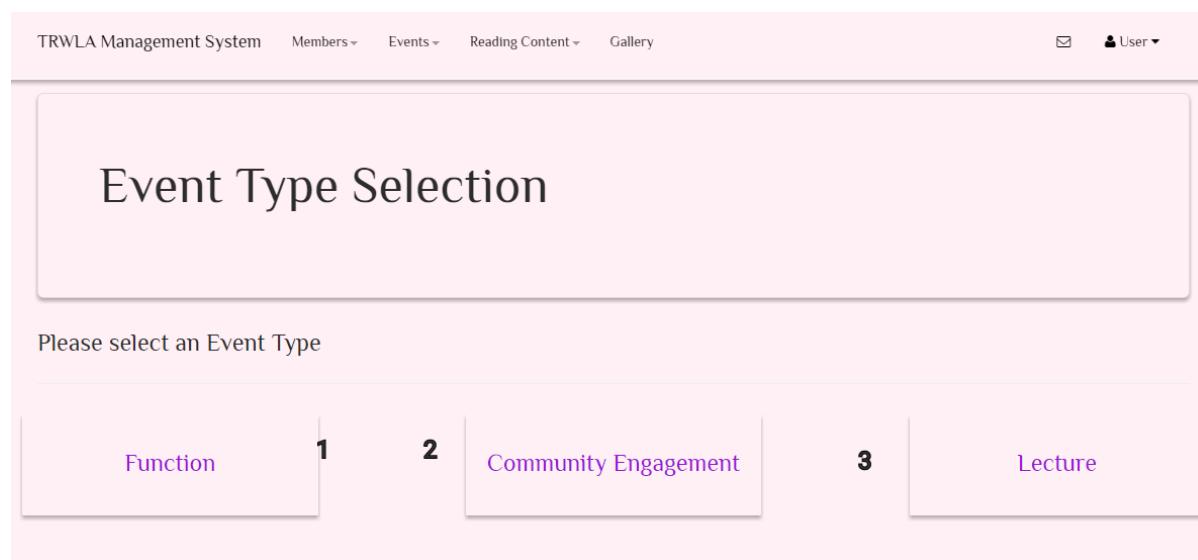
7 **8**

9 **10** Create **11** Update **12** Delete **13** Return Log Attendance

Figure 235: 5.0.1 Events Home VA

Number	Component Name	Functionality
1	Search textbox.	The user will enter their event search parameters that has to be a string and can only search by name.
2	Search upcoming events button.	Once clicked the system will use a linq query to match the parameters with a record of the event in the events table.
3	Event Name Sort Button.	Once clicked, it will sort the list of events in the event timeline by name in alphabetical order.

4	Date Sort Button	Once clicked, it will sort the list of event in the event timeline by date. Events that are listed first are ones that are upcoming in the calendar.
5	Event Timeline pane:	All the created events by volunteers in TWRLA will appear here.
6	My event timeline pane	all the events the user has rsvp'd to will appear here.
7	Details Hyperlink	Once clicked, this will redirect the user to details page of the particular event.
8	RSVP Hyperlink	Once clicked, this will allow the user to rsvp to the selected event.
9	Create Button:	once clicked, it will direct the user to the select event type screen to proceed in creating their event.
10	Update Button:	Once clicked, the system will redirect the user to update event page of the selected event to allow them to update the event.
11	Return Button:	Will redirect the user to the previous they were on.
12	Log Attendance Button:	Once clicked this will redirect the user to log attendance page for the particular event.



Please select an Event Type

1 2 3

Function Community Engagement Lecture

Figure 236: 5.0.2 Events Home VA

Number	Component Name	Functionality
1	Function Button:	Will redirect the user to the create function event page.
2	Community Engagement Button:	Will redirect the user to the create community engagement event page.
3	Lecture button	Will redirect the user to the create lecture event page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) User ▾

Create a Function

Event

Name	Looking Forward	1	 11	
Guest Speaker	Christopher	2		
Register Guest Speaker 3				
<input type="radio"/> Accepted Invitation <input checked="" type="radio"/> Rejected Invitation				4
Time	02:30 PM	5		
Date	2017/07/29	6		
Venue	Duxburv Palace	7		
Add Venue 8				
Summary	Taking that step forward	9		
Description	This is where you will look forward	10		

12 [Browse](#)

13 [Create](#) **14** [Return](#) **15** [Cancel](#)

Figure 237: 5.1.1 Create Function VA

Number	Component Name	Functionality
1	Name textbox	The user will enter the name of the event. Must be a string value of no more than 35 characters.)
2	Guest Speaker Dropdown:	The user will select a guest speaker. Must be a value selected from the Guest Speaker dropdown list which is populated by the GuestSpeaker table
3	Register guest speaker hyperlink	Once clicked the user will be redirected to the register guest speaker page.

4	Guest speak radio buttons	If the guest speaker has confirmed their attendance, the accepted invitation will be checked. If not the rejected invitation will be checked.
5	Time picker	The user will enter their preferred time for the event. Must be a valid time format i.e HH:MM as clicked from the time picker.
6	Date picker	The user will enter their preferred date for the event. Must be a valid date in the format i.e. CCYY-MM-DD as clicked from the date picker
7	Venue dropdown	The user will select a venue from the dropdown list. Must be a value clicked from the Venue dropdown list which is populated by the <u>Venue</u> table.
8	Add venue hyperlink:	once clicked, the user will be directed to the add venue page.
9	Summary textbox:	The user can provide an event summary. Must be a string value of no more than 100 characters
10	Description Textbox:	The user can provide an event description. Must be a string value of no more than 300 characters.
11	Image box:	This is where the poster of the event will be displayed. Must be in a valid image format png or svg.
12	Browse Button:	Once clicked, the system will open up an 'Open File Dialog' that will allow the user to get an image off their machine.
13	Create button:	Once clicked the system will launch a confirmation Modal
14	Return Button:	Will redirect the user to events home page.

15

Cancel button:

Will cancel the creation of the event

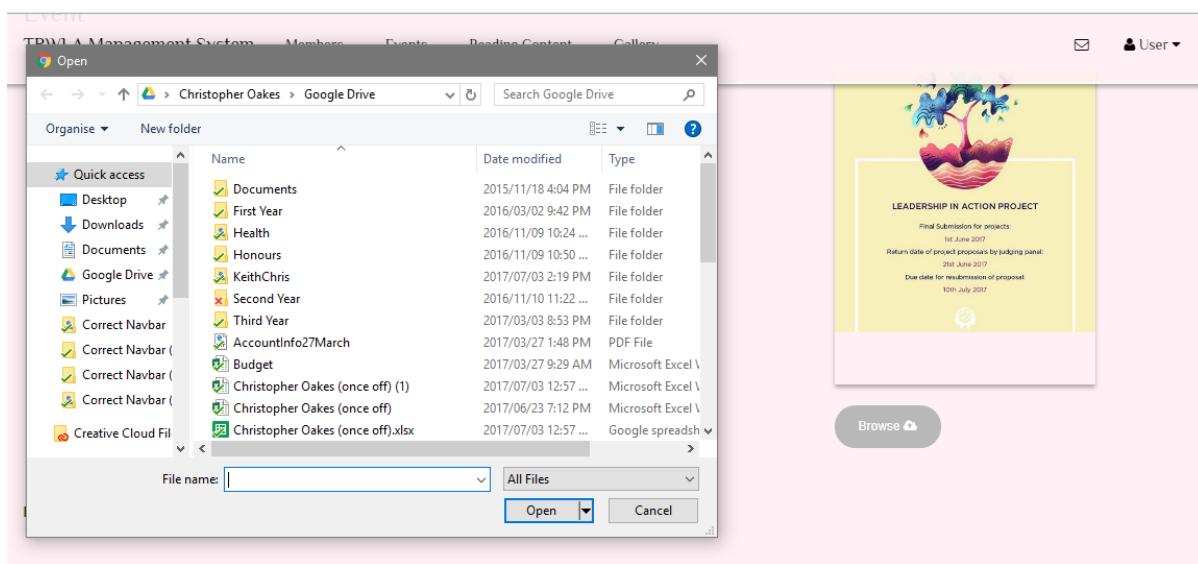


Figure 238: 5.1.2 Create Function VA

Number	Component Name	Functionality
1	Open file dialog	This will allow the user to select an image on the computer to upload as an event poster.

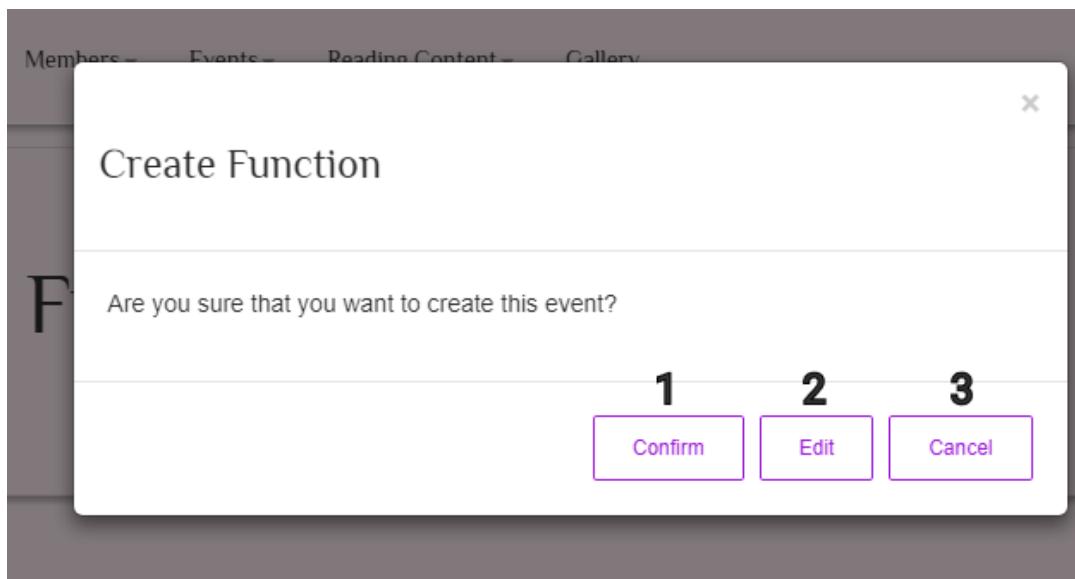


Figure 239: 5.1.3 Create Function VA

Number	Component Name	Functionality
1	Confirm button:	The system will validate all the fields and use a LINQ query to insert the event into the event and function tables.
2	Edit button:	Will allow the user to go back to the create function page to edit any information
3	Cancel button:	Will cancel the creation of a function event.

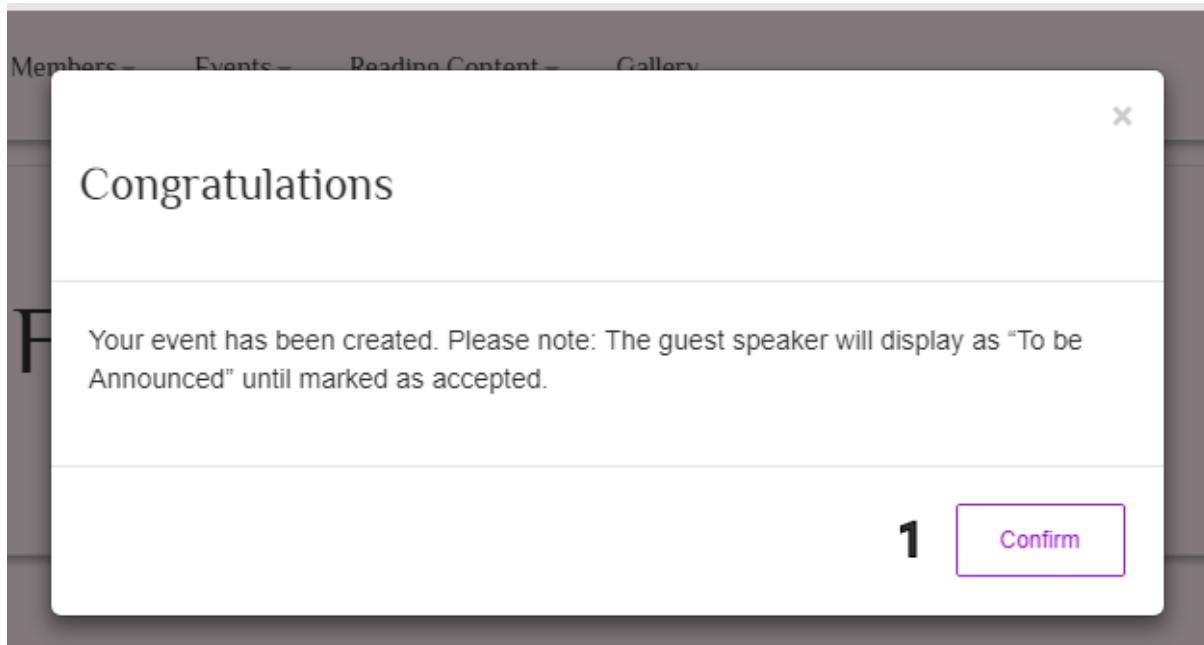


Figure 240: 5.1.4 Create Function VA

Number	Component Name	Functionality
1	Confirm button modal:	Will direct the user to events home page and also display the event they have just created.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Create a Community Engagement

Name	<input type="text" value="Looking Forward"/> 1
Time	<input type="text" value="02:30 PM"/> 2
Date	<input type="text" value="2017/07/29"/> 3
Venue	Duxbury Palace 4 Add Venue 5
Summary	<input type="text" value="This is where you will find"/> 6
Description	<input type="text" value="Where you want to look forward"/> 7
Content	<input type="text" value="Looking Forward"/> 8 Upload Content 9



10

11 [Browse](#)

12 [Create](#) **13** [Return](#) **14** [Cancel](#)

Figure 241: 5.2.1 Create Community Outreach VA

Number	Component Name	Functionality
1	Name textbox:	The user will enter the name of the event. Must be a string value of no more than 35 characters.)
2	Time picker:	The user will enter their preferred time for the event. Must be a valid time format i.e HH:MM as clicked from the time picker.
3	Date picker:	The user will enter their preferred date for the event. Must be a valid date in the format i.e. CCYY-MM-DD as clicked from the date picker.
4	Venue dropdown:	The user will select a venue from the dropdown list. Must be a value clicked from the Venue dropdown list which is populated by the <u>Venue</u> table.

5	Add venue hyperlink:	once clicked, the user will be directed to the add venue page.
6	Summary textbox:	The user can provide an event summary. Must be a string value of no more than 100 characters
7	Description Textbox:	The user can provide an event description. Must be a string value of no more than 300 characters.
8	Content Dropdown:	The user will select content to accompany the event. Must be a value clicked from the Content dropdown list which is populated by the <u>Community outreachContent</u> table.
9	Upload Content Hyperlink:	Will Redirect the user to the upload content page.
10	Image box:	This is where the poster of the event will be displayed. Must be in a valid image format png or svg.
11	Browse Button:	Once clicked, the system will open up an 'Open File Dialog' that will allow the user to get an image off their machine.
12	Create button:	Once clicked the system will launch a confirmation Modal.
13	Return Button:	Will redirect the user to events home page.
14	Cancel button:	Will cancel the creation of the event.

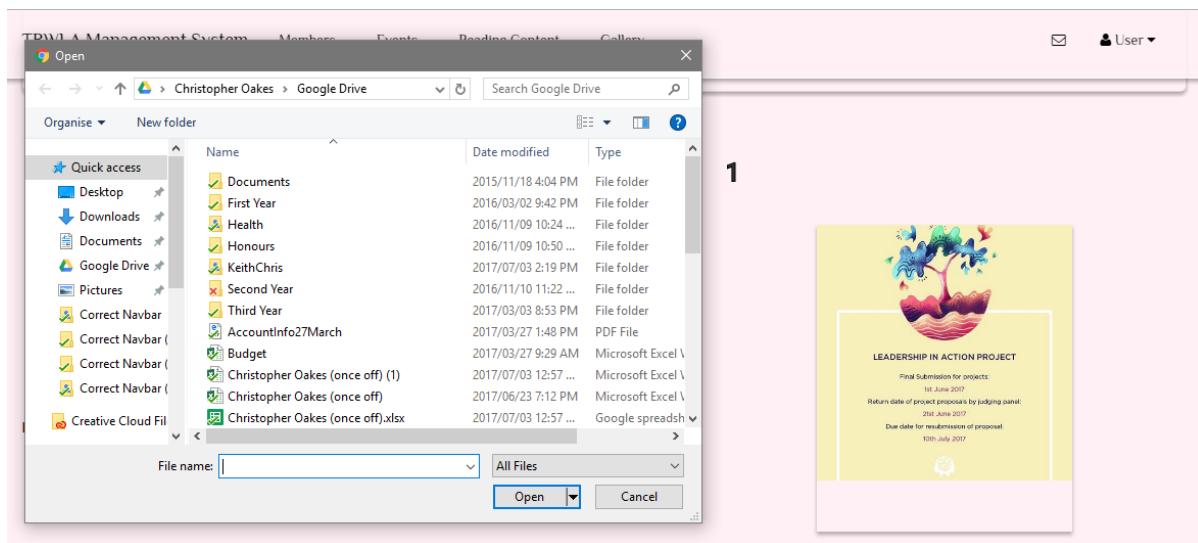


Figure 242: 5.2.2 Create Community Outreach VA

Number	Component Name	Functionality
1	Open file dialog	This will allow the user to select an image on the computer to upload as an event poster.

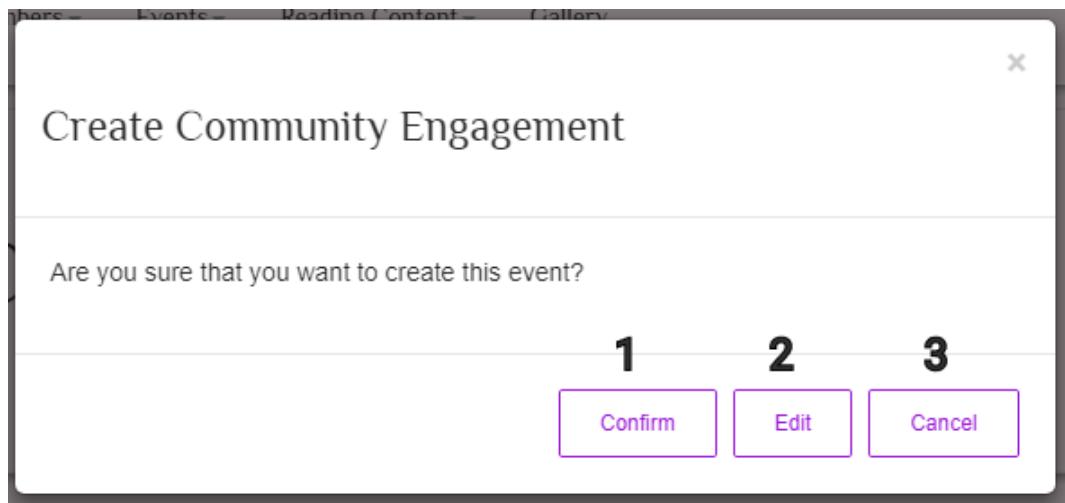


Figure 243: 5.2.3 Create Community Outreach VA

Number	Component Name	Functionality
1	Confirm button	The system will validate all the fields and use a LINQ query to insert the event into the <u>Event and Community Outreach tables.</u>
2	Edit button	Will allow the user to go back to the create community outreach page to edit any information
3	Cancel button	Will cancel the creation of the community outreach event.

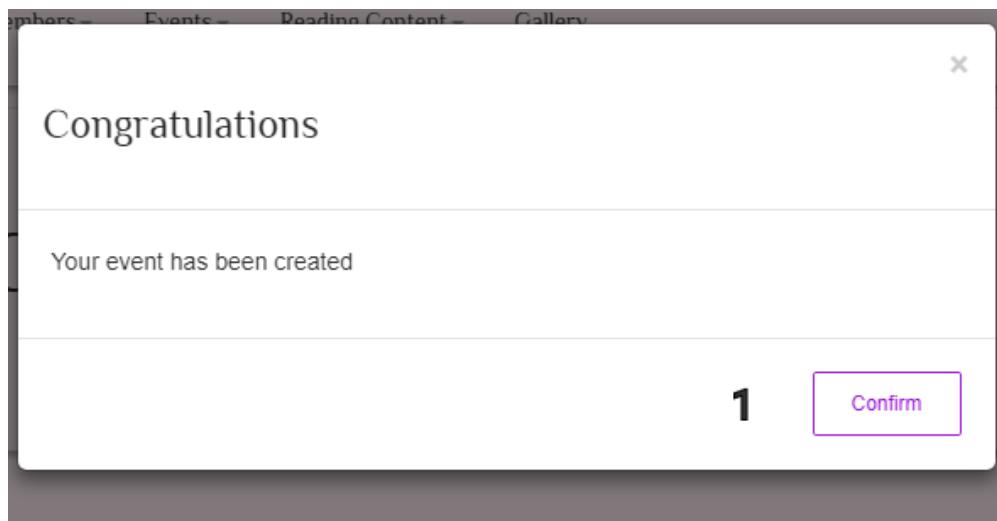


Figure 244: 5.2.4 Create Community Outreach VA

Number	Component Name	Functionality
1	Confirm button modal	Will direct the user to events home page and also display the event they have just created.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) User ▾

Create a Lecture

Name	Looking Forward	1
Time	02:30 PM	2
Date	2017/07/29	3
Venue	Duxbury Palace	4
	Add Venue 5	
Summary	This is where you will see	6
Description	Taking that step forward	7
Content	Looking Forward	8
	Upload Content 9	
Residence	Mauritie	10

11


12 [Browse](#)

13 [Create](#)

14 [Return](#)

15 [Cancel](#)

Figure 245: 5.3.1 Create Lecture VA

Number	Component Name	Functionality
1	Name textbox	The user will enter the name of the event. Must be a string value of no more than 35 characters.)
2	Time picker	The user will enter their preferred time for the event. Must be a valid time format i.e HH:MM as clicked from the time picker.
3	Date picker:	The user will enter their preferred date for the event. Must be a valid date in the format i.e. CCYY-MM-DD as clicked from the date picker.
4	Venue dropdown:	The user will select a venue from the dropdown list. Must be a value clicked

		from the Venue dropdown list which is populated by the <u>Venue</u> table.
5	Add venue hyperlink:	once clicked, the user will be directed to the add venue page.
6	Summary textbox:	The user can provide an event summary. Must be a string value of no more than 100 characters
7	Description Textbox:	The user can provide an event description. Must be a string value of no more than 300 characters.
8	Content Dropdown:	The user will select content to accompany the event. Must be a value clicked from the Content dropdown list which is populated by the <u>LectureContent</u> table.
9	Upload Content Hyperlink:	Will Redirect the user to the upload content page.
10	Image box:	This is where the poster of the event will be displayed. Must be in a valid image format png or svg.
11	Residence dropdown:	The user will select residence from the dropdown list that is populated from the <u>Residence</u> table.
12	Browse Button:	Once clicked, the system will open up an ‘Open File Dialog’ that will allow the user to get an image off their machine.
13	Create button:	Once clicked the system will launch a confirmation Modal.
14	Return Button:	Will redirect the user to events home page.
15	Cancel button:	Will cancel the creation of the event.

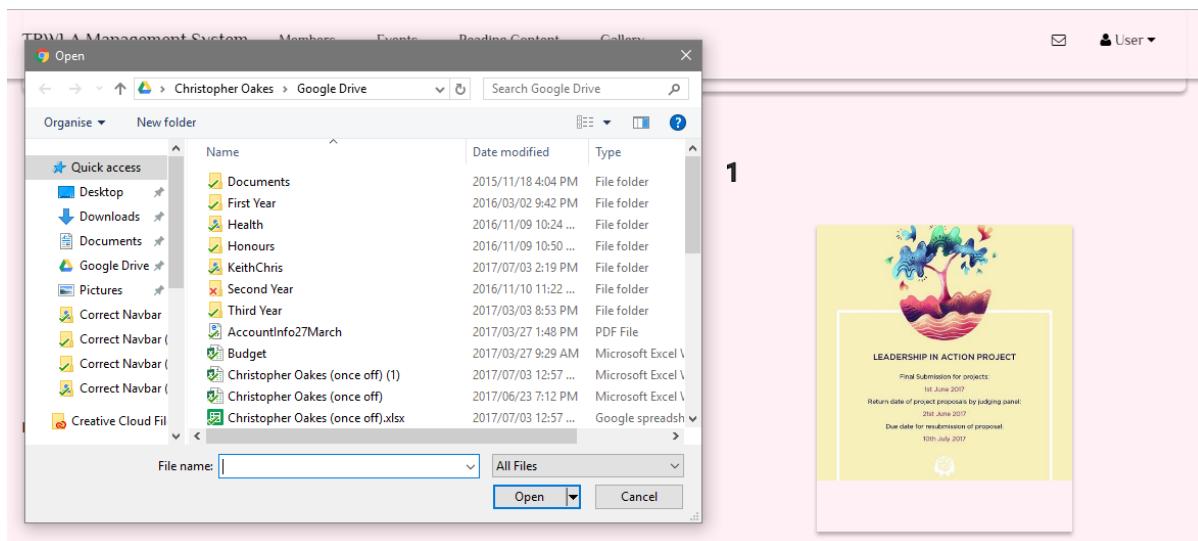


Figure 246: 5.3.2 Create Lecture VA

Number	Component Name	Functionality
1	Open file dialog	This will allow the user to select an image on the computer to upload as an event poster.

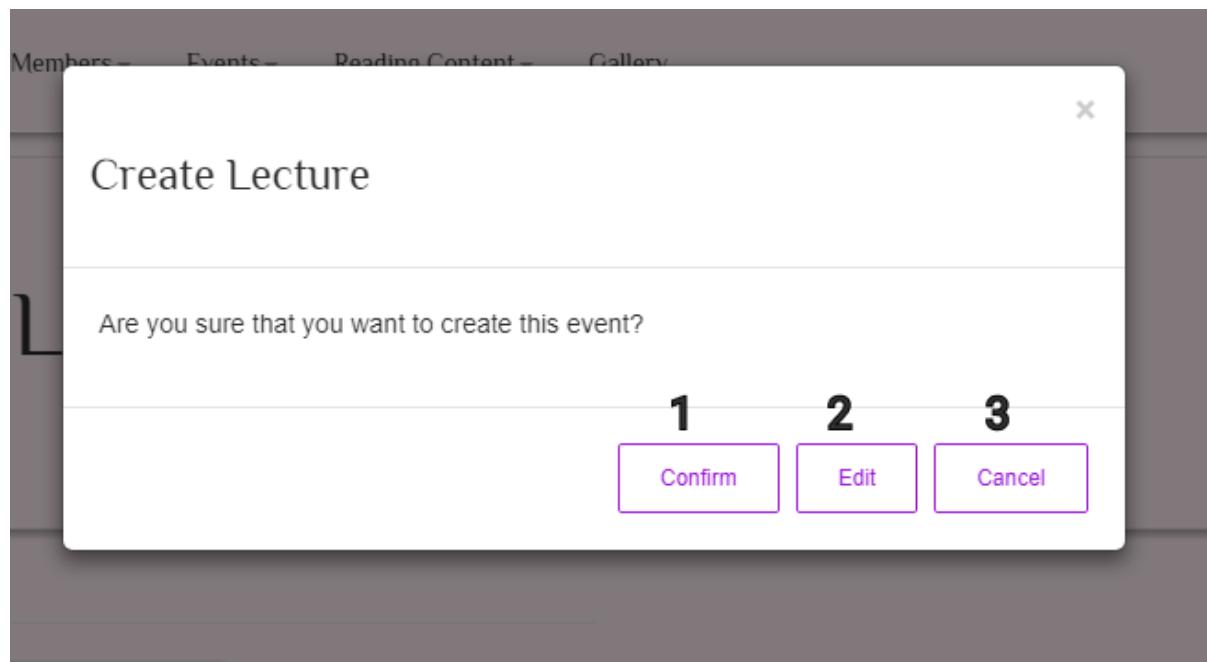


Figure 247:5.3.3 Create Lecture VA

Number	Component Name	Functionality
1	Confirm button	The system will validate all the fields and use a LINQ query to insert the event into the <u>Event and Lecture tables</u> .
2	Edit button	Will allow the user to go back to the create Lecture page to edit any information
3	Cancel button	Will cancel the creation of a lecture event.

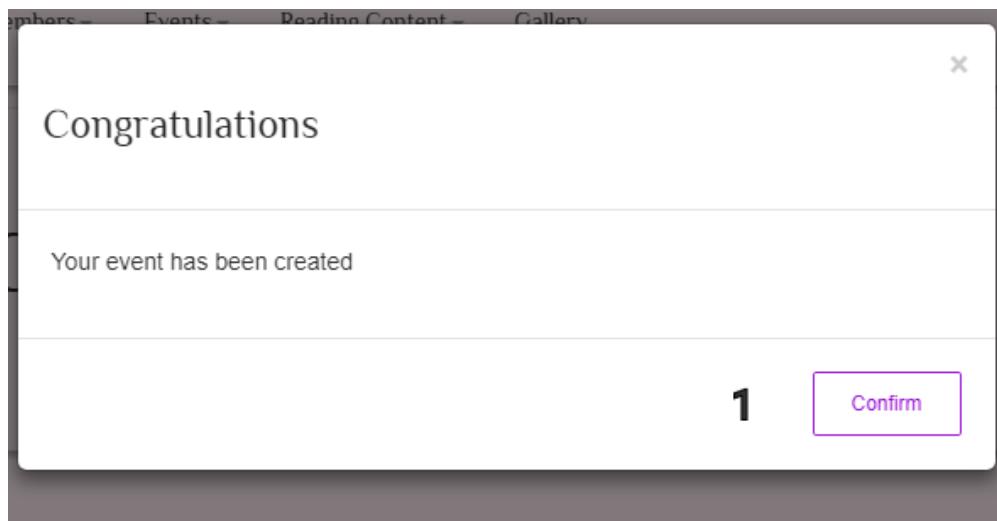


Figure 248: 5.3.4 Create Lecture VA

Number	Component Name	Functionality
1	Confirm button modal:	Will direct the user to events home page and also display the event they have just created.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Welcome Cailin

1 Finding Yourself

2 Search upcoming events by Name

3 SORT WITH THE FOLLOWING:

4 Event Name Date

5 Upcoming TRWLA Events

6 My Upcoming Events

7 Details **8** RSVP

9 Create **10** Update **11** Delete **12** Return **13** Log Attendance

Figure 249: 5.4.1 Search Event VA

Number	Component Name	Functionality
1	Search textbox.	The user will enter their event search parameters that has to be a string and can only search by name.
2	Search upcoming events button.	Once clicked the system will use a linq query to match the parameters with a record of the event in the events table.
3	Event Name Sort Button.	Once clicked, it will sort the list of events in the event timeline by name in alphabetical order.

4	Date Sort Button	Once clicked, it will sort the list of event in the event timeline by date. Events that are listed first are ones that are upcoming in the calendar.
5	Event Timeline pane:	All the created events by volunteers in TWRLA will appear here.
6	My event timeline pane	all the events the user has rsvp'd to will appear here.
7	Details Hyperlink	Once clicked, this will redirect the user to details page of the particular event.
8	RSVP Hyperlink	Once clicked, this will allow the user to rsvp to the selected event.
9	Create Button:	once clicked, it will direct the user to the select event type screen to proceed in creating their event.
10	Update Button:	Once clicked, the system will redirect the user to update event page of the selected event to allow them to update the event.
11	Return Button:	Will redirect the user to the previous they were on.
12	Log Attendance Button:	Once clicked this will redirect the user to log attendance page for the particular event.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Finding Yourself Details

Name	<input type="text" value="Finding Yourself"/>	1
Date	<input type="text" value="08/09/2017"/>	2
Time	<input type="text" value="20:00"/>	3
Summary	<input type="text" value="This is where you will"/>	4
Description	<input type="text" value="Finding yourself is the most important attribute to have..."/>	5
Content	<input type="text" value="Looking Inward"/>	6
Venue	<input type="text" value="Duxbury Palace"/>	7

9 [RSVP](#)
10 [Update](#)
11 [Cancel Event](#)
12 [Return](#)

Figure 250: 5.4.2 Search Event VA ComEng

Number	Component Name	Functionality
1	Name textbox:	The event's name attribute is retrieved from the <u>Event and Community outreach</u> tables and displayed in the textbox.
2	Time textbox:	The event's time attribute is retrieved from the <u>Event and Community outreach</u> tables and displayed in the textbox.
3	Date textbox:	The event's date attribute is retrieved from the <u>Event and Community outreach</u> tables and displayed in the textbox.
4	Venue textbox:	The event's venue attribute is retrieved from the <u>Event and Community outreach</u>

		<u>outreach</u> tables and displayed in the textbox.
5	Summary textbox:	The event's summary attribute is retrieved from the <u>Event and Community outreach</u> tables and displayed in the textbox.
6	Description Textbox:	The event's description attribute is retrieved from the <u>Event and Community outreach</u> tables and displayed in the textbox.
7	Content textbox:	The event's content attribute is retrieved from the <u>Event, Community Engagement Content and Community outreach</u> tables and displayed in the textbox.
8	Image box:	The events poster is displayed here, it is retrieved from the <u>Event and Community outreach</u> tables.
9	RSVP button:	Once clicked, it will allow the user to RSVP to an event.
10	Update button:	Once clicked it will direct user to the update community engagement page.
11	Return Button:	Will redirect the user to the events home page.
12	Cancel Event button:	Will allow the user to cancel the event.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Finding Yourself Details

Name	<input type="text" value="Finding Yourself"/>	1
Date	<input type="text" value="08/09/2017"/>	2
Time	<input type="text" value="20:00"/>	3
Summary	<input type="text" value="This is where you will"/>	4
Description	<input type="text" value="Finding yourself is the most important attribute to have..."/>	5
Guest Speaker	Christopher Oakes	6
<input type="radio"/> Accepted Invitation <input checked="" type="radio"/> Pending Invitation		7
Venue	<input type="text" value="Duxbury Palace"/>	8
RSVP Update Cancel Event Return		10 11 12 13
		

Figure 251: 5.4.2 Search Event VA Function

Number	Component Name	Functionality
1	Name textbox:	The event's name attribute is retrieved from the <u>Event and Function</u> tables and displayed in the textbox.
2	Time textbox:	The event's time attribute is retrieved from the <u>Event and Function</u> tables and displayed in the textbox.
3	Date textbox:	The event's date attribute is retrieved from the <u>Event and Function</u> tables and displayed in the textbox.
5	Venue textbox:	The event's venue attribute is retrieved from the <u>Event and Function</u> tables and displayed in the textbox.

6	Summary textbox:	The event's summary attribute is retrieved from the <u>Event and Community</u> tables and displayed in the textbox.
7	Description Textbox:	The event's description attribute is retrieved from the <u>Event and Function</u> tables and displayed in the textbox.
8	Content textbox:	The event's content attribute is retrieved from the <u>Event, Function Content and Function</u> tables and displayed in the textbox.
9	Image box:	The events poster is displayed here, it is retrieved from the <u>Event and Function</u>
10	Guest Speaker Textbox:	the event's guest speaker will be retrieved from the <u>GuestSpeaker</u> table and displayed in the textbox.
11	Guest speak radio buttons:	If the guest speaker has confirmed their attendance, the accepted invitation will be checked. If not the rejected invitation will be checked.
12	RSVP button:	Once clicked, it will allow the user to RSVP to an event.
13	Update button:	Once clicked it will direct user to the update Function Event page.
14	Return Button:	Will redirect the user to the events home page.
15	Cancel Event button:	Will allow the user to cancel the event.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Finding Yourself Details

Name	<input type="text" value="Finding Yourself"/>	1
Date	<input type="text" value="08/09/2017"/>	2
Time	<input type="text" value="20:00"/>	3
Summary	<input type="text" value="This is where you will"/>	4
Description	<input type="text" value="Finding yourself is the most important attribute to have..."/>	5
Content	<input type="text" value="Looking Inward"/>	6
Venue	<input type="text" value="Duxbury Palace"/>	7
Residence	<input type="text" value="Magritje"/>	8



9

10 [RSVP](#)
11 [Update](#)
12 [Cancel Event](#)
13 [Return](#)

Figure 252: 5.4.2 Search Event VA Lecture

Number	Component Name	Functionality
1	Name textbox:	The event's name attribute is retrieved from the <u>Event and Lecture</u> tables and displayed in the textbox.
2	Time textbox:	The event's time attribute is retrieved from the <u>Event and Lecture</u> tables and displayed in the textbox.
3	Date textbox:	The event's date attribute is retrieved from the <u>Event and Lecture</u> tables and displayed in the textbox.
4	Venue textbox:	The event's venue attribute is retrieved from the <u>Event and Lecture</u> tables and displayed in the textbox.

5	Summary textbox:	The event's summary attribute is retrieved from the <u>Event and Lecture</u> tables and displayed in the textbox.
6	Description Textbox:	The event's description attribute is retrieved from the <u>Event and Lecture</u> tables and displayed in the textbox.
7	Content textbox:	The event's content attribute is retrieved from the <u>Event, Lecture Content and Lecture</u> tables and displayed in the textbox.
8	Image box:	The events poster is displayed here, it is retrieved from the <u>Event and Lecture</u> .
9	Residence Textbox:	the event's residence attribute will be retrieved from the <u>Residence</u> table and displayed in the textbox.
10	RSVP button: Once clicked, it will allow the user to RSVP to an event. Update button:	Once clicked it will direct user to the update Lecture Event page.
11	Return Button:	Will redirect the user to the events home page.
12	Cancel Event button:	Will allow the user to cancel the event.

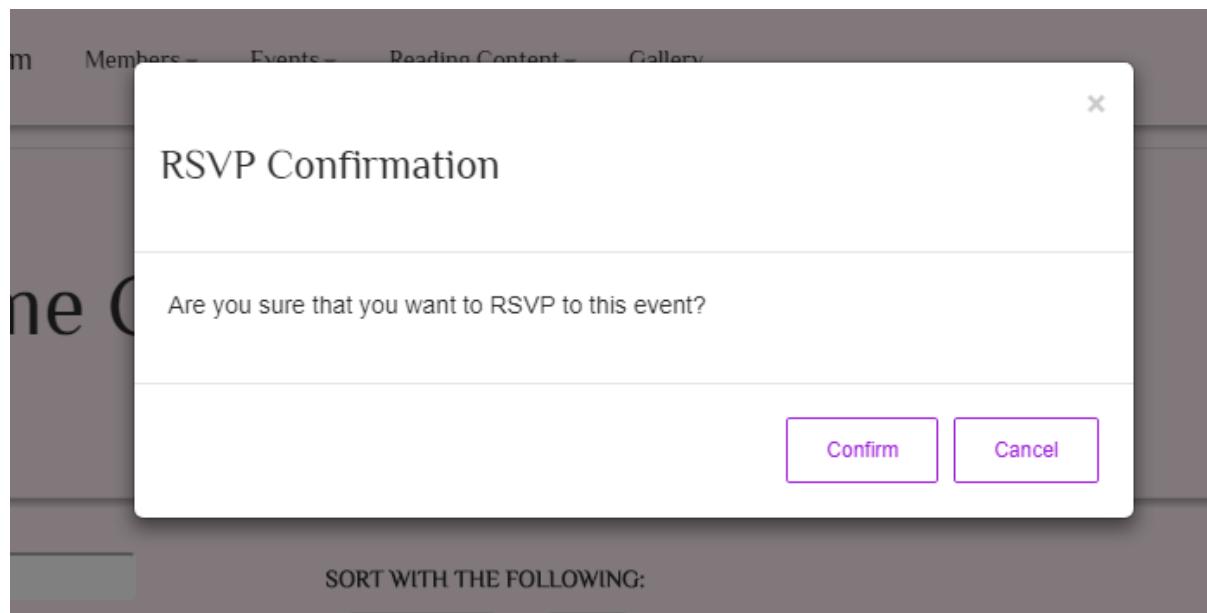


Figure 253: 5.5.1 RSVP to an Event VA ComEng

Number	Component Name	Functionality
1	Confirm Button:	Will update the user's RVSP status as going, and will show the event on the users 'My upcoming events' pane.
2	Cancel Button:	Will abort the user's confirmation of their RSVP to the event.

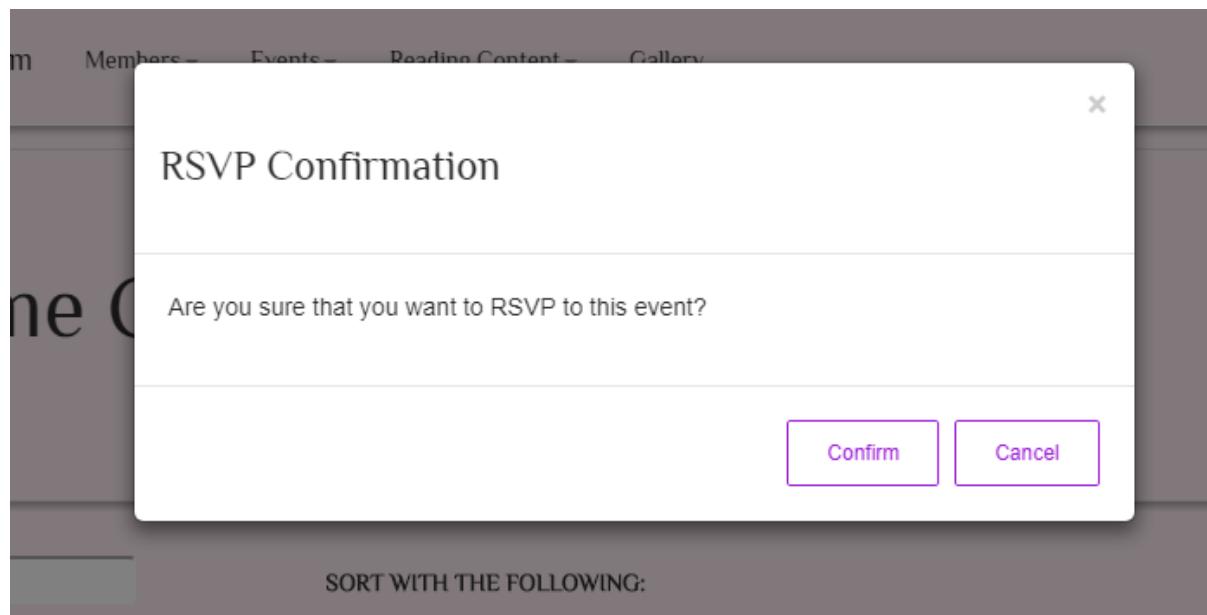


Figure 254: 5.5.1 RSVP to an Event VA Lecture

Number	Component Name	Functionality
1	Confirm Button:	Will update the user's RVSP status as going, and will show the event on the users 'My upcoming events' pane.
2	Cancel Button:	Will abort the user's confirmation of their RSVP to the event.

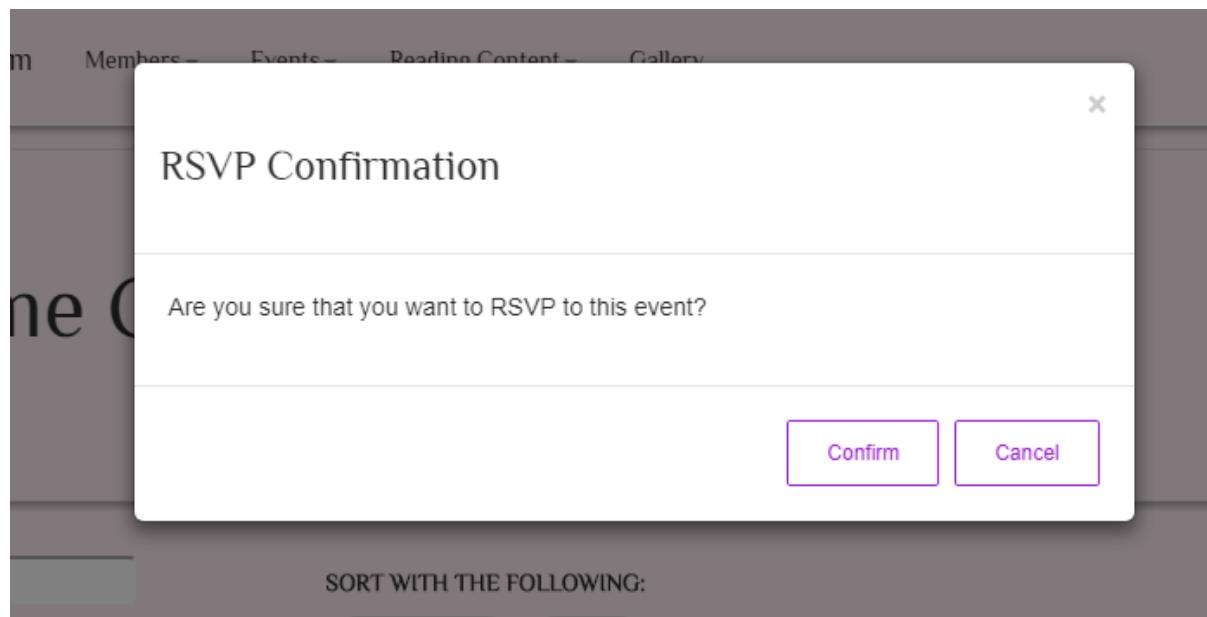


Figure 255: 5.5.2 RSVP to an Event VA Function

Number	Component Name	Functionality
1	Confirm Button:	Will update the user's RVSP status as going, and will show the event on the users 'My upcoming events' pane.
2	Cancel Button:	Will abort the user's confirmation of their RSVP to the event.

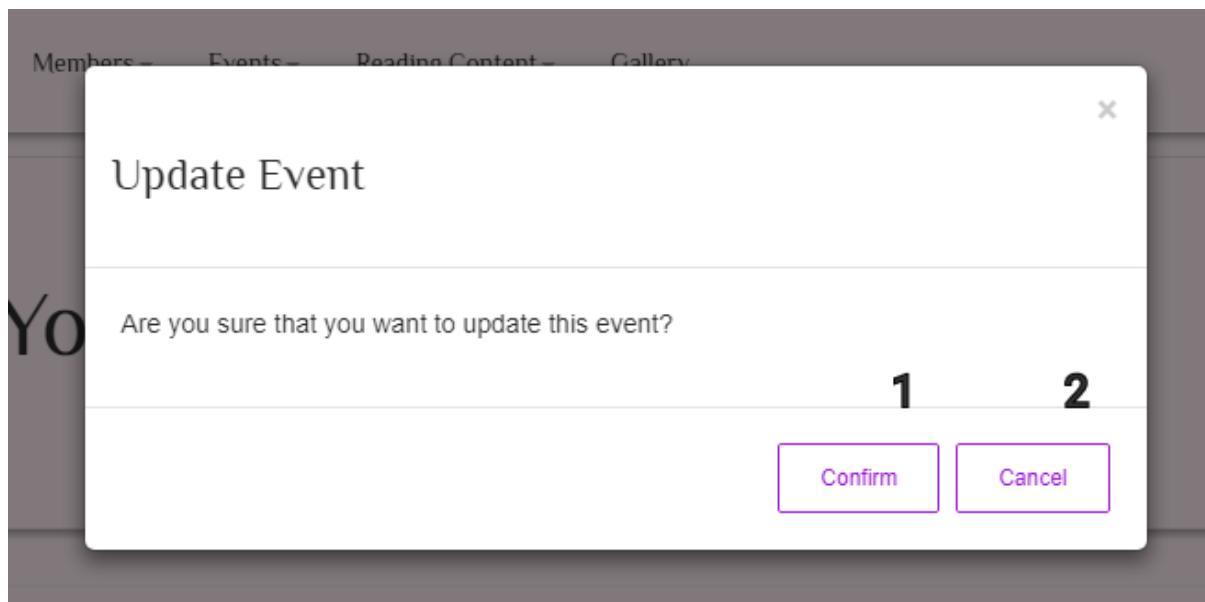


Figure 256: 5.6.1 Update Event Information VA ComEng

Number	Component Name	Functionality
1	Confirm Button:	The system updates the relevant fields in the <u>Event and Community Engagement tables.</u>
2	Cancel Button:	The system will abort the updating of the selected event.

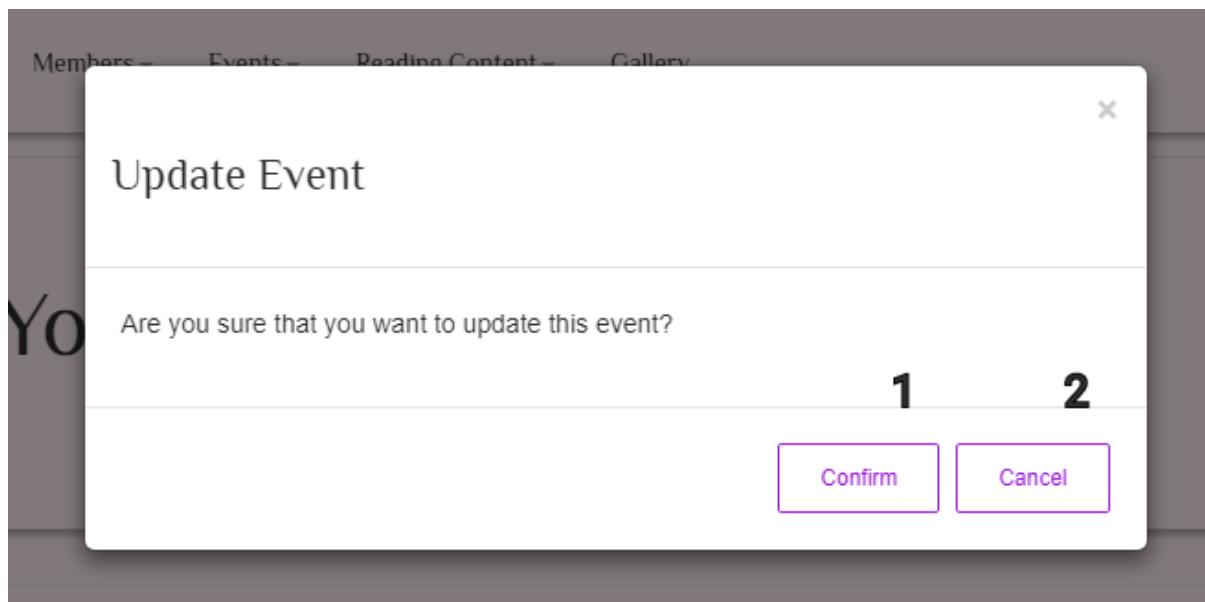


Figure 257: 5.6.1 Update Event Information VA Lecture

Number	Component Name	Functionality
1	Confirm Button:	The system updates the relevant fields in the <u>Event and Lecture tables</u> .
2	Cancel Button:	The system will abort the updating of the selected event.

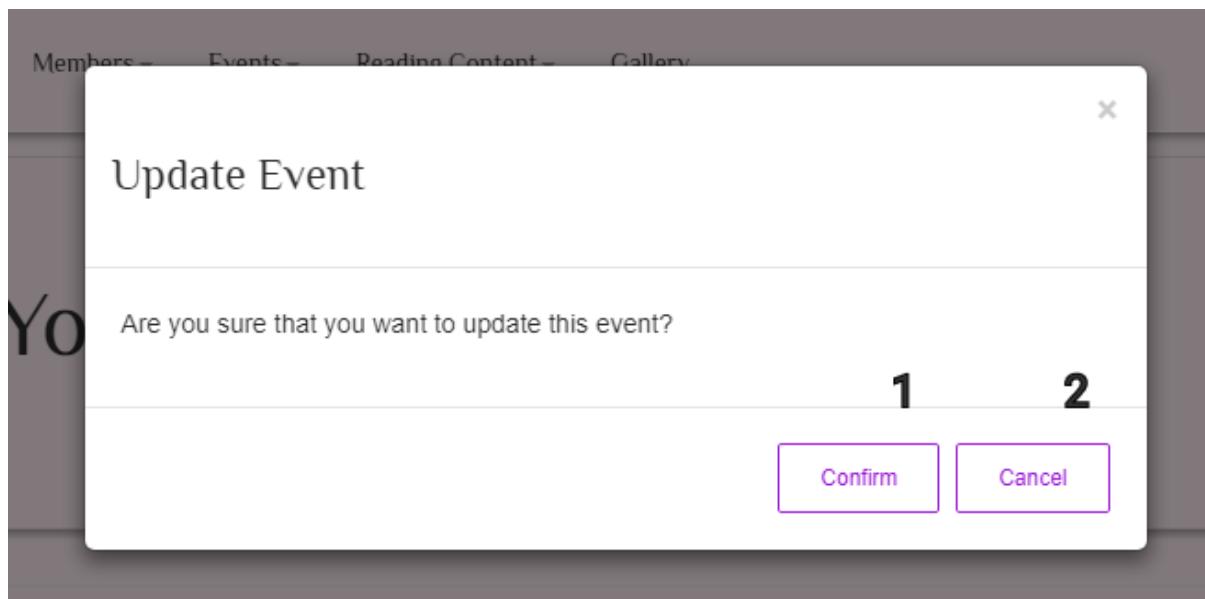


Figure 258: 5.6.2 Update Event Information VA Function

Number	Component Name	Functionality
1	Confirm Button:	The system updates the relevant fields in the <u>Event and function Engagement tables.</u>
2	Cancel Button:	The system will abort the updating of the selected event.

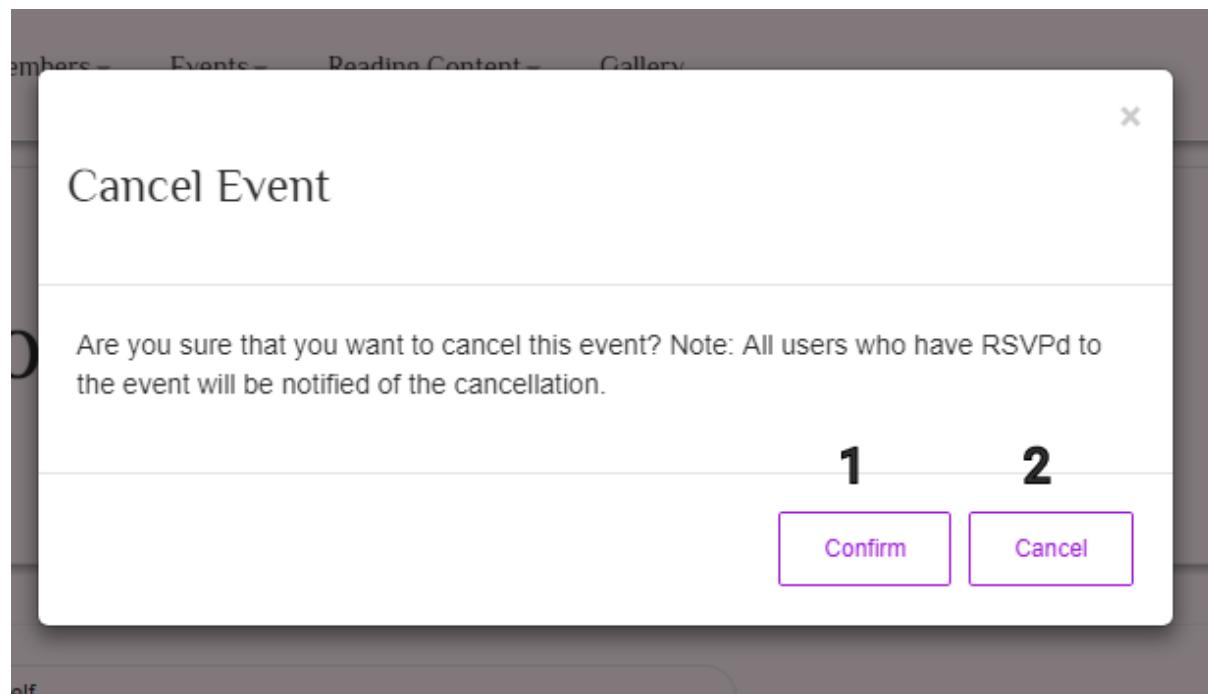


Figure 259: 5.7.1 Cancel Event VA ComEng

Number	Component Name	Functionality
1	Confirm Button:	The system will delete the relevant event from the <u>Event and Community Engagement</u> Tables.
2	Cancel Button:	The system will abort the cancelation of an event.

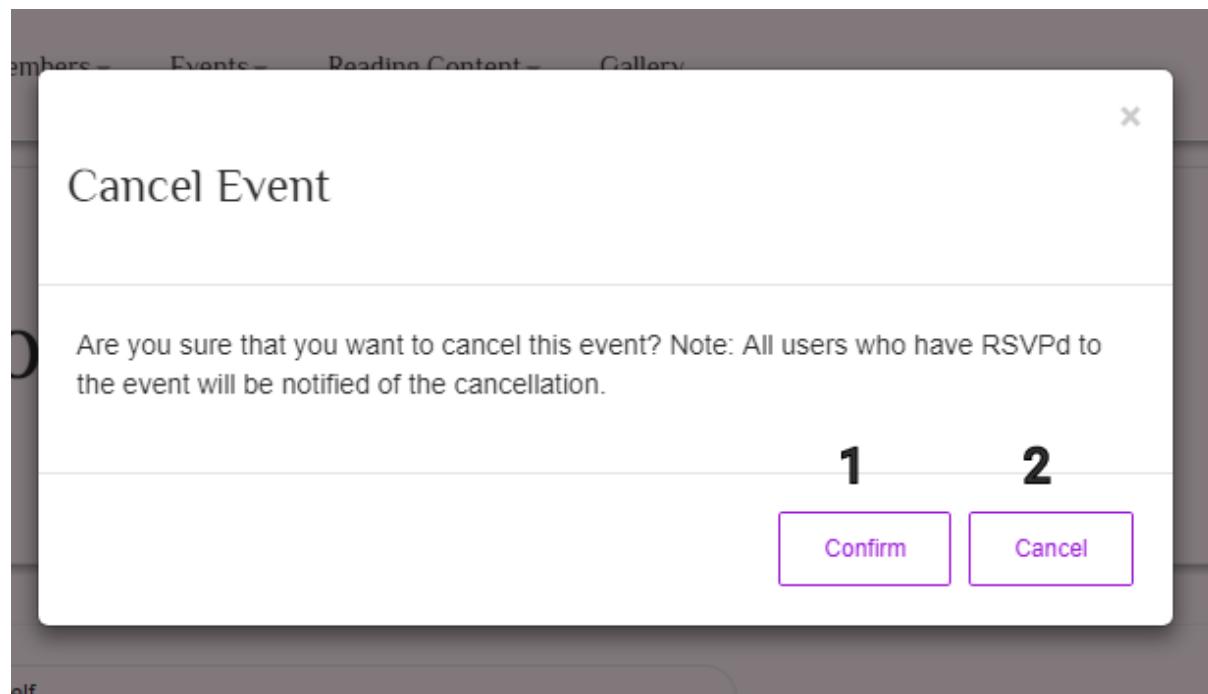


Figure 260: 5.7.1 Cancel Event VA Lecture

Number	Component Name	Functionality
1	Confirm Button:	The system will delete the relevant event from the <u>Event and Lecture</u> Tables.
2	Cancel Button:	The system will abort the cancelation of an event

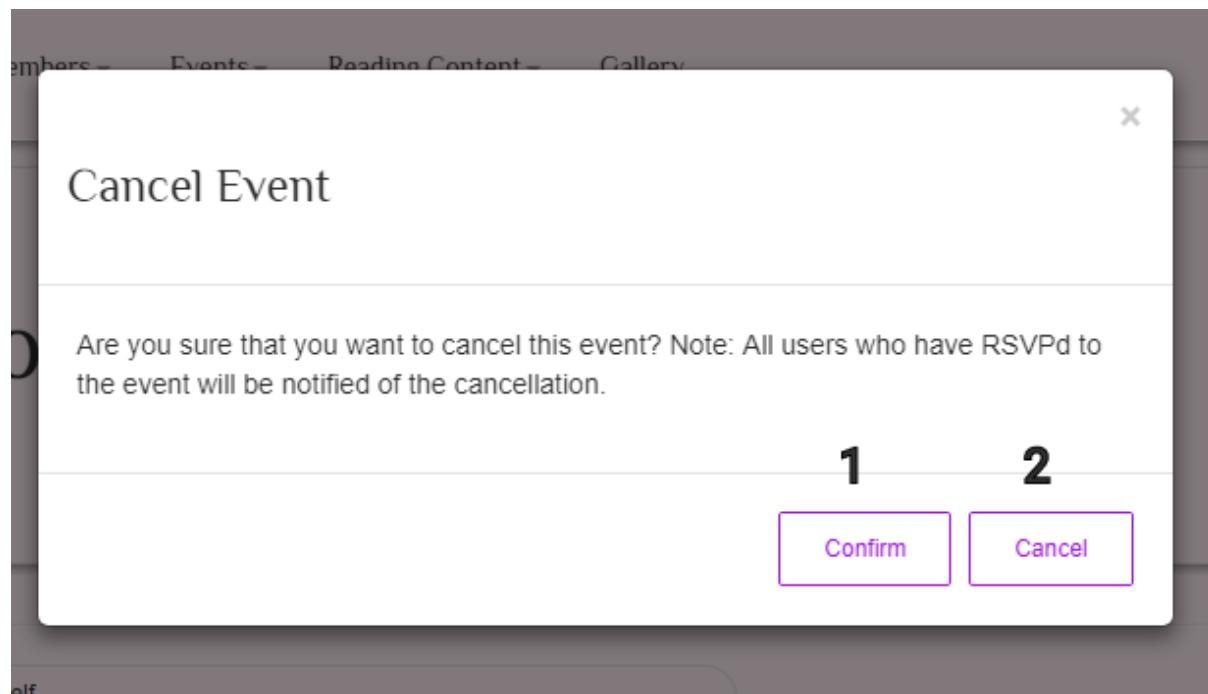


Figure 261: 5.7.2 Cancel Event Function VA

Number	Component Name	Functionality
1	Confirm Button:	The system will delete the relevant event from the <u>Event and Function</u> Tables.
2	Cancel Button:	The system will abort the cancelation of an event

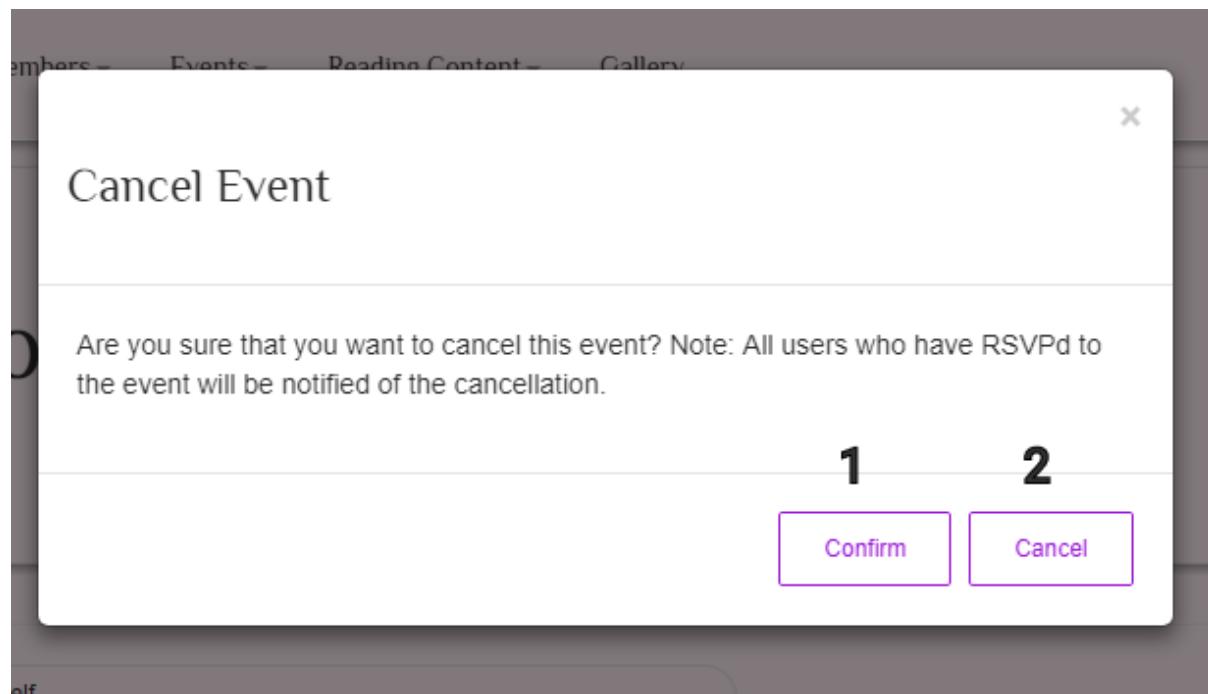


Figure 262: 5.7.2 Cancel Event VA From home page

Number	Component Name	Functionality
1	Confirm Button:..	The system will delete the relevant event from the <u>Event and Community Engagement</u> Tables
2	Cancel Button:	The system will abort the cancelation of an event

The screenshot shows the 'Log Attendance' page of the TRWLA Management System. At the top, there is a navigation bar with links for 'TRWLA Management System', 'Members', 'Events', 'Reading Content', and 'Gallery'. On the right side of the navigation bar are icons for email and user profile, with a dropdown menu labeled 'User'. Below the navigation bar, the main content area has a pink header with the title 'Log Attendance'. The main body of the page is white and contains the following elements:

- 1**: A search bar with the placeholder text 'Search for a Student'.
- 2**: A button labeled 'Search for a Student'.
- 3**: A table listing student information. The columns are 'Name', 'Surname', 'Student Number', and 'Log Attendance' (which is a hyperlink). The data rows are:

Name	Surname	Student Number	Action
Achal	Seechoonparsad	15223023	Log Attendance
Jackie	Lawler	15213656	Log Attendance
Cailin	Smith	12012563	Log Attendance
Amolishwa	Moloko	12312623	Log Attendance
Christine	Oakes	12031236	Log Attendance
Susan	Botha	12312365	Log Attendance
Leselie	Lovegood	12546236	Log Attendance
Debbie	Debshire	15554236	Log Attendance
Lora	Lona	15236459	Log Attendance
- 4**: A button labeled 'Add a New Student'.
- 5**: A button labeled 'Return'.

Figure 263: 5.8.1 Log Event Attendance VA

Number	Component Name	Functionality
1	Search textbox:	The user will enter their search parameters for a student here. It can only be string value.
2	Search for student button:	The system will then take the string parameter and execute a LINQ query to retrieve the data from the Student table.
3	Table:	The table's columns will inherit the attributes of the student table: Name, Surname and student number.
4	Log attendance hyper link:	Once clicked, it will direct the user to the Log attendance modal.

5	Add new Student Button:	This button will allow the volunteer to add a new student.
	Return button:	

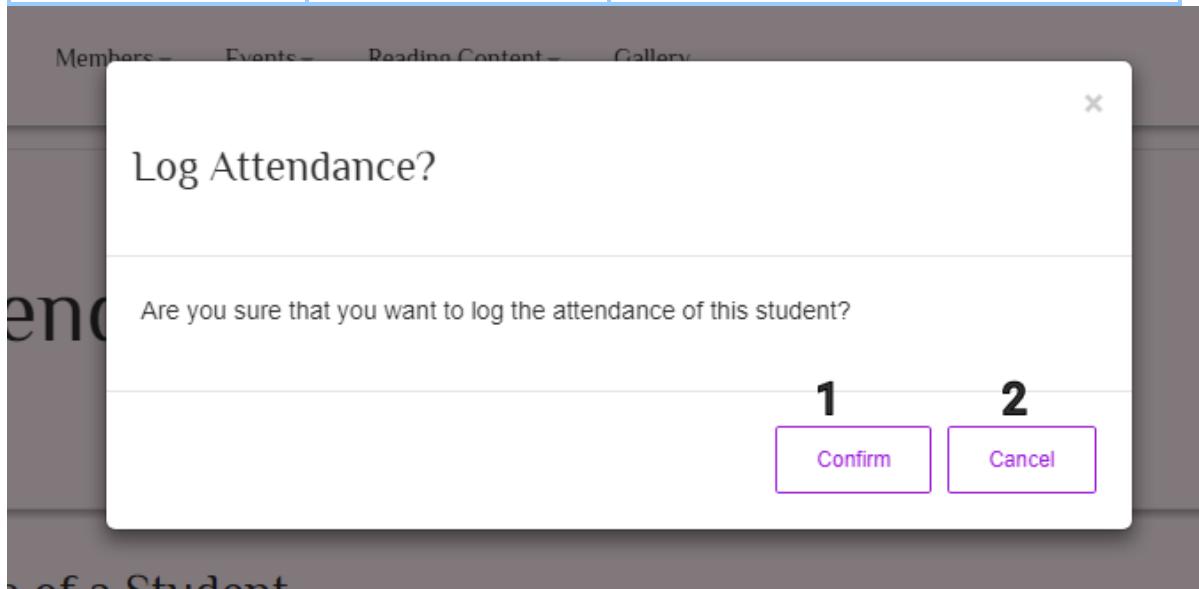


Figure 264: 5.8.3 Log Event Attendance VA

Number	Component Name	Functionality
1	Confirm button:	This will update the student's attendance in the <u>Attendance</u> table.
2	Cancel button:	This will abort the logging of the student's attendance.

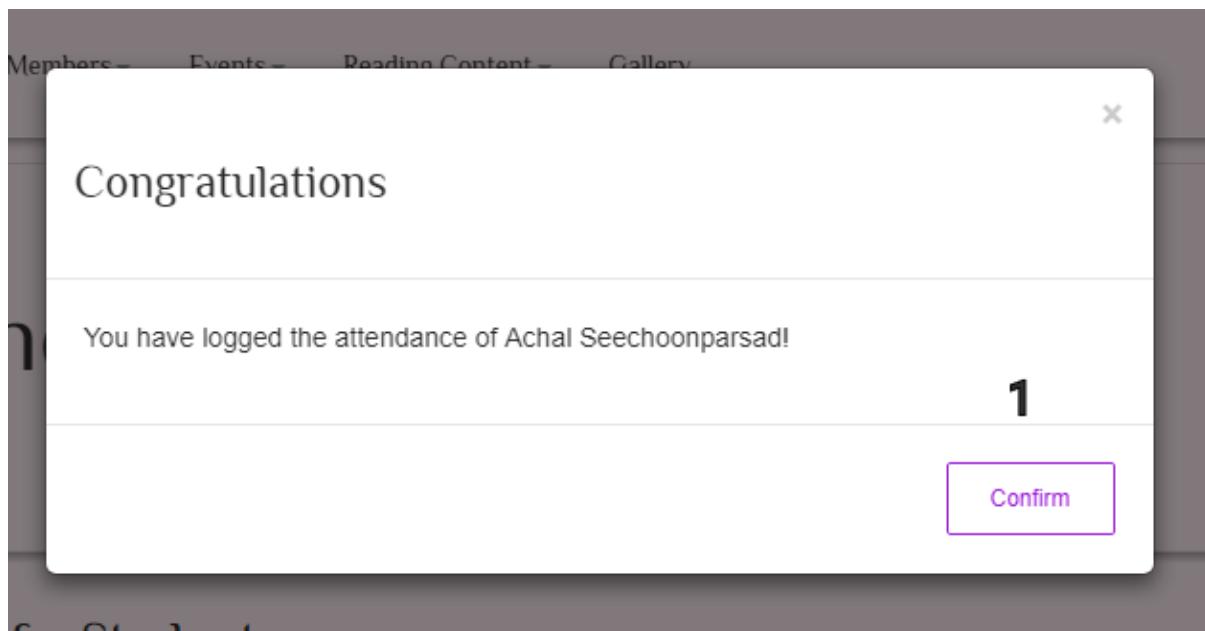


Figure 265: 5.8.4 Log Event Attendance VA

Number	Component Name	Functionality
1	Confirm button:	Will redirect the user to Log Attendance Page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Send a Notification

Please select your recipients

1

2

3

Name	Surname	Student Number
Achal	Seechoonparsad	15223023
Jackie	Lawler	15213656
Cailin	Smith	12012563
Amolishwa	Moloko	12312623
Christine	Oakes	12031236
Susan	Botha	12312365
Leselie	Lovegood	12546236
Debbie	Debishiie	15554236
Lora	Lona	15236459

4 [Confirm Recipients](#) [Return](#) 5

Figure 266: 5.9.1 Send Notification VA

Number	Component Name	Functionality
1	Search textbox:	The user will enter their search parameters for a recipient here. It can only be string value.
2	Search recipient's events by residence button:	This will retrieve the volunteer's name by residence. Will execute a LINQ query by matching the search parameters in the Residence table.

3	Table:	
4	Confirm Recipients Button:	This will hold the selected recipients in local storage to allow for a bulk notification to be sent.
5	Return button:	Will redirect the user to the events home page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Send Notification

Please fill in your message

This is a message to all those who I would like to send this message to

1
2
3

[Send Notification](#)
[Return](#)

Figure 267: 5.9.3 Send Notification VA

Number	Component Name	Functionality
1	Text area:	The user will enter their message here. Has to be string with maximum 300 characters.
2	Send notification button:	Will launch a send message modal.
3	Return button:	Will redirect the user the to the Send notification page.

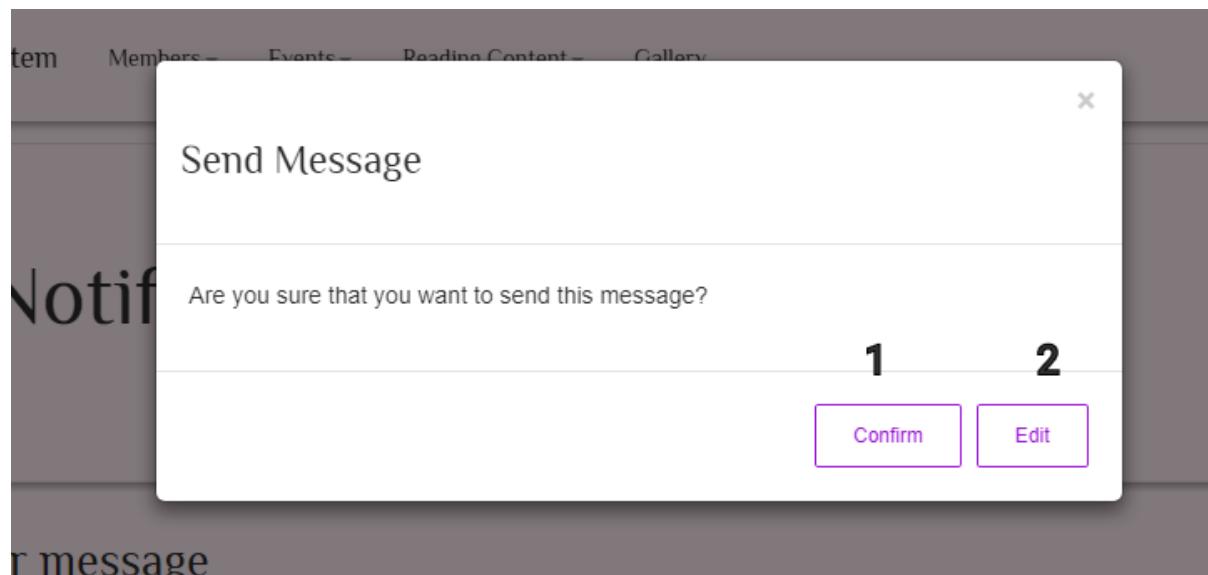


Figure 268: 5.9.4 Send Notification VA

Number	Component Name	Functionality
1	Confirm button:	Will post an HTTP request to the server and send the notification to the recipient's accounts.
2	Edit button:	Will allow the user to go back and edit the textarea.

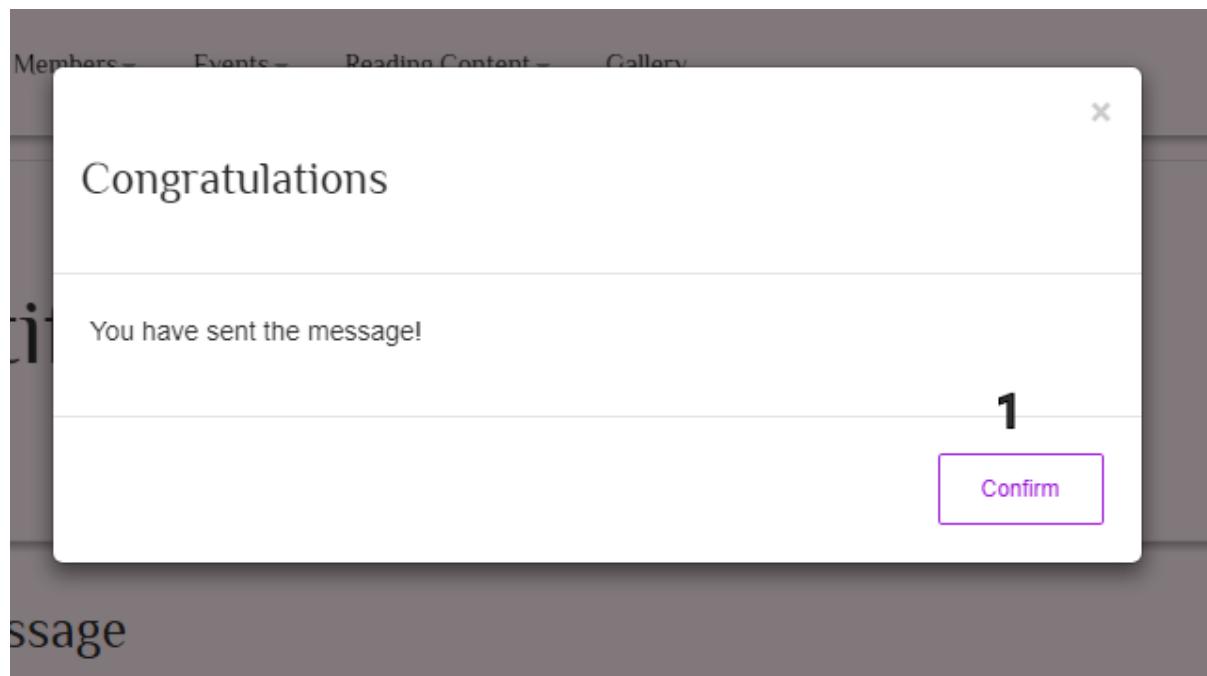


Figure 269: 5.9.5 Send Notification VA

Number	Component Name	Functionality
1	Confirm button:	Will redirect the user to the send notification page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Guest Speakers

1
2 Search for a Guest Speaker by First Name

FirstName	LastName	Phone	EmailAddress	PictureLink	3
Christopher	Oakes	834074027	u15213626@tuks.co.za	http://hdimages.org/wp-content/uploads/2017/03/placeholder-image10.jpg	4 Details
Achal	Seechoonparsad	834024023	u13623727@tuks.co.za		Details
Keith	Kichenbrand	834872256	keithl@gmail.com		Details
None	None	839999999	None		Details
Sarah	Oakes	832364563	sarah@gmail.com	www.google.com	Details

5 Register Guest Speaker **6** Update **7** Delete **8** Return

Figure 270: 6.0.1 Function VA

Number	Component Name	Functionality
1	Search textbox:	The user will enter their search parameters for the guest speaker. It can only be a string value.
2	Search Guest Speaker Textbox:	The system will match the string search parameters with the guest speakers in the Guest Speaker table.
3	Table:	The columns inherit the attributes of the Guest Speaker Table: First Name, Last Name, Phone, Email Address and Picture Link.
4	Details hyperlink:	This will redirect the volunteer to the details page of the Guest Speaker.
5	Register Guest Speaker Button:	This button will redirect the user to the Register Guest speaker page.

6	Update button:	This will direct the user to the update guest speaker page for the selected guest speaker.
7	Delete button:	Once clicked it directs the user to the delete guest speaker page for the selected guest speaker in the table.
8	Return Button:	Will direct the user to their previous page.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Register Guest Speaker

GuestSpeaker

Name	Christopher	1
Surname	Oakes	2
Phone Number	834074027	3
Email Address	u15213626@tuks.co.za	4
Picture URL	www.google.com	5

6 [Create](#) 7 [Return](#)

Figure 271: 6.1.1 Register Guest Speaker VA

Number	Component Name	Functionality
1	Name textbox:	The user will enter the name of the guest speaker, which must be a string value 35 characters.
2	Surname Textbox:	The user will enter the guest speaker's surname which must be a string value 35 characters.

3	Phone Number Textbox:	The user will enter the guest speaker's phone number which must be an integer value.
4	Email Address Textbox:	the user will enter the guest speaker's email address which must be a string 255 characters.
5	Picture URL Textbox:	The user will insert a picture URL of the guest speaker.
6	Create Button:	Once clicked the system will store the data in the local storage of the browser and launch the confirmation modal.
7	Return button:	Will redirect the user to the Guest Speaker main page.

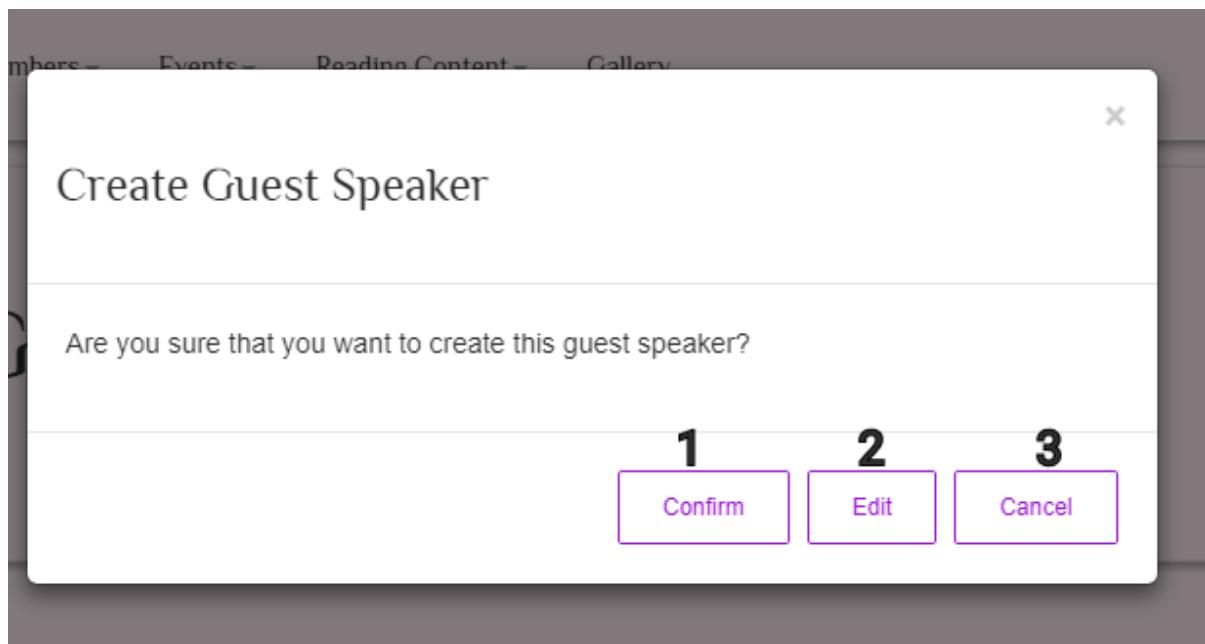


Figure 272: 6.1.2 Register Guest Speaker VA

Number	Component Name	Functionality
1	Confirm button:	The system will validate the fields and then execute a LINQ query and insert the data into the guest speaker table.
2	Edit Button:	Will allow the user to go back and editing any fields they would like.
3	Cancel Button:	Will abort the action of creating a Guest Speaker.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Christopher Oakes Details

FirstName	<input type="text" value="Christopher"/>	1
LastName	<input type="text" value="Oakes"/>	2
Phone	<input type="text" value="834074027"/>	3
EmailAddress	<input type="text" value="u15213626@tuks.co.za"/>	4
PictureLink	<input type="text" value="http://hdimages.org/wp-content/uploads/2017/03/placeholder-image10.jpg"/>	5

[6 Update](#) [7 Delete](#) [8 Return](#)

Figure 273: 6.2.2 Search Guest Speaker VA

Number	Component Name	Functionality
1	Name textbox:	The guest speaker's name will be retrieved from the <u>Guest Speaker</u> table and displayed in the textbox.
2	Surname TextBox:	The guest speaker's surname will be retrieved from the <u>Guest Speaker</u> table and displayed in the textbox.
3	Phone Number TextBox:	The guest speaker's phone number will be retrieved from the <u>Guest Speaker</u> table and displayed in the textbox.
4	Email Address TextBox:	The guest speaker's email address will be retrieved from the <u>Guest Speaker</u> table and displayed in the textbox.
5	Picture URL TextBox:	The guest speaker's picture URL will be retrieved from the <u>Guest Speaker</u> table and displayed in the textbox.

6	Update Button:	Once clicked the system will store the data in the local storage of the browser and launch the confirmation modal.
7	Delete Button:	Will launch a modal to confirm the deletion of the guest speaker.
8	Return button:	Will redirect the user to the Guest Speaker main page. Return button: Will redirect the user to the Guest Speaker main page.

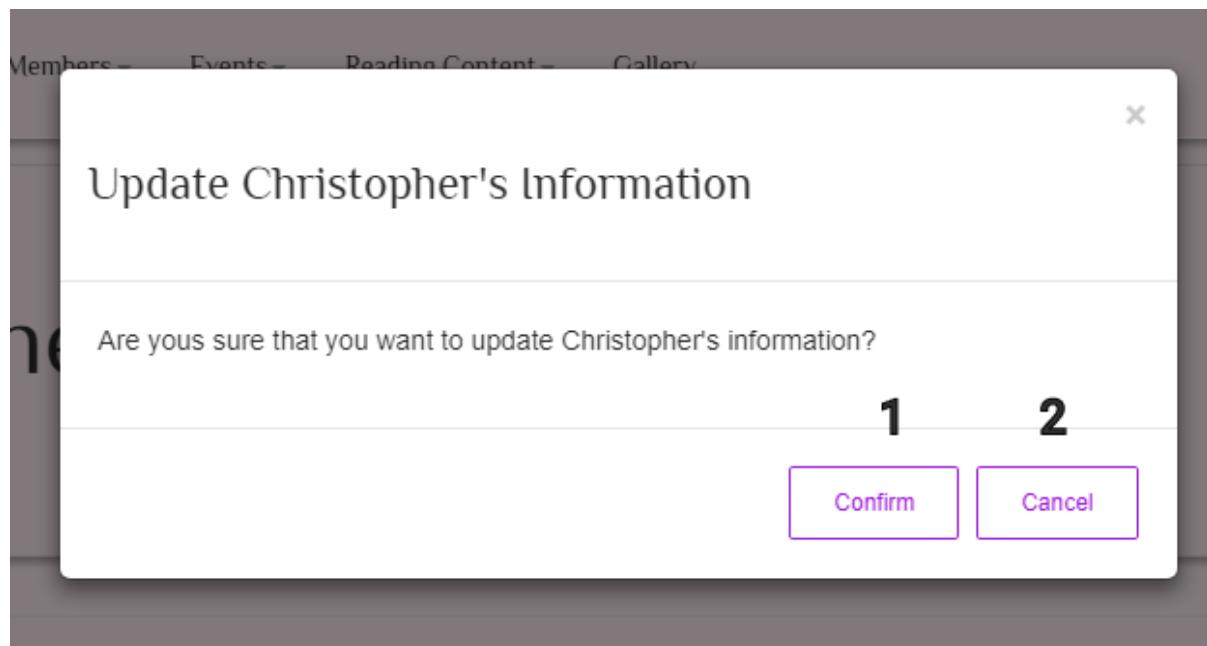


Figure 274: 6.3.1 Update Guest Speaker VA

Number	Component Name	Functionality
1	Confirm Button:	The system will validate the fields and then execute a LINQ query and insert the data into the <u>guest speaker</u> table.
2	Cancel Button:	Will abort the action of updating a Guest Speaker.

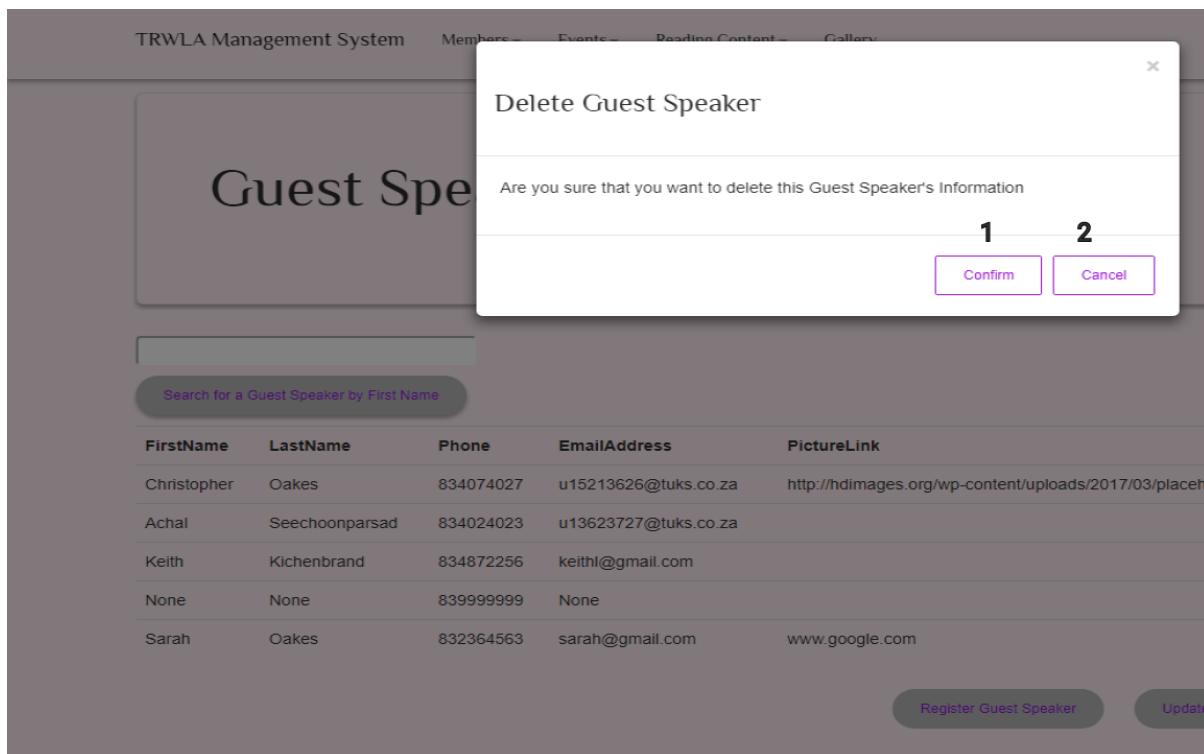


Figure 275: 6.4.1 Delete Guest Speaker VA

Number	Component Name	Functionality
1	Confirm Button:	Once clicked, the system will delete the selected guest speaker from the Guest Speaker Table.
2	Cancel Button:	Will abort the action of deleting a Guest Speaker.

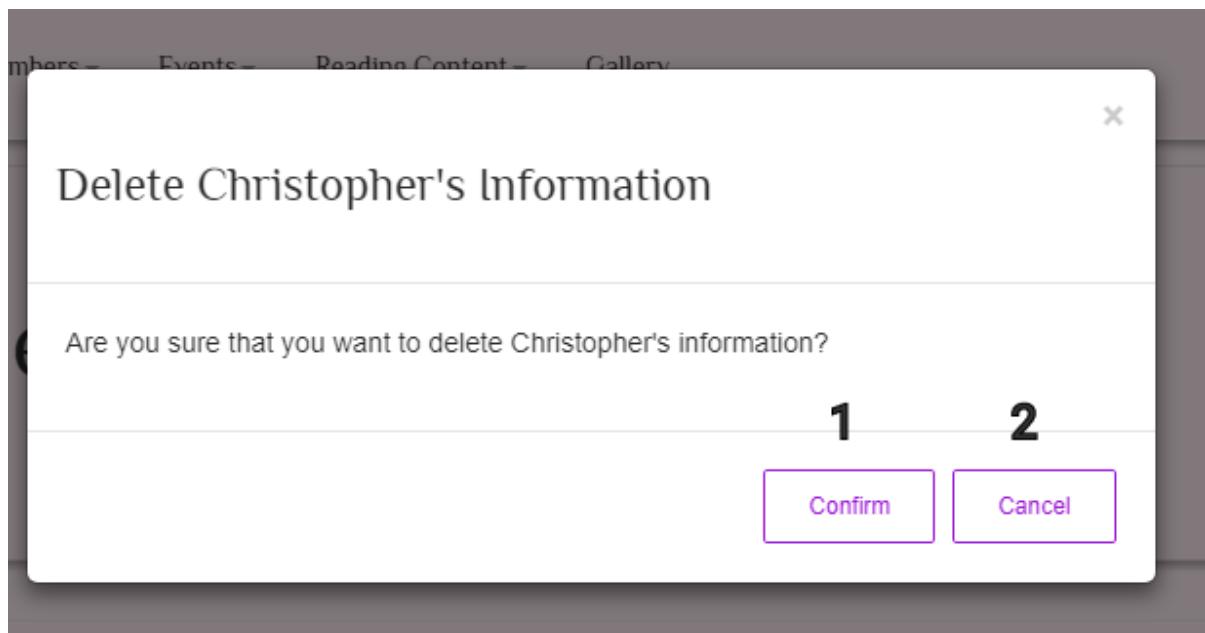


Figure 276: 6.4.2 Delete Guest Speaker VA

Number	Component Name	Functionality
1	Confirm Button:	Once clicked, the system will delete the selected guest speaker from the <u>Guest Speaker</u> Table.
2	Cancel Button:	Will abort the action of deleting a Guest Speaker.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) [User ▾](#)

Content Management

Content Type	Name	Author	Date Published	Link	Theme	Status	Description	Details
Lecture	Looking Forward	Chris Oakes	1905/28/00	www.google.com	The Future	1	Take a step into the future and plan your life	4

[5 Upload Content](#)
 [6 Update](#)
 [7 Delete](#)
 [8 Return](#)

Figure 277: 7.0.1 Content VA

Number	Component Name	Functionality
1	Search Textbox:	The user will enter their search parameters for the content. It is a search string value that will be accepted by the textbox.
2	Search Content Button:	Once clicked, the system will execute a LINQ query to match the search parameters with attribute in the Content Table
3	Table:	The columns of the table will inherit the attributes of the content table: Content Type, Name, Author, Date Published, Link, Theme, Status and Description.
4	Details hyperlink:	Once clicked, it will display the details page of the selected content item from the table.
5	Upload Content Button:	Once clicked the upload content page will be displayed.

6	Update Button:	Once clicked the update content page will be displayed.
7	Delete Button:	Once clicked the delete content page will be displayed.
8	Return button:	Once clicked the system will redirect the user back to the page they were on.

TRWLA Management System Members ▾ Events ▾ Reading Content ▾ Gallery [✉](#) User ▾

Create Content

Content Type Lecture **1**

Name Looking Forward **2**

Author Chris Oakes **3**

Date Published 2017/07/27 **4**

Content Link www.google.com **5**

Theme Acceptance **6**

Status **7**

Description This is where you will look forward **8**

9 **10**

[Create](#) [Return](#)

Figure 278: 7.1.1 Upload Content VA

Number	Component Name	Functionality
1	Content Type Dropdown:	The user will select between lecture or function for the content type in the dropdown list.
2	Name Textbox:	The user will enter the name of the content. Name must be a string up to 35 character long.
3	Author Textbox:	The user will enter the name of the author content. Author must be a string up to 70 characters long
4	Date Published Date picker:	The user will select the publishing date from the picker. Date Published must be in the valid date format - 10 characters in CCYY-MM-DD format.

5	Content link textbox:	The user will supply the link to the content. Content Link may be no longer than 100 characters.
6	Theme textbox:	The user will supply the name of the theme for the content. Theme must be a string no longer than 35 characters.
7	Status toggle switch:	The user will toggle the switch to determine if they want to hold onto releasing the content by locking it.
8	Description textbox:	The user will provide a description about the content. Description must be a string.
9	Create Button:	The data stored in the textboxes will be stored in local storage, then launch a confirmation modal.
10	Return Button:	Will redirect the user to Content Management Page

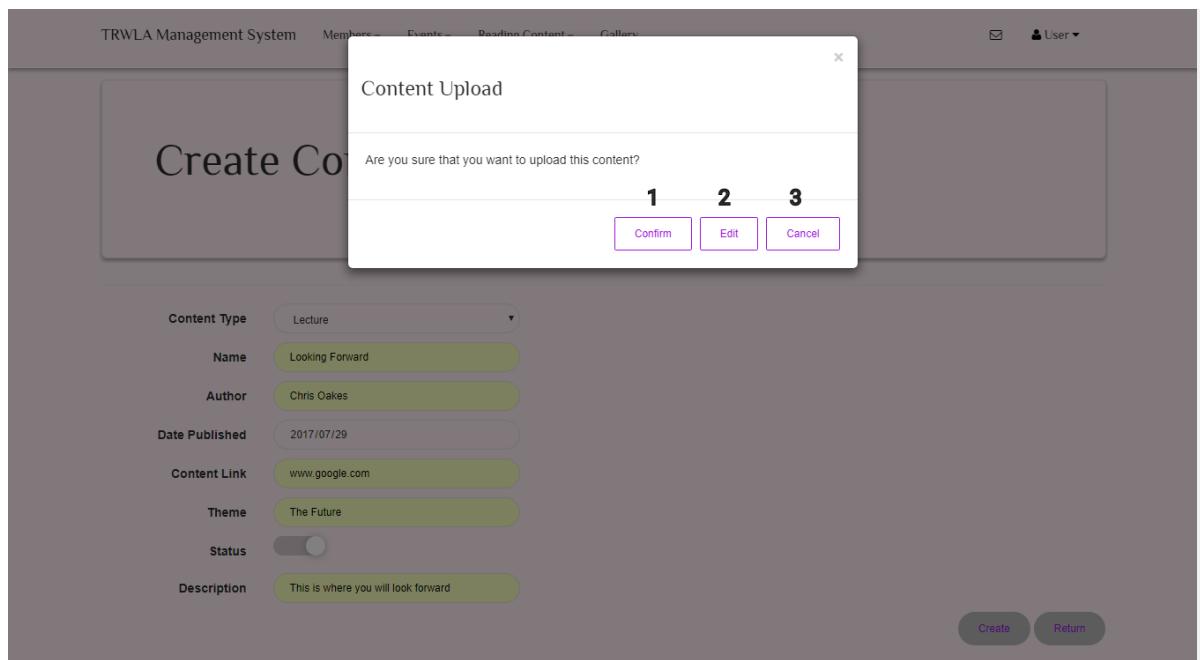


Figure 279: 7.1.2 Upload Content VA

Number	Component Name	Functionality
1	Confirm Button:	Once clicked, the system will validate the entered fields and then insert the new record in the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables
2	Edit Button:	Will allow the user to go back and edit things they want to change.
3	Cancel button:	Will Abort the uploading of content.

TRWLA Management System Home Volunteers Reading Content ▾ Gallery [✉](#) [User ▾](#)

Looking Forward Details

ContentType	Lecture	1
Name	Looking Forward	2
Author	Chris Oakes	3
DatePublished	1905/28/00	4
Link	www.google.com	5
Theme	The Future	6
Status	<input type="checkbox"/>	7
Description	Take a step into the future and plan your life 8	
>		9
;		10
		11

[Update](#) [Delete](#) [Return](#)

Figure 280: 7.2.1 Search Content VA Locked

Number	Component Name	Functionality
1	Content Type Textbox:	The system will retrieve the content type from the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables and display it in the textbox.
2	Name Textbox:	The system will retrieve the name from the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables and display it in the textbox.
3	Author Textbox:	The system will retrieve the Author from the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables and display it in the textbox.
4	Date Published Textbox:	The system will retrieve the Date Published from the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables and display it in the textbox.

5	Content link textbox:	The system will retrieve the Content link from the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables and display it in the textbox.
6	Theme textbox:	The system will retrieve the Theme from the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables and display it in the textbox.
7	Status toggle switch:	The system will retrieve the Status from the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables and display it in the textbox.
8	Description textbox:	The system will retrieve the Description from the <u>Content</u> and/or <u>LectureContent/CommContent</u> Tables and display it in the textbox.
9	Update Button:	Will enable the user to save and update the existing content record in the <u>Content</u> Table.
10	Delete Button:	Will enable the user to delete the existing record of the content from the <u>Content table.</u>
11	Return Button:	Once clicked, the user will be returned to their previous page.

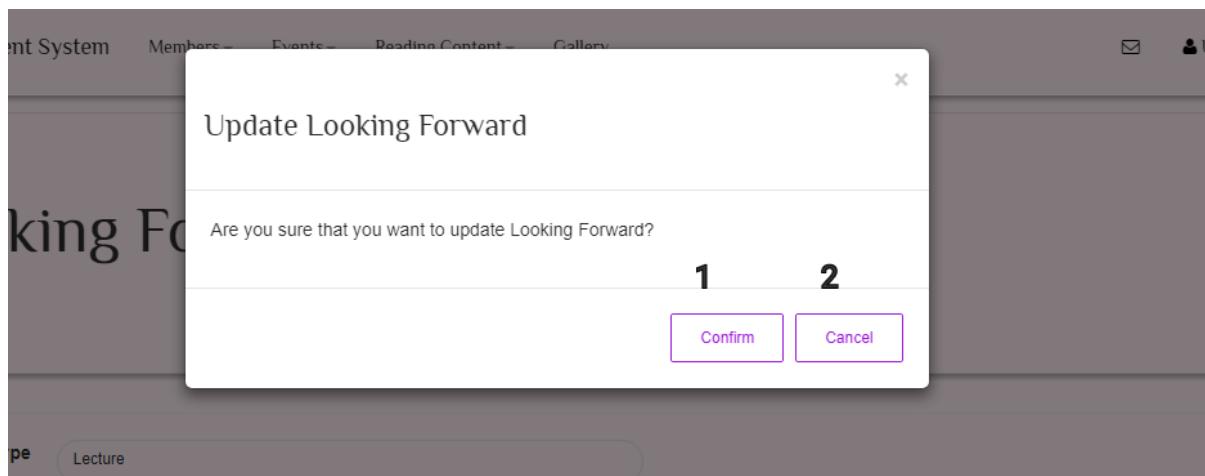


Figure 281: 7.3.2 Update Content VA

Number	Component Name	Functionality
1	Confirm Button:	The system will execute a LINQ query and update the existing record in the <u>Content</u> and/or <u>LectureContent/CommContent tables.</u>
2	Cancel Button:	Will abort the updating of any lecture content.

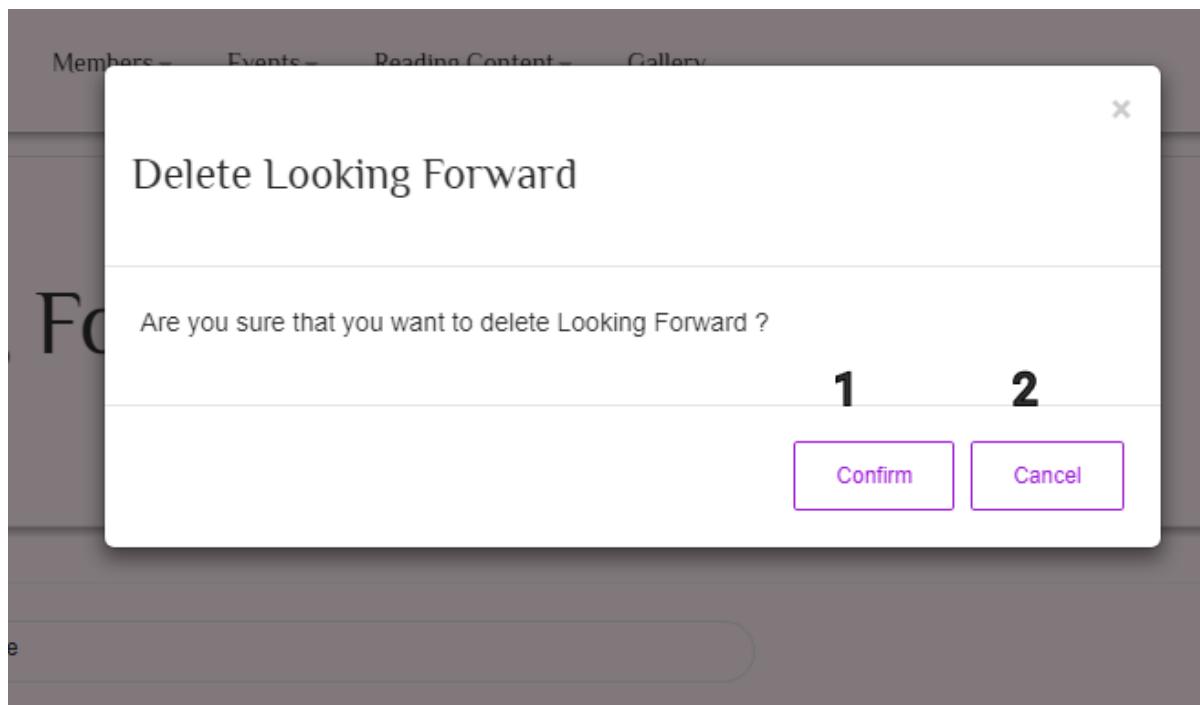


Figure 282: 7.4.1 Delete Content VA

Number	Component Name	Functionality
1	Confirm Button:	The system will execute a LINQ query and delete the existing record in the Content and/or LectureContent/CommContent tables .
2	Cancel Button:	Will abort the deletion of any lecture content.

TRWLA Management System Home Volunteers Reading Content ▾ Gallery [✉](#) [User ▾](#)

Looking Forward Details

Content Type Lecture

Name Looking Forward

Author Chris Oakes

Date Published 1905/28/00

Link www.google.com

Theme The Future

Status

Description Take a step into the future and plan your life

Review the Content of this Lecture

Leave feedback on the content:

1

a review. This is a review. This is a review.
This is a review. This is a review. This is a review.
This is a review. This is a review. This is a review.
This is a review. This is a review. This is a review.
This is a review. This is a review. This is a review.
This is a review. This is a review. This is a review.
This is a review. This is a review. This is a review.
This is a review. This is a review. This is a review.
This is a review. This is a review.

Rate the Content out of 5:

1 2 3 4

[Review Content](#) [Return](#)

Figure 283: 7.5.1 Review Lecture Content S

Number	Component Name	Functionality
1	Review Text Area:	The user will be able to leave a review on the content, which has to be a string. Maximum 300 characters.
2	Review Content Button:	Once clicked, the system will launch a confirmation modal.

3	Rating Dropdown:	The user can supply a rating to the content out of 5. Must be selected from the drop down list.
4	Return Button:	Once clicked, the user will be returned to their previous page.

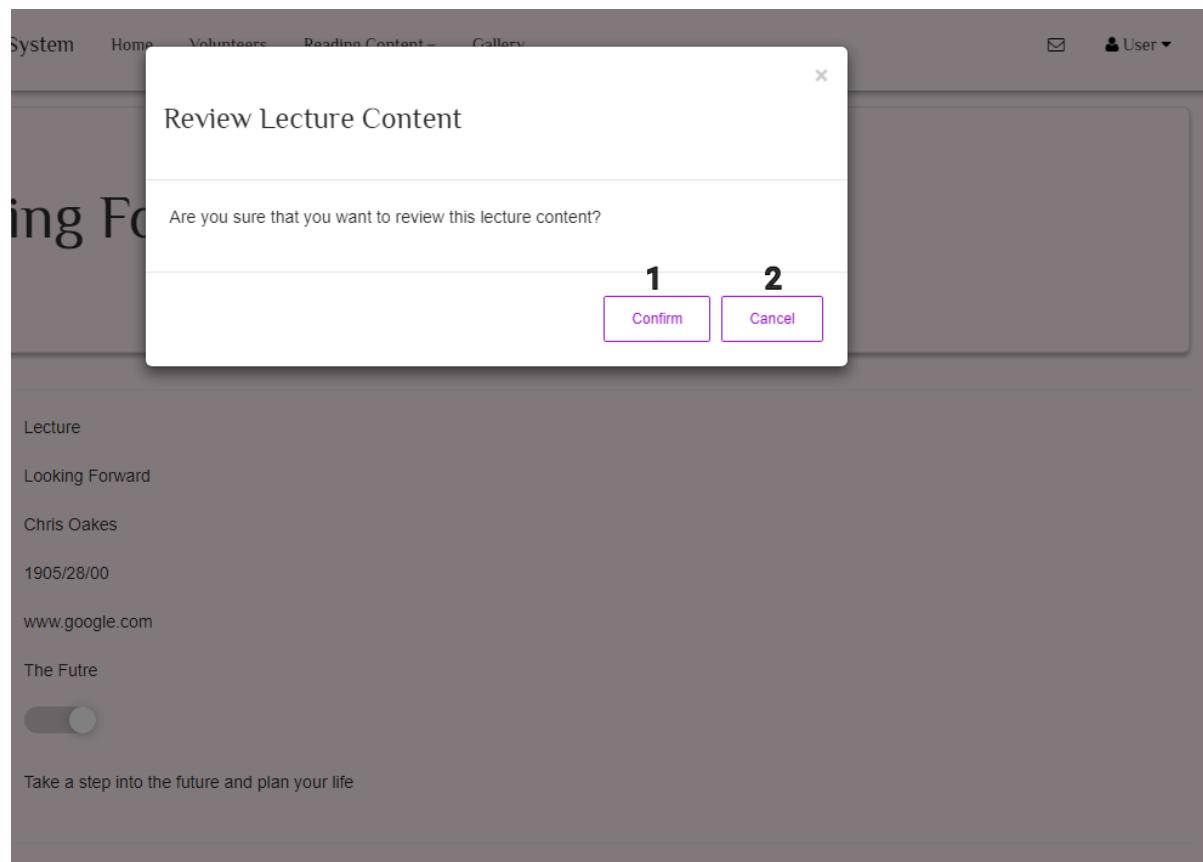


Figure 284: 7.5.2 Review Lecture Content S

Number	Component Name	Functionality
1	Confirm button:	The system will execute a LINQ query which will add the review to the Review table.
2	Cancel Button:	Will abort the completion of leaving a review.

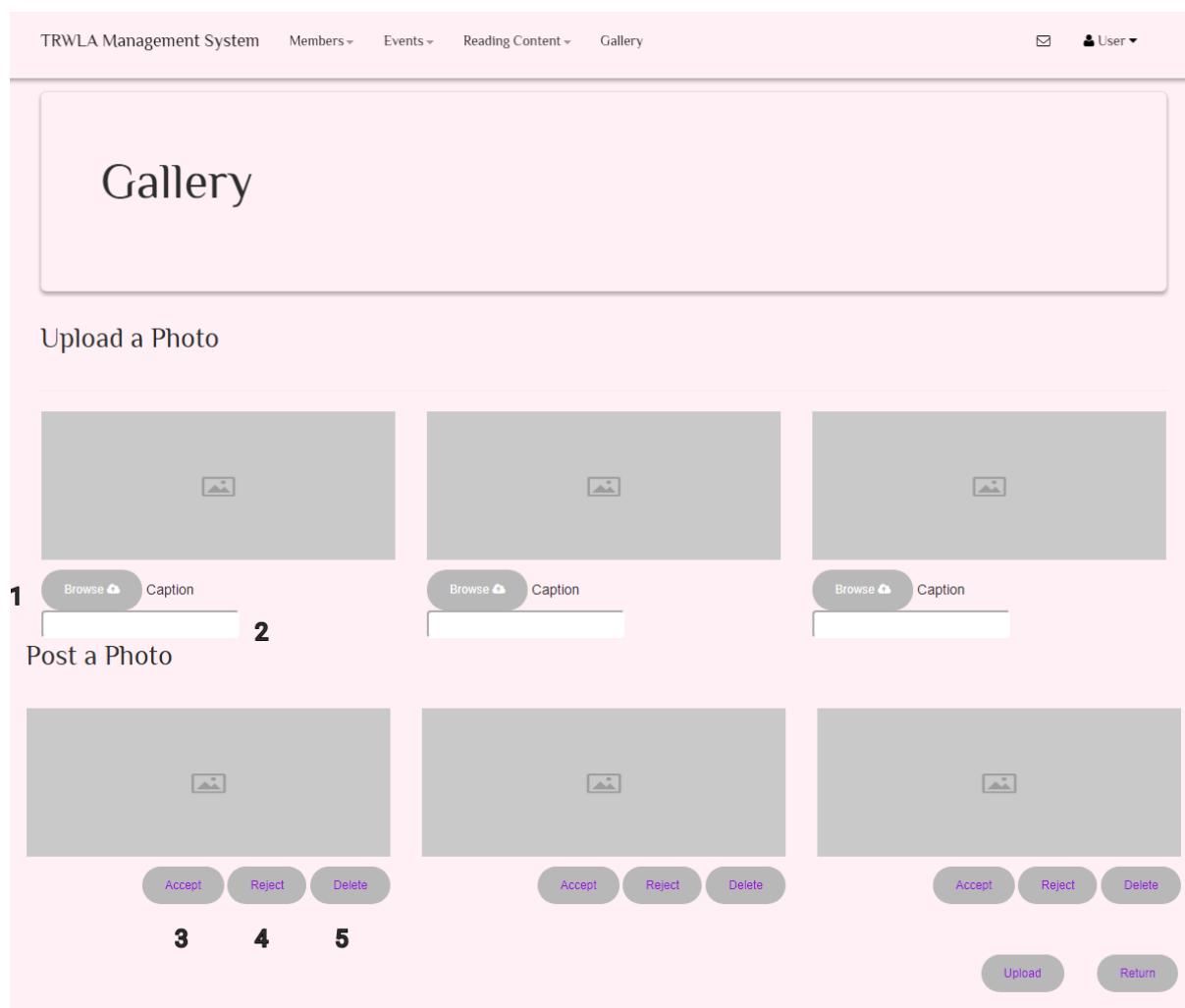


Figure 285: 8.1.1Upload Photo Admin

Number	Component Name	Functionality
1	Browse Button	Once clicked, the system will launch the 'Open File Dialog' to allow the user to upload the picture.
2	Caption Textbox	The user can supply a string value for the caption. It cannot be more than 140 characters.
3	Upload Button	The system will launch a confirmation modal.
4	Accept Button:	Once Clicked, the photo will be launched onto the gallery.

5	Reject Button:	Once clicked the photo will be allowed to be posted.
6	Delete button:	Once clicked it will delete the photo off of the system.

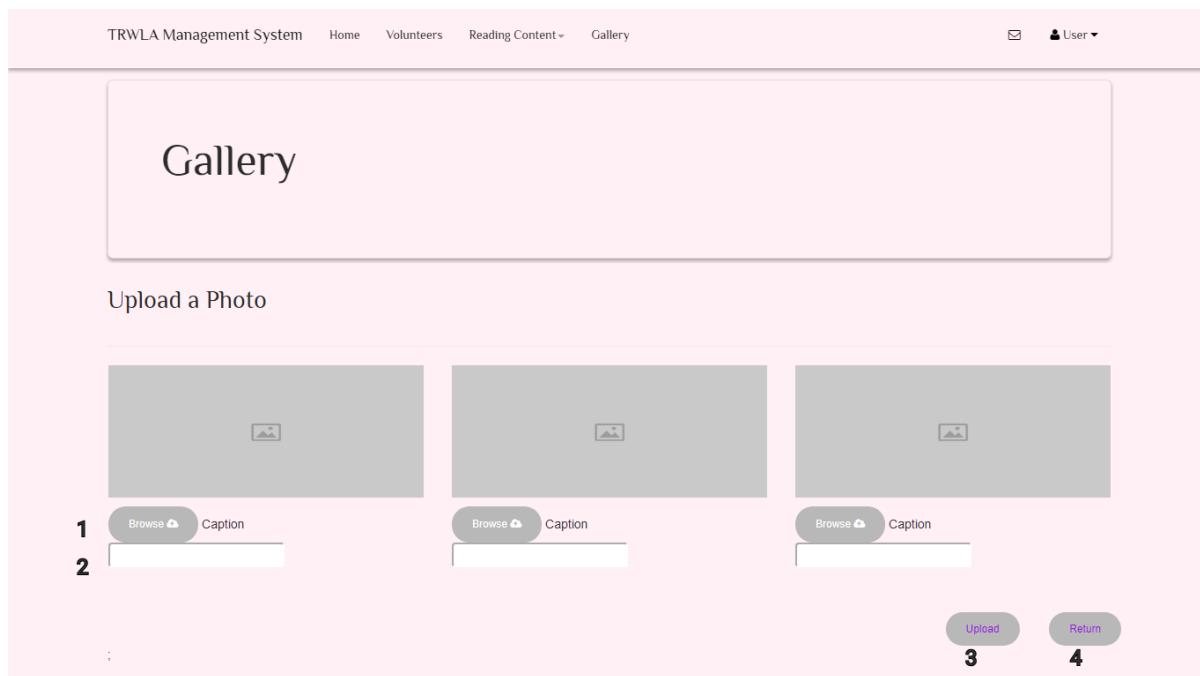


Figure 286: 8.1.1 Upload Photo S

Number	Component Name	Functionality
1	Browse Button	Once clicked, the system will launch the 'Open File Dialog' to allow the user to upload the picture.
2	Caption Textbox	The user can supply a string value for the caption. It cannot be more than 140 characters.
3	Upload Button	The system will launch a confirmation modal.
4	Return button	Once clicked, the user will be returned to their previous page.

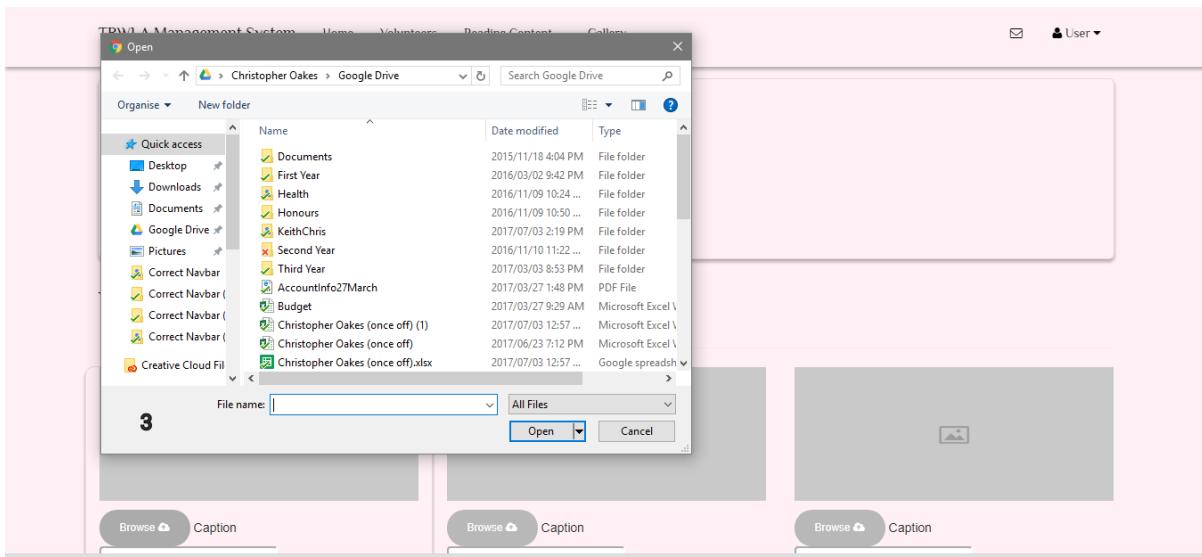


Figure 287: 8.1.2 Upload Photo S

Number	Component Name	Functionality
3	Upload File dialog	Upload File dialog: The dialog will allow the user to upload a JPG/SVG image with the size restriction of 5 megabytes.

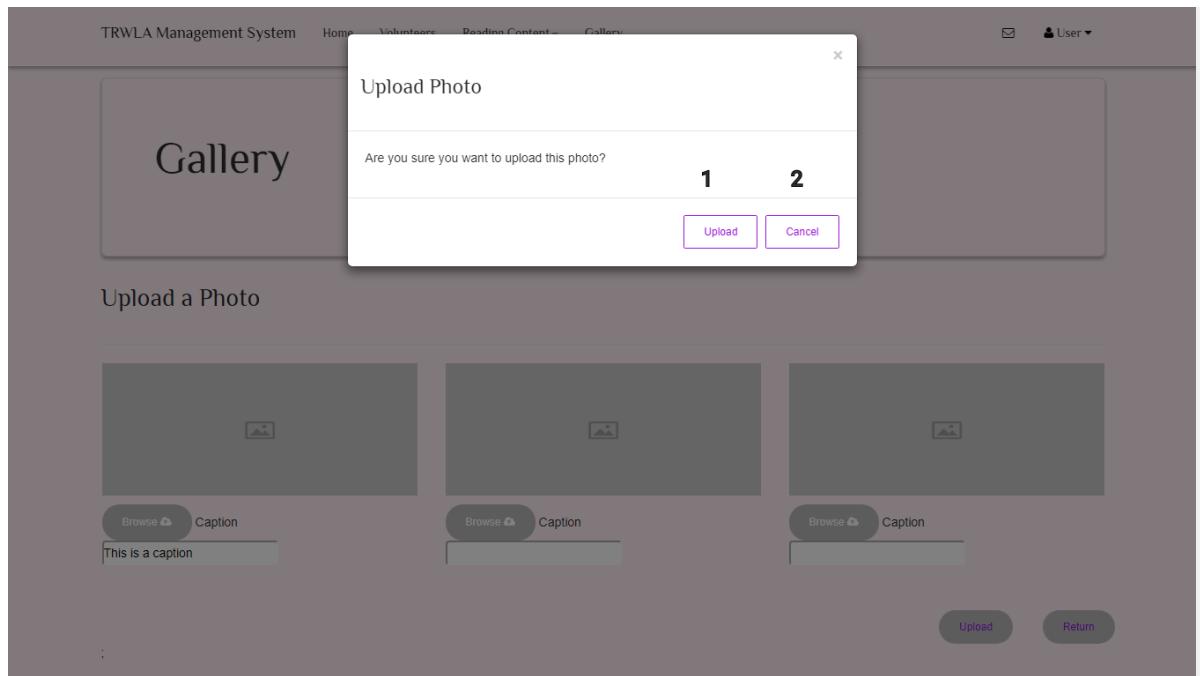


Figure 288: 8.1.4 Upload Photo S

Number	Component Name	Functionality
1	Upload button	Once clicked the system will successfully upload the image onto the system for it to be reviewed.
2	Cancel Button	The system will abort the uploading of photos.

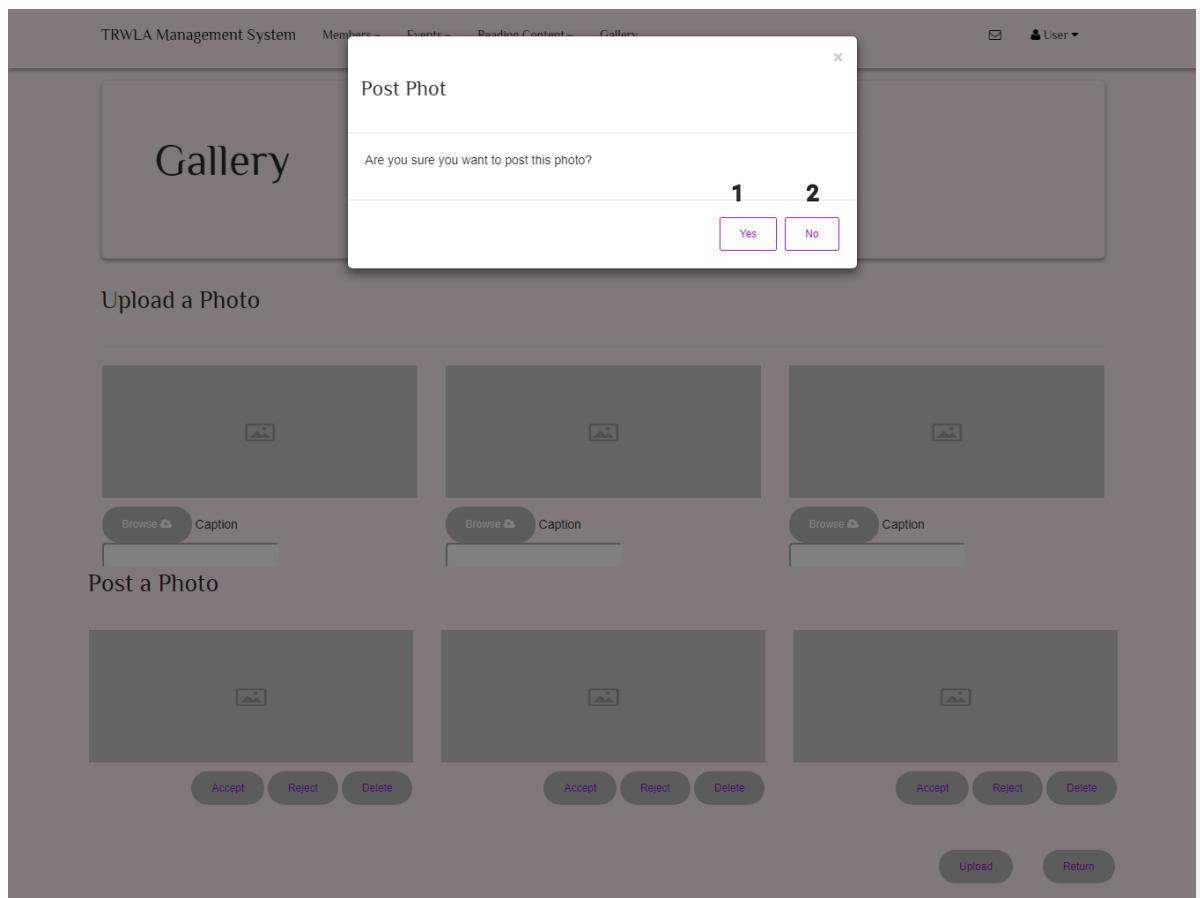


Figure 289: 8.2.1 Post Photo

Number	Component Name	Functionality
1	Yes button:	Will not post photo onto the gallery.
2	No Button:	Will redirect user to previous page.

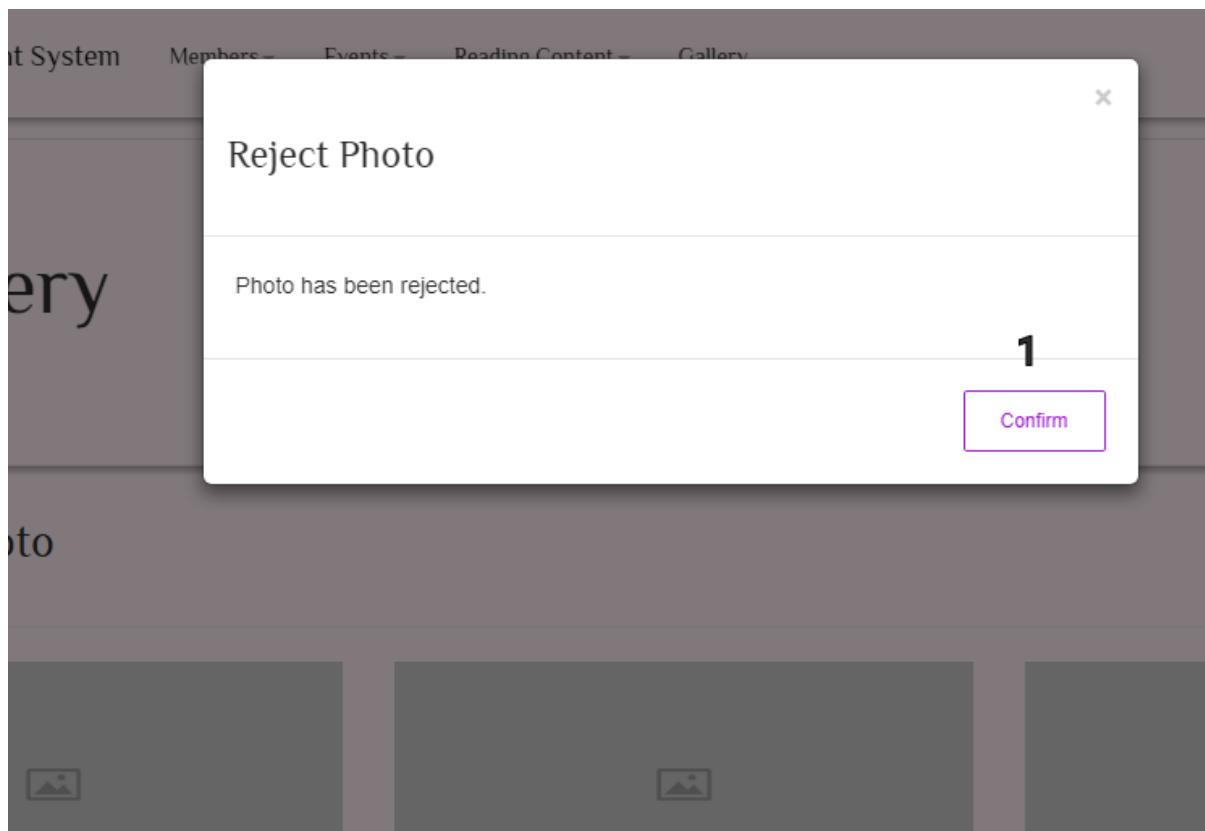


Figure 290: 8.2.2 Post Photo

Number	Component Name	Functionality
1	Confirm button	Will redirect user to previous page.

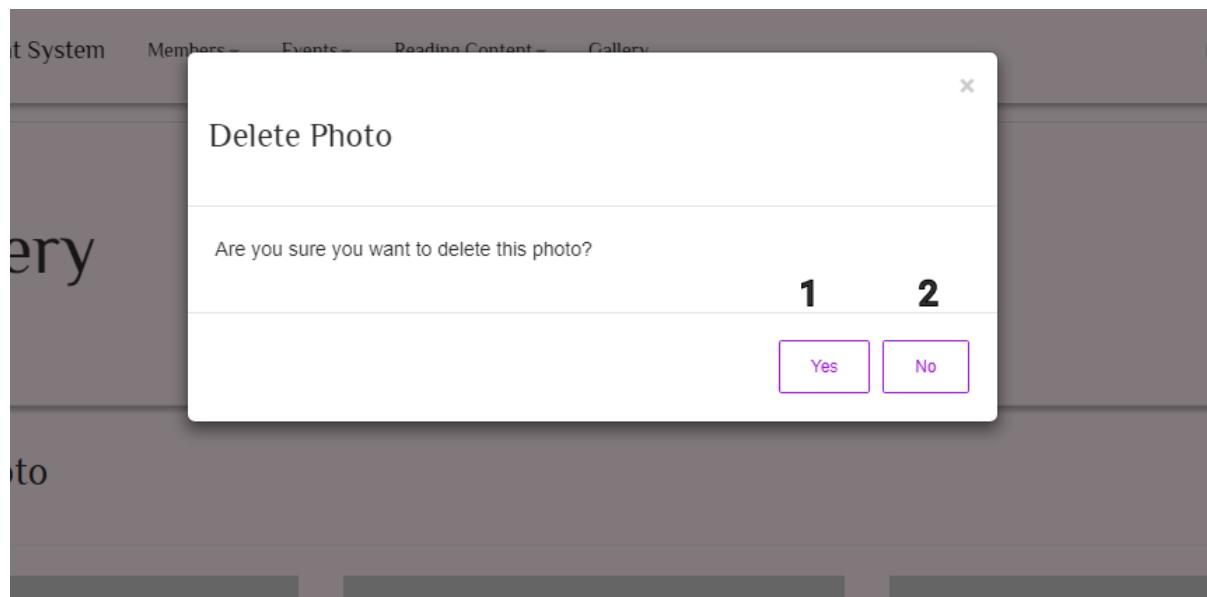


Figure 291: 8.3.1 Delete Photo

Number	Component Name	Functionality
1	Yes button:	Will Delete the photo from the system permanently.
2	No Button:	Will redirect user to previous page.

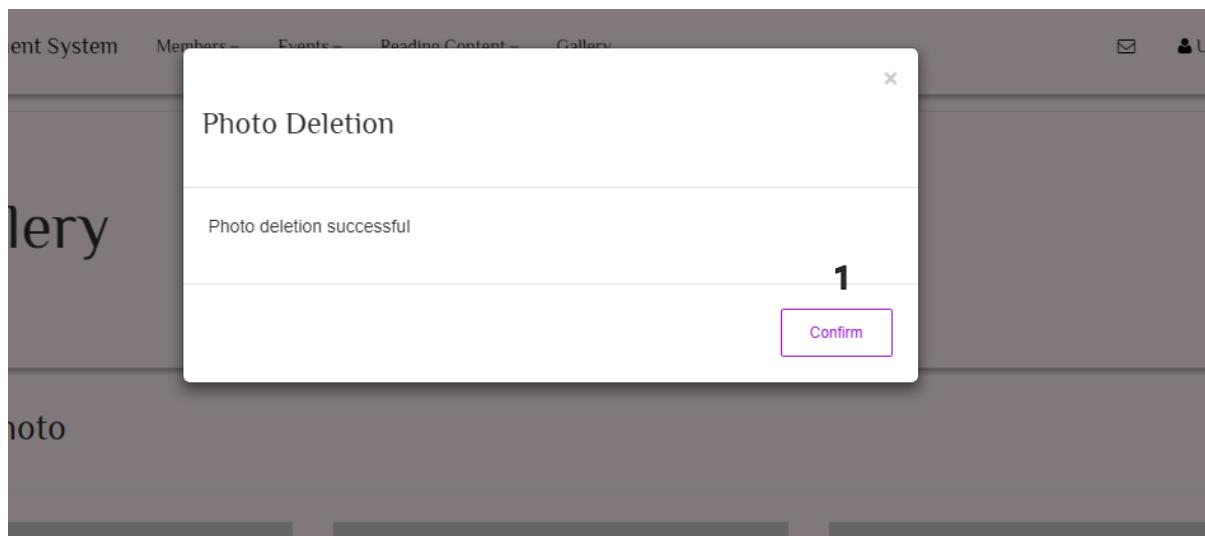


Figure 292: 8.3.2 Delete Photo

Number	Component Name	Functionality
1	Confirm button:	Will redirect user to gallery page.



Figure 293: 9.1 Generate Class Attendance

Number	Component Name	Functionality
1	Tabs: Class attendance tab.	This tab has the generated report that tracked the class attendance of registered TWRLA students.

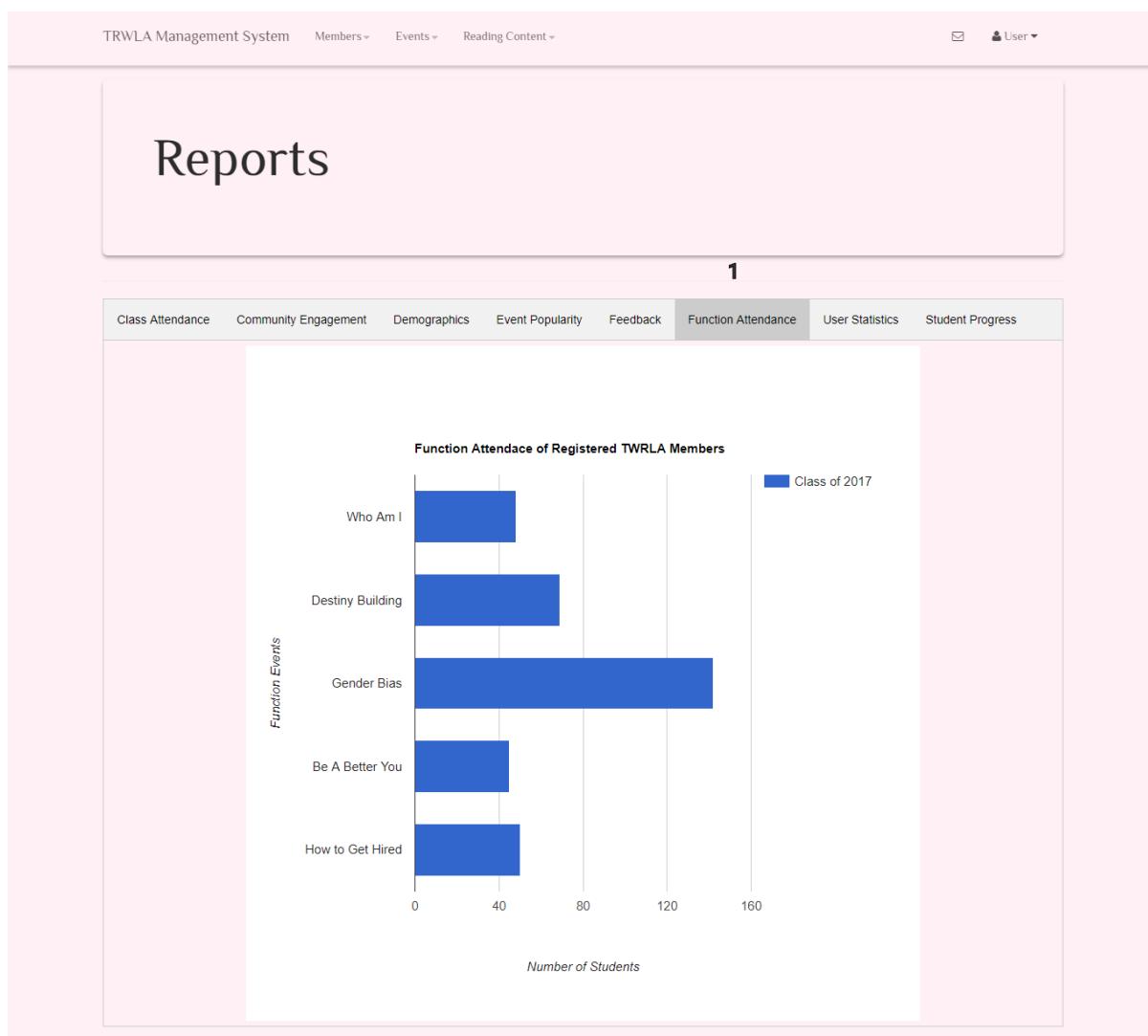


Figure 294: 9.2 Generate Function Attendance

Number	Component Name	Functionality
1	Function Attendance Tab:	This reports generates the recorded attendance of students at function attendance events.

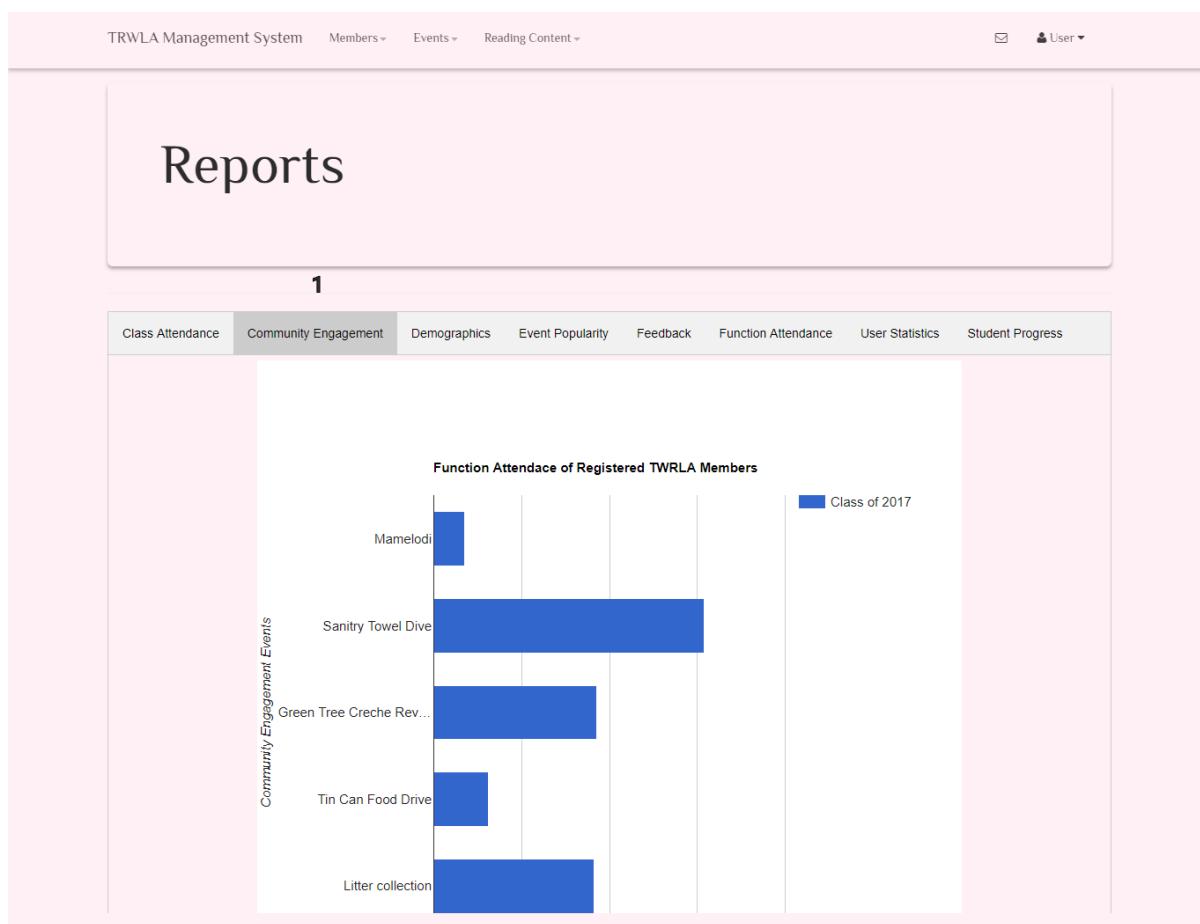


Figure 295: 9.3 Generate Com Engagement

Number	Component Name	Functionality
1	Community Engagement Tab	This reports generates the recorded attendance of students at community engagement events.

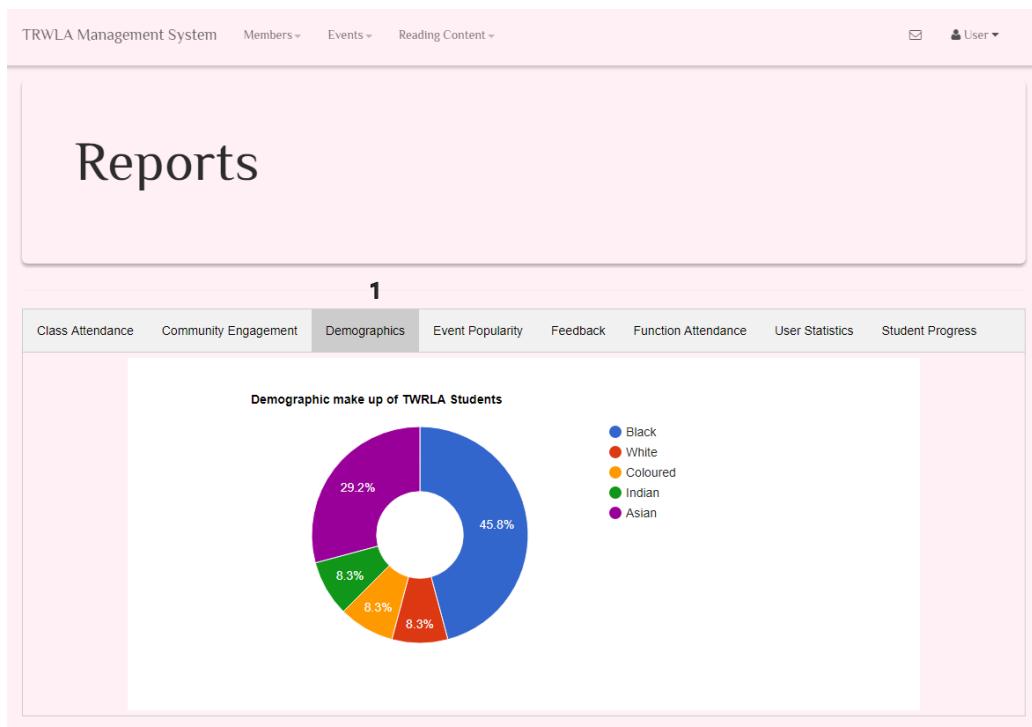


Figure 296:9.4 Generate Demographics

Number	Component Name	Functionality
1	Demographics tab:	This report generates the demographic make of all TWRLA members.

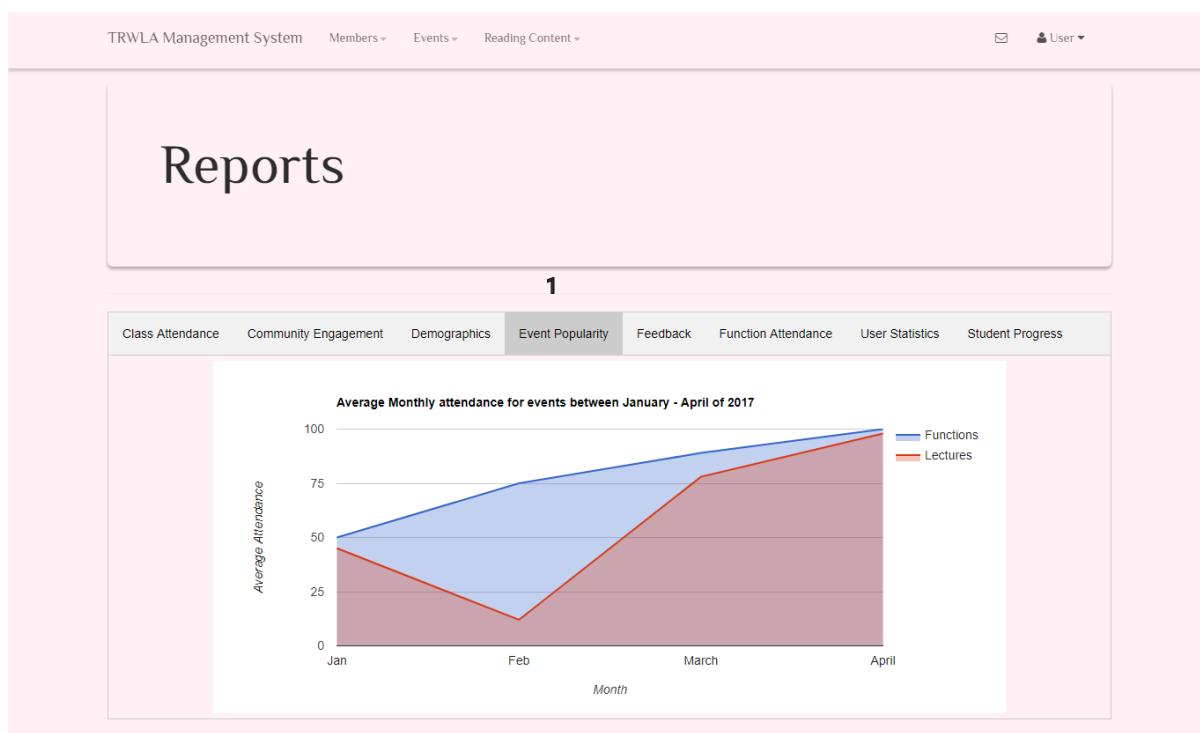


Figure 297:9.5 Generate Event Popularity

Number	Component Name	Functionality
1	Event Popularity tab:	This report generates the average monthly attendance for functions and lectures for a particular month.

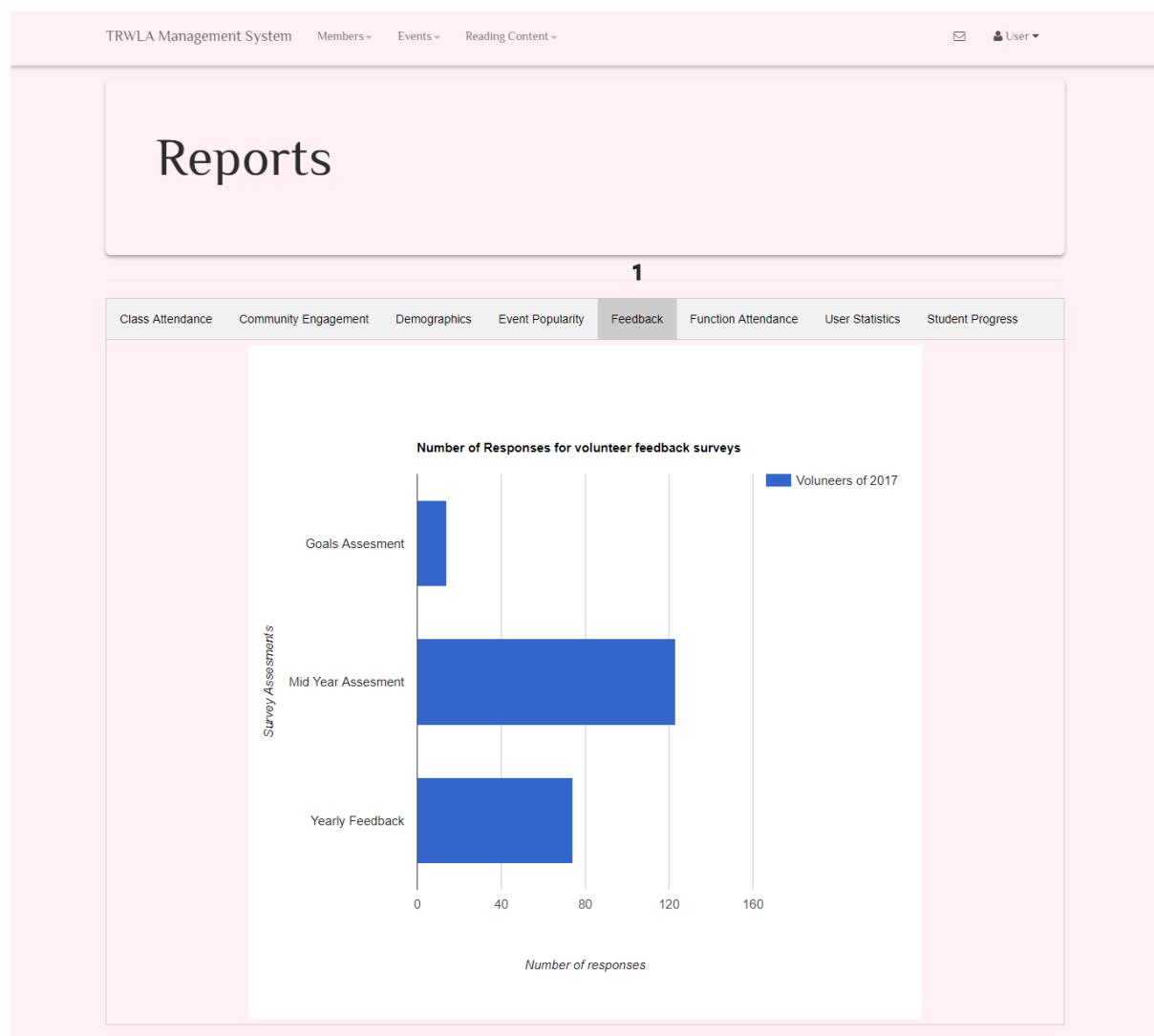


Figure 298: 9.6 Generate Feedback

Number	Component Name	Functionality
1	Feedback tab:	This report will generate the number of responses from the feedback surveys created for the volunteers.

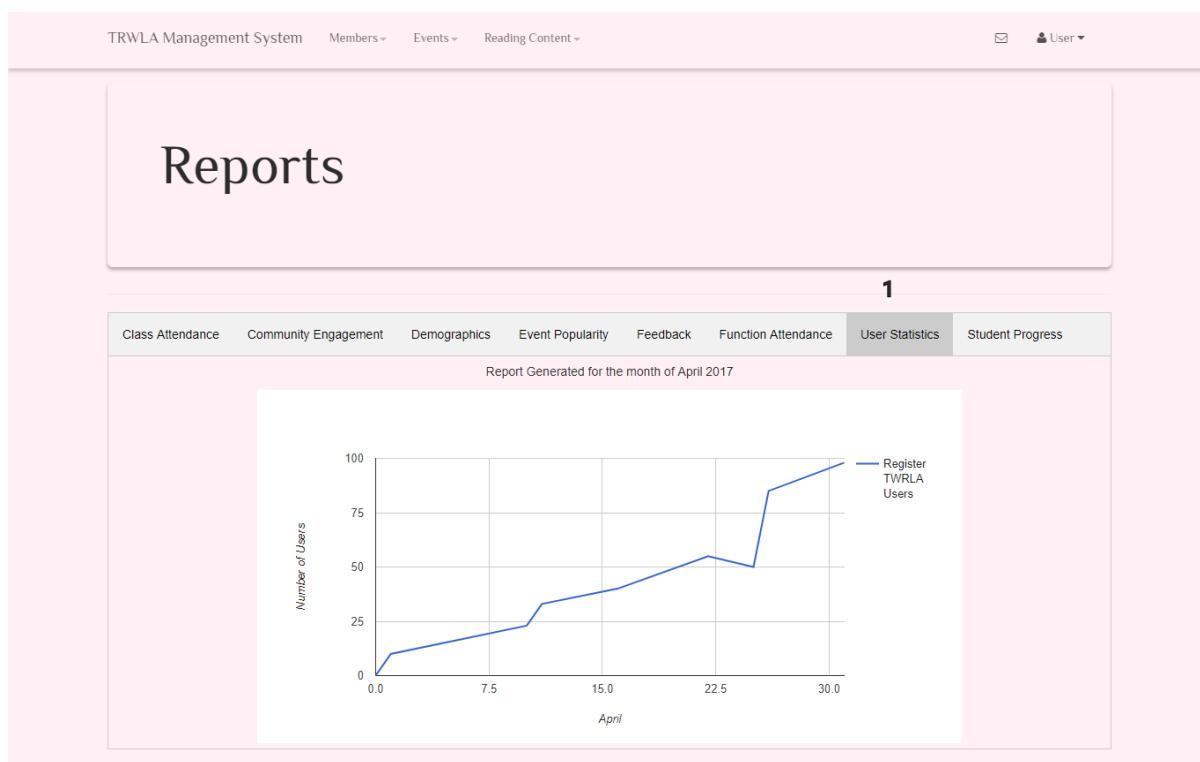
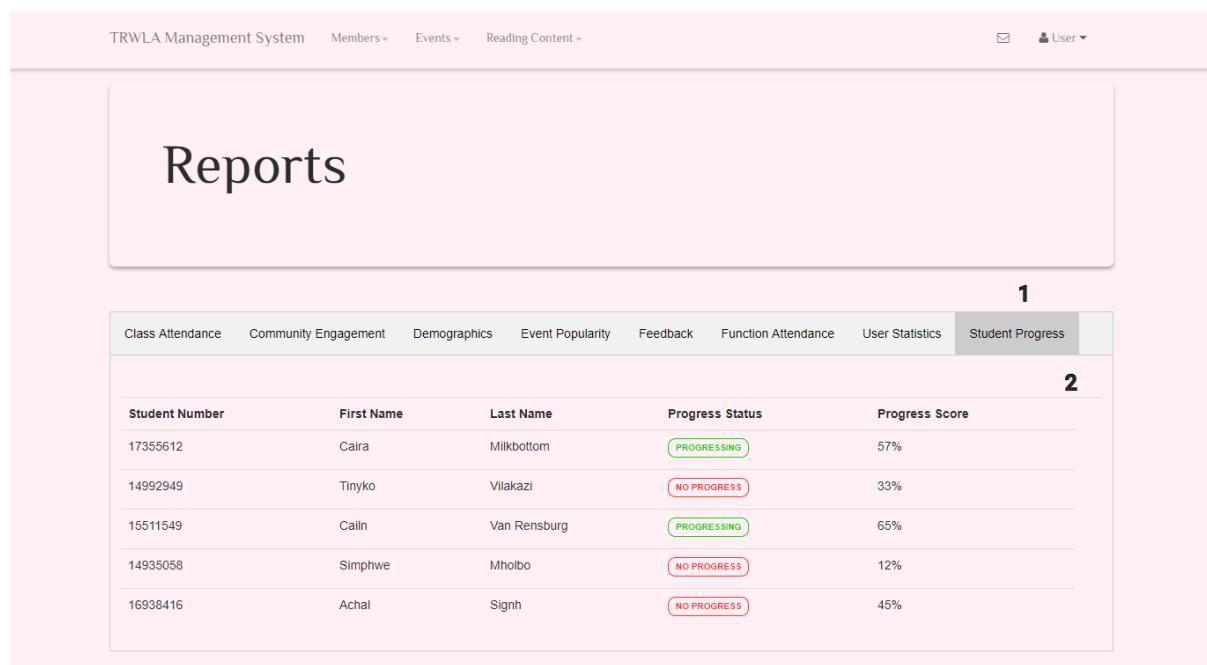


Figure 299: 9.7 Generate User Statistics

Number	Component Name	Functionality
1	User Statistics tab:	This will generate a report based on the number of active users in the database.



Student Number	First Name	Last Name	Progress Status	Progress Score
17355612	Caira	Milkbottom	PROGRESSING	57%
14992949	Tinyko	Vilakazi	NO PROGRESS	33%
15511549	Cain	Van Rensburg	PROGRESSING	65%
14935058	Simpewe	Mholbo	NO PROGRESS	12%
16938416	Achal	Signh	NO PROGRESS	45%

Figure 300: 9.8: Generate Student Progress

Number	Component Name	Functionality
1	Student Progress tab:	Will display a generated report of the registered students' progress.
2	Table:	This table has columns with attributes from the student table – Student Number, First Name, Last Name and progress Status. The last column gives the student a progress score so that the facilitators can alert students about their progress in the program.

6.3 Conclusion

6.3.1 The above section described each screen that the user interacts with which also corresponded with the use case narratives depicted above. Each screen was numbered accordingly with their description.

7. Output Design

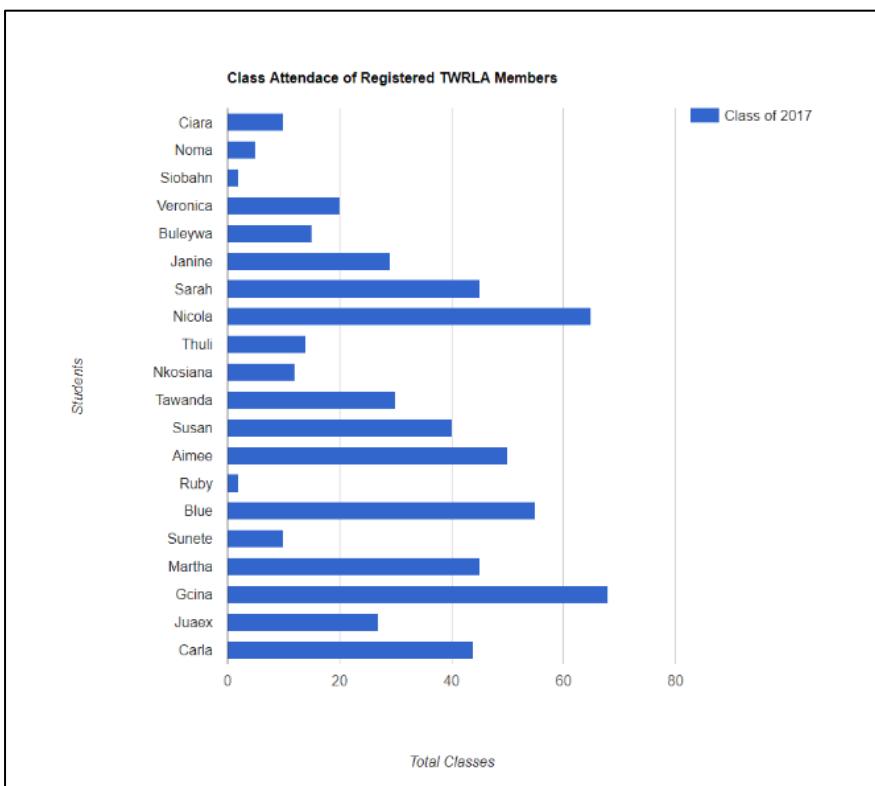
7.1 Introduction

7.1.1 The following section describes the general outputs that the system produces for the user as well as the reports that affect business decisions for the future. Each description explains what type of output it is, as well as the features it contains. The reports are presented at the end of the document as Appendix A as to not interfere with the format of the overall document.

7.2 Reports

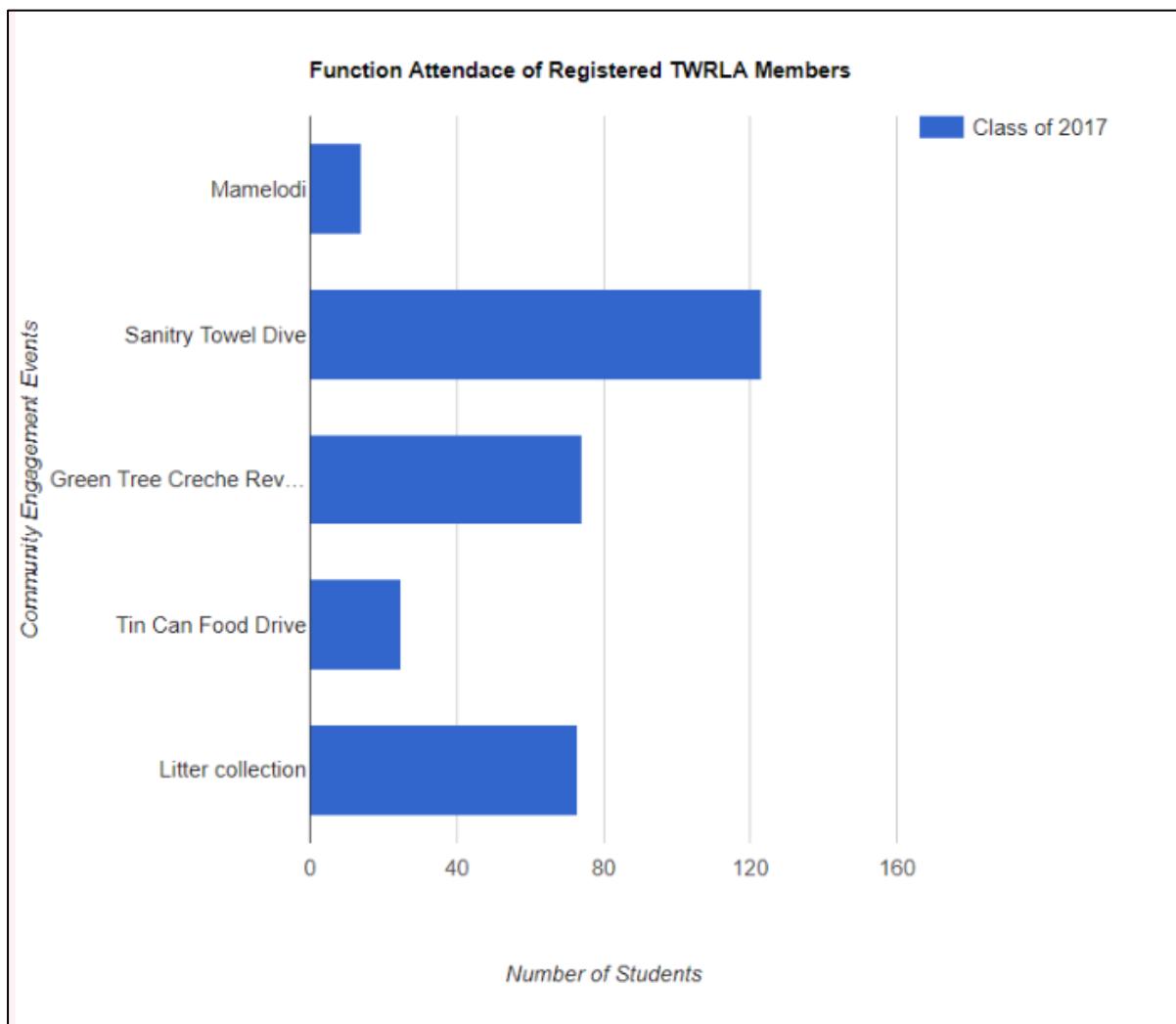
General	
Report No	9.1
Report Name	Class Attendance Report
Description	A report that determines the amount of lectures each student at the Academy attends.
Frequency	Ad hoc
Special Features	N/A
Page Orientation	Landscape
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Bar graph	Class Attendance of Registered TRWLA Members	x-axis: Total classes y-axis: Students legend: Class of ##### (year)	N/A	N/A	Calculated by counting the amount of times a student's attendance was captured at a lecture.



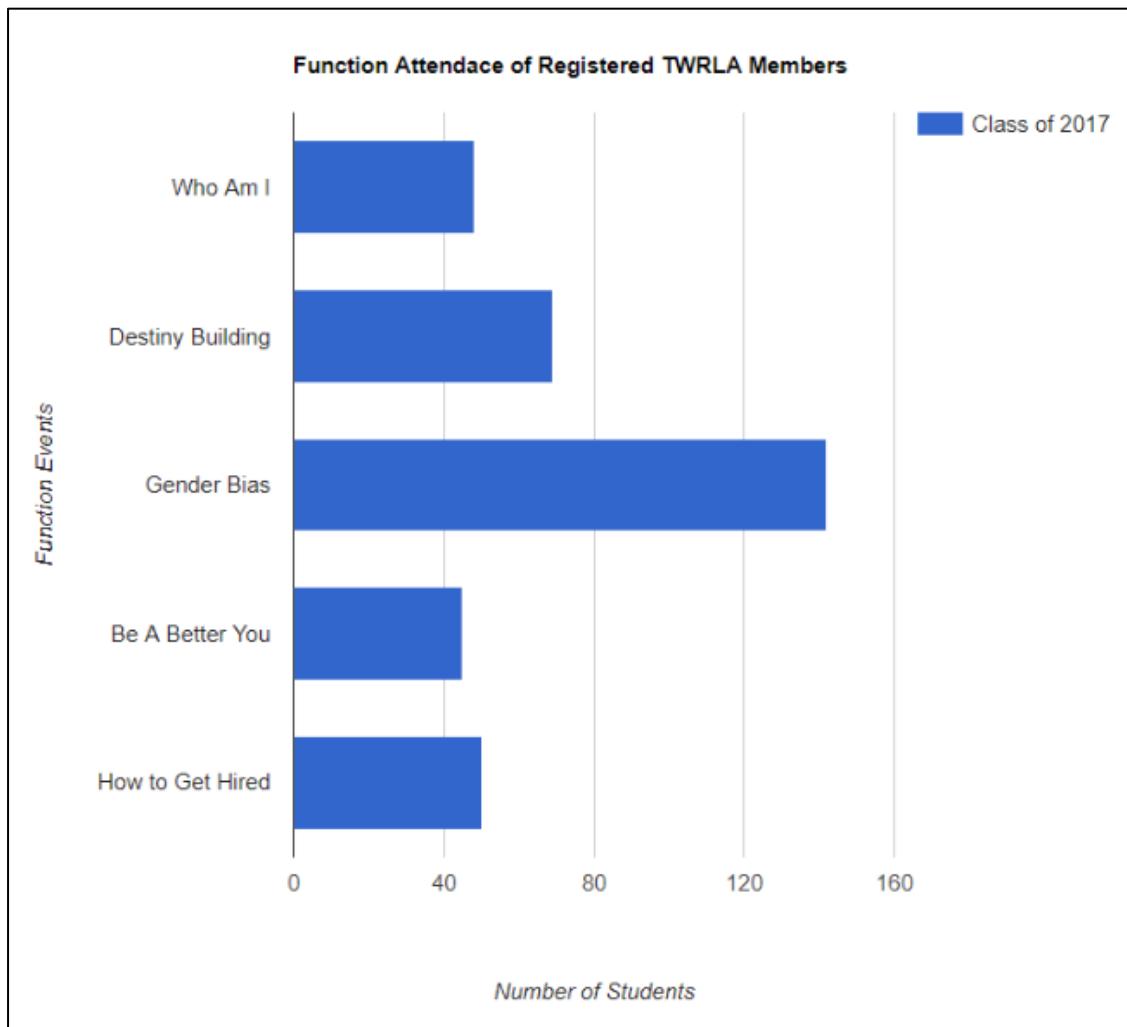
General	
Report No	9.2
Report Name	Community Engagement Attendance Report
Description	A report that determines the amount of students that attend community engagement events.
Frequency	Ad hoc
Special Features	N/A
Page Orientation	Landscape
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Bar graph	Community Engagement Attendance of Registered TRWLA Members	x-axis: Number of Students y-axis: Community Engagement Events legend: Class of #### (year)	N/A	N/A	Calculated by counting the amount of students that attend community engagement events.



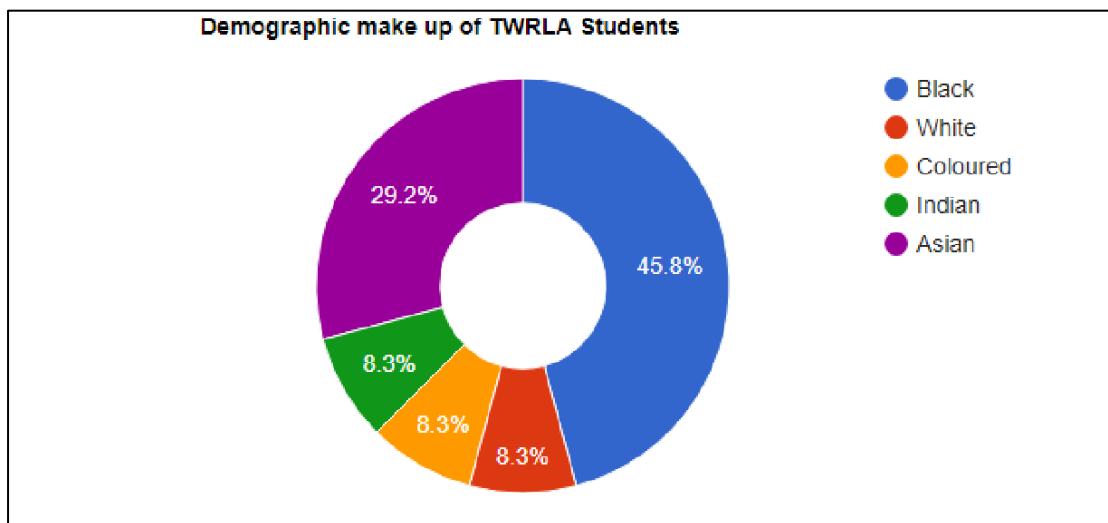
General	
Report No	9.3
Report Name	Function Attendance Report
Description	A report that determines the amount of students that attend functions of the academy.
Frequency	Ad hoc
Special Features	N/A
Page Orientation	Landscape
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Bar graph	Function Attendance of Registered TRWLA Members	x-axis: Number of Students y-axis: Function Events legend: Class of #### (year)	N/A	N/A	Calculated by counting the amount of students that attend functions.



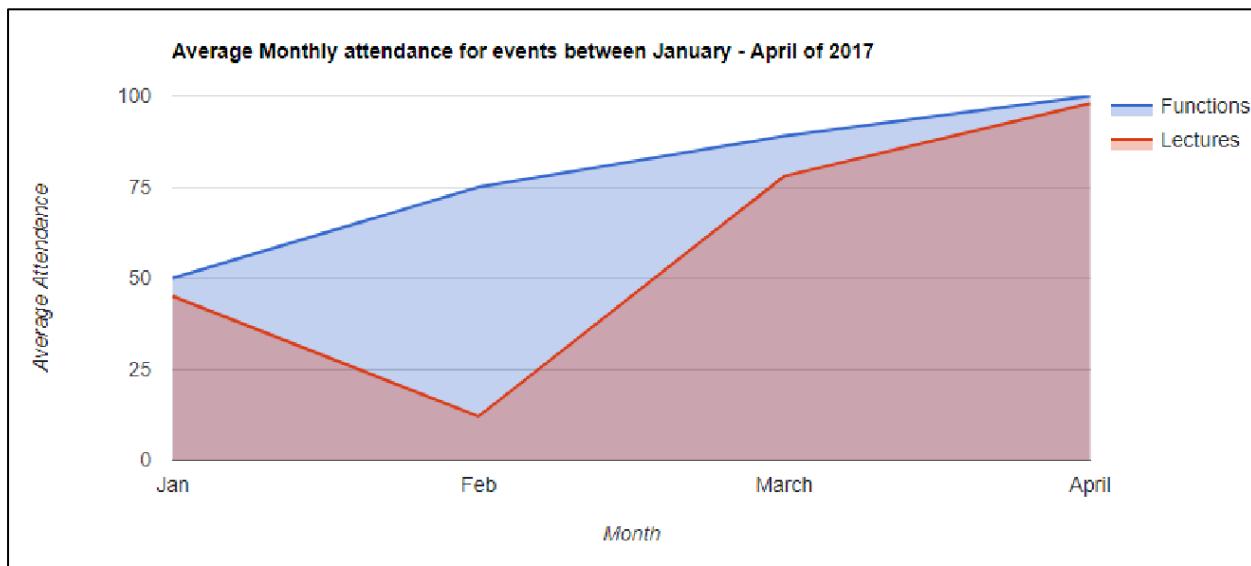
General	
Report No	9.4
Report Name	Demographics Report
Description	A report that determines the percentage of students that belong to a specific race.
Frequency	Ad hoc
Special Features	N/A
Page Orientation	Landscape
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Pie chart	Demographic Make Up of TRWLA Students	legend: Black White Coloured Indian Asian	N/A	N/A	Calculated by counting the amount of students that belong to a certain race based on their indicated race on their profiles.



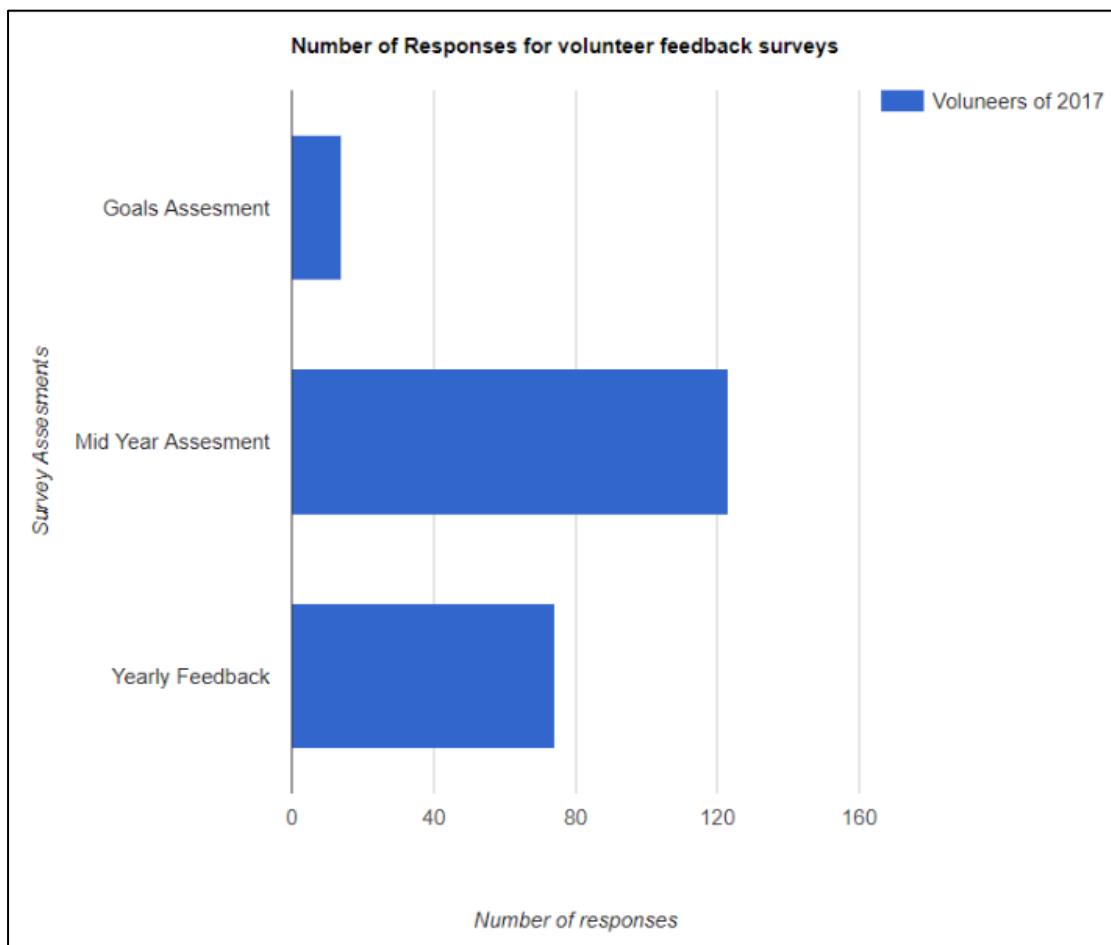
General	
Report No	9.5
Report Name	Event Popularity Report
Description	A report that compares the attendance of functions and lectures during a four month period of the current year.
Frequency	Ad hoc
Special Features	N/A
Page Orientation	Landscape
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Area chart	Average Monthly Attendance for events between ##### - ##### of ##### (months and year)	X-axis: Month Y-axis: Average Attendance Legend: Functions Lectures	N/A	N/A	Calculated by counting the average amount of attendance per function and lecture event over a four month period.



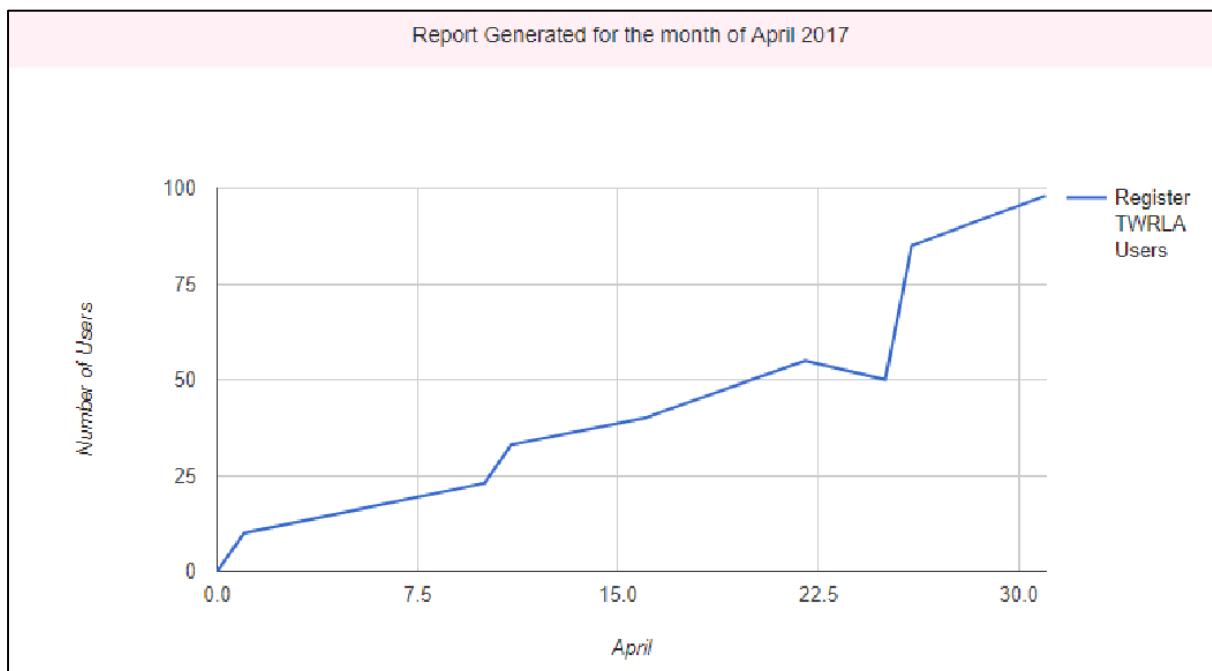
General	
Report No	9.6
Report Name	Feedback Report
Description	A report that determines the amount of feedback that volunteers make throughout the year about the academy.
Frequency	Ad hoc
Special Features	N/A
Page Orientation	Landscape
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Bar graph	Number of responses for volunteer feedback surveys.	X-axis: Number of responses Y-axis: Survey Assessments Legend: Volunteers of ##### (year)	N/A	N/A	Calculated by counting the amount of feedback responses received from volunteers.



General	
Report No	9.7
Report Name	User Statistics Report
Description	A report that determines the amount of times a user logs into the system within a specific month.
Frequency	Ad hoc
Special Features	N/A
Page Orientation	Landscape
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Line chart	Report Generated for the month of #####	X-axis: Month Y-axis: Number of Users Legend: Registered TRWLA Users	N/A	N/A	Calculated by counting the amount of times users log onto the system.



General	
Report No	9.8
Report Name	Student Progress Report
Description	A report that determines the progress of each student up until the current date.
Frequency	Ad hoc
Special Features	N/A
Page Orientation	Landscape
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Table column	Student Number	Numeric, Calibri, 12 pt.	8 characters	Left	Saved student number on the system.
Table column	First Name	Text, Calibri, 12 pt.	20 characters	Left	Saved first name on the system.
Table column	Last Name	Text, Calibri, 12 pt.	30 characters	Left	Saved last name on the system.
Table column	Progress status	Text, Calibri, 12 pt., Coloured red or green	N/A	Left	Changes label based on student's progress.
Table column	Progress Score	Percentage, Calibri, 12 pt.	N/A	Left	Calculated based on the student's attendance of lectures.

Student Number	First Name	Last Name	Progress Status	Progress Score
17355612	Caira	Milkbottom	PROGRESSING	57%
14992949	Tinyko	Vilakazi	NO PROGRESS	33%
15511549	Cailn	Van Rensburg	PROGRESSING	65%
14935058	Simpewe	Mholbo	NO PROGRESS	12%
16938416	Achal	Sighn	NO PROGRESS	45%

7.3 Outputs

General	
Report No	1
Report Name	Creation Confirmation Message
Description	A general modal message that displays once something has been created on the system.
Frequency	Upon successfully creation.
Special Features	Confirm Button to close the Modal
Page Orientation	N/A
Page Size	N/A

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Modal	Success "This ##### (what is affected) has been successfully created."	Text, Calibri, 12pt.	N/A	N/A	Displayed when an item has been successfully created on the system.

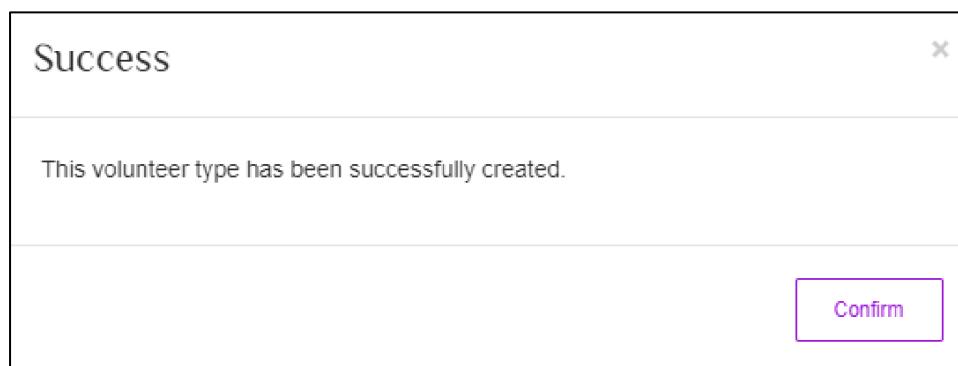


Figure 301: Suggested Confirmation Message Example

General	
Report No	2
Report Name	Updated Confirmation Message
Description	A general modal message that displays once something has been updated on the system.
Frequency	Upon successful update.
Special Features	Confirm Button to close the Modal
Page Orientation	N/A
Page Size	N/A

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Modal	Success "This ##### (what is affected) has been successfully updated."	Text, Calibri, 12pt.	N/A	N/A	Displayed when an item has been successfully created on the system.

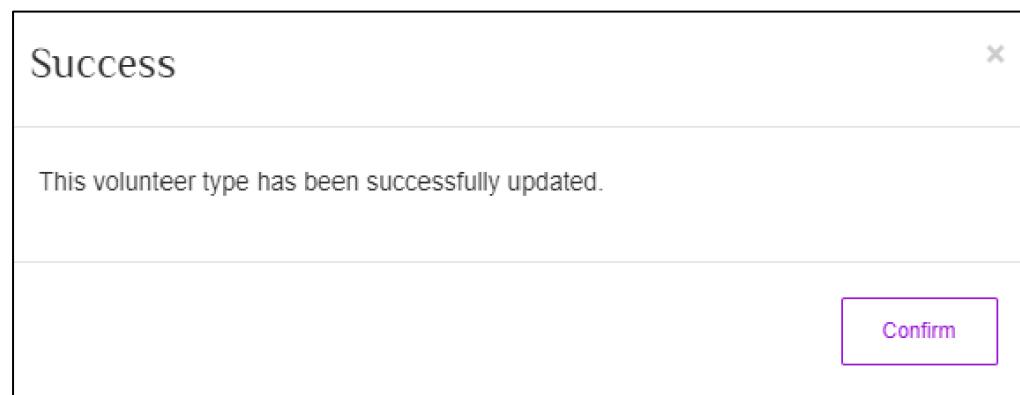


Figure 302: Suggested Successful Update Message Example

General	
Report No	3

Report Name	Deletion Warning Message
Description	A general modal message that displays once something is to be deleted on the system.
Frequency	Upon deletion of an item.
Special Features	Confirm Button and Cancel Button.
Page Orientation	N/A
Page Size	N/A

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Modal	Delete ##### (item being deleted) “Are you sure you want to delete ##### (what is affected)?”	Text, Calibri, 12pt.	N/A	N/A	Displayed when an item is to be deleted.

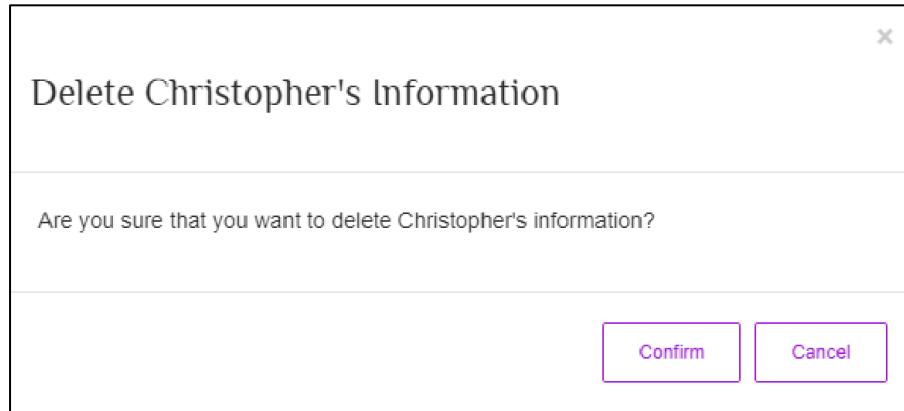


Figure 303: Suggested Deletion Warning Message Example

General	
Report No	4
Report Name	Update Warning Message

Description	A general modal message that displays once something is to be updated on the system.
Frequency	Upon update of an item.
Special Features	Confirm Button and Cancel button
Page Orientation	N/A
Page Size	N/A

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Modal	Update ##### (What is being updated) “Are you sure that you want to update this ##### (what is affected)?”	Text, Calibri, 12pt.	N/A	N/A	Displayed when an item is to be updated.

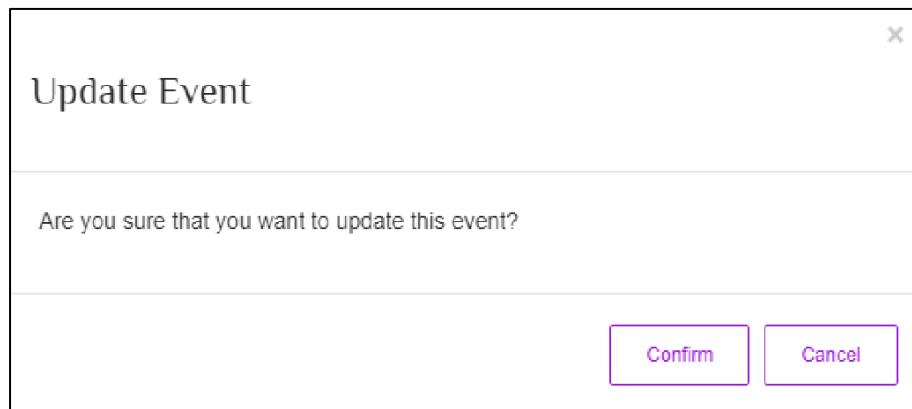


Figure 304: Suggested Update Warning Message Example

General	
Report No	5
Report Name	Error message
Description	A general modal message that displays once something is in the incorrect format.
Frequency	Upon creation or update of an item.
Special Features	Try Again Button
Page Orientation	N/A
Page Size	N/A

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Modal	"The required fields have not been completed or the information entered is invalid.)	Text, Calibri, 12pt.	N/A	N/A	Displayed when an item is created or updated and in the incorrect format.

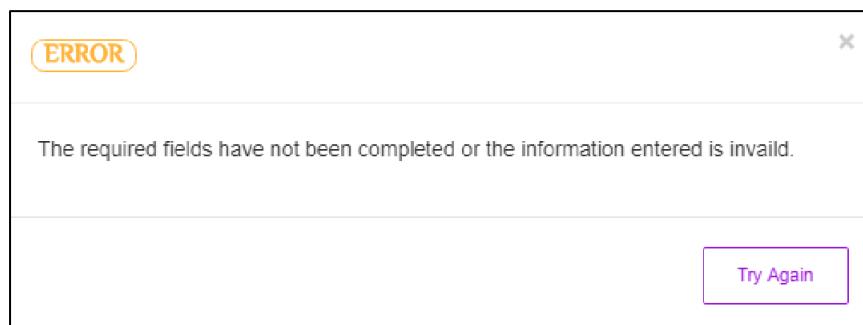


Figure 305: Suggested Error Message Example

General	
Report No	6
Report Name	No Search Results Modal

Description	A general modal message that displays once the system has processed a search request and has found no results.
Frequency	Upon a search.
Special Features	Try Again Button
Page Orientation	N/A
Page Size	N/A

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Modal	"No ##### (what is affected) match your search criteria."	Text, Calibri, 12pt.	N/A	N/A	Displayed when the system cannot locate what the user has searched for.

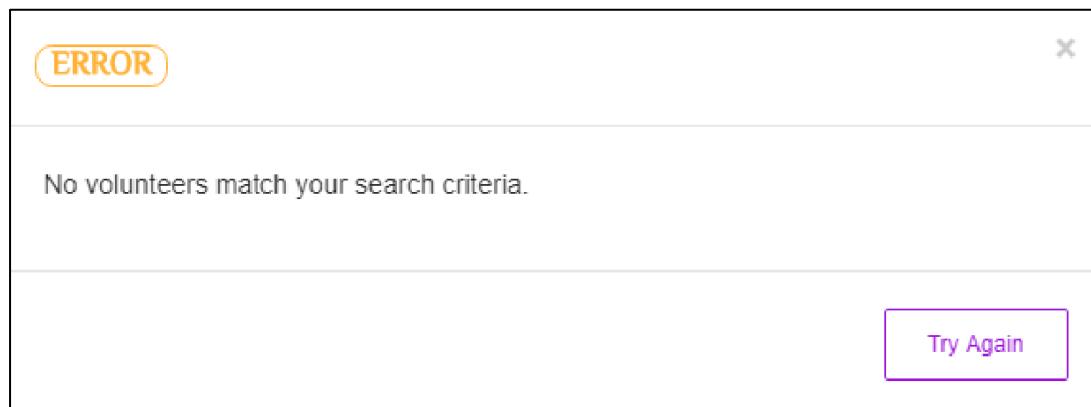


Figure 306: Suggested No Search Results Message Example

General	
Report No	7
Report Name	Email
Description	A general email that is generated when requested to by a volunteer.
Frequency	Ad hoc

Special Features	Connects to local email servers such as Gmail and Hotmail.
Page Orientation	N/A
Page Size	A4

Report Fields					
Field	Field Name	Format (character type, font, font size)	Maximum Length	Justified	Source
Email	"This email is generated from the TuksRes Women In Leadership Academy System"	Text, Calibri, 12pt.	N/A	N/A	Email Servers : Gmail, Hotmail, Yahoo etc.

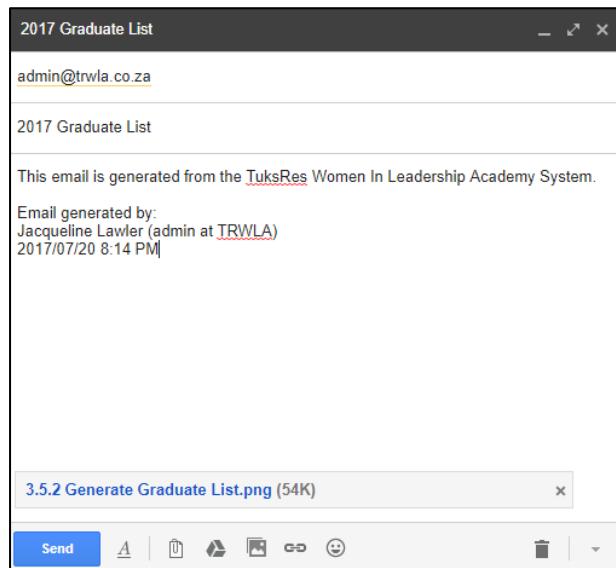


Figure 307: Example of email

FirstName	Surname	Phone Number	Email Address
Ciara	Mash	0147258786	u14284795@tuks.co.za
Una	De Burger	07895847	u14254786@tuks.co.za
Sinco	Renwillie	0174536985	u18789654@tuks.co.za
Martina	Khiosi	0214785963	u103658@tuks.co.za
Ellie	Martuns	0612587963	u10285478@tuks.co.za

Figure 308: Suggested Graduate List Attachment in Email Example

7.4 Conclusion

7.4.1 The above section described all of the general outputs that the system produces due to a user's interactions with the system. At the end of the document the reports are presented in a suggested format in order for the organisation to make informed business decisions.

8. Test Specification

8.1 Introduction

8.1.1 In this section the focus will be on the necessary procedures to test whether our system meets the business requirements stipulated and fulfils the outlined needs that our client had requested. Our test plan will cover how we will test the different components of our system during the implementation and maintenance phase.

8.2 Test plan

8.2.1 Usability Testing

8.2.1.1 The need for usability testing is to make sure that all components and features created will work in a way that satisfies our client. Once the initial build of the system is complete, non-tester who people as similar our client's users will evaluate whether different components and features will impact the system in a positive or negative way.

8.2.2 Unit Testing

8.2.2.1 This will be carried out by the developers throughout the development of the system to make sure that there is correct functionality between all code modules created.

8.2.3 Testing of Database

8.2.3.1 Our database will be tested to ensure that our referential integrity is correct. As well observing how our schemas handle data that is inserted into it.

8.2.4 Final release Testing

8.2.4.1 Milestones will be created whereby testers will assume that they are the end users of the system. This will ensure that complete coverage of our requirements are covered. From the test we anticipate to pick up on different types of bugs that may have not occurred in other forms of testing. Also, to make sure that our system adheres to HCI design principles that will seek to delight our client's users.

8.3 Test Scenarios and Procedures

1.3 Login	
Test Case ID: 1	Test Designed by: Amo Moloko
Test Priority (Low/Medium/High): High	Test Designed date: 06 July 2017
Module Name: TWRLA login screen	Test Executed by: Amo Moloko
Test Title: Verify login with valid username and password	Test Execution date: To be confirmed.
Description: The objective is to test that the user can login successfully into the system with the registered email address and password that they have provided.	
Pre-conditions: User must have a valid username and password that is stored inside of the database.	
Dependencies: Connection to the database.	

Step	Test Step	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
1	Navigate to the login page.		None	Login Page should load.		Login Page loads in timely manner.		
2	Enter Email address.		<u>xxxxxxxx@tuks.co.za</u>	Textbox should accept entered email address.		The system validates the entered password.		
3	Enter Password		MySQL#123	Textbox hides the characters in the password.		The system verifies that the password adheres to the restrictions set by the database.		
4	Click 'Log in' button.		Data from the email and password textboxes should be parsed to the database:	A modal should appear indicating that they have successfully		They have successfully logged in and appear on the student's home page.		

Step	Test Step	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
				logged in.				

Post Condition

User is validated with database and successfully logged into their account. The account session details are logged in database.

2.1 Register Volunteer	
Test Case ID: 2	Test Designed by: Amo Moloko
Test Priority (Low/Medium/High): Med	Test Designed date: 06 July 2017
Module Name: Register Volunteer Screen	Test Executed by: Amo Moloko
Test Title: Verify register volunteer.	Test Execution date: To Be Confirmed
Description: The purpose of this test is to verify if a user can register successfully as new volunteer with a unique code.	
Pre-conditions: The pending volunteer needs to have been hired by TWRLA.	
Dependencies: This test needs the module code from the unique code generation fully functional.	

Step	Test Step	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
1	Navigate to the register as new user page		None.	None.		Login Page loads in timely manner.		
2	Click radio Button		User type = volunteer	System displays a modal asking the user to enter a unique code, with a 'unique code textbox' and 'continue button'.		The system validates the entered password.		
3	Enter Unique code in Textbox		78958	System verifies that the numeric character count is valid against the restrictions provided by the database.		The system verifies that the password adheres to the restrictions set by the database.		
4	Click 'Continue Button'		<input type="submit" value="uniquecode" >	The system marks the unique code as		They have successfully logged in and appear		

Step	Test Step	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
				'Used'. As well as displaying a form if textboxes for the volunteer to register.		on the student's home page.		
5	Enter information in text fields.		<pre> <input type="text" name="name"> <input type="text" name="Surname"> <input type="numerical" name="phonenumber"> <input type="text" name="emailaddress"> <input type="text" class="form-control date-picker" value="10/05/2016" / data-datepicker-color=""> <ul class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink1" > VolunteerType1 </pre>	The system will display a modal notifying the user that their details are correct and are okay to proceed.				

Step	Test Step	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
			<pre> VolunteerType2 <ul class="dropdown-menu" aria- labelledby="navbarD ropdownMenuLink1" > Residence1 Residence2 </pre>					
6	Verify that system has successfully added a volunteer.		None	None		There needs to be a newly inserted record in the relevant table of the database.		

Post Condition

The volunteer's unique code has been marked as used and that their registration is successful and visible within the database.

2.9 Generate Unique Code	
Test Case ID: 3	Test Designed by: Amo Moloko
Test Priority (Low/Medium/High): Med	Test Designed date: 06 July 2017
Module Name: Generate Unique Code	Test Executed by: Amo Moloko
Test Title: Verify that a unique code can be generated	Test Execution date: To be confirmed
Description: Test that the system can generate a unique code for a volunteer to use. This will allow them to register onto the system. The code has to be a five-digit code.	
Pre-conditions: Admin users need to be logged into the system.	
Dependencies: None.	

Step	Test Steps	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
1	Navigate to the Unique generation page		None	The Generate Unique code page should be displayed.		None		
2	Click on "Generate Unique Code" Button		Code = "58747" & Status = 'Active'.	The page should populate the table with the newly generated unique code.		The database should have a new entry within the unique code table and it has be five digits.		

Post Condition

A five digit unique code has been generated and assigned to a volunteer.

7.1 Upload Content

Test Case ID: 4	Test Designed by: Amo Moloko
Test Priority (Low/Medium/High): High	Test Designed date: 06 Jul. 17
Module Name: Upload Content	Test Executed by: Amo Moloko
Test Title: Verify uploaded content	Test Execution date: Amo Moloko
Description: Test whether the content of the various types can be successfully uploaded onto the system.	
Pre-conditions: An event has to be created first for the content to be uploaded.	
Dependencies: None.	

Step	Test Steps	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
1	Navigate to content dropdown.		None	The system displays manage content page. With the following buttons : “Upload Content”, “Update”, “Delete”, “Return”		None		
2	Click “Upload Content Button”		None	The system displays upload Content page. With the following : “Content Type” (Label & textbox), “Name” (Label & Textbox), “Author” (Label& Textbox), “Date” (Label & Datepicker), “Content Link” (Label & Textbox), “Theme” (Label & Textbox),		None		

				"Status" (label & switch), "Description" (label & textbox) buttons : "Create" & "Return".				
3	Click "Create" button		Content Type = "Lecture" Name = "Be you" Author = "Sinqo Mabuza" Date = "03/09/2017" Content Link = "beyou.pdf" Theme = "Self Upliftment" Status = "Locked" Description = "xxxxxxxxxxxxxxxxxxxxx" xxxxx"	All the data that is entered into the fields must be validated.		The data entered into the fields is inserted into the relevant content table. Thus creating a new record of the specific content.		

Post-conditions:

The user has uploaded the relevant content successfully.

5.1 Create Function

Test Case ID: 5	Test Designed by: Christopher Oakes
Test Priority (Low/Medium/High): High	Test Designed date: 07 July 2017
Module Name: Create Function	Test Executed by: <Name>
Test Title: Verify that an Event of type Function has been created.	Test Execution date: <Date>
Description: Test if a function can be created through the use of the system and adequately be uploaded onto the TRWLA timeline for students to see.	
Pre-conditions: A volunteer has logged onto the system successfully.	
Dependencies: None.	

Step	Test Steps	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
1	Navigate to Create Event under Event Drop down		None	The system displays the 'Create Event' page with the following buttons: 'Function', 'Community Engagement' and 'Lecture'		None		
2	Click 'Function' button		None	The system displays the 'Create Function' page with the following: Name: Label and Text box Guest Speaker: Label and Dropdown Add Guest Speaker link Accept Invitiation		None		

Step	Test Steps	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
				label radio button Reject invitation label radio button which is selected Time: Label and time picker Date: Label and date picker Venue: Label and Dropdown Add Venue link Summary: Label and text box Description : Label and text box Image box with a browse button and a cloud icon Create button Return button Cancel button				
3	Click the browse button	No poster to be selected. Go to Step 4.	<ul style="list-style-type: none"> The selected image that the user selects from their device in which they are creating the event 	The system displays the image that the user selected on the page in the image box above the browse button.		None		

Step	Test Steps	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
4	Click the 'Create' button		<ul style="list-style-type: none"> • Name = "Looking Forward" • Guest Speaker = "Christopher Oakes" • Accept Invitation radio button = Not Selected • Reject Invitation radio button = Selected • Time = "12:00 PM" • Date = "2017/10/12" • Venue = "Duxbury Palace" • Summary = "xxxxxxxxxxxxxx xxxxx" • Description = "xxxxxxxxxxxxxx xxxxx" 	All the data that is entered into the fields must be validated. The system displays a Create Event modal requesting if the user wants to create this event.		None		
5	Click the 'Confirm' button		None	The system displays a 'Congratulations' modal stipulating that the event has been created. The system also details that the Guest Speaker invitation will display as 'Invitation to be Announced' until marked as		The data entered into the fields are inserted into the <u>Event</u> , <u>Function</u> , <u>Function Speaker</u> , <u>Schedule</u> tables respectively based on the information provided. This creates a new record in the database of TRWLA and sends and displays it		

Step	Test Steps	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
				accepted.		on the TRWLA timeline for students and for volunteers to see. Inserts the image into the servers storage facilities.		
6	Click the 'Confirm' button							

Post-conditions:

The volunteer has successfully created a Function Event and its information has been loaded into the Event, Function, FunctionSpeaker and Schedule tables in the TRWLA System Database.

5.1 RSVP to Event

Test Case ID: 6	Test Designed by: Christopher Oakes
Test Priority (Low/Medium/High): High	Test Designed date: 7 July 2017.
Module Name: RSVP to a Function Event	Test Executed by: <Name>
Test Title: Verify that a user of the system (Student, Volunteer or Admin) is able to successfully able to RSVP to an existing event on the TRWLA system.	Test Execution date: <Date>
Description: Test if a user is able to RSVP to an event that exists on the TRWLA system from the timeline of TRWLA as well as the details of the event viewed from the TRWLA timeline. Once the user has RSVP'd to the event the event must be transferred from the TRWLA timeline to the users 'Upcoming Events' timeline.	
Pre-conditions: Student has logged onto the system and there is an existing event in the TRWLA Timeline on the homepage once a User has logged in.	
Dependencies: An Event has been created, and the user has yet to RSVP to the event.	

Step	Test Steps	Alt Step	Test Data	Expected Result	Actual Result	Technical Result	Status (Pass/Fail)	Notes
1	Select an existing events 'Details' from the main menu page showing the TRWLA timeline.	Click the 'RSVP' Button directly from the Main Menu page once logged in. Go to Step 2.	None	<p>The system displays the events details with the following information:</p> <ul style="list-style-type: none"> • Name of the Event (Label and Text) • Date of the Event (Label and Date) • Time of the event (Label and Time (24hr Time)) • Summary of the event (Label and Text) • Description of the Event (Label and Text) • Guest Speaker of the Event 		None		

				<p>(Label and Text)</p> <ul style="list-style-type: none"> • Venue of the Event (Label and Text) • Poster of the Event (Image) • RSVP button • Return Button 				
2	Click the 'RSVP' Button		None	Modal appears asking the student to confirm that they want to RSVP to the event: 'Are you sure you want to RSVP to this event?' 'Confirm' and 'Cancel' buttons are displayed		None		
3	The student clicks 'Confirm'		None	The system displays the main menu to the student again with the Event that the student RSVPd removed from the 'Upcoming TRWLA Events' and placed on the 'My Upcoming Events' side of the display. The student has successfully RSVPd to the event.		The system updates the <u>RSVP_Event</u> table in the TRWLA system database with the <u>PersonID</u> and <u>EventID</u> of the student and the event respectively and updates the <u>Schedule</u> Table with the <u>PersonID</u> of the student and the <u>EventID</u> of the event to update the 'My Upcoming Events' for the		

						students personal Main Menu.		
--	--	--	--	--	--	------------------------------------	--	--

Post-conditions:

The student has successfully RSVPd to an event and their information has been added to the [RSVP_Event](#) and [Schedule](#) tables in the TRWLA database

8.4 Test Data

Use Case	1.3 Log In
Subsystem	User
Description	This use case enables users to login into the TWRLA System using the email and password they used to sign up.

UI Components	Parsed Data	Expected Result	Technical Assumptions
@Html.TextBoxFor(m => m.Email, new { @class = "form-control" })	u14285747@tuks.co.za	Validation of textbox successful.	The email address is inside the database to allow for successful authentication.
@Html.PasswordFor(m => m.Password, new { @class = "form-control" })	IloveINF370	Validation of textbox successful.	That the password exists within the database to validated the users authentication.
<u>!@Html.CheckBoxFor(m => m.RememberMe, new { @class = "checkbox label" })</u>	checked="True"	Checkbox should indicate that it is checked.	None.
<input class="btn btn-default btn-round" data-bbox="204 1256 473 1404" type="submit" value="Log in"/>	value="Submit"	The system should validate the inputs against the records in the database and log the user into the system .	The button calls the correct functions from the relevant controller.

Use Case	2.1 Register Volunteer
Subsystem	Volunteer
Description	This use case allows the user to register as a volunteer onto the system.

UI Components	Parsed Data	Expected Result	Technical Assumptions
@Html.TextBoxFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })	Thuli	Validation of textbox successful.	That this record of data is unique.
@Html.TextBoxFor(model => model.Surname, new { htmlAttributes = new { @class = "form-control" } })	Dasener	Validation of textbox successful.	That this record of data is unique.
@Html.TextBoxFor(model => model.PhoneNumber, new { htmlAttributes = new { @class = "form- control" } })	+074125897	Validation of textbox successful.	That the data entered is numeric
@Html.TextBoxFor(model => model.EmailAddress, new { htmlAttributes = new { @class = "form- control" } })	u10458749@tuks.co.za	Validation of textbox successful.	There are no duplicated versions of the record.
<input class="form-control date-picker" data-datepicker-color="" data-kind="parent" type="text" value="10/05/2016"/>	02/05/2017	The system should verify that a valid date has been entered.	That the date picker does not allow the user to selected dates past the current date.
@Html.DropDownList("VolunteerTypeID", null, htmlAttributes: new { @class = "form-control" })	Administrator	Droplist should be populated with the relevant options.	Dropdown list is fed with data from the database from the correct table.
@Html.DropDownList("Residence", null, htmlAttributes: new { @class = "form- control" })	Nerina	Droplist should be populated with the relevant options.	Dropdown list is fed with data from the database from the correct table.

UI Components	Parsed Data	Expected Result	Techincal Assumptions
<pre><input type="submit" value="Create" class="btn btn-default" /></pre>	value="Submit"	The system should validate the inputs against the records in the database and log the user into the system .	The button calls the correct fuctions from the relevant controller.

Use Case	2.9 Generate Unique Code
Subsystem	User
Description	

UI Components	Parsed Data	Expected Result	Techincal Assumptions
<button class="btn btn-lg">Generate Unique Code</button>	value="RAND(1,5)"	Calls a function that generates random numbers.	The function generates a unique 5 digit code.
<table class="table">	458478,Status='Active'	Table should be populated with Unique code values.	None.
<input type="text" value="" placeholder="Search" class="form-control" />	"Ela Freder"	Search the volunteers in the table.	Uses a linq query with a string to search for the volunteers.

Use Case	7.1 Upload Content
Subsystem	Content
Description	

UI Components	Parsed Data	Expected Result	Techincal Assumptions
@Html.TextBox(model => model.ContentType, new { htmlAttributes = new { @class = "form-control" } })	Lecture	The system should validate the textbox successfully.	That the a lecture content table exists withing the database.
@Html.Textbox(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })	Emotional intellegence	The system should validate the textbox successfully.	The name does not conflict with any other records in the database
@Html.Textbox(model => model.Author, new { htmlAttributes = new { @class = "form-control" } })	Emily Castro	The system should validate the textbox successfully.	The author is authorised to publish the content.
@Html.Datepicker(model => model.DatePublished, new {	4/5/2017	The system should verify that a valid	That the date picker does not allow the user to selected

UI Components	Parsed Data	Expected Result	Techincal Assumptions
htmlAttributes = new { @class = "form-control" } })		date has been entered.	dates past the current date.
@Html.Textbox(model => model.Link, new { htmlAttributes = new { @class = "form-control" } })	"emotionalintellegence.pdf"	The link should allow a user to access to open the file.	The link is a hyperlink.
@Html.Textbox(model => model.Theme, new { htmlAttributes = new { @class = "form-control" } })	Becoming Me	The system should verify that a valid date has been entered.	none.
<input type="checkbox" name="checkbox" class="bootstrap-switch" data-on-label="ON" data-off label="OFF"/>	"1"	The switch should be turn on.	That the data is rendered as binary.
@Html.Textbox(model => model.Description, new { htmlAttributes = new { @class = "form-control" } })	This lecture illustrates how we can overcome emotional intellegence to better navigate sticky situations.	The system should verify that a valid date has been entered.	None.
<input type="submit" value="Create" class="btn btn-default btn-round" />	Value ="Submit"	The system should display a modal to show that content has been successfully uploaded. And that the relevant data from the input form is successfully inserted into the database.	That a record of the uploaded content does not exist.

Use Case	5.1 Create Function
Subsystem	5
Description	Test if a function can be created through the use of the system and adequately be uploaded onto the TRWLA timeline for students to see.

UI Components	Parsed Data	Expected Result	Technical Assumptions
@Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control form-control-danger" } })	Looking Forward	Validation of textbox successful.	That the event name is unique
@Html.DropDownList("GuestSpeakerID", null, htmlAttributes: new { @class = "form-control" })	Christopher Oakes	Validation of dropdownlist item checked.	That the guest speaker has already been loaded onto the system.
@Html.ActionLink("Register Guest Speaker", "Create", "GuestSpeakers")	None	If clicked, goes to Register Guest Speaker	The button will take the user to Register Guest Speaker
<input type="radio" name="radio1" id="radio2" value="option2" checked="">	Value = 'Checked'	Checkbox should indicate that it is checked.	The button will be checked at 'Invitation Rejected' during function creation if a Guest Speaker is selected.
@Html.EditorFor(model => model.EventStartTime, new { htmlAttributes = new { @class = "form-control" } })	8:00 pm	Validation of time format	The time selected is not at an unrealistic time of day nor does it clash with any other event happening, based on the date selected.
@Html.EditorFor(model => model.EventDate, new { htmlAttributes = new { @class = "form-control" } })	29/8/2017	Validation of date format	The datepicker does not allow the user to select dates that have already passed.
@Html.DropDownList("VenueID", null, htmlAttributes: new { @class = "form-control" })	Duxbury Palace	Validation of dropdownlist item checked.	The venue already exists on the system as the list is prepopulated.

UI Components	Parsed Data	Expected Result	Technical Assumptions
@Html.ActionLink("Add Venue", "Index", "Events", new { @style = "margin-top:10px; text-align:center; margin-left:25px;" })	None	If clicked, goes to Add New Venue	The button will take the user to Add New Venue.
@Html.EditorFor(model => model.Summary, new { htmlAttributes = new { @class = "form-control" } })	This is where you will find yourself...	Validation of textbox successful.	The user is only allocated to enter a certain number of characters before the system tells the user that they may not enter any more.
@Html.LabelFor(model => model.Description, htmlAttributes: new { @class = "control-label col-md-2" })	Taking a step into the future is one of the biggest things we have to face today, and "Looking Forward" is step one.	Validation of textbox successful.	The user is only allocated to enter a certain number of characters before the system tells the user that they may not enter any more.
	Value = Selected Image	The div is populated with the selected iamge that volunteer wanted to upload	
<input type="file" id="selectedFile" style="display: none;" />	value = "Browse"	The system displays a file dialogue prompting the user to search for the poster they want to upload.	
<button class="btn btn-default btn-round" data-toggle="modal" data-target="#Create" style="color:#9b16d8;">Create</button>	value = 'Submit'	The system should validate the inputs against the records in the	The click of this button will call the appropriate functions from the controller.

UI Components	Parsed Data	Expected Result	Technical Assumptions
		database and Create a New Function on the system.	
@Html.ActionLink("Return", "Index", "Events")	None	If clicked returns volunteer to the main menu.	The system will take the volunteer back to the main menu.
@Html.ActionLink("Cancel", "Index", "Events")	None	If clicked returns volunteer to the main menu.	The system will take the volunteer back to the main menu.

Use Case	5.5 RSVP to an Event
Subsystem	5
Description	Test if a user is able to RSVP to an event that exists on the TRWLA system from the timeline of TRWLA as well as the details of the event viewed from the TRWLA timeline. Once the user has RSVP'd to the event the event must be transferred from the TRWLA timeline to the users 'Upcoming Events' timeline.

UI Components	Parsed Data	Expected Result	Techincal Assumptions
<button class="btn-primary" data-toggle="modal" data-target="#NotGoing">RSVP</button>	None	The system prompts the user asking them to confirm that they want to go to the event.	The system has access to the users information as they have logged into the system as well as the event information regarding the event that the user wants to RSVP to. Thus both records exist in the database and can be accessed by the system. The appropriate function in the controller is called.
@Html.ActionLink("Details", "Details", new { id = item.EventID })	EventID = 1	The system displays the details of the event to the user.	The system validates that the event exists on the system and displays the appropriate information to the user drawn from the relevant tables.
@Html.DisplayFor(model => model.Name)	Finding Yourself	System displays the Name of the event	The system draws the information from the correct table = <u>Event</u> , <u>Function</u> and <u>FunctionSpeaker</u> .
@Html.DisplayFor(model => model.EventDate)	08/09/2017	System displays the date of the event	The system draws the information from the correct table = <u>Event</u> , <u>Function</u> and <u>FunctionSpeaker</u> .

UI Components	Parsed Data	Expected Result	Techincal Assumptions
@Html.DisplayFor(model => model.EventStartTime)	20:00	The system displays the time of the event	The system draws the information from the correct table = <u>Event</u> , <u>Function</u> and <u>FunctionSpeaker</u> .
@Html.DisplayFor(model => model.Summary)	This is where you will	The system displays the summary of the event	The system draws the information from the correct table = <u>Event</u> , <u>Function</u> and <u>FunctionSpeaker</u> .
@Html.DisplayFor(model => model.Description)	Finding yourself is the most important attribute to have...	The system displays the description of the event	The system draws the information from the correct table = <u>Event</u> , <u>Function</u> and <u>FunctionSpeaker</u> .
@Html.ActionLink("Christopher Oakes", "Index", "Events", new { id = item.GuestSpeakers.FirstName })	Christopher Oakes	The system displays the name of the guest speaker of the evnet in a link format allowing the user to follow the link to get details on the guest speaker	The system draws the information from the correct table = <u>Event</u> , <u>Function</u> and <u>FunctionSpeaker</u> .
@Html.DisplayFor(model => model.Venue.Name)	Duxbury Palace	The system displays the venue of the event.	The system draws the information from the correct table = <u>Event</u> , <u>Function</u> and <u>FunctionSpeaker</u> .
		The system displays the image of the event.	The system draws the infromation from the external server.
<button class="btn btn-default btn-round" data-toggle="modal" data-target="#NotGoing" style="color:#9b16d8">RSVP</button>	None	The system prompts the user asking them to	The system has access to the users information as they have logged into the system as well as the

UI Components	Parsed Data	Expected Result	Techincal Assumptions
		confirm that they want to go to the event.	event information regarding the event that the user wants to RSVP to. Thus both records exist in the database and can be accessed by the system. The appropriate function in the controller is called.
@Html.ActionLink("Return", "Index", "Events")	None	The system returns the user to the main menu displaying the different upcoming and RSVPd to events.	The system calls the controller to display the main menu again to the user.

8.5 Conclusion

What was covered in our various test procedures will enable us to forecast what the intended functionality of the system will be. The system's weak points will also be highlighted and will enable us to improve or fix any of them.

9. Hardware and Software Requirements

9.1 Introduction

This section will cover the necessary hardware and software requirements required by our client to run the system. These requirements need to be adhered to, to allow the maximum value to be unlocked while using the system.

9.2 Requirements

9.2.1 Hardware

Below we outline the minimum and recommended requirements necessary for the system to function once implemented.

Internet connection Specifications

Development:

During development of the system, no internet connection is required in terms of testing the system as a local host is used to run the web application on a browser of choosing. The browsers that are available are those that are installed on the developer's computer and as such are listed to:

1. Google Chrome
2. Firefox
3. Internet Explorer
4. Microsoft Edge
5. Opera Mini
6. Safari

Being the most popular options on desktop and laptop computers at present. Mobile offerings are:

1. Safari
2. Opera
3. Opera mini
4. Google Chrome
5. Samsung Browser

Deployed Application:

1. The deployed application will require an internet connection to run. As such, it will be required to handle the load of the database of the TRWLAS system at maximum which is limited to a total of 40 MB per the calculations and size estimation of the database.

Computer specifications

As the system will not be locally run on a single (or multiple) machine/s, but rather be run on an internet browser it is suggested that the user maintains the minimum requirements to run a browser on their computer/tablet/cell phone.

The most popular browsers were explored in terms of the minimum requirements laid out for them to run on a desktop and the results are as follows:

1. Opera, Firefox and Chrome are the only three browsers that are available on all major operating systems including Windows, Mac and Linux.
2. Out of the three browsers, the one that requires the highest computer requirements and as such will be the recommended requirements for the system is: Firefox. **Invalid source specified.**

	Minimum Requirements	Recommended Requirements
CPU		Windows: 1. Pentium 4 or newer Mac: 1. Mac Computer with an Intel x86 processor
RAM		Windows and Mac: 1. 512 MB
Storage	40 MB	Windows and Mac: 1. 210 MB of hard drive space
Operating System	Windows 7/ Mac OS X Lion	Windows: 1. Windows 7 2. Windows 8

	Minimum Requirements	Recommended Requirements
		3. Windows 10 Mac: 1. MacOS 10.9 2. MacOS 10.10 3. MacOS 10.11 4. MacOS 10.12
Optical Drive	None	None

9.2.2 Software

Operating System	Windows: 1. Windows 7 2. Windows 8 3. Windows 10 Mac: 1. MacOS 10.9 2. MacOS 10.10 3. MacOS 10.11 4. MacOS 10.12
Browser	1. Google Chrome 2. Firefox 3. Internet Explorer 4. Microsoft Edge 5. Opera Mini 6. Safari
Developer Environment	Microsoft Visual Studio
Storage/Server	Amazon S3 Standard: Ideal for dynamic website due to high durability, availability and performance object storage for frequently accessed data. Provides free tier for one year. Ideal for

	capturing visual data such as images which is a necessity for the TRWLA System due to the fact that all images will be stored directly in the server instead of in the database in order to optimise database functionality.
--	--

9.3 Conclusion

These specifications are merely a guide we recommend that be followed strictly to avoid unnecessary instances where the system fails on the client. The specifications clearly illustrate what is needed for the system to operational.

10. Network / Web Layout Specifications

10.1 Introduction

Our system is a web based system that will be host by web Africa. Below we illustrated the network layout of our system for our client.

10.2 Diagram

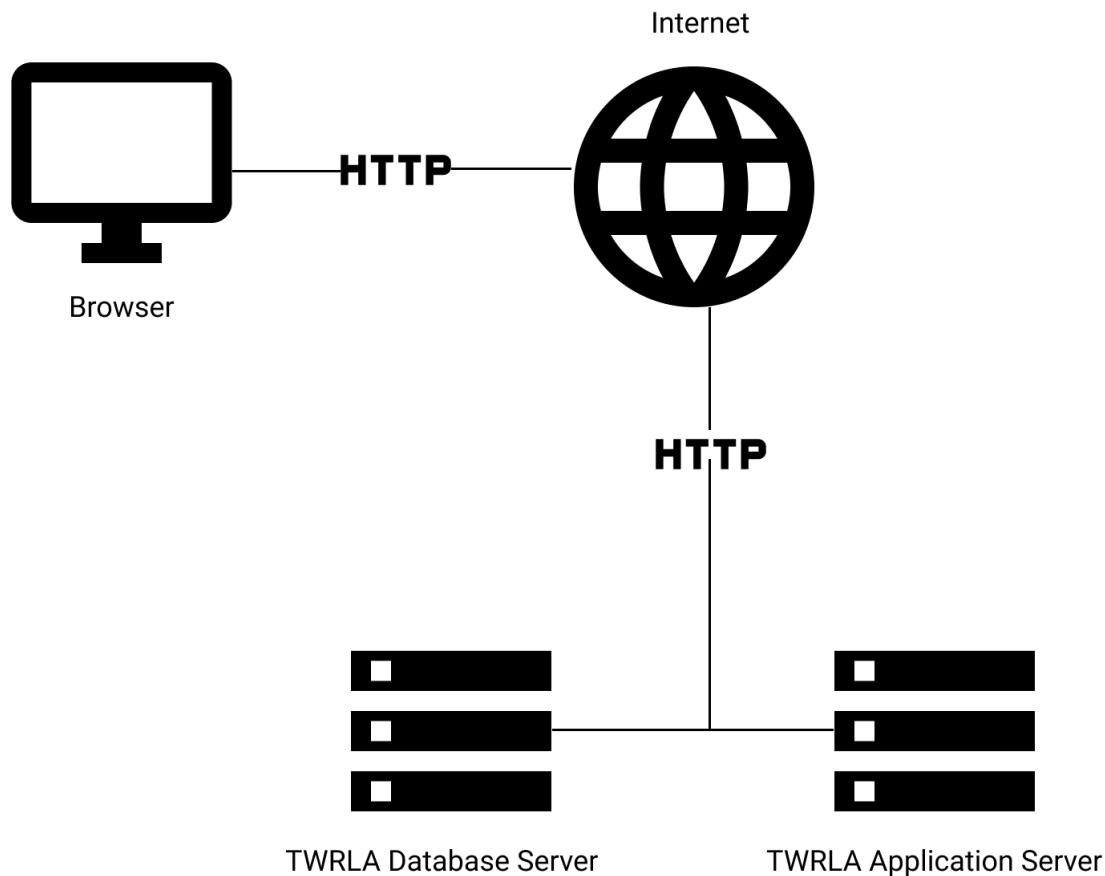


Figure 309: Network Layout Structure

10.3 Conclusion

The network layout clearly illustrates how the networks of system will function once it is developed and deployed for our client.

11. Validation

11.1 Introduction

11.1.1 The purpose of the validation is to ensure that the technical specifications detailed above are in line with all the requirements of the system. It displays the requirements for each subsystem, the use cases that pertain to that requirement, each process of the primitive level diagram that link to the use case, the entities that are affected by each process, the screen that is displayed for the process as well as every report linked to the process.

11.2 Validation

11.2.1 Subsystem 1

Requirement	Use Case	Processes	Entities	Screens	Reports
1.1 Check Forgotten Password	1.1 Check Forgotten Password	1.1.1	Person		
		1.1.2		Home Page	
		1.1.3			
		1.1.4		1.3.1	
		1.1.5			
		1.1.6		1.1.1	
		1.1.7			
		1.1.8			
		1.1.9	SecurityAnswer	1.1.2	
		1.1.10			
		1.1.11	SecurityAnswer		
1.2 Change Password	1.2 Change Password	1.2.1			
		1.2.2		General My Profile Page	
		1.2.3			

Requirement	Use Case	Processes	Entities	Screens	Reports
		1.2.4		1.2.2	
		1.2.5			
		1.2.6	Person		
		1.2.7	Person		
		1.2.8			Report 2
1.3 Login	1.3Login	1.3.1			
		1.3.2		1.3.1	
		1.3.3			
		1.3.4	Person		
		1.3.5	UserType		
		1.3.6		5.0.1	
		1.3.7	AuditLog		
1.4 Logout	1.4 Logout	1.4.1			
		1.4.2			
		1.4.3			
		1.4.4			
		1.4.5			
		1.4.6	AuditLog	1.3.1	
1.5 Deactivate Account	1.5 Deactivate Account	1.5.1			
		1.5.2		1.4.1	
		1.5.3			
		1.5.4	Person, Student, Residence		

Requirement	Use Case	Processes	Entities	Screens	Reports
		1.5.5			
		1.5.6		1.5.2	
		1.5.7			
		1.5.8	Person	1.3.1	
1.6 Create User Type	1.6 Create User Type	1.6.1			
		1.6.2			
		1.6.3			
		1.6.4	UserType	1.6.1	
		1.6.5			
		1.6.6		1.3.1	
		1.6.7			
		1.6.8	UserType		
		1.6.9		1.3.1	Report 1
		Alt 1.6.6	Event	5.0.1	
1.7 Update User Type	1.7 Update User Type	1.7.1			
		1.7.2			
		1.7.3			
		1.7.4	UserType	1.6.1	
		1.7.5			
		1.7.6		1.7.1	Report 4
		1.7.7			
		1.7.8	UserType		

Requirement	Use Case	Processes	Entities	Screens	Reports
		1.7.9	UserType	1.7.1	Report 2
1.8 Delete User Type	1.8 Delete User Type	Alt 1.7.6	Event	5.0.1	
		1.8.1			
		1.8.2			
		1.8.3			
		1.8.4	UserType	1.6.1	
		1.8.5			
		1.8.6		1.8.1	
		1.8.7			
		1.8.8	UserType		Report 4
		1.8.9			
		1.8.10	UserType	1.6.1	
		Alt 1.8.6	Event	5.0.1	
1.9 Search Alumni	1.9 Search Alumni	1.9.1			
		1.9.2			
		1.9.3			
		1.9.4		1.9.1	
		1.9.5			
		Alt 1.9.5.1			
		Alt 1.9.5.2			
		Alt 1.9.5.3			
		1.10.1		Home page	

Requirement	Use Case	Processes	Entities	Screens	Reports
1.10 View Static Webpage	1.10 View Static Webpage	1.10.2			
		1.10.3		About page	
		1.10.3a		Contact page	
		1.10.3b		Gallery Page	
		1.10.3c			
		1.10.3d			
		1.10.3e			
		1.10.3f			
		1.10.3g			

11.2.2 Subsystem 2

Requirement	Use Case	Processes	Entities	Screens	Reports
2.1 Register Volunteer	2.1 Register Volunteer, AUC 1	AUC 1.1			
	Register User	AUC1.2		Home page	
		AUC1.3			
		AUC1.4		1.3.1	
		AUC 1.5			
		AUC 1.6		2.1	
		AUC 1.7			
		AUC 1.8	Person		
		AUC 1.9	Person, SecurityAnswer		
		AUC 1.10			
	2.1.1			2.1.1	
	2.1.2			2.1.2	
	2.1.3				
	2.1.4				
	2.1.5		UniqueCode		
	2.1.6		UniqueCode		
	2.1.7		VolunteerType, Residence	2.1.7	
	2.1.8				
	2.1.9				
	2.1.10		Volunteer		
	2.1.11				Report 1

Requirement	Use Case	Processes	Entities	Screens	Reports
2.2 Search Volunteer	2.2 Search Volunteer	2.2.1			
		2.2.2			
		2.2.3			
		2.2.4		2.2.1	
		2.2.5	Person, Volunteer		
		2.2.6	Volunteer	2.2.1	
2.3 Update Volunteer	2.3 Update Volunteer	2.3.1			
		2.3.2			
		2.3.3			
		2.3.4	VolunteerType, Residence, Volunteer, Person	General Profile page	
		2.3.5			
		2.3.6			
		2.3.7			Report 4
		2.3.8			
		2.3.9	Volunteer, Person		
		2.3.10			Report 2
2.4 Delete Volunteer	2.4 Delete Volunteer	2.4.1			
		2.4.2		2.4.1	
		2.4.3			
		2.4.4			

Requirement	Use Case	Processes	Entities	Screens	Reports
		2.4.5	Volunteer, Person		
		2.4.6			Report 3
2.5 Create Volunteer Type	2.5Create Volunteer Type	2.5.1			
		2.5.2	Volunteer	2..2.1	
		2.5.3			
		2.5.4	VolunteerType	2.6.1	
		2.5.5			
		2.5.6		2.5.2	
		2.5.7			
		2.5.8			
		2.5.9			
		2.5.10			
		2.5.11	VolunteerType		
		2.5.12			Report 1
		2.5.13		2.6.1	
2.6 Search Volunteer Type	2.6 Search Volunteer Type	2.6.1			
		2.6.2	Volunteer	2.2.1	
		2.6.3			
		2.6.4	VolunteerType	2.6.1	
		2.6.5			
		2.6.6	VolunteerType	2.6.1	
		2.7.1			

Requirement	Use Case	Processes	Entities	Screens	Reports
2.7 Update Volunteer Type	2.7 Update Volunteer Type	2.7.2	Volunteer	2.2.1	
		2.7.3			
		2.7.4	VolunteerType	2.6.1	
		2.7.5			
		2.7.6	VolunteerType	2.7.1	
		2.7.7			
		2.7.8			
		2.7.9			
		2.7.10			Report 4
		2.7.11			
2.8 Delete Volunteer Type	2.8 Delete Volunteer Type	2.8.1			
		2.8.2	Volunteer	2.2.1	
		2.8.3			
		2.8.4	VolunteerType	2.6.1	
		2.8.5			
		2.8.6	VolunteerType	2.8.1	
		2.8.7			
		2.8.8			
		2.8.9			
		2.8.10			Report 3
		2.8.11			

Requirement	Use Case	Processes	Entities	Screens	Reports
		2.8.12	VolunteerType	2.6.1	
2.9 Generate Unique Code	2.9 Generate Unique Code	2.9.1			
		2.9.2			
		2.9.3			
		2.9.4	Volunteer	2.2.1	
		2.9.5			
		2.9.6		2.9	
		2.9.7			
		2.9.8	UniqueCode		
		2.9.9	UniqueCode		
2.10 Register Admin	2.10 Register Admin	2.10.1			
		2.10.2			
		2.10.3			
		2.10.4	UserType, Volunteer	2.10	
		2.10.5			
		2.10.6	UserType		
		Alt 2.10.6	Event	5.0.1	
2.11 Deregister Admin	2.11 Deregister Admin	2.11.1			
		2.11.2			
		2.11.3			
		2.11.4	UserType, Volunteer	2.10	
		2.11.5			

Requirement	Use Case	Processes	Entities	Screens	Reports
		2.11.6	UserType		
		Alt 2.11.6	Event	5.0.1	

11.2.3 Subsystem 3

Requirement	Use Case	Processes	Entities	Screens	Reports
3.1 Register Student	3.1 Register Student, AUC 1 Register User	AUC 1.1			
		AUC 1.2		Home Page	
		AUC 1.3			
		AUC 1.4		1.3.1	
		AUC 1.5			
		AUC 1.6		3.1.1	
		AUC 1.7			
		AUC 1.8	Person		
		AUC 1.9	Person, SecurityAnswer		
		AUC 1.10		3.1.2	
		3.1.1			
		3.1.2	Residence	3.1.3	
		3.1.3			
		3.1.4	Residence		
3.2 Search Student	3.2 Search Student	3.2.1			
		3.2.2			
		3.2.3			
		3.2.4	Student	3.2.1	
		3.2.5			

Requirement	Use Case	Processes	Entities	Screens	Reports
		3.2.6	Student, Person	3.3.1	
		Alt 3.2.5		3.2.1	
		Alt 3.2.5			
		Alt 3.2.6		5.0.1	
3.3 Update Student	3.3 Update Student	3.3.1			
		3.3.2			
		3.3.3			
		3.3.4	StudentType, Residence, Student	General Profile page	
		3.3.5			
		3.3.6			
		3.3.7			Report 4
		3.3.8			
		3.3.9	Student		Report 2
		3.3.10			
		3.3.11	Event	5.0.1	
3.4 Delete Student	3.4 Delete Student	3.4.1			
		3.4.2			
		3.4.3			
		3.4.4	Student	3.2.1	
		3.4.5			
		3.4.6			

Requirement	Use Case	Processes	Entities	Screens	Reports
		3.4.7			
		3.4.8			
		3.4.9	Student		Report 3
		3.4.10			
		3.4.11	Student	3.2.1	
3.5 Generate Graduate List	3.5 Generate Graduate List	3.5.1			
		3.5.2			
		3.5.3			
		3.5.4	Student	3.2.1	
		3.5.5			
		3.5.6	Student, Graduate	3.5	
		3.5.7			
		3.5.8			
		3.5.9			
		3.5.10			
		3.5.11			Report 7
		3.5.12			
		3.5.13			
		3.5.14	Student	3.2.1	
3.6 Add Student Type	3.6 Add Student Type	3.6.1			
		3.6.2	Person, Student	3.2.1	
		3.6.3			

Requirement	Use Case	Processes	Entities	Screens	Reports
		3.6.4	StudentType	3.7	
		3.6.5			
		3.6.6		3.6.2	
		3.6.7			
		3.6.8			
		3.6.9			
		3.6.10			
		3.6.11	StudentType	3.7	
3.7 Search Student Type	3.7 Search Student Type	3.7.1			
		3.7.2	Person, Student	3.2.1	
		3.7.3			
		3.7.4	StudentType	3.3.7	
		3.7.5			
		3.7.6	StudentType	3.7	
3.8 Update Student Type	3.8 Update Student Type	3.8.1			
		3.8.2	Student, Person	3.2.1	
		3.8.3			
		3.8.4	StudentType	3.7	
		3.8.5			
		3.8.6	StudentType	3.8.1	
		3.8.7			
		3.8.8			

Requirement	Use Case	Processes	Entities	Screens	Reports
		3.8.9			
		3.8.10			Report 4
		3.8.11			
		3.8.12			
		3.8.13	StudentType	3.7	
3.9 Delete Student Type	3.9 Delete Student Type	3.9.1			
		3.9.2	Student, Person	3.2.1	
		3.9.3			
		3.9.4	StudentType	3.7	
		3.9.5			
		3.9.6	StudentType	3.9.1	
		3.9.7			
		3.9.8			
		3.9.9			
		3.9.10			
		3.9.11			
		3.9.12	StudentType	3.7	

11.2.4 Subsystem 4

Requirement	Use Case	Processes	Entities	Screens	Reports
4.1 Add New Venue	4.1 Add New Venue	4.1.1			
		4.1.2			
		4.1.3			
		4.1.4		4.2	
		4.1.5			
		4.1.6	VenueType	4.1	
		4.1.7			
		4.1.8			
		4.1.9	Venue, Address		
		4.1.10			Report 1
4.2 Search Venue	4.2 Search Venue	4.2.1			
		4.2.2			
		4.2.3			
		4.2.4	Venue	4.2	
		4.2.5			
		4.2.6	Venue		
		4.2.7		4.2	
		4.2.8			
		4.2.9	Address	4.3.1	
4.3 Update Venue	4.3 Update Venue	4.3.1		4.3.1	
		4.3.2			
		4.3.3			

Requirement	Use Case	Processes	Entities	Screens	Reports
		4.3.4			Report 4
		4.3.5			
		4.3.6	Venue, Address		
		4.3.7			Report 2
4.4 Delete Venue	4.4 Delete Venue	4.4.1		4.4.1	
		4.4.2			
		4.4.3			
		4.4.4			
		4.4.5	Venue, Address		
		4.4.6			Report 3
4.5 Add Venue Type	4.5 Add Venue Type	4.5.1			
		4.5.2			
		4.5.3			
		4.5.4	Venue	4.2	
		4.5.5			
		4.5.6	VenueType	4.6	
		4.5.7			
		4.5.8		4.5	
		4.5.9			
		4.5.10			
		4.5.11	VenueType		
		4.5.12			

Requirement	Use Case	Processes	Entities	Screens	Reports
		4.5.13	VenueType	4.6	
4.6 Search Venue Type	4.6 Search Venue Type	4.6.1			
		4.6.2			
		4.6.3			
		4.6.4	Venue	4.2	
		4.6.5			
		4.6.6	VenueType	4.6	
		4.6.7			
		4.6.8		4.7.1	
		4.6.9			
		4.6.10	VenueType	4.6	
4.7 Update Venue Type	4.7 Update Venue Type	Alt 4.6.7			
		Alt 4.6.8			
		4.7.1			
		4.7.2			
		4.7.3			
		4.7.4	Venue	4.2	
		4.7.5			
		4.7.6	VenueType	4.6	
		4.7.7			
		4.7.8		4.7.1	
		4.7.9			

Requirement	Use Case	Processes	Entities	Screens	Reports
		4.7.10			
		4.7.11			Report 4
		4.7.12			
		4.7.13	VenueType		Report 2
		4.7.14			
		4.7.15	VenueType	4.6	
		Alt 4.7.7			
		Alt 4.7.8			
4.8 Delete Venue Type	4.8 Delete Venue Type	4.8.1			
		4.8.2			
		4.8.3			
		4.8.4	Venue	4.2	
		4.8.5			
		4.8.6	VenueType	4.6	
		4.8.7			
		4.8.8			
		4.8.9			
		4.8.10	VenueType		Report 3
		4.8.11			
		4.8.12	VenueType	4.6	
		Alt 4.8.7			
		Alt 4.8.7			
		Alt 4.8.8		4.8.1	

Requirement	Use Case	Processes	Entities	Screens	Reports
4.9 Add Residence	4.9 Add Residence	4.9.1			
		4.9.2	Venue		
		4.9.3			
		4.9.4	Venue	4.2	
		4.9.5			
		4.9.6		4.10	
		4.9.7			
		4.9.8		4.9.2	
		4.9.9			
		4.9.10			
		4.9.11	Residence		Report 1
4.10 Search Residence	4.10 Search Residence	4.10.1			
		4.10.2	Venue	4.2	
		4.10.3			
		4.10.4	Residence	4.10	
		4.10.5			
		4.10.6	Residence		
4.11 Update Residence	4.11 Update Residence	4.11.1			
		4.11.2	Venue	4.2	
		4.11.3			
		4.11.4	Residence	4.10	
		4.11.5			

Requirement	Use Case	Processes	Entities	Screens	Reports
		4.11.6	Residence	4.11.1	
		4.11.7			
		4.11.8			
		4.11.9			
		4.11.10			Report 4
		4.11.11			
		4.11.12			Report 2
		4.11.13	Residence	4.10	
4.12 Delete Residence	4.12 Delete Residence	4.12.1			
		4.12.2	Venue	4.2	
		4.12.3			
		4.12.4	Residence	4.10	
		4.12.5			
		4.12.6	Residence	4.12.1	
		4.12.7			
		4.12.8			
		4.12.9			
		4.12.10			Report 3
		4.12.11			
		4.12.12	Residence	4.10	

11.2.5 Subsystem 5

Requirement	Use Case	Processes	Entities	Screens	Reports
5.1 Create Function	5.1 Create Function, AUC 2 Create Event	AUC 2.1			
		AUC 2.2			
		AUC 2.3			
		AUC 2.4		5.0.2	
		Alt AUC 2.5			
		5.1.1		5.1.1	
		5.1.2			
		5.1.3	Venue, GuestSpeaker		
		5.1.4			
		5.1.5			
		5.1.6			
		5.1.7		5.1.2	
		5.1.8			
		5.1.9			
		5.1.10			
		5.1.11		5.1.3	
		5.1.12			
		5.1.13	Event, Function	5.1.4	
		5.1.14			
		5.1.15	Event	5.0.1	
		Alt 5.1.5			
		AUC 2.1			

Requirement	Use Case	Processes	Entities	Screens	Reports
5.2 Create Community Outreach	5.2 Create Community Outreach, AUC 2 Create Event	AUC 2.2			
		AUC 2.3			
		AUC 2.4		5.0.2	
		Alt AUC 2.5			
		5.2.1			
		5.2.2			
		5.2.3			
		5.2.4			
		5.2.5			
		5.2.6			
		5.2.7		5.2.2	
		5.2.8			
		5.2.9			
		5.2.10			
		5.2.11		5.2.3	
		5.2.12			
		5.2.13	Event, Content	5.2.4	
		5.2.14			
		5.2.15	Event		
5.3 Create Lecture	5.3 Create Lecture, AUC 2 Create Event	AUC 2.1			
		AUC 2.2			
		AUC 2.3			
		AUC 2.4		5.0.2	

Requirement	Use Case	Processes	Entities	Screens	Reports
		Alt AUC 2.5			
		5.3.1			
		5.3.2			
		5.3.3	Venue, Residence, LectureContent	5.3.1	
		5.3.4			
		5.3.5		5.3.2	
		5.3.6			
		5.3.7			
		5.3.8			
		5.3.9		5.3.3	
		5.3.10			
		5.3.11	Residence, Venue, Lecture		
		5.3.12	Event, Lecture		
		5.3.13		5.3.4	Report 1
5.4 Search Event	5.4 Search Event	5.4.1			
		5.4.2			
		5.4.3			
		5.4.4	Event	5.0.1	
		5.4.5			
		5.4.6		5.4.1	
		5.4.7			
		5.4.8		5.4.2	

Requirement	Use Case	Processes	Entities	Screens	Reports
		5.4.9			
		5.4.10		5.0.1	
5.5 RSVP to an Event	5.5 RSVP to an Event	5.5.1	Event	5.0.1	
		5.5.2			
		5.5.3		5.5.1	
		5.5.4			
		5.5.5	RSVP_Event		
		5.5.6			
		5.5.7			
		Alt 5.5.2		5.0.1	
5.6 Update Event Information	5.6 Update Event Information	5.6.1			
		5.6.2			
		5.6.3			
		5.6.4	Event	5.0.1	
		5.6.5			
		5.6.6			
		5.6.7		5.4.2	
		5.6.8			
		5.6.9			
		5.6.10		5.6.1	
		5.6.11			
		5.6.12	Function, Lecture, CommunityOutreach, Event		

Requirement	Use Case	Processes	Entities	Screens	Reports
		5.6.13			
		5.6.14	Event	5.0.1	
5.7 Cancel Event	5.7 Cancel Event	5.7.1			
		5.7.2			
		5.7.3			
		5.7.4	Event	5.0.1	
		5.7.5			
		5.7.6			
		5.7.7		5.4.2	
		5.7.8			
		5.7.9		5.7.1	
		5.7.10			
		5.7.11	Lecture, Function, Event, CommunityOutreach		
		5.7.12			
		5.7.13	Event	5.0.1	
5.8 Log Event Attendance	5.8 Log Event Attendance	5.8.1			
		5.8.2		5.8.1	
		5.8.3			
		5.8.4	RSVP_Event, Student		
		5.8.5	Student		
		5.8.6			
		5.8.7			

Requirement	Use Case	Processes	Entities	Screens	Reports
		5.8.8		5.8.3	
		5.8.9			
		5.8.10	Attendance, StudentMilestone	5.8.4	
		Alt 5.8.5			
		Alt 5.8.5b		3.2.1	
5.9 Send Notification	5.9 Send Notification	5.9.1			
		5.9.2		5.9.1	
		5.9.3			
		5.9.4		5.9.2	
		5.9.5			
		5.9.6		5.9.3	
		5.9.7			
		5.9.8			
		5.9.9	Notification	5.9.4	

11.2.6 Subsystem 6

Requirement	Use Case	Processes	Entities	Screens	Reports
6.1 Register Guest Speaker	6.1 Register Guest Speaker	6.1.1			
		6.1.2	GuestSpeaker	6.0.1	
		6.1.3			
		6.1.4		6.1.1	
		6.1.5			
		6.1.6		6.1.2	
		6.1.7			
		6.1.8			
		6.1.9	GuestSpeaker		
6.2 Search Guest Speaker	6.2 Search Guest Speaker	6.2.1			
		6.2.2	Guest Speaker	6.0.1	
		6.2.3			
		6.2.4	GuestSpeaker	6.2.2	
6.3 Update Guest Speaker	6.3 Update Guest Speaker	6.3.1			
		6.3.2	GuestSpeaker	6.0.1	
		6.3.3			
		6.3.4	GuestSpeaker		
		6.3.5			
		6.3.6	GuestSpeaker	6.2.2	
		6.3.7			
		6.3.8		6.3.1	
		6.3.9			

Requirement	Use Case	Processes	Entities	Screens	Reports
		6.3.10			
		6.3.11	GuestSpeaker		
6.4 Delete Guest Speaker	6.4 Delete Guest Speaker	6.4.1			
		6.4.2	GuestSpeaker	6.0.1	
		6.4.3			
		6.4.4	GuestSpeaker		
		6.4.5			
		6.4.6		6.3.1	
		6.4.7			
		6.4.8		6.4.1	
		6.4.9			
		6.4.10	GuestSpeaker		

11.2.7 Subsystem 7

Requirement	Use Case	Processes	Entities	Screens	Reports
7.1 Upload Content	7.1Upload Content	7.1.1			
		7.1.2			
		7.1.3			
		7.1.4		7.0.1	
		7.1.5			
		7.1.6		7.1.1	
		7.1.7		7.1.2	
		7.1.8			
		7.1.9	Content		
		7.1.10			Report 1
7.2 Search Content	7.2Search Content	7.2.1			
		7.2.2			
		7.2.3			
		7.2.4	Content	7.0.1	
		7.2.5			
		7.2.6	Content	7.2.1	
7.3 Update Content	7.3Update Content	7.3.1			
		7.3.2			
		7.3.3		7.3.2	
		7.3.4			
		7.3.5			
		7.3.6	Content		

Requirement	Use Case	Processes	Entities	Screens	Reports
		7.3.7			Report 3
7.4 Delete Content	7.4Delete Content	7.4.1			
		7.4.2			
		7.4.3		7.4.1	
		7.4.4			
		7.4.5	Content		
		7.4.6			Report 3
7.5 Review Lecture Content	7.5Review Lecture Content	7.5.1			
		7.5.2			
		7.5.3			
		7.5.4	Content	7.0.1	
		7.5.5			
		7.5.6		7.5.1	
		7.5.7			
		7.5.8			
		7.5.9			
		7.5.10		7.5.2	
		7.5.11			
		7.5.12	Review		

11.2.8 Subsystem 8

Requirement	Use Case	Processes	Entities	Screens	Reports
8.1 Upload Photo	8.1 Upload Photo	8.1.1			
		8.1.2		8.1.1	
		8.1.3			
		8.1.4		8.1.2	
		8.1.5			
		8.1.6			
		8.1.7			
		8.1.8			
		8.1.9		8.1.4	
		8.1.10			
		8.1.11			Report 1
		8.1.12			
		8.1.13		8.1.1	
		Alt 8.1.4	Event	5.0.1	
8.2 Post Photo	8.2 Post Photo	8.2.1			
		8.2.2		8.1.1	
		8.2.3			
		8.2.4		8.2.1	
		8.2.5			
		8.2.6			
		8.2.7			
		8.2.8		8.1.1	

Requirement	Use Case	Processes	Entities	Screens	Reports
		Alt 8.2.4	Event	5.0.1	
8.3 Delete Photo	8.3 Delete Photo	8.3.1			
		8.3.2		8.1.1	
		8.3.3			
		8.3.4		8.3.1	
		8.3.5			
		8.3.6		8.3.2	
		8.3.7			
		8.3.8			
		Alt 8.3.4	Event	5.0.1	

11.2.9 Subsystem 9

Requirement	Use Case	Processes	Entities	Screens	Reports
9.1 Generate Class Attendance Report	9.1 Generate Class Attendance Report	9.1.1			
		9.1.2			
		9.1.3			
		9.1.4		9.1	
		9.1.5			
		9.1.6	Event, Lecture, Attendance		
		9.1.7			Report 9.1
		Alt 9.1.6	Event	5.0.1	
9.2 Generate Function Attendance Report	9.2 Generate Function Attendance Report	9.2.1			
		9.2.2			
		9.2.3			
		9.2.4		9.1	
		9.2.5			
		9.2.6	Attendance, Event, Function		
		9.2.7		9.2	Report 9.2
		Alt 9.2.6	Event	5.0.1	
9.3 Generate Community Engagement Attendance Report	9.3 Generate Community Engagement Attendance Report	9.3.1			
		9.3.2			
		9.3.3			
		9.3.4		9.1	

Requirement	Use Case	Processes	Entities	Screens	Reports
		9.3.5			
		9.3.6	Attendance, Event, CommunityOutreach		
		9.3.7		9.3	Report 9.3
		Alt 9.3.6	Event	5.0.1	
9.4 Generate Demographics Report	9.4 Generate Demographics Report	9.4.1			
		9.4.2			
		9.4.3			
		9.4.4		9.1	
		9.4.5			
		9.4.6	Student	9.4	
		9.4.7			Report 9.4
		Alt 9.4.6	Event	5.0.1	
9.5 Generate Event Popularity Report	9.5 Generate Event Popularity Report	9.5.1			
		9.5.2			
		9.5.3			
		9.5.4		9.1	
		9.5.5			
		9.5.6	Attendance, Event, Lecture, Function, CommunityOutreach		
		9.5.7		9.5	Report 9.5

Requirement	Use Case	Processes	Entities	Screens	Reports
9.6 Generate Feedback Report	9.6 Generate Feedback Report	9.6.1			
		9.6.2			
		9.6.3			
		9.6.4		9.1	
		9.6.5			
		9.6.6	VolunteerFeedback		
		9.6.7		9.6	Report 9.6
9.7 Generate User Statistics Report	9.7 Generate User Statistics Report	9.7.1			
		9.7.2			
		9.7.3			
		9.7.4		9.1	
		9.7.5			
		9.7.6	Person		
		9.7.7		9.7	Report 9.7
		Alt 9.7.6	Event	5.0.1	
9.8 Generate Student Progress Report	9.8 Generate Student Progress Report	9.8.1			
		9.8.2			
		9.8.3			
		9.8.4		9.1	
		9.8.5			

Requirement	Use Case	Processes	Entities	Screens	Reports
		9.8.6	Student, Attendance, StudentMilestone		
		9.8.7		9.8	Report 9.8
		Alt 9.8.6	Event	5.0.1	

11.3 Conclusion

11.3.1 The purpose of the validation was to ensure that the functional specifications detailed above are in line with all the requirements of the system. It displayed the requirements for each subsystem, the use case linked to each subsystem, each process in the primitive level that pertain to that use case, the entities that are affected by each process, the screens that are displayed for each process and every report linked to the process.

12. Complexity

Topic	Level		Marks	M
1. Special GUI	Appropriate MDS and SDI form design of the system	*	3	Yes
	Appropriate use of grids		3	Yes
	Appropriate use of tabs		3	Yes
	Use of graphs in an appropriate business context		3	Yes
	The storage and display of graphical information, like photos with a good business reason		3	Yes
	Working e-mail automatically generated from the database in an appropriate business context		3	Yes
	SMS messages automatically generated from the system in an appropriate business context		3	Yes
	Extensive user-friendly search facility		3	Yes
	At least one use of a tree to display data		3	Yes

Topic	Level		Marks	M
2. Database access	At least one use of a calendar view of data		3	Yes
	Uploading a file into the system with appropriate business reason		3	Yes
	The use of multimedia in an appropriate business context		3	Yes
	At least one use of a timer in an appropriate business context		3	Yes
3. Reports	At least 30 tables used (4 member groups) or 40 tables used (5 member groups)	*	6	Yes
	Full referential integrity on all tables	*	6	Yes
	At least one use of master-detail table relationships	*	3	Yes
3. Reports	At least 5 simple list reports in a reporting tool (no control breaks, no graphs, single table)	*	6	Yes
	At least 1 transactional report with 2 or more control breaks (with heading and total lines, multiple tables)		3	No

Topic	Level	Marks	M
	At least 1 management report using a graph	3	Yes
4. Flexibility	All data that can change in future should not be hard coded but maintained in a sub-module of the system (e.g. Lookup tables)	6	Yes
	Some business rules are not hard coded, but maintained in a sub-module of the system.	6	Yes
5. Error handling	All system-generated errors are trapped and consistent, user-friendly error messages are displayed	6	Yes
	Appropriate data validation on all input fields	6	Yes
6. Help	At least one menu item or other control that opens up a complete help document (HTML, PDF, Help-file)	3	Yes
	Extensive context-sensitive help. E.g. calling Help on a specific screen/function will automatically open	6	No

Topic	Level	Marks	M
7. Security	the specific help for that screen/function.		
	Search Facility on Help	3	Yes
	Extensive use of hints	3	Yes
8. Audit Trail	Logon screen with user ID and password and fixed user profiles	3	Yes
	Encrypted passwords in database	3	Yes
	Flexible user profiles (i.e. you can dynamically add user profiles that will enable/disable access to certain parts of the system)	6	Yes
	An audit trail of all transactions in the system showing at least date, time, user, transaction type, critical data (such as amount and quantity of transaction)	6	Yes
	Able to search the audit trail on any of the following: date, user, transaction type	3	Yes

Topic	Level	Marks	M
9. Installation	Fully functional installation disks that take care of application installation requirements (install and uninstall)	6	No
	Fully functional installation disks that take care of database installation requirements (including database settings)	6	No
10. Backup and Restore	A backup and restore subsystem exists that backup/restore all data (system may exit during restore)	6	Yes
11. Import/Export Data	OLE: Opens Word or Excel and automatically places data in it based on the selected data in the calling screen (with good business reason)	6	Yes
	Text File: At least 1 text file for Importing or Exporting of data (with good business reason)	3	Yes
	XML: At least 1 XML file for Importing or Exporting of data (with good business reason)	3	Yes

Topic	Level	Marks	M
12. External INPUT device	Simple Link to an external INPUT device using Windows plug-and-play technology. (This could include a swipe card reader, bar code reader, etc.)	3	No
	Loose Link to an external INPUT device using device specific software. Data or images must seamlessly be stored in the database but device specific software is visible to the user. (This could include a digital camera, scanner, voice recording device, thump print reader, etc.)	6	No
	Tight Link to an external INPUT device using device specific software. Data or images must seamlessly be stored in the database but device specific software is not visible to the user. (This could include a digital camera, scanner, voice recording device, thump print reader, etc.)	9	No
13. External APPLICATION / Services	Integrate an existing web service into your application (with good business reason)	3	Yes

Topic	Level	Marks	M
14. Web processing	A fully functional link to an installed external application system exists and the interface must be shown to work on the external system. Note that this excludes Microsoft Office Applications	6	No
	At least one appropriate business use of static Web-pages (e.g. Help files or an advertisement for your client)	3	Yes
	Substantial Web-server processing – Display data from a database on the browser	3	Yes
	Substantial Web-server processing – add data from a browser into the system	6	Yes
	Substantial Web-server processing – uploading a file from a local PC to the web-server (integrated into the system)	6	Yes
	Substantial mobile device processing integrated into the system (e.g. Smartphone, Tablet)	9	Yes

Topic	Level	Marks	M
15. Programming Principles	The system consists of three distinct tiers: data; business; and presentation. Each of the levels consists of a separate application object.	6	Yes
	Basic interfacing to the Windows system registry for appropriate application data and settings	3	No
	Comprehensive use of stored procedures and/or triggers	3	Yes

13. Sign-off by Team

The team hereby agrees that each member has contributed towards and agrees with the contents of the above document.

Signed on this _____ of July 2017 at _____.

Cailin Smith**Christopher Oakes**

Amogelang Moloko**Achal Seechoonparsad**

Jacquiline Lawler

14. Sign-off by Client

I _____ hereby agree to the contents of the above document.

Signed on this the _____ of July 2017 at _____.

Nerene Grobler