System and Software Architecture Description (SSAD)

GOTRLA

Team 15

Team members	Roles
Deepak Earayil	Prototyper ,Project Manager & Tester
Ankith Nagarle	Prototype , Developer and Software/System Architect
Aayush Jain	Feasibility Analyst & Tester
Presha Thakkar	Prototyper ,Developer and Operational Concept Engineer
Suhani Vyas	Life Cycle Planner and Requirements Engineer
Anushila Dey	Developer and Software/System Architect
Nidhi Baheti	Operational Concept Engineer & Tester
Elaine Lo	Quality Analyst, Shaper & Tester

December 08, 2014

Version History

Date	Author	Version	Changes made	Rationale
08/25/05	PA	2.0	• Original template for use with Instructional ICM-Sw v1.0	• Initial draft for use with Instructional ICM-Sw v1.0
05/22/09	SK	2.1	• Embedded description in each Table	• To be consistent with ICM EPG template set standard V2.1
10/13/14	AD	2.2	• Use case description	• Detailed information regarding use cases
10/20/14	AD, AJ	2.3	• Updated all the diagrams	• Changes were made according to feedback given during ARB
11/30/14	NB, SV,EL	3.0	• Updated all the diagrams, added sequence diagrams	• Based on the feedback from ARB review and exit condition for Draft foundation review
12/08/14	NB, SV	3.1	• Updated section 3.2 and section 4	• Based on the feedback from TRR and exit condition for Final Foundation commitment package.

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1 Purpose of the SSAD

The purpose of this SSAD is to document the architecture and design of GOTRLA Smartphone application being developed. This SSAD gives the details of how the application is implemented and how are the stakeholders involved in various use cases of this application. While the developers and testers will use this document as a reference to the system architecture during development phase, other stakeholders like project staff and clients use this as a reference to understand the structure of the system once the proposed system is delivered. This system being developed will be faithful to the architecture specified in the SSAD.

1.2 Status of the SSAD

This is version 3.0 of the SSAD document.

2. System Analysis

2.1 System Analysis Overview

The GOTRLA application has two parts, namely, the mobile application and the web application. The primary purpose of the GOTRLA smartphone app is to display rosters of registered girls and allow volunteer coaches to enter attendance information of girls registered with the roster. The web application allows the program staff to login and view the girls' attendance information. It also allows the program staff to enter and manage rosters of various schools as well as details of registered girls. The application allows the program staff to export the attendance data of each roster and evaluate them. The program staff can also keep track of the attendance of the volunteer coaches.

2.1.1 System Context

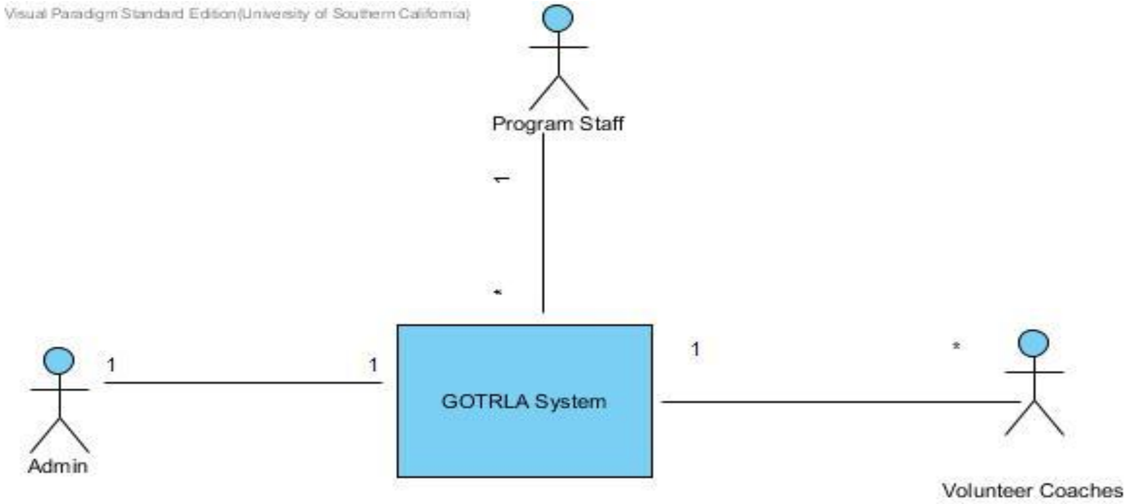

Visual Paradigm Standard Edition(University of Southern California)
Program Staff
1
*
Admin
1
1
GOTRLA System
1
*
Volunteer Coaches

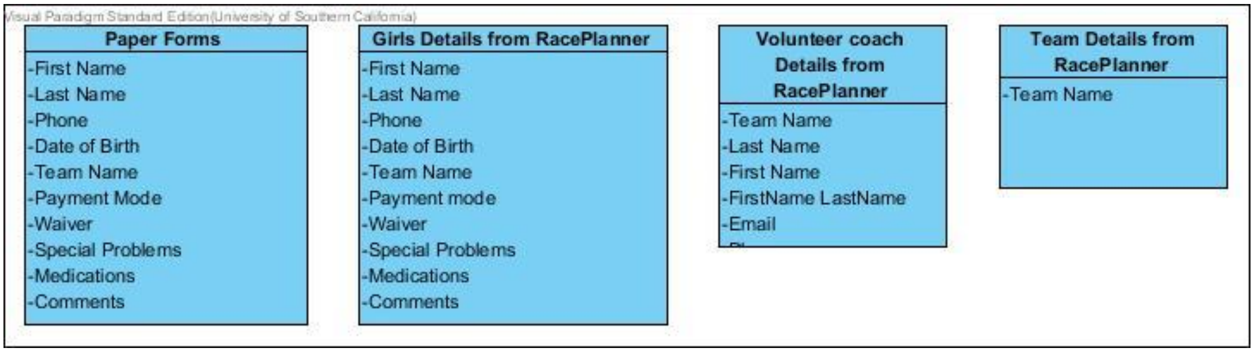
Figure 1: System Context Diagram

Table 1: Actors Summary

Actor	Description	Responsibilities
Mobile app User		

Actor	Description	Responsibilities
Volunteer Coach	Volunteer coaches are the users who are working as coaches for GOTRLA and conducting the various program sessions.	• Check in the girls and other coaches attendance. • Views attendance details • Modifies attendance details
Web app User		
Admin	Admin is the user who control the entire GOTRLA management	• Adds Users for the web application. • Modifies/ Deletes User details for web application. • Monitors/Views attendance details of Girls and Volunteer coaches
Program Staff	Employees who work for the GOTRLA management	• Adds/Modifies/Deletes Volunteer Coaches' details • Adds/Modifies/Deletes details of Registered Girls • Monitors/View attendance details of girls and volunteer coaches

2.1.2 Artifacts & Information



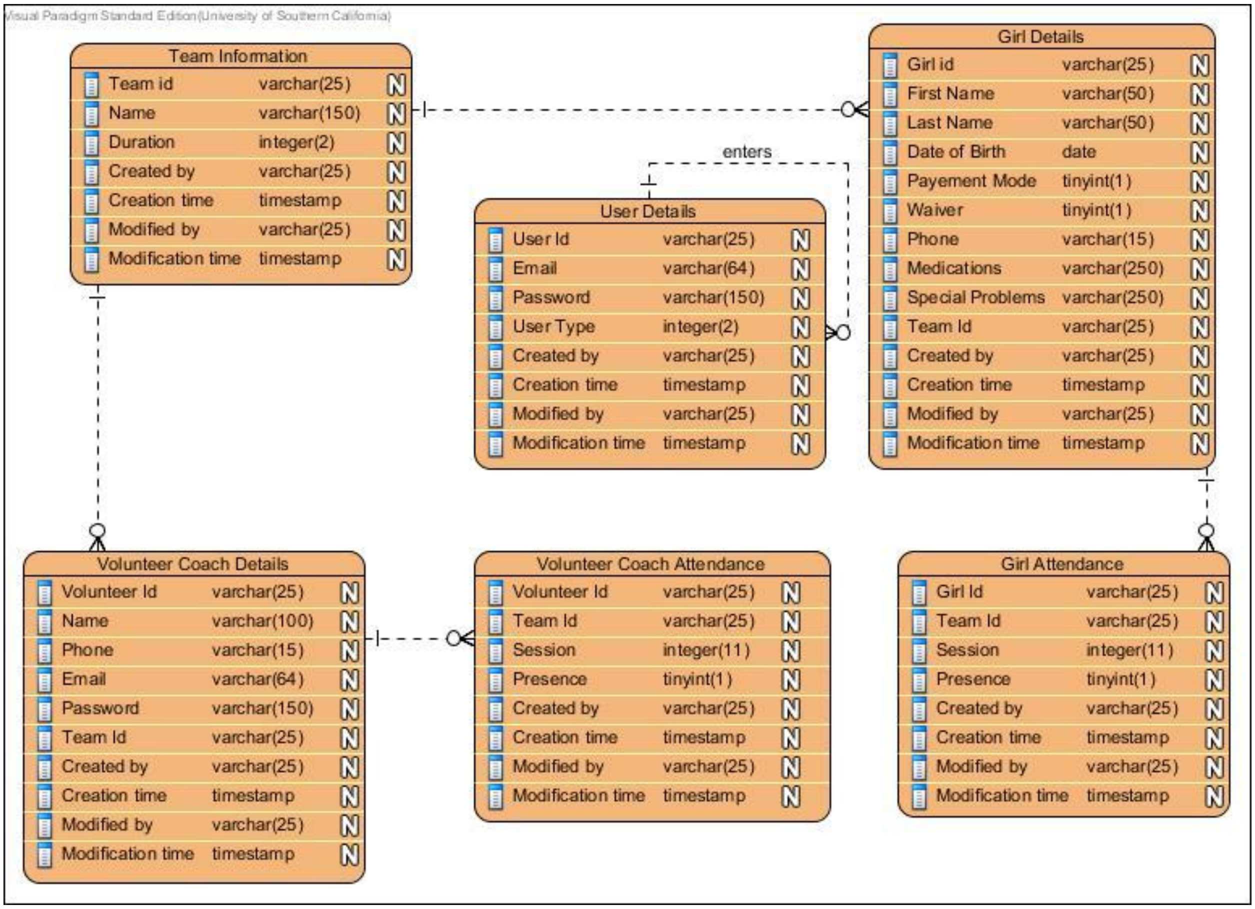


Figure 2: Artifacts and Information Diagram

Table 2: Artifacts and Information Summary

Artifact	Purpose
Girl Details from Race Planner	The information supplied by the race planner is used to collectively create girl’s profile and the list of girls enrolled.
Volunteer Details from Race Planner	This is used to import the volunteer details from Race Planner to our system.
Team Details from Race Planner	This is used to import the name of Teams from the Race Planner.
Paper forms	The information supplied by the paper forms is used to collectively create girl’s profile and the list of girls enrolled.
Girls Details	It stores the details of each girl such as name, date of birth, medical condition which is of business importance to the client.
Girl Attendance	Attendance of girls.
User Details	Stores the details of all the users of web application.
Volunteer Coach Details	It stores the contact information of volunteer.
Volunteer Coach	Attendance of volunteer coaches.

Attendance	
Team Details	Name and duration of team is stored.

2.1.3 Behavior

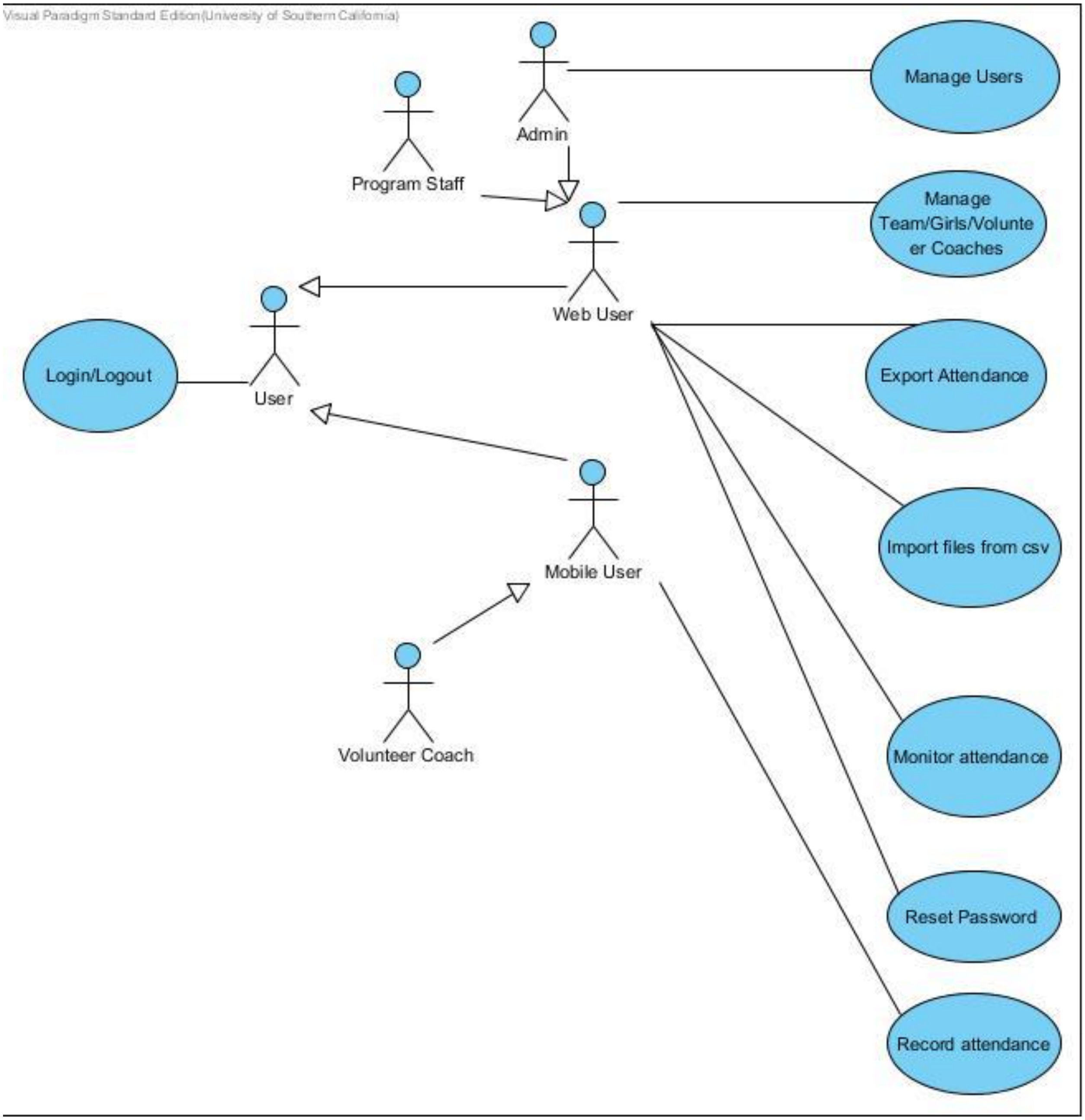


Figure 3: Process Diagram

2.1.3.1 User Management

2.1.3.1.1 Login/Logout

Table 3: Process Description

Identifier	UC -1 Login/logout
Purpose	Logs the user in or out of the system. Based on user's status, he/she will be given access to certain parts of the application
Requirements	WC_3276, WC_3275, WC_3279
Development Risks	None
Pre-conditions	Database is set up with user email address and password
Post-conditions	System operational state, no error condition not handled by the system

Table 4: Typical Course of Action: User Login Successfully

Seq#	Actor's Action	System's Response
1	User either enters http://gotrla.net/ into a web browser or opens the mobile application	Login page is displayed
2	User enters email address and password	
3	User clicks log-in button	
4		Checks database for matching email address and password
5		On success, system displays the home screen

Table 5: Typical Course of Action: User Logout Successfully

Seq#	Actor's Action	System's Response
1	User clicks log out	
2		Removes the user from the current session
		Redirects the user to a new login page

Table 6: Exceptional Course of Action: Incorrect Email entered

Seq#	Actor's Action	System's Response
1	User enters email address and/or password	
2		If invalid email address (not valid

		format), display an error message

Table 7: Exceptional Course of Action: Email and Password not matching.

Seq#	Actor's Action	System's Response
1	User enters email address and/or password	
2		Send all information to database and check for existing entry with the given email address or username
3		If email address and password do not match that in the database, display an error message

Table 8: Exceptional Course of Action: Web User forgets password.

Seq#	Actor's Action	System's Response
1	User either enters http://gotrla.net/ into a web browser or opens the mobile application	
2	Selects forgot password option.	
3		Redirects to a forgot password page.
4		Prompts to enter Email ID
5	Enters Email ID.	
6		Password reset link sent at the Email ID.
7	Opens Inbox and Selects the Link.	
8		Redirected to a Password reset page.
9	Enter New password.	
10		System sends all information to database and update the entry for the Password reset.
11		Password reset successfully display.

2.1.3.2 Web Application

2.1.3.2.1 Manage users

Table 9: Process Description

Identifier	UC -2 Manage users

Purpose	Allows admin to add/edit/delete data on users
Requirements	WC_3277
Development Risks	None
Pre-conditions	- SW/HW ready and internet is available - System database is properly initialized and user has correct access level to make adjustments
Post-conditions	- System operational state, no error condition not handled by the system - Data is saved to the database

Table 10: Typical Course of Action: User Added Successfully

Seq#	Actor's Action	System's Response
1	Admin open Add User page	
2	Admin enters details for the user on the form	
3	Admin clicks save	
4		System sends all information to database and update the entry for the user.

Table 11:Typical Course of Action: User deleted successfully

Seq#	Actor's Action	System's Response
1	Admin opens View Users Page	
2	Admin finds the user they want to delete	
3	User clicks the delete button	
4		The user information gets deleted from the database.

Table 12: Typical Course of Action: User information modified successfully

Seq#	Actor's Action	System's Response
1	Admin opens View Users Page	
2	Admin finds the user whose information he wants to edit	
3	Admin clicks the edit button	
4		System displays the add data form
5		System pulls the data to be edited and populates it in the add data form
6	Admin adjust the data on the form	
7	Clicks save	

8		System sends all information to database and update the entry for the user.

Table 13: Exceptional Course of Action: Incorrect User information entered

Seq#	Actor's Action	System's Response
1	Admin opens View Users Page	
2	Admin finds the user whose information he wants to edit	
3	Admin clicks the edit button	
4	Admin enters data on a form	
5		If incorrect or missing input data, display an error message

2.1.3.2.2 Manage Team/Girls/Volunteer coaches

Table 14: Process Description

Identifier	UC -3 Manage teams/girls/volunteer coaches
Purpose	Allows staff/admin to add/edit/delete data on teams, girls, and volunteer coaches
Requirements	WC_3269, WC_3270, WC_3271, WC_3265, WC_3273
Development Risks	None
Pre-conditions	- SW/HW ready and internet is available - System database is properly initialized and user has correct access level to make adjustments
Post-conditions	- System operational state, no error condition not handled by the system - Data is saved to the database

Table 15: Typical Course of Action: Girl/Volunteer/Team information added successfully

Seq#	Actor's Action	System's Response
1	Web user opens add page of Girls/Volunteer/Team	
2	Web user enters data on a form	
3	Clicks save	
4		System sends all information to database and update the entry for the Girl/Volunteer/Team.

Table 16: Typical Course of Action: Girl/Volunteer/Team information modified successfully

Seq#	Actor's Action	System's Response
1	Web user opens view page of Girls/Volunteer/Team.	
2	Finds the Girls/Volunteer/Team whose information user wants to modify	
3		System displays the add data form
4		System pulls the data to be edited and populates it in the add data form
5	Web user modifies the field.	
6	Clicks Save	
7		System sends all information to database and update the entry for the Girl/Volunteer/Team.

Table 17: Typical Course of Action: Girl/Volunteer/Team information deleted successfully

Seq#	Actor's Action	System's Response
1	Web user opens view page of Girls/Volunteer/Team.	
2	Finds the Girls/Volunteer/Team whom user wants to delete	
3	Clicks delete	
4		The Girls/Volunteer/Team information gets deleted from the database.

Table 18: Exceptional Course of Action: Incorrect Information entered

Seq#	Actor's Action	System's Response
1	Web user enters details for Girls/Volunteer/Team	
2	Clicks Save	
2		If incorrect or missing input data(not valid format), display an error message

Table 19: Exceptional Course of Action: Required Information missing

Seq#	Actor's Action	System's Response
1	Web user enters details for Girls/Volunteer/Team.	
2	Clicks Save	
2		Required field missing, display an error message.

2.1.3.2.3 Import files from CSV

Table 20: Process Description

Identifier	UC -5 Import files from CSV
Purpose	Allows staff/admin to import attendance on volunteers and girls
Requirements	WC_3269, WC_3270, WC_3271, WC_3265, WC_3273
Development Risks	None
Pre-conditions	- SW/HW ready and internet is available - CSV file is created with proper formatting and data
Post-conditions	- System operational state, no error condition not handled by the system - Data is saved to the database

Table 21: Typical Course of Action: Girls/Volunteer Data added successfully from csv

Seq#	Actor's Action	System's Response
1	Web User open Import Girls/Volunteer page	
2	Click choose file	
3	User select which file to import.	
4		System reads CSV file and pulls the data.
5		System sends all information to database and update the entry for the Girl/Volunteer.

Table 22: Exceptional Course of Action: csv file does not get imported

Seq#	Actor's Action	System's Response

1	Web User open Import Girls/Volunteer page.	
2	Click choose file	
3	User select which file to import	
4		System reads CSV file and determines that the data does have proper formatting.
5		Incorrect or missing input data, display an error message

2.1.3.2.4 Export attendance

Table 23: Process Description

Identifier	UC -4 Export attendance
Purpose	Allows staff/admin to export attendance on volunteers and girls
Requirements	WC_3278
Development Risks	None
Pre-conditions	- SW/HW ready and internet is available - System database is properly initialized and user has correct access level
Post-conditions	System operational state, no error condition not handled by the system

Table 24: Typical Course of Action: Attendance data gets exported successfully.

Seq#	Actor's Action	System's Response
1	Web user selects the attendance page	
2	User selects type of attendance to export (Girls/Volunteer)	
3	User clicks export	
4		Data gets compiled into an CSV file
5		System prompts user for destination folder
6	User selects destination	
7		File gets saved to user's computer

2.1.3.2.5 Monitor Attendance

Table 25: Process Description

Identifier	UC -6 Monitor attendance
Purpose	Allows staff/admin to view attendance data

Requirements	WC_3263
Development Risks	None
Pre-conditions	Attendance data is saved correctly in the database
Post-conditions	System operational state, no error condition not handled by the system

Table 26: Typical Course of Action: Girls/Volunteer attendance data displayed successfully

Seq#	Actor's Action	System's Response
1	Web User clicks Girls/Volunteer attendance page.	
2		System pulls the data to be displayed
3		Displays the attendance data of Girl/Volunteer.

2.1.3.3 Mobile Application

2.1.3.3.1 Record Attendance

Table 27: Process Description

Identifier	UC -7 Record attendance
Purpose	Allows coaches to record whether or not a student showed up on a particular day
Requirements	WC_3268, WC_3274, WC_3262
Development Risks	None
Pre-conditions	SW/HW ready and internet and GPRS is available
`	- System operational state, no error condition not handled by the system - Data is saved to the database

Table 28: Typical Course of Action: Girls presence recorded successfully

Seq#	Actor's Action	System's Response
1	Volunteer coach login successfully	
2	Selects a session	
3	Enters attendance details of each girl present (check in)	
4		System sends all information to

		database and update the entry for the Girl.

Table 29: Typical Course of Action: Volunteer presence recorded successfully

Seq#	Actor's Action	System's Response
1	Volunteer coach login successfully	
2	Selects session	
3	Volunteer coach checks in his presence.	
4		System sends all information to database and update the entry for the Volunteer.

Table 30: Exceptional Course of Action: Girls attendance overwritten.

Seq#	Actor's Action	System's Response
1	Volunteer coach enters attendance details of each girl	
2		If data for that session already exists, the previous data is overwritten with the new data.

2.1.4 Modes of Operation

The Web Application part of the system has two modes of operation:

- **Admin**: It has a different view wherein admin can modify program staff's information apart from all the other access rights which program staff has.
- **Program Staff**: Program staff cannot view other program staff's data.

2.2 System Analysis Rationale

Based on our analysis of how the users interact with the system, we have identified 2 classes of operational stakeholders.

1. **Admin and Program staff** - These users include Administrator and Program Staff of GOTRLA. The administrator is only user who has privilege to add/edit/delete the users of the web application. The program staffs have privilege to add/edit/delete the girl's data, volunteer data and team data. The users can use the web application to access the system with various permissions and functionalities based on the type of user.
2. **Volunteer Coaches** - These are users who can use the mobile application to check in girls attendance. The volunteer can access the application only when they are added to the system by program staff via web application. This would generate their username and password for the mobile application. They can use it after they get their details.

There is 1 external system:

Race Planner – this is the legacy system used by the client to store details of the girls; the administrator would extract data from the race planner and import the data into the developed system.

3. Technology-Specific System Design

3.1 Design Overview

3.1.1 System Structure

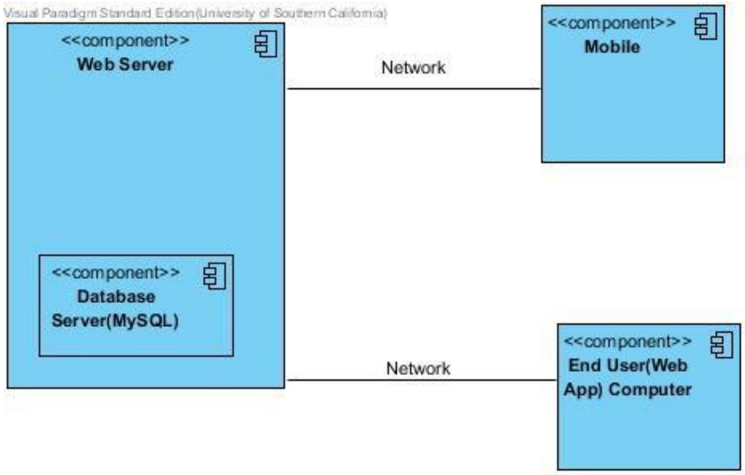


Figure 4: Hardware Component Class Diagram

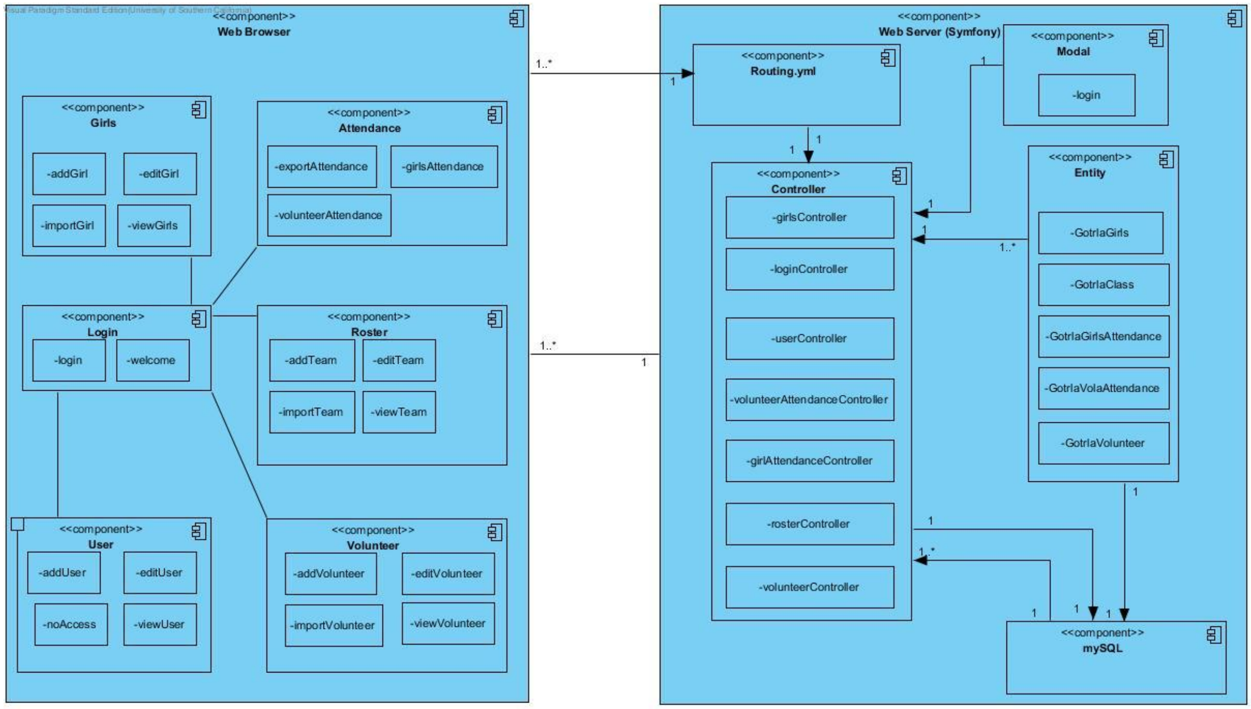


Figure 5: Software Component Class Diagram

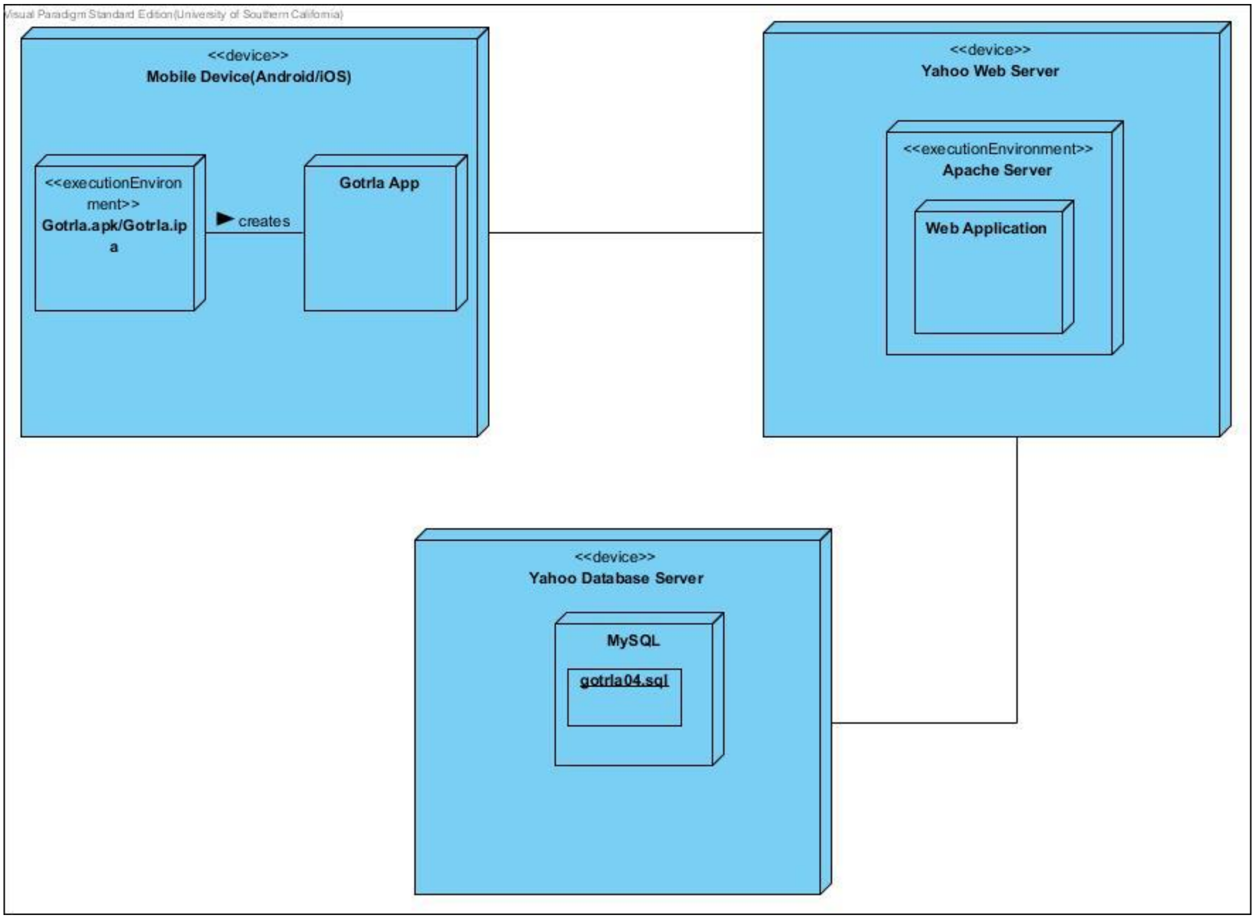


Figure 6: Deployment Diagram

Table 31: Hardware Component Description

Hardware Component	Description
Web Application Server	Used for the web application.
Mobile Device	Used by the volunteer to run the mobile application
Computer	Used by the program staff and admin to access web application..

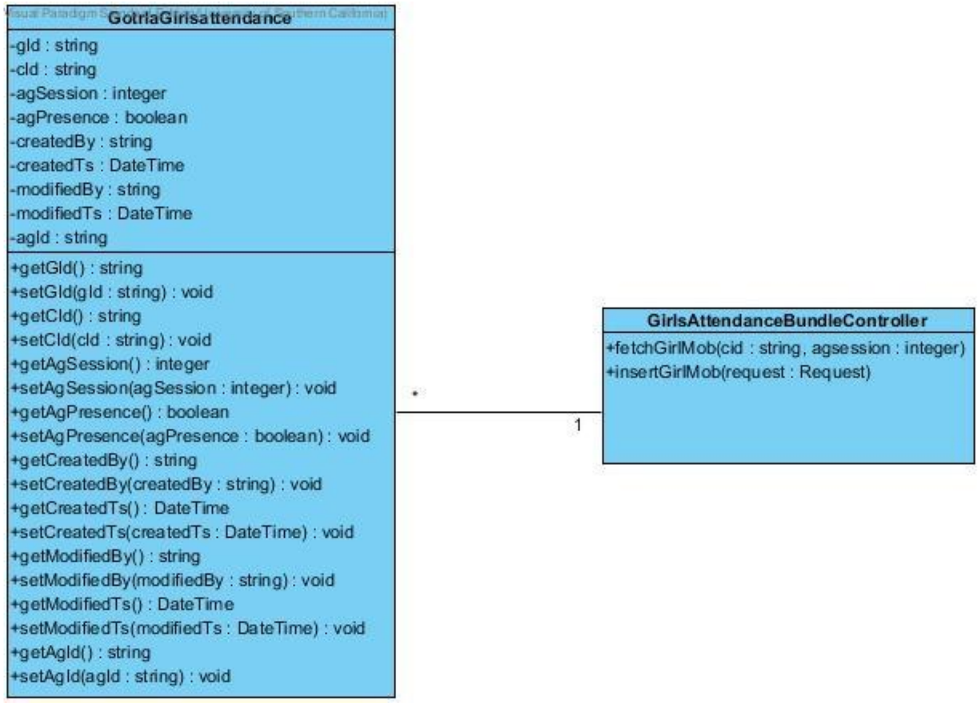
Table 32: Software Component Description

Software Component	Description
Login Page	User would use this to login to the web app.
Welcome Page	It is the page that displays when the user successfully logins.
Add Girls Page	This page is used to add girls to the database
View Girls Page	Once the girls are added, they can be viewed on this page.
Edit Girls Page	To change the details of any girls, user use this page.
Import Girls Page	To import girl’s data user will use this page.

Add Volunteer Page	This page is used to add volunteer to the database
View Volunteer Page	Once the volunteer are added, they can be viewed on this page.
Edit Volunteer Page	To change the details of any volunteer, user use this page.
Import Volunteer Page	To import volunteer's data user will use this page.
Add Users Page	This page is used to add users and their privilege to the database by the admin.
View Users Page	Once the users are added, they can be viewed on this page.
Edit Users Page	To change the details of any users, admin use this page.
Girls Attendance Page	This page shows the number of girls present and absent in a particular session.
Volunteer Attendance Page	This page shows the number of volunteer present and absent in a particular class.
Export Attendance Page	This page is used to export complete attendance.
Add Team Page	This page is used to add team to the database
View Team Page	Once the teams are added, they can be viewed on this page.
Edit Team Page	To change the details of any team, user use this page.
Import Team Page	To import team's data user will use this page.

3.1.2 Design Classes

3.1.2.1 Girls Attendance Management

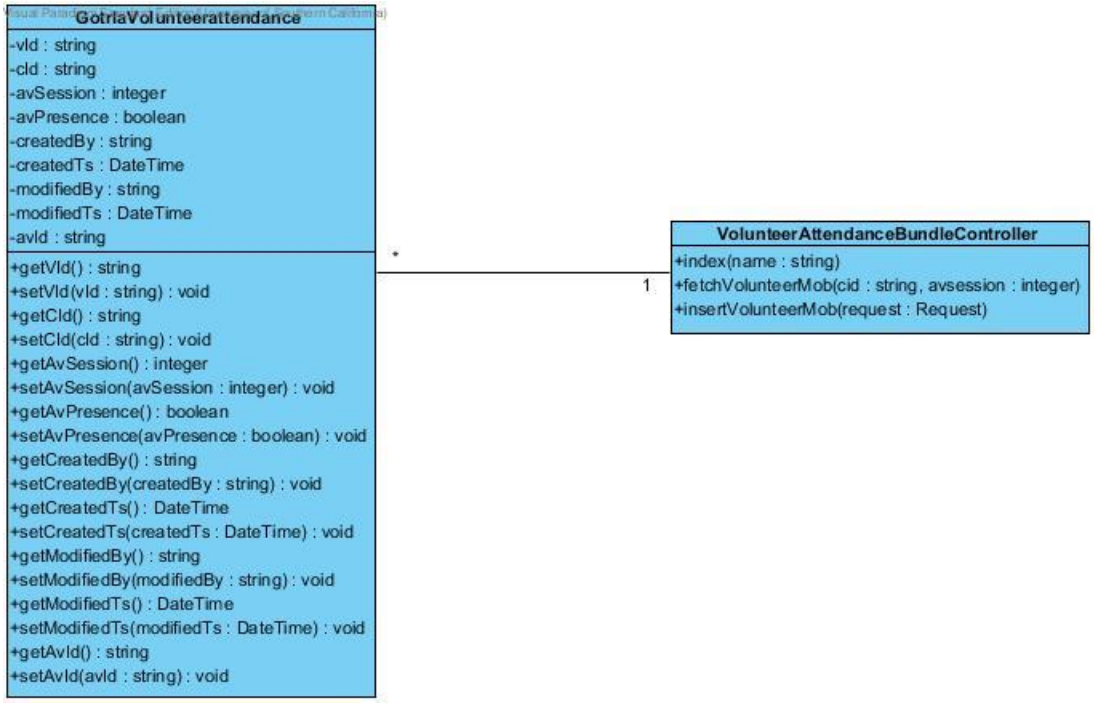

GotrlaGirlsattendance
-gId : string
-cId : string
-agSession : integer
-agPresence : boolean
-createdBy : string
-createdTs : DateTime
-modifiedBy : string
-modifiedTs : DateTime
-agId : string
+getGId() : string
+setGId(gId : string) : void
+getCId() : string
+setCId(cId : string) : void
+getAgSession() : integer
+setAgSession(agSession : integer) : void
+getAgPresence() : boolean
+setAgPresence(agPresence : boolean) : void
+getCreatedBy() : string
+setCreatedBy(createdBy : string) : void
+getCreatedTs() : DateTime
+setCreatedTs(createdTs : DateTime) : void
+getModifiedBy() : string
+setModifiedBy(modifiedBy : string) : void
+getModifiedTs() : DateTime
+setModifiedTs(modifiedTs : DateTime) : void
+getAgId() : string
+setAgId(agId : string) : void
*
1
GirlsAttendanceBundleController
+fetchGirlMob(cid : string, agsession : integer)
+insertGirlMob(request : Request)

Class	Type	Description
GotrlaGirlsAttendance	Entity	It represents all the information related to

		Girls attendance.
GirlsAttendanceBundle	Controller	Contains all the functions related to Girls attendance monitoring

Table 33: GotrlaGirlsAttendance Class Description

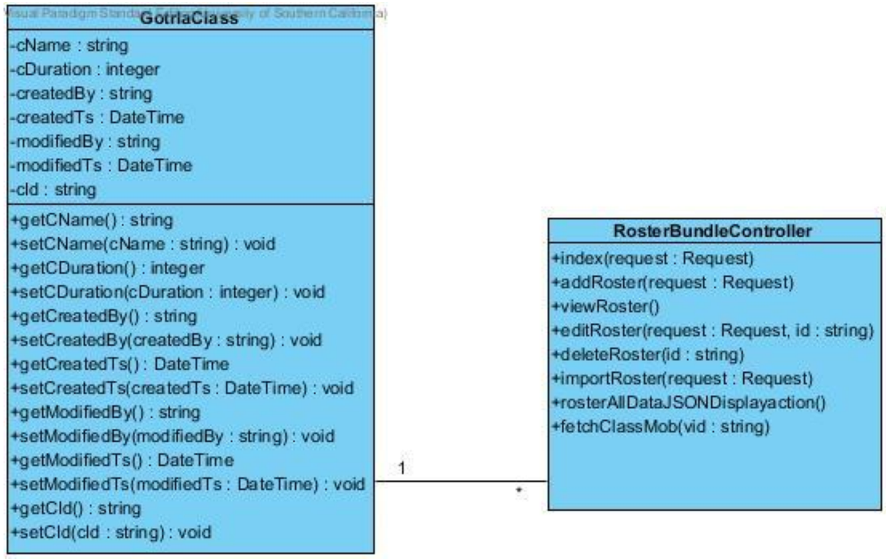
3.1.2.2 GotrlaVolunteer Management


GotrlaVolunteerattendance
-vId : string
-cId : string
-avSession : integer
-avPresence : boolean
-createdBy : string
-createdTs : DateTime
-modifiedBy : string
-modifiedTs : DateTime
-avId : string
+getVId() : string
+setVId(vId : string) : void
+getCId() : string
+setCId(cId : string) : void
+getAvSession() : integer
+setAvSession(avSession : integer) : void
+getAvPresence() : boolean
+setAvPresence(avPresence : boolean) : void
+getCreatedBy() : string
+setCreatedBy(createdBy : string) : void
+getCreatedTs() : DateTime
+setCreatedTs(createdTs : DateTime) : void
+getModifiedBy() : string
+setModifiedBy(modifiedBy : string) : void
+getModifiedTs() : DateTime
+setModifiedTs(modifiedTs : DateTime) : void
+getAvId() : string
+setAvId(avId : string) : void
*
1
VolunteerAttendanceBundleController
+index(name : string)
+fetchVolunteerMob(cid : string, avsession : integer)
+insertVolunteerMob(request : Request)

Class	Type	Description
GotrlaVolunteer	Entity	It represents all the information related to Volunteer attendance.
VolunteerAttendanceBundle	Controller	Contains all the functions related to Volunteer attendance monitoring

Table 34: GotrlaVolunteerAttendance Class Description

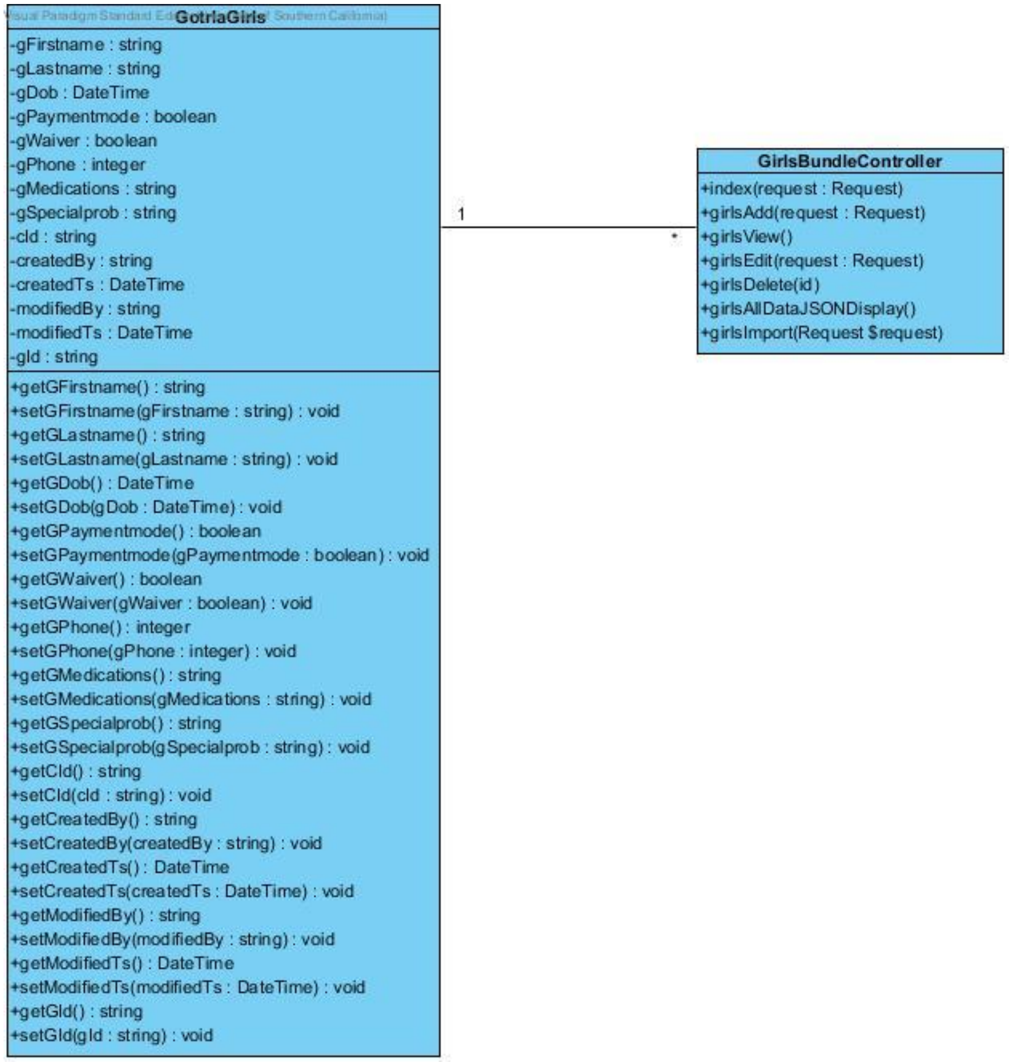
3.1.2.3 GotrlaClass Management


GotrlaClass
-cName : string
-cDuration : integer
-createdBy : string
-createdTs : DateTime
-modifiedBy : string
-modifiedTs : DateTime
-cId : string
+getCName() : string
+setCName(cName : string) : void
+getCDuration() : integer
+setCDuration(cDuration : integer) : void
+getCreatedBy() : string
+setCreatedBy(createdBy : string) : void
+getCreatedTs() : DateTime
+setCreatedTs(createdTs : DateTime) : void
+getModifiedBy() : string
+setModifiedBy(modifiedBy : string) : void
+getModifiedTs() : DateTime
+setModifiedTs(modifiedTs : DateTime) : void
+getCId() : string
+setCId(cId : string) : void
1
*
RosterBundleController
+index(request : Request)
+addRoster(request : Request)
+viewRoster()
+editRoster(request : Request, id : string)
+deleteRoster(id : string)
+importRoster(request : Request)
+rosterAllDataJSONDisplayaction()
+fetchClassMob(vid : string)

Class	Type	Description
GotrlaClass	Entity	It represents all the information related to Tlass session..
RosterBundle	Controller	Contains all the functions related to Girls attendance monitoring

Table 35: GotrlaClass Class Description

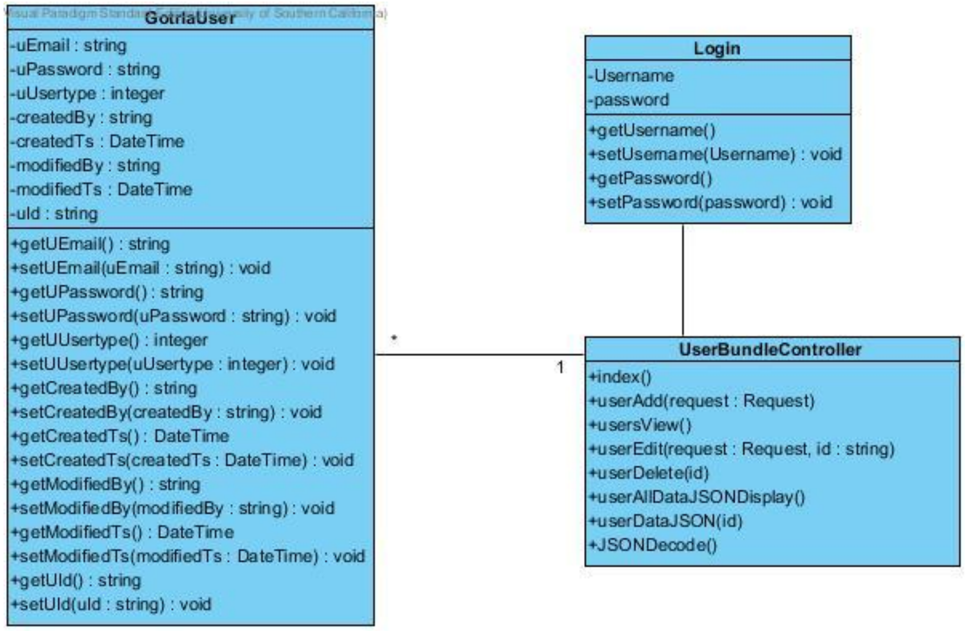
3.1.2.4 GotrlaGirls Management


GotrlaGirls
-gFirstname : string
-gLastname : string
-gDob : DateTime
-gPaymentmode : boolean
-gWaiver : boolean
-gPhone : integer
-gMedications : string
-gSpecialprob : string
-cId : string
-createdBy : string
-createdTs : DateTime
-modifiedBy : string
-modifiedTs : DateTime
-gId : string
+getGFirstname() : string
+setGFirstname(gFirstname : string) : void
+getGLastname() : string
+setGLastname(gLastname : string) : void
+getGDob() : DateTime
+setGDob(gDob : DateTime) : void
+getGPaymentmode() : boolean
+setGPaymentmode(gPaymentmode : boolean) : void
+getGWaiver() : boolean
+setGWaiver(gWaiver : boolean) : void
+getGPhone() : integer
+setGPhone(gPhone : integer) : void
+getGMedications() : string
+setGMedications(gMedications : string) : void
+getGSpecialprob() : string
+setGSpecialprob(gSpecialprob : string) : void
+getCId() : string
+setCId(cId : string) : void
+getCreatedBy() : string
+setCreatedBy(createdBy : string) : void
+getCreatedTs() : DateTime
+setCreatedTs(createdTs : DateTime) : void
+getModifiedBy() : string
+setModifiedBy(modifiedBy : string) : void
+getModifiedTs() : DateTime
+setModifiedTs(modifiedTs : DateTime) : void
+getGId() : string
+setGId(gId : string) : void
1
*
GirlsBundleController
+index(request : Request)
+girlsAdd(request : Request)
+girlsView()
+girlsEdit(request : Request)
+girlsDelete(id)
+girlsAllDataJSONDisplay()
+girlsImport(Request $request)

Class	Type	Description
GotrlaGirls	Entity	It represents all the information related to all registered Girls.
GirlsBundle	Controller	Contains all the functions to access Girls information.

Table 36: GotrlaGirls Class Description

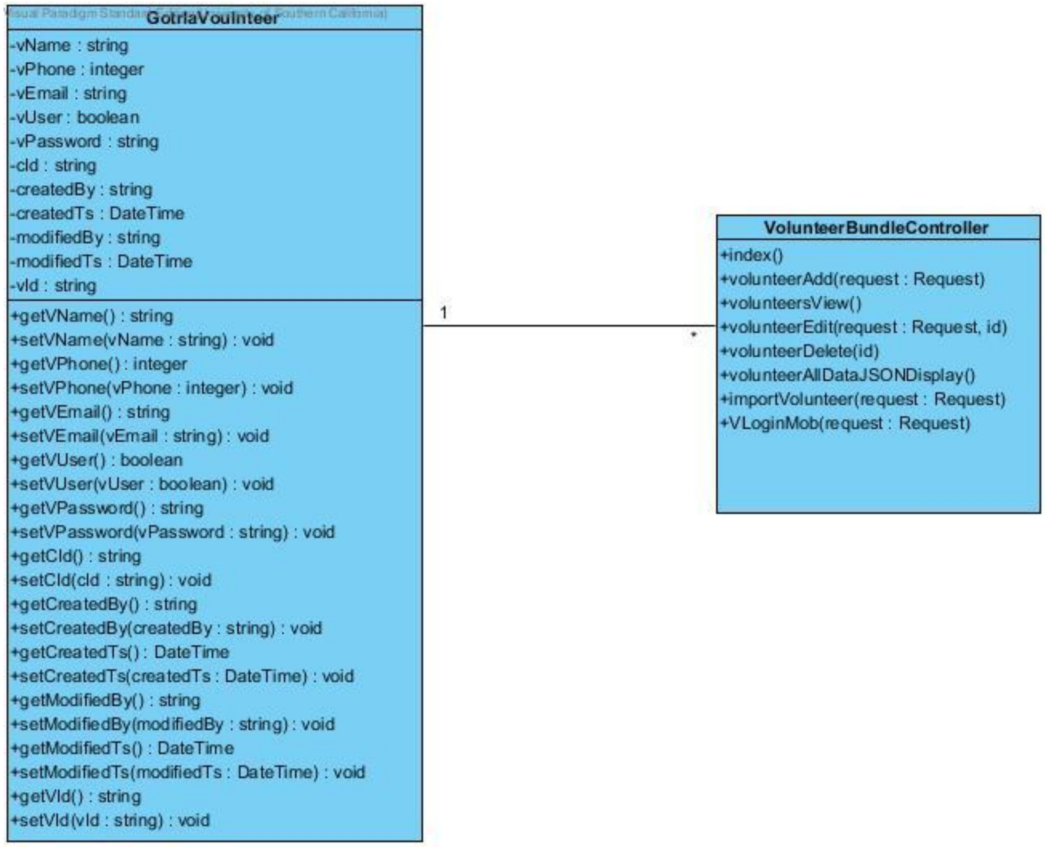
3.1.2.5 GotrlaUser Management


Visual Paradigm Standard Edition(University of Southern California)
GotrlaUser
-uEmail : string
-uPassword : string
-uUsertype : integer
-createdBy : string
-createdTs : DateTime
-modifiedBy : string
-modifiedTs : DateTime
-uId : string
+getUEmail() : string
+setUEmail(uEmail : string) : void
+getUPassword() : string
+setUPassword(uPassword : string) : void
+getUUsertype() : integer
+setUUsertype(uUsertype : integer) : void
+getCreatedBy() : string
+setCreatedBy(createdBy : string) : void
+getCreatedTs() : DateTime
+setCreatedTs(createdTs : DateTime) : void
+getModifiedBy() : string
+setModifiedBy(modifiedBy : string) : void
+getModifiedTs() : DateTime
+setModifiedTs(modifiedTs : DateTime) : void
+getUId() : string
+setUId(uId : string) : void
*
1
Login
-Username
-password
+getUsername()
+setUsername(Username) : void
+getPassword()
+setPassword(password) : void
UserBundleController
+index()
+userAdd(request : Request)
+usersView()
+userEdit(request : Request, id : string)
+userDelete(id)
+userAllDataJSONDisplay()
+userDataJSON(id)
+JSONDecode()

Class	Type	Description
GotrlaUser	Entity	It represents all the information related to any user of the system.
UserBundle	Controller	Contains all the functions related to User management .
Login	Modal	It stores all the information regarding user authentication during login/logout.

Table 37: GotrlaUser Class Description

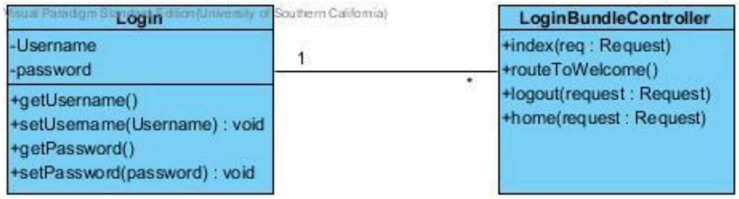
3.1.2.6 GotrlaVolunteer Management


GotrlaVoulnteer
-vName : string
-vPhone : integer
-vEmail : string
-vUser : boolean
-vPassword : string
-cId : string
-createdBy : string
-createdTs : DateTime
-modifiedBy : string
-modifiedTs : DateTime
-vId : string
+getVName() : string
+setVName(vName : string) : void
+getVPhone() : integer
+setVPhone(vPhone : integer) : void
+getVEmail() : string
+setVEmail(vEmail : string) : void
+getVUser() : boolean
+setVUser(vUser : boolean) : void
+getVPassword() : string
+setVPassword(vPassword : string) : void
+getCId() : string
+setCId(cId : string) : void
+getCreatedBy() : string
+setCreatedBy(createdBy : string) : void
+getCreatedTs() : DateTime
+setCreatedTs(createdTs : DateTime) : void
+getModifiedBy() : string
+setModifiedBy(modifiedBy : string) : void
+getModifiedTs() : DateTime
+setModifiedTs(modifiedTs : DateTime) : void
+getVId() : string
+setVId(vId : string) : void
1
*
VolunteerBundleController
+index()
+volunteerAdd(request : Request)
+volunteersView()
+volunteerEdit(request : Request, id)
+volunteerDelete(id)
+volunteerAllDataJSONDisplay()
+importVolunteer(request : Request)
+VLoginMob(request : Request)

Class	Type	Description
GotrlVolunteer	Entity	It represents all the information related to Volunteer management .
VolunteerBundle	Controller	Contains all the functions related to volunteer management.

Table 38: GotrlaVolunteer Class Description

3.1.2.7 Login/Logout Management



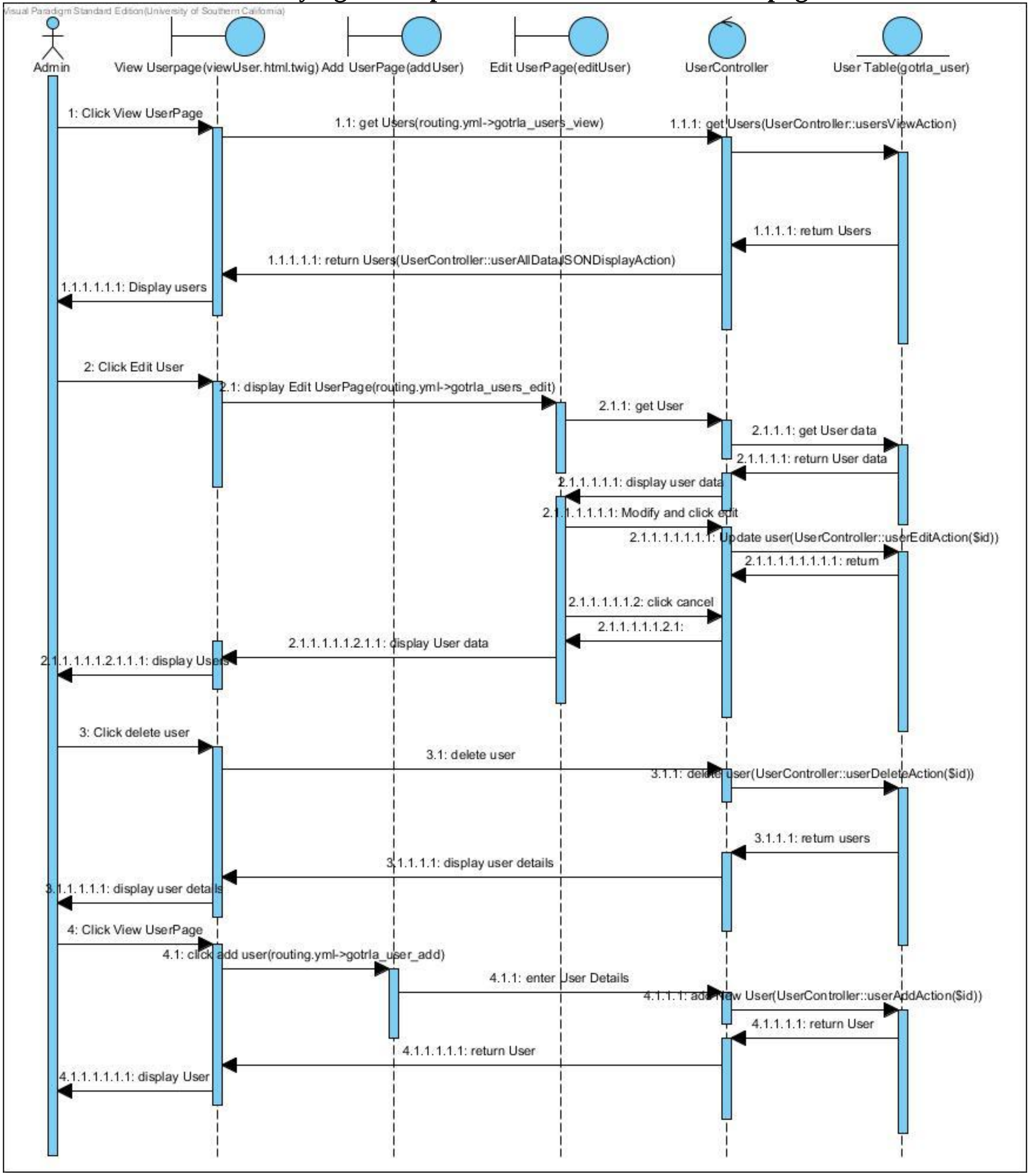
Class	Type	Description
Login	Modal	It stores the information regarding the username and password of different users.

LoginBundle	Controller	Contains all the functions related to login/logout session of application.

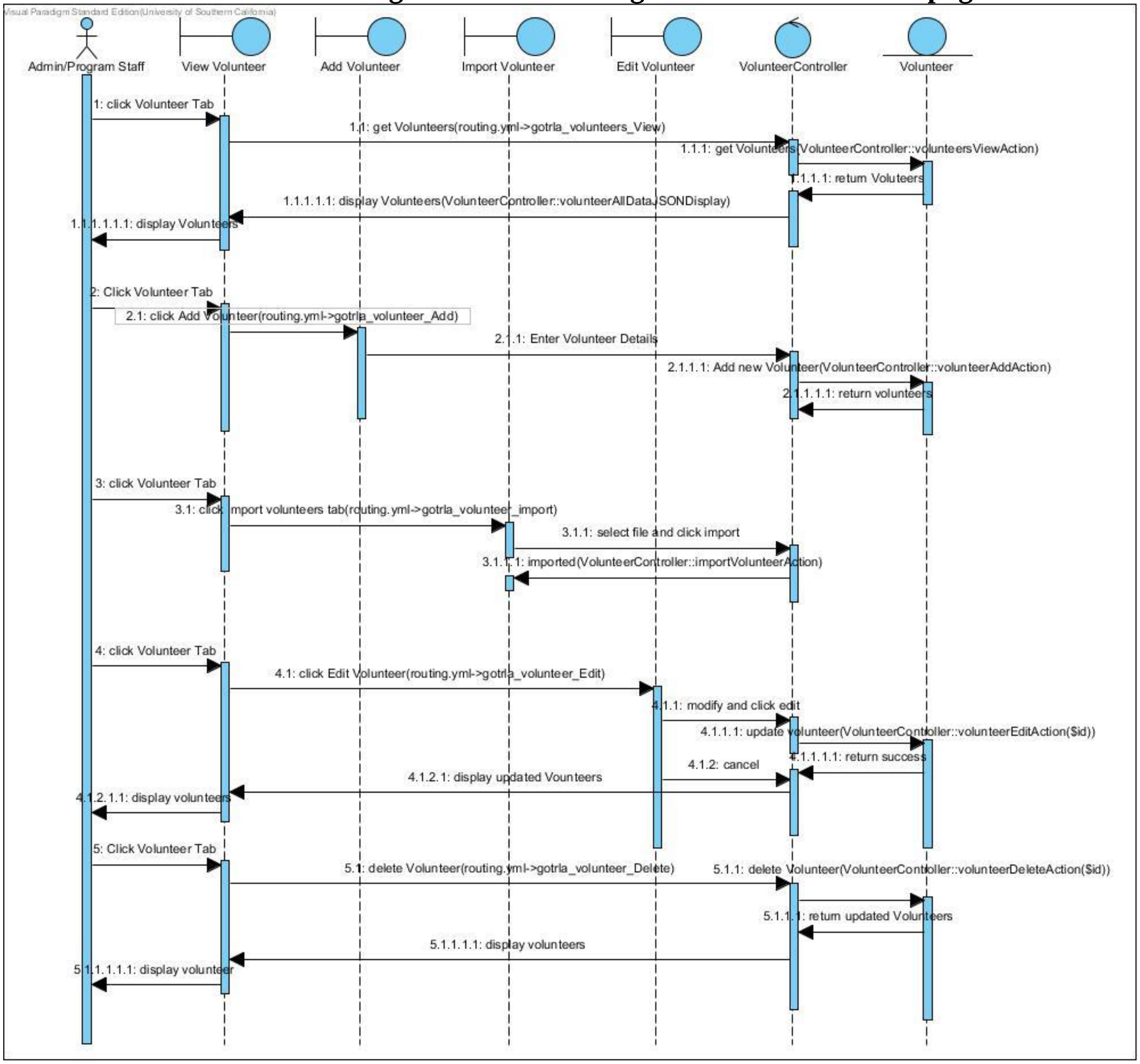
Table 39: Login Modal Description

3.1.3 Process Realization

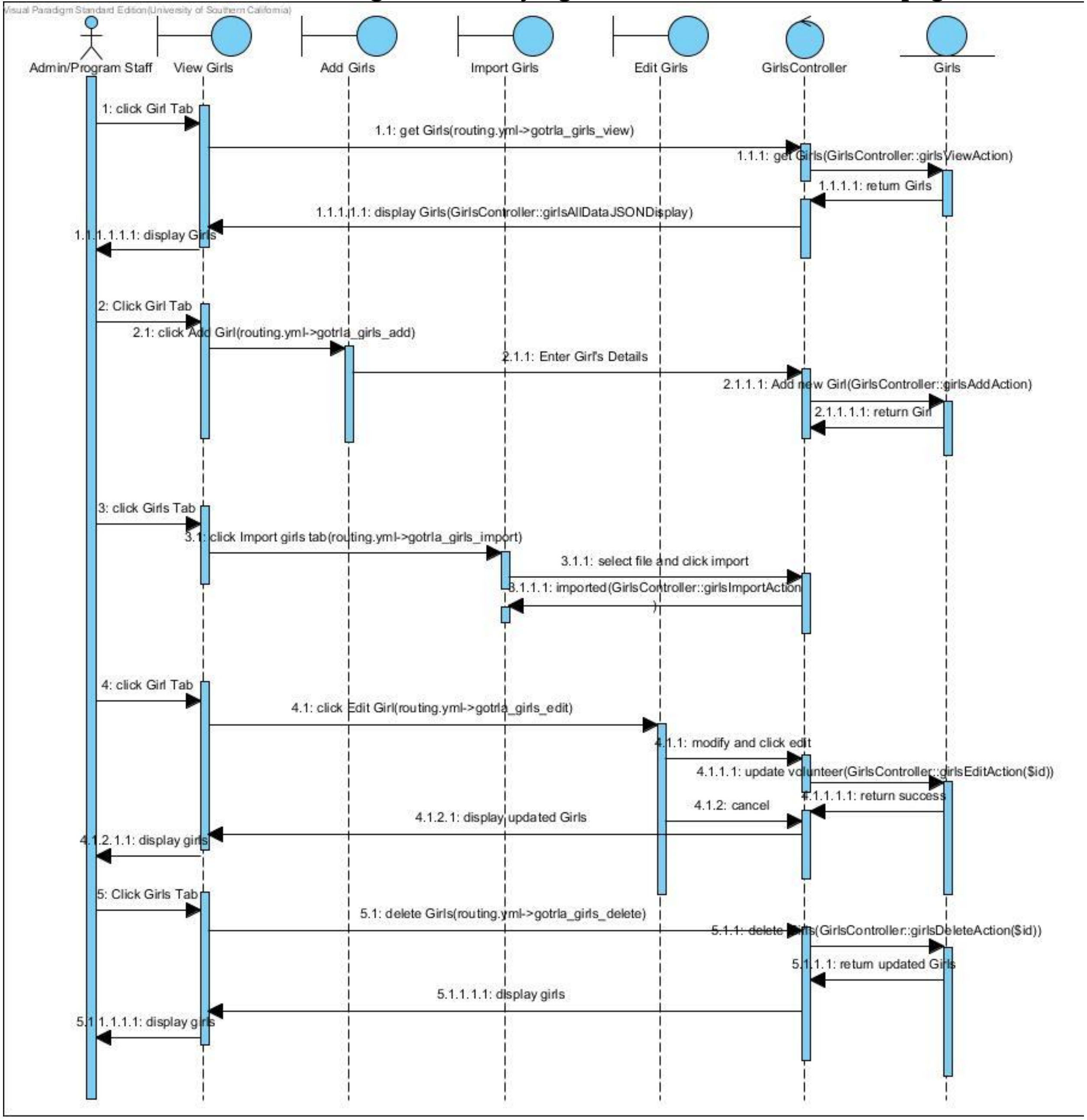
3.1.3.1 Scenario: Admin trying to do operations on User Tab of Webpage


Visual Paradigm Standard Edition(University of Southern California)
Admin
View Userpage(viewUser.html.twig)
Add UserPage(addUser)
Edit UserPage(editUser)
UserController
User Table(gotrla_user)
1: Click View UserPage
1.1: get Users(routing.yml->gotrla_users_view)
1.1.1: get Users(UserController::usersViewAction)
1.1.1.1: return Users
1.1.1.1.1: return Users(UserController::userAllDataJSONDisplayAction)
1.1.1.1.1.1: Display users
2: Click Edit User
2.1: display Edit UserPage(routing.yml->gotrla_users_edit)
2.1.1: get User
2.1.1.1: get User data
2.1.1.1.1: return User data
2.1.1.1.1.1: display user data
2.1.1.1.1.1.1: Modify and click edit
2.1.1.1.1.1.1.1: Update user(UserController::userEditAction($id))
2.1.1.1.1.1.1.1.1: return
2.1.1.1.1.1.2: click cancel
2.1.1.1.1.1.2.1:
2.1.1.1.1.1.2.1.1: display User data
2.1.1.1.1.1.2.1.1.1: display Users
3: Click delete user
3.1: delete user
3.1.1: delete user(UserController::userDeleteAction($id))
3.1.1.1: return users
3.1.1.1.1: display user details
3.1.1.1.1.1: display user details
4: Click View UserPage
4.1: click add user(routing.yml->gotrla_user_add)
4.1.1: enter User Details
4.1.1.1: add New User(UserController::userAddAction($id))
4.1.1.1.1: return User
4.1.1.1.1.1: return User
4.1.1.1.1.1.1: display User

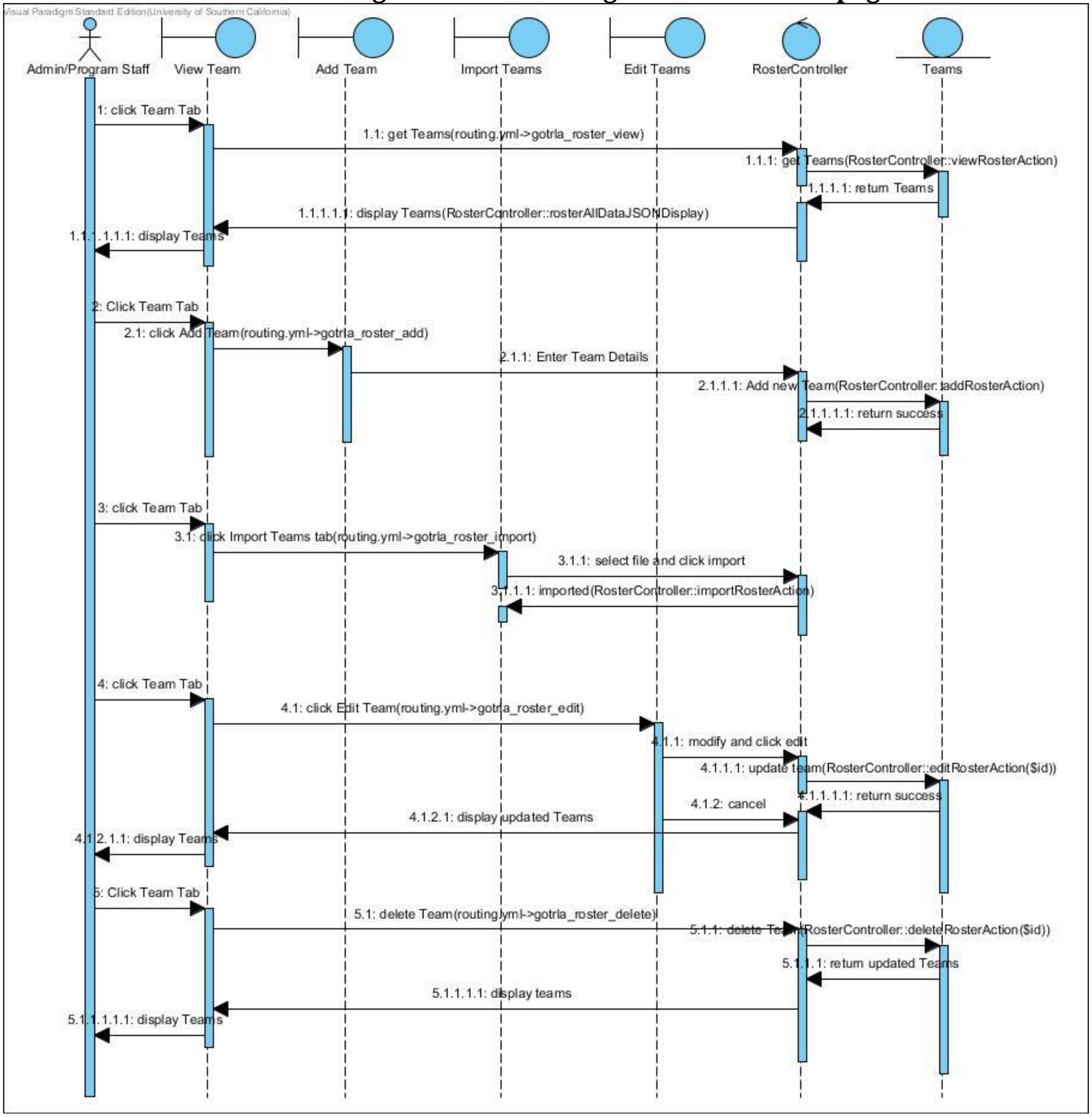
3.1.3.2 Scenario: Admin or Program Staff accessing Volunteer tab on Webpage


Visual Paradigm Standard Edition(University of Southern California)
Admin/Program Staff
View Volunteer
Add Volunteer
Import Volunteer
Edit Volunteer
VolunteerController
Volunteer
1: click Volunteer Tab
1.1: get Volunteers(routing.yml->gotrla_volunteers_View)
1.1.1: get Volunteers(VolunteerController::volunteersViewAction)
1.1.1.1: return Voluteers
1.1.1.1.1: display Volunteers(VolunteerController::volunteerAllDataJSONDisplay)
1.1.1.1.1.1: display Volunteers
2: Click Volunteer Tab
2.1: click Add Volunteer(routing.yml->gotrla_volunteer_Add)
2.1.1: Enter Volunteer Details
2.1.1.1: Add new Volunteer(VolunteerController::volunteerAddAction)
2.1.1.1.1: return volunteers
3: click Volunteer Tab
3.1: click import volunteers tab(routing.yml->gotrla_volunteer_import)
3.1.1: select file and click import
3.1.1.1: imported(VolunteerController::importVolunteerAction)
4: click Volunteer Tab
4.1: click Edit Volunteer(routing.yml->gotrla_volunteer_Edit)
4.1.1: modify and click edit
4.1.1.1: update volunteer(VolunteerController::volunteerEditAction($id))
4.1.1.1.1: return success
4.1.2: cancel
4.1.2.1: display updated Vounteers
4.1.2.1.1: display volunteers
5: Click Volunteer Tab
5.1: delete Volunteer(routing.yml->gotrla_volunteer_Delete)
5.1.1: delete Volunteer(VolunteerController::volunteerDeleteAction($id))
5.1.1.1: return updated Volunteers
5.1.1.1.1: display volunteers
5.1.1.1.1.1: display volunteer

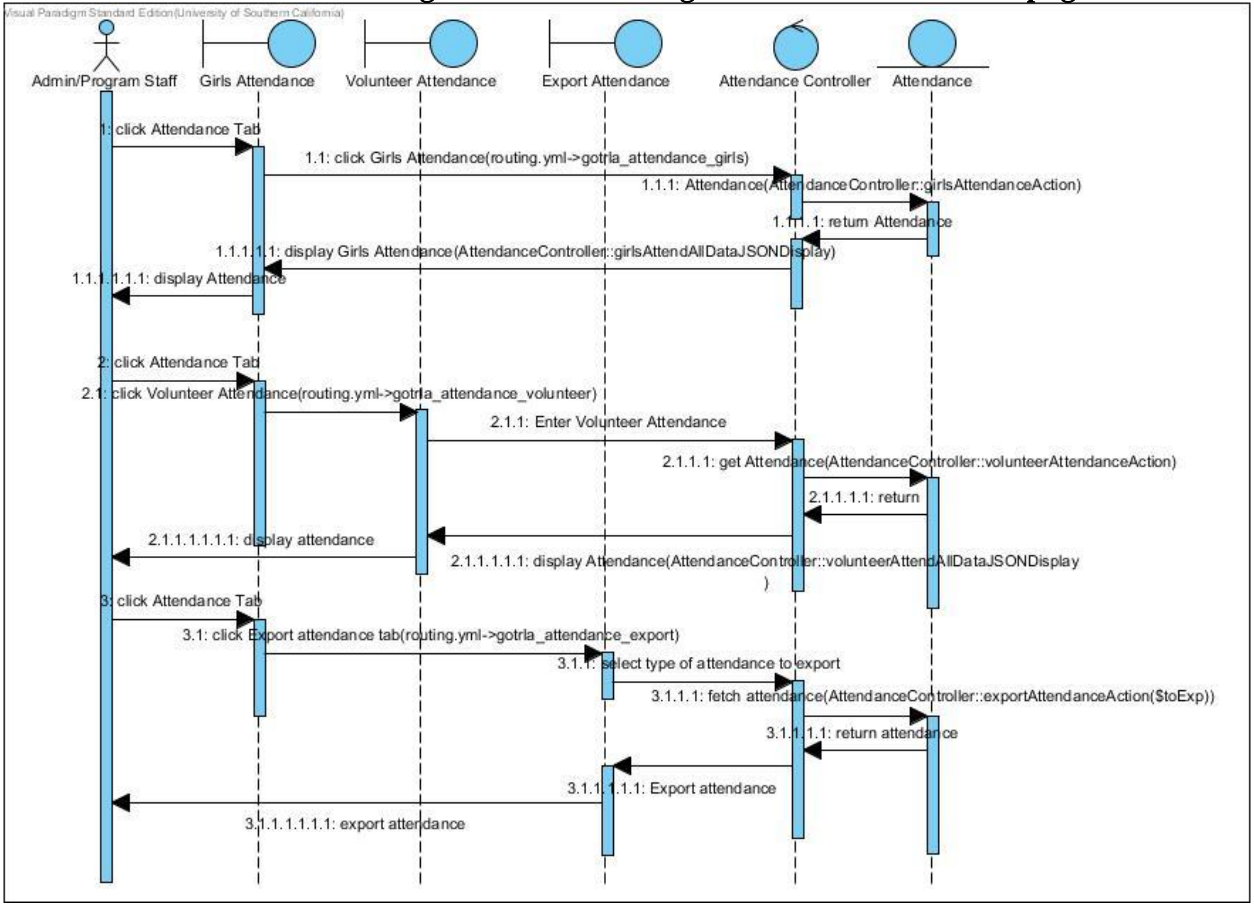
3.1.3.3 Scenario: Admin or Program staff trying to access Girls Tab on Webpage


Visual Paradigm Standard Edition(University of Southern California)
Admin/Program Staff
View Girls
Add Girls
Import Girls
Edit Girls
GirlsController
Girls
1: click Girl Tab
1.1: get Girls(routing.yml->gotrla_girls_view)
1.1.1: get Girls(GirlsController::girlsViewAction)
1.1.1.1: return Girls
1.1.1.1.1: display Girls(GirlsController::girlsAllDataJSONDisplay)
1.1.1.1.1.1: display Girls
2: Click Girl Tab
2.1: click Add Girl(routing.yml->gotrla_girls_add)
2.1.1: Enter Girl's Details
2.1.1.1: Add new Girl(GirlsController::girlsAddAction)
2.1.1.1.1: return Girl
3: click Girls Tab
3.1: click Import girls tab(routing.yml->gotrla_girls_import)
3.1.1: select file and click import
3.1.1.1: imported(GirlsController::girlsImportAction)
4: click Girl Tab
4.1: click Edit Girl(routing.yml->gotrla_girls_edit)
4.1.1: modify and click edit
4.1.1.1: update volunteer(GirlsController::girlsEditAction($id))
4.1.1.1.1: return success
4.1.2: cancel
4.1.2.1: display updated Girls
4.1.2.1.1: display girls
5: Click Girls Tab
5.1: delete Girls(routing.yml->gotrla_girls_delete)
5.1.1: delete Girls(GirlsController::girlsDeleteAction($id))
5.1.1.1: return updated Girls
5.1.1.1.1: display girls
5.1.1.1.1.1: display girls

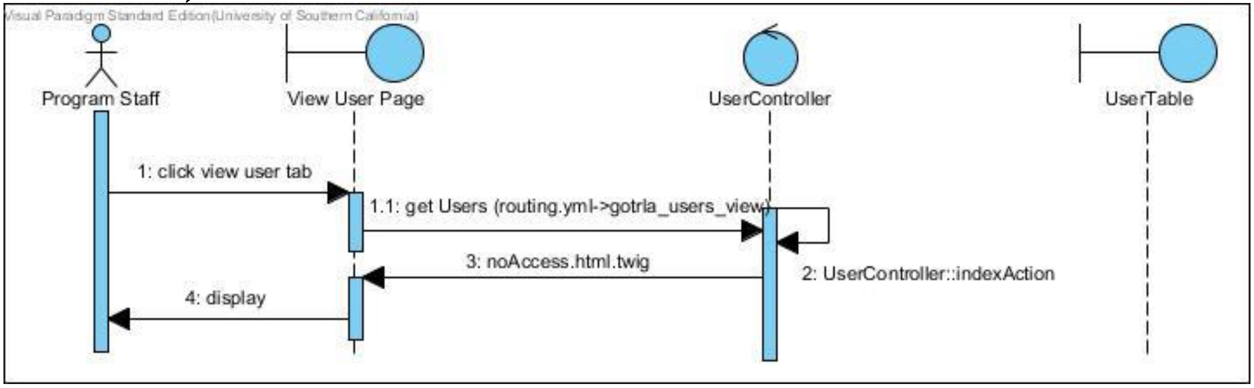
3.1.3.4 Scenario: Admin or Program Staff accessing Team tab on Webpage


Visual Paradigm Standard Edition(University of Southern California)
Admin/Program Staff
View Team
Add Team
Import Teams
Edit Teams
RosterController
Teams
1: click Team Tab
1.1: get Teams(routing.yml->gotrla_roster_view)
1.1.1: get Teams(RosterController::viewRosterAction)
1.1.1.1: return Teams
1.1.1.1.1: display Teams(RosterController::rosterAllDataJSONDisplay)
1.1.1.1.1.1: display Teams
2: Click Team Tab
2.1: click Add Team(routing.yml->gotrla_roster_add)
2.1.1: Enter Team Details
2.1.1.1: Add new Team(RosterController::addRosterAction)
2.1.1.1.1: return success
3: click Team Tab
3.1: click Import Teams tab(routing.yml->gotrla_roster_import)
3.1.1: select file and click import
3.1.1.1: imported(RosterController::importRosterAction)
4: click Team Tab
4.1: click Edit Team(routing.yml->gotrla_roster_edit)
4.1.1: modify and click edit
4.1.1.1: update team(RosterController::editRosterAction($id))
4.1.1.1.1: return success
4.1.2: cancel
4.1.2.1: display updated Teams
4.1.2.1.1: display Teams
5: Click Team Tab
5.1: delete Team(routing.yml->gotrla_roster_delete)
5.1.1: delete Team(RosterController::deleteRosterAction($id))
5.1.1.1: return updated Teams
5.1.1.1.1: display teams
5.1.1.1.1.1: display Teams

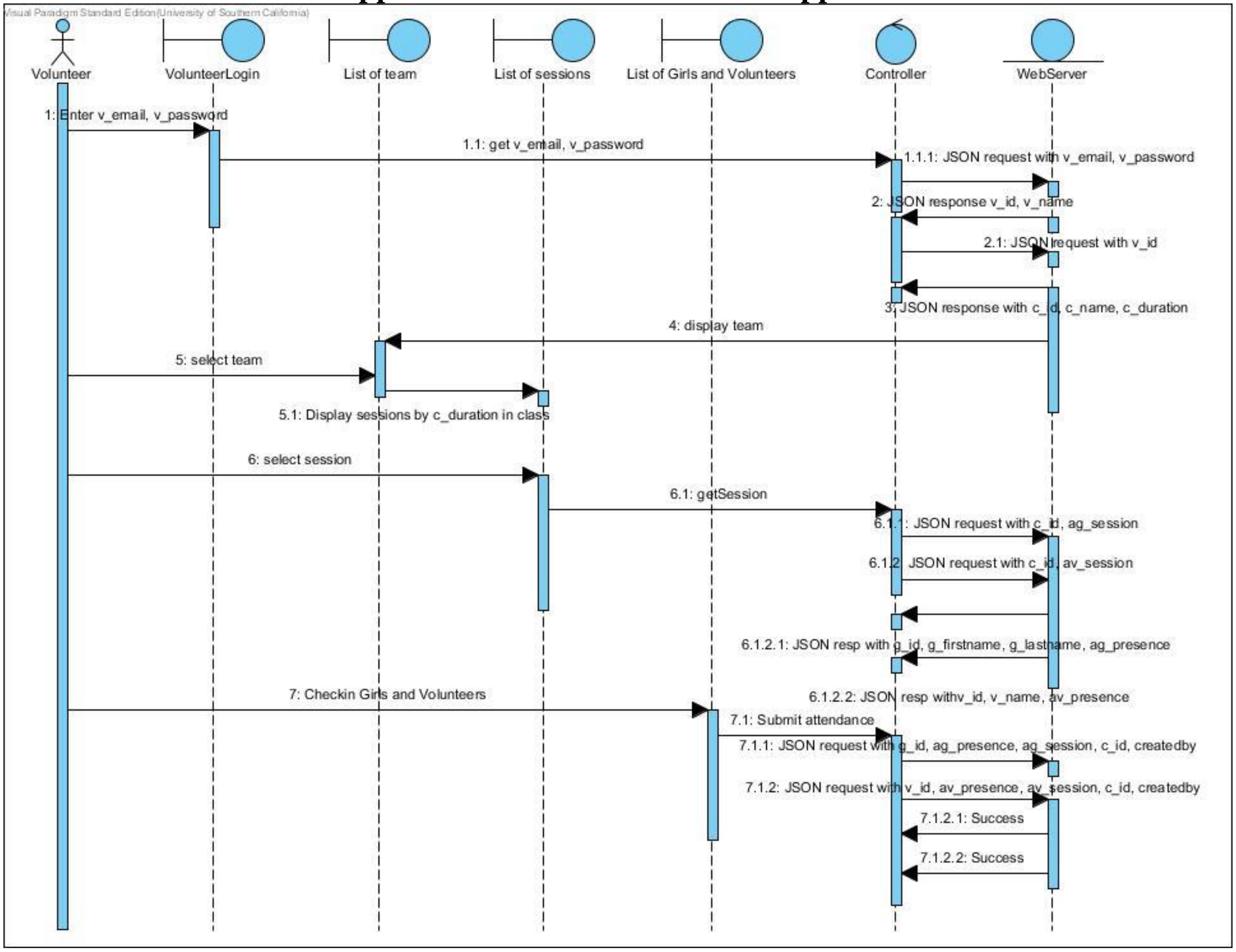
3.1.3.5 Scenario:Admin or Program staff accessing Attendance tab on Webpage


Visual Paradigm Standard Edition(University of Southern California)
Admin/Program Staff
Girls Attendance
Volunteer Attendance
Export Attendance
Attendance Controller
Attendance
1: click Attendance Tab
1.1: click Girls Attendance(routing.yml->gotrla_attendance_girls)
1.1.1: Attendance(AttendanceController::girlsAttendanceAction)
1.1.1.1: return Attendance
1.1.1.1.1: display Girls Attendance(AttendanceController::girlsAttendAllDataJSONDisplay)
1.1.1.1.1.1: display Attendance
2: click Attendance Tab
2.1: click Volunteer Attendance(routing.yml->gotrla_attendance_volunteer)
2.1.1: Enter Volunteer Attendance
2.1.1.1: get Attendance(AttendanceController::volunteerAttendanceAction)
2.1.1.1.1: return
2.1.1.1.1.1: display Attendance(AttendanceController::volunteerAttendAllDataJSONDisplay)
2.1.1.1.1.1.1: display attendance
3: click Attendance Tab
3.1: click Export attendance tab(routing.yml->gotrla_attendance_export)
3.1.1: select type of attendance to export
3.1.1.1: fetch attendance(AttendanceController::exportAttendanceAction($toExp))
3.1.1.1.1: return attendance
3.1.1.1.1.1: Export attendance
3.1.1.1.1.1.1: export attendance

3.1.3.6 Scenario: Rainy Day Case: When a volunteer (who does not have access right to view user tab) access the user tab


Visual Paradigm Standard Edition(University of Southern California)
Program Staff
View User Page
UserController
UserTable
1: click view user tab
1.1: get Users (routing.yml->gotrla_users_view)
2: UserController::indexAction
3: noAccess.html.twig
4: display

3.1.3.2 Scenario: Mobile App: Volunteer works on mobile app.


Visual Paradigm Standard Edition(University of Southern California)
Volunteer
VolunteerLogin
List of team
List of sessions
List of Girls and Volunteers
Controller
WebServer
1: Enter v_email, v_password
1.1: get v_email, v_password
1.1.1: JSON request with v_email, v_password
2: JSON response v_id, v_name
2.1: JSON request with v_id
3: JSON response with c_id, c_name, c_duration
4: display team
5: select team
5.1: Display sessions by c_duration in class
6: select session
6.1: getSession
6.1.1: JSON request with c_id, ag_session
6.1.2: JSON request with c_id, av_session
6.1.2.1: JSON resp with g_id, g_firstname, g_lastname, ag_presence
6.1.2.2: JSON resp withv_id, v_name, av_presence
7: Checkin Girls and Volunteers
7.1: Submit attendance
7.1.1: JSON request with g_id, ag_presence, ag_session, c_id, createdby
7.1.2: JSON request with v_id, av_presence, av_session, c_id, createdby
7.1.2.1: Success
7.1.2.2: Success

4. Architectural Styles, Patterns and Frameworks

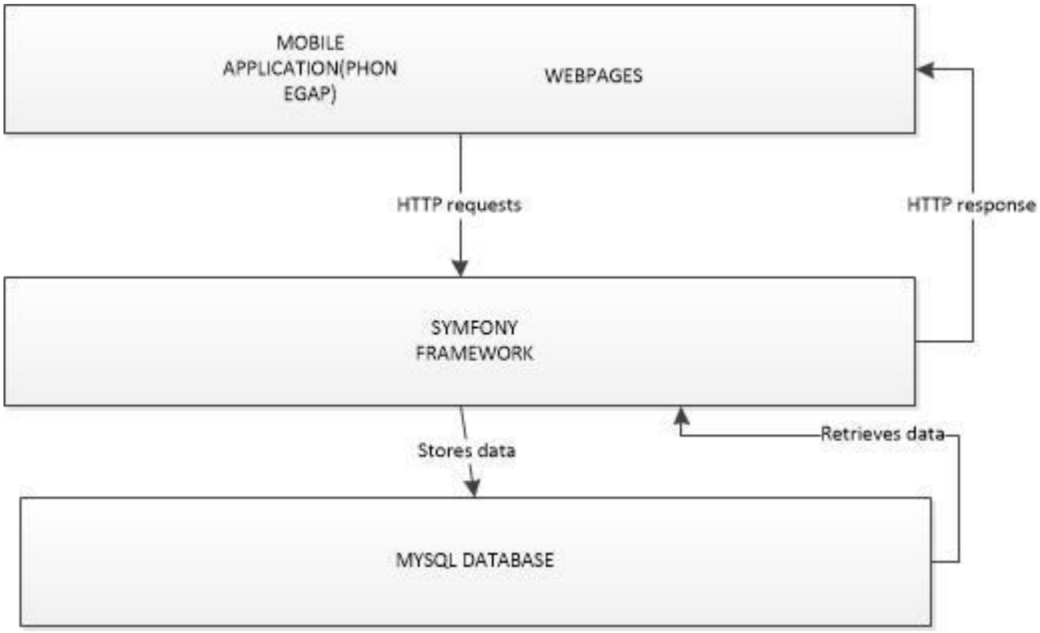


Figure 7: 3 Tier Architecture

Table 40: Architectural Styles, Patterns, and Frameworks

Name	Description	Benefits, Costs, and Limitations
3 Tier Architecture		Benefit: It provides modularity. Costs: Limitations:
Layer 1	Mobile Application and Website in Application Layer	
Layer 2	Symfony Framework builds the Business Logic	
Layer3	MySQL database	