# System and Software Architecture Description (SSAD)

**TOUR CONDUCTOR**

**TEAM - 05**

| Name | Role |
|---|---|
| **Ankush H Prasad** | **System Architect , Project Manager, Prototyper/Builder** |
| **Ajay Kumar G C** | **Project Manager, Life Cycle Planner Prototyper/Builder.** |
| **Aadithya B K** | **Requirements Engineer, Prototyper/Builder.** |
| **Andrew Han** | **IIV & V, Quality Focal Point, Prototyper/Builder.** |
| **Joseph Mouawad** | **Operations Concept Engineer, Prototyper/Builder.** |
| **Manas Yadav** | **Feasibility Analyst, Prototyper/Builder.** |
| **Rohith Ravindra** | **Life Cycle Planner, Prototyper/Builder.** |

**11/30/2015**

# Version History

| Date | Author | Version | Changes made | Rationale |
|---|---|---|---|---|
| 10/12/15 | Ankush | 0.1 | • Updated section 1 | • To identify the purpose and status of SSAD |
| 10/19/15 | Ankush | 1.0 | • Updated section 2 | • To identify system context, artifacts and information, behavior, system analysis rationale |
| 11/30/15 | Ankush | 2.0 | • Updated section 2,3,4,5 | • Technology independent model, Technology specific system model, Architectural frameworks, design & patterns |

# Table of Contents

# Table of Tables

# Table of Figures

# 1. Introduction

## 1.1 Purpose of the SSAD

This document provides a comprehensive architectural overview of the Tour Conductor system, using a number of different architectural views to depict different aspects of the system and also explains the different perspectives of the system in-terms of Tour taker, Owner and Tour Creator. It is intended to capture and convey the significant architectural decisions which have been made on the system.

## 1.2 Status of the SSAD

This document is the final version of SSAD where we are discussing in detail, the architectural overview of the system in depth. The areas that this document tries to explain in detail are the overview of the system, system context defining the boundary between system or part of the system and its environment, artifacts, behavior, modes of operation and systems analysis rationale. This also explains the design we opted for the project, why did we chose such a design, by referring to both Technology Independent and Technology specific design.

# 2. System Analysis

## 2.1 System Analysis Overview

The primary purpose of the Tour Conductor is to help tour takers to take tours of places of their current location or by searching for a specific location. The tours are previously uploaded to the system by Tour Creators, who access a website to upload the information. However, these Tour Creators need to be pre-authorized by the Owner of the system to access the system or upload the tours. The Tour Conductor application return a set of tours or stops based on the tour taker's search and the tour taker can decide to view a tour or a stop based on his/her interest.

### 2.1.1 System Context

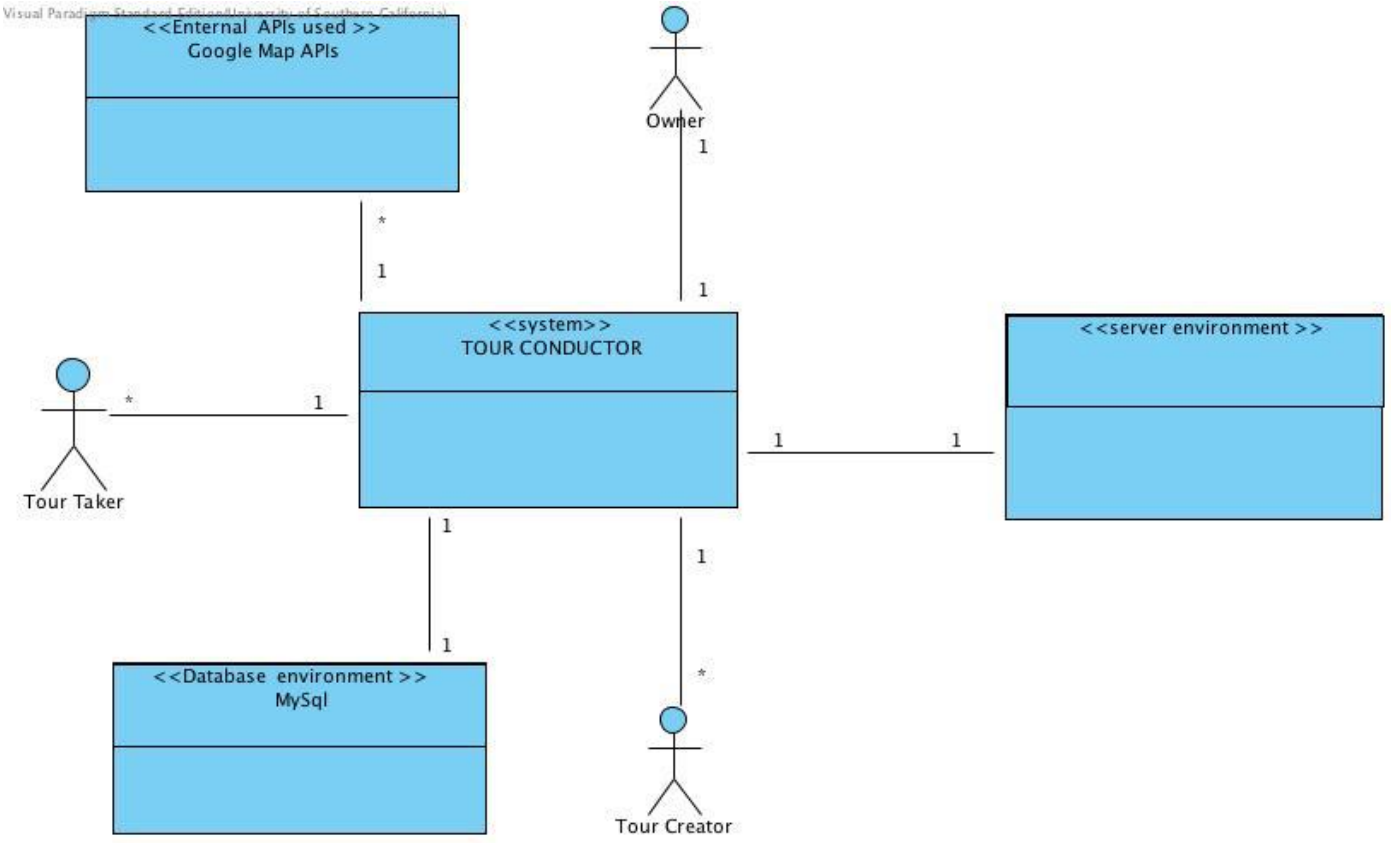



**Figure 1: System Context Diagram**

**Table 1: Actors Summary**

| Actor | Description | Responsibilities |
|---|---|---|
| Owner | Owner of the system | Authorize Tour Creator, Delete tours |
| Tour Creator | Feeds/Updates tours in system | Upload/Update tours, update stops |
| Tour Taker | End user who views tours in system | Search for a tour, select a tour based on current location/results of search, view tour |

## 2.1.2 Artifacts & Information

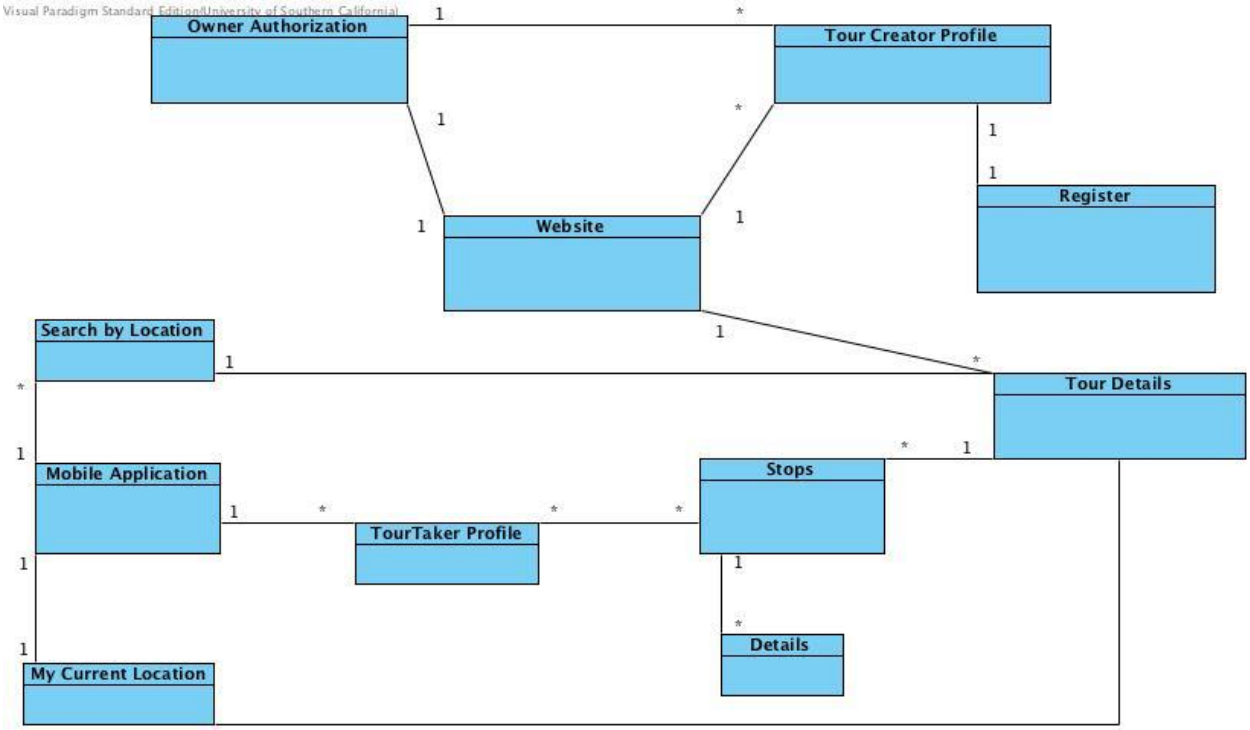



**Figure 2: Artifacts and Information Diagram**

**Table 2: Artifacts and Information Summary**

| Artifact | Purpose |
|---|---|
| Owner Authorization | To authorize Tour Creator in to the system |
| Tour Creator Profile | To identify the tour creator who uploads/updates the tours |
| Website | Accessed by tour creator and owner to upload/delete tours |
| Tour Details | To hold the tour to stop mappings and tour details |
| Stops | To hold stop ids, tours that stops are part of |
| Details | Holds details of stops like URL, description |
| Tour taker Profile | Users of the tour conductor mobile app who search/view tours |
| Mobile Application | The Android application with which tour taker takes tours |
| Search by location | Tour taker can use this artifact to view tours at remote location |

| | |
|---|---|
| My Current Location | This artifact helps tour takers to take tours based on their current location. |
| Register | Artifact indicating Tour Creator being registered into the system. |

### 2.1.3 Behavior

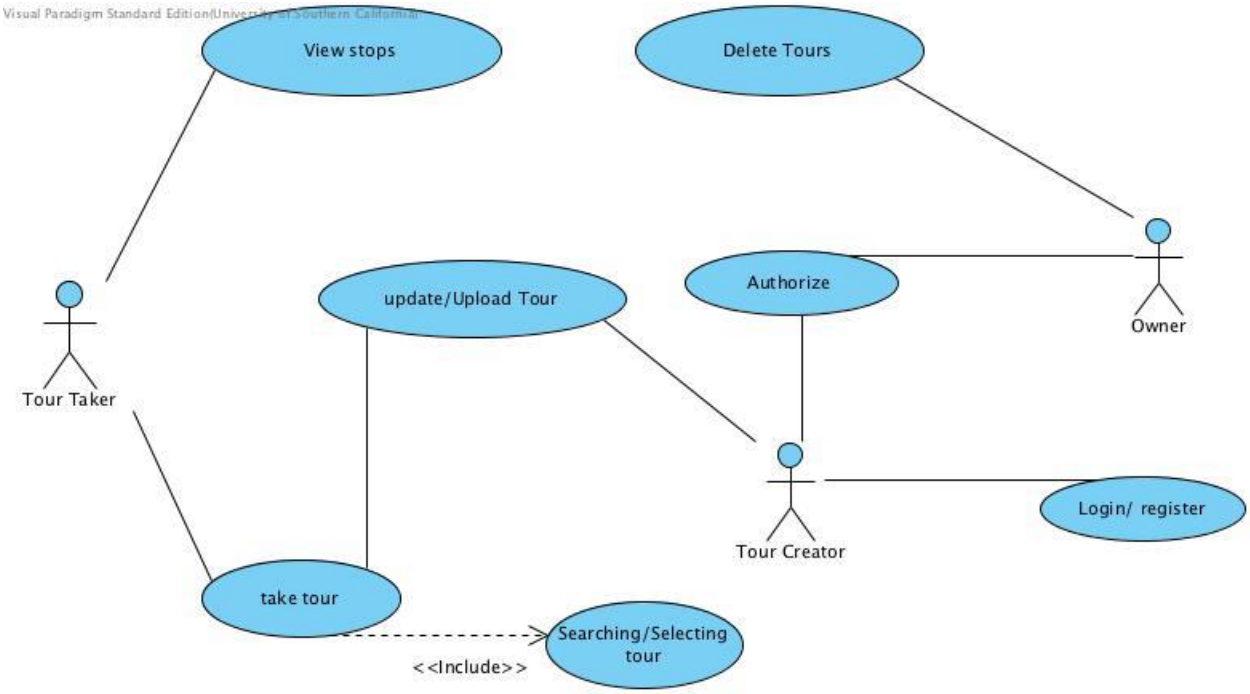



**Figure 3: Process Diagram**

#### 2.1.3.1 Capability x

##### 2.1.3.1.1 Process y

**Table 3a: Process Description**

| | |
|---|---|
| **Identifier** | Retrieval of tour details |
| **Purpose** | To obtain the tours based on search location or the current location by the tour taker |
| **Requirements** | Web server to process requests and database system that has tours uploaded by tour creator |
| **Development Risks** | Mapping efficiency of latitude and longitude information from google map APIs obtained from tour taker location with the tours |
| **Pre-conditions** | Relevant tours for the location are uploaded by tour taker and the |

| | |
|---|---|
| | database is always connected to tour taker system and is compatible |
| **Post-conditions** | Tour taker is shown a list of tours available for his location of search and he is given an option to view a tour or a stop |

**Table 3b: Typical Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Owner Authorizes a tour creator | Tour creator can access the system and upload/modify tours |
| **2** | Owner can delete irrelevant tours | The tours are deleted permanently on database |
| **3** | Tour Creator logs in or registers to the system | On successful login/registration, tour creator is granted access to the system to upload tours. |
| **4** | Tour Creator uploads tours | The tours are reflected on the database and the tour taker can view these tours if the locations match |
| **5** | Tour taker searches for a tour | A list of tours are returned based on the location match |
| **6** | Tour taker views a particular stop | Details on the stop such as URL, photos, description are shown. |

**Table 3c: Alternate Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | | |
| **2** | | |
| **3** | | If registration/login is unsuccessful, the tour creator is not granted access to the system and a notification is sent to the owner |
| **4** | | |
| **5** | | If exact location match is not found, the tour at nearby location is returned (the near metric is still under discussion) |
| **6** | | |

**Table 3d: Exceptional Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | | |
| **2** | | |
| **3** | | Redirect to login page after showing error |

| | | |
|---|---|---|
| **4** | | |
| **5** | | If no nearby tours (with in the decided distance) is not found, then the application would receive null results from the webserver and a message indicating unavailability of tours at the location is displayed. |
| **6** | | It is tour takers wish to choose to download the URL content based on the network bandwidth available. |

**Table 4a: Process Description**

| | |
|---|---|
| **Identifier** | Retrieval of stops details |
| **Purpose** | To obtain the stops based on the tour selected by the tour taker |
| **Requirements** | Tour has relevant stops in the database |
| **Development Risks** | Retrieval of stops JSON object, parsing and displaying it on the map as markers |
| **Pre-conditions** | Tour has stops which are uploaded by the Tour Creator |
| **Post-conditions** | Tour taker is shown stops with tis details, such as description, photo image and other details |

**Table 4b: Typical Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Tour taker clicks on a stop | Stop details such as description, photo are presented to the user |
| **2** | Tour taker clicks on a stop displayed | The stop details are displayed |

**Table 4c: Alternate Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | | If no stops are found an empty map is displayed |
| **2** | | If the details are not present then empty description and no photo is shown |

**Table 4d: Exceptional Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | | If internet is disconnected during the request, a null or empty object is |

| | | |
|---|---|---|
| | | initiated |
| **2** | | |

**Table 5a: Process Description**

| | |
|---|---|
| **Identifier** | Updating tours |
| **Purpose** | To add a new stop or delete stop |
| **Requirements** | Existence of a Tour with stops |
| **Development Risks** | Maintain database consistency on a delete or modification of a tour |
| **Pre-conditions** | The tour for which modification needed exists in the database |
| **Post-conditions** | The stop would be updated or deleted or added and reflected to the database. |

**Table 5b: Typical Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Tour Creator selects option to update tour | Update tour boundary page is displayed |
| **2** | Tour creator modifies tour details | The modified details are reflected to the database |

**Table 5c: Alternate Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | | |
| **2** | | If any mistake made while entering new details, like, missing few details, an error is thrown and Tour Creator is asked to reenter the details |

**Table 5d: Exceptional Course of Action**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | | |
| **2** | | If database connection is lost, then no updates are made and an error is thrown |

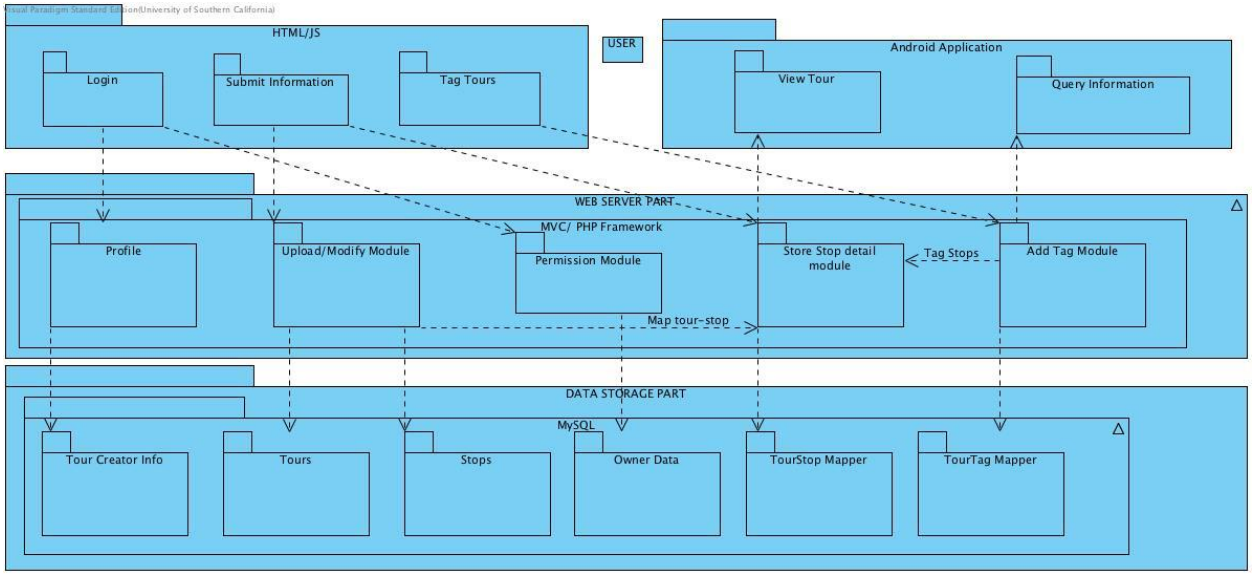
### 2.1.4 Modes of Operation

Tour Conductor System has two modes of operation:

1. Tour Conductor Website mode: This mode is used by Owner and the tour creator. Owner uses this mode to authorize tour creators and delete any irrelevant tours. Tour creator uses this mode to upload/modify any tours by logging into the system
2. Tour Conductor application: Tour takers use this mode to take the tours by searching for a location or using their current location to fetch the tours. Tour takers can then select a particular tour and view it.

## 2.2 System Analysis Rationale

Analyzing our proposed architecture below:




Referring to the above architecture from bottom to top;

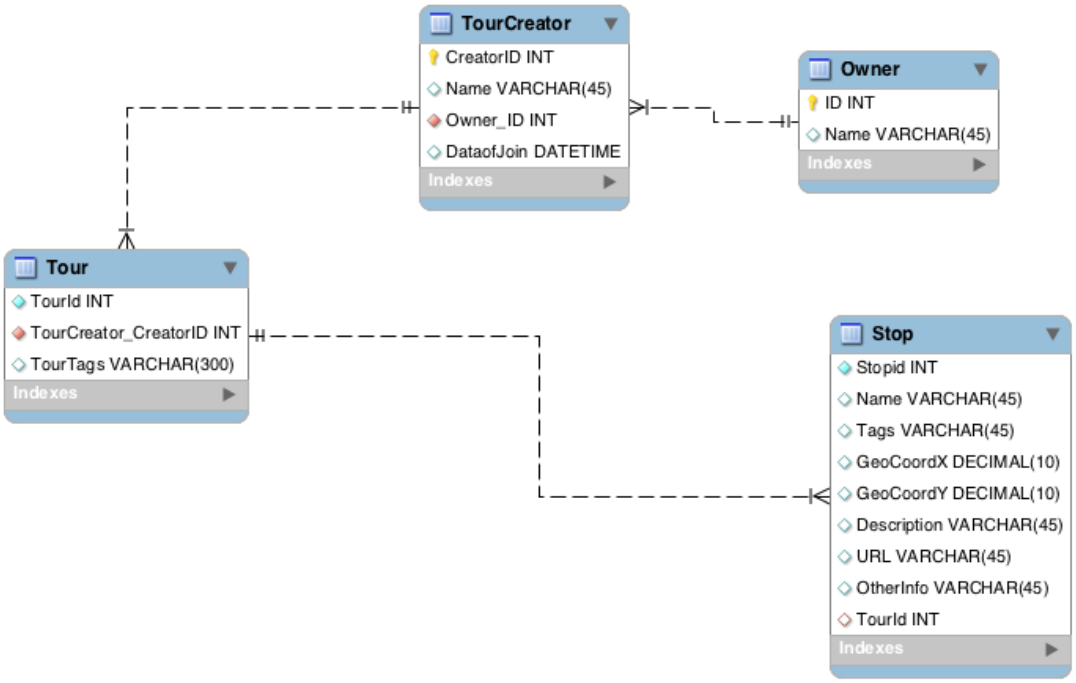
1. Data/Storage part
   This provides the necessary SQL relational database infrastructure for storing details about tour creator (including whether he is registered to the system by a field that denotes whether he is authorized or not), tours (details of the tours and its tags), Stops (details of stops, stop id, name and other information), Owner data( the owner who authorizes a tour creator to the system), Tour Stop mapper (a table that tags tours and the related stops), Tour tag mapper (to update tags to the tours, to match user search string with tours)

2. Web Server part
   We are using Model, View, Controller framework for designing the web backend. It offers profile module and permission module (to validate a tour creator and authorize tour

creator), Upload/Modify module (to upload/modify the tours by the tour creator), Store stop detail module (to store the stops and its details as description, URL etc. ) and Tag module to tag stops and tours.

3. User part
   This can be divided into two packages: Website (Tour creator – HTML/JS) and Android (Tour taker – Version ICS and later).
   Tour Creator uses login module to login to website, with submit information module – he submits the details of stops, tours and updates tours. (Arrows are depicted outward as tour taker enters information to the web server)
   Tour taker uses query module to retrieve information or a set of tours using a search string or current location and views a particular tour using the view tour module. (Hence, the arrows are inward, indicating data is flowing into view tour and query modules)

Our proposed database structure:




Owner table maintains the owner of the system who authorizes the tour creator into the system.
Tour Creator table has tour creator details on how he signs into the system (email and password) and a field Authorized indicates whether the tour creator is authorized by the owner of the system or not.
Tour table has tour ids, the id of its creator and the tags that match the tour with the search string of tour taker.
Stop table has details of stops like name, tags, location coordinates, urls, description and other details. Each stop entry has TourId which connects Tour and Stops.

# 3. Technology-Independent Model

## 3.1 Design Overview

### 3.1.1 System Structure

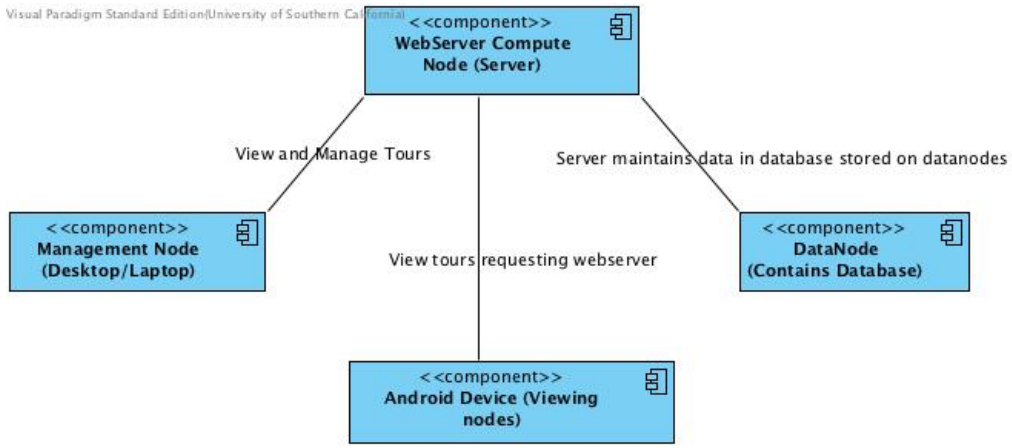



**Figure 4: Hardware Component Class Diagram**

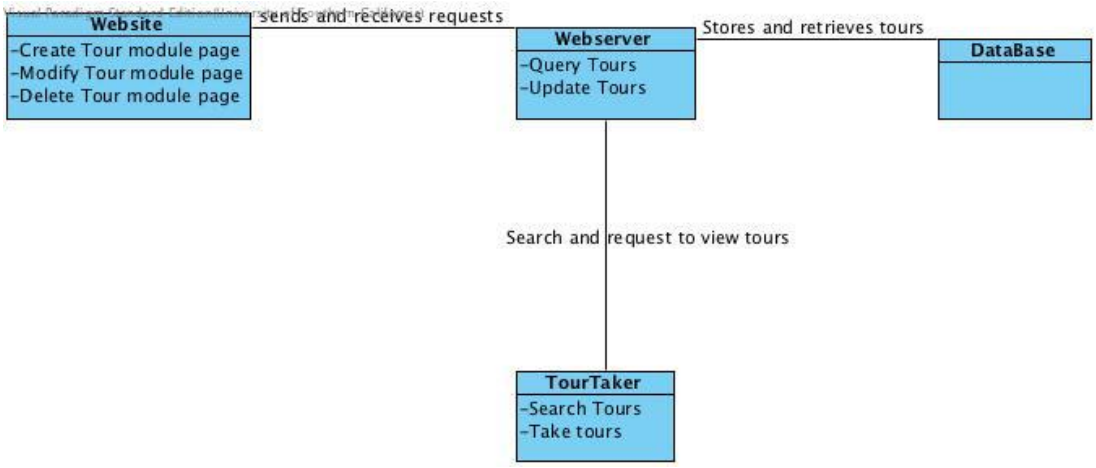



**Figure 5: Software Component Class Diagram**

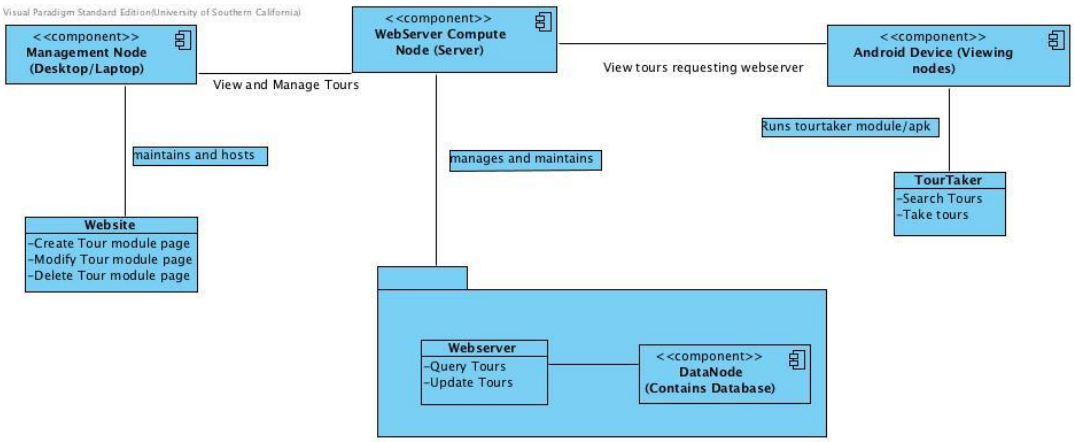



**Figure 6: Deployment Diagram**

**Table 6: Hardware Component Description**

| Hardware Component | Description |
|---|---|
| Management node | Management nodes are the computers/laptops that the Tour Creator uses to access the Tour Conductor Website and manage the entire system(adding tours/updating tours or even delete tours). They decide on what Tour Takers can see when they search for a tour. Management nodes are connected to webserver which processes all the requests. |
| WebServer Compute node | Compute nodes are computers/servers that run the webserver which listens to users requests and processes them. If the request is from a website for updating/deleting/create tours, the requests are processed and status is sent back. If the request is from an android application(from Tour Taker), the respective tours are searched and the resulting contents are send back to the requestor. |
| Data Node | Data Nodes are centralized computers/storage that hosts the database required to serve the requests. |
| Android device (View nodes) | Any android device used by the Tour taker to view the tour taker |

**Table 7: Software Component Description**

| Software Component | Description |
|---|---|
| Website | Website software component class holds the Display logic code. It mainly concerns on how users can view the website and the features provided by the website in managing the tours for the system |
| Webserver | This software component class is used to listen to the requests from the |

| | |
|---|---|
| | website and the mobile application and process them and provide relevant results. |
| DataBase | This software component generated queries specific to the operations requested (add/modify/delete tours) and sends those queries as transaction to the database and reflects the changes on the database. |
| TourTaker | This software component is the one that sends a request to the webserver demanding the tours near-by the location specified or tours based on search string. |

## 3.1.2 Design Classes

### 3.1.2.1 <Classes n>

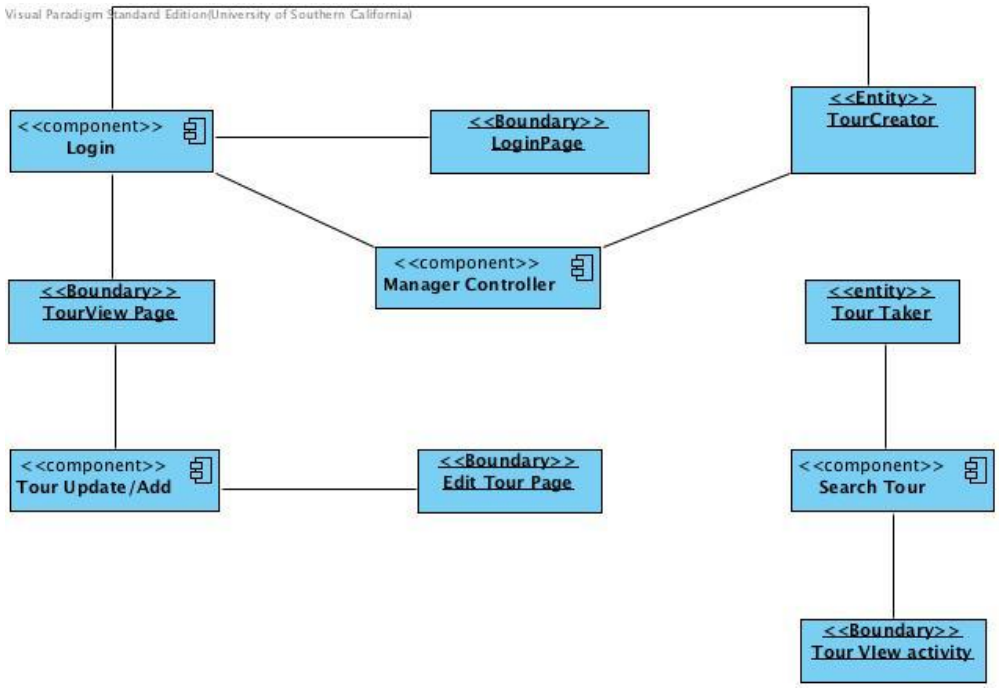



**Figure 7: Design Class Diagram**

**Table 8: Design Class Description**

| Class | Type | Description |
|---|---|---|
| Login Page | Boundary | The Tour Creators use this page to provide their credentials to login to the system. |
| Login | Controlling component | This component accepts Tour Creators' credentials for accessing the website and validates the user. If the validation fails, the Tour Creator is redirected back to this module. If the validation is successful, the management controlling module is called, with which the Tour Creator is redirected to a page where he can view or update the tours. |
| Tour View page | Boundary | The Tour Creators use this page to view the tours they have created. |
| Tour Update/Add | Controlling component | This component takes in the login Id of the tour creator and redirects him/her to a page that allows him/her to add or update a tour. |
| Edit Tour Page | Boundary | The Tour creators use this page to edit/update/add the tours they have created or want to create. |
| Search Tour | Controlling component | This component is used to search the tours either by accepting tour taker's current location or the search string. |
| Tour View Activity | Boundary | This activity displays the tours as requested by the Tour taker and also redirects to google maps activity(COTS) if Tour taker wants detailed instructions on how to reach from one stop to other. |
| Tour Creator | Entity | This would store the tour creator's id which would be appended to all the tours the tour creator creates/modifies and to track the tour creator activities |
| Tour Taker | Entity | This would accept the requests from the user on whether the user wants to search tours based on search string or location. |

### 3.1.3 Process Realization

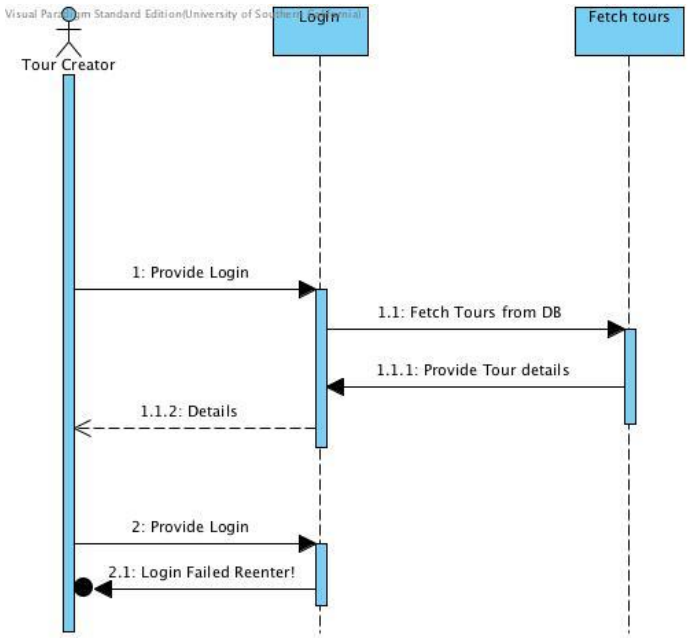



**Figure 8a: Process Realization Diagram**

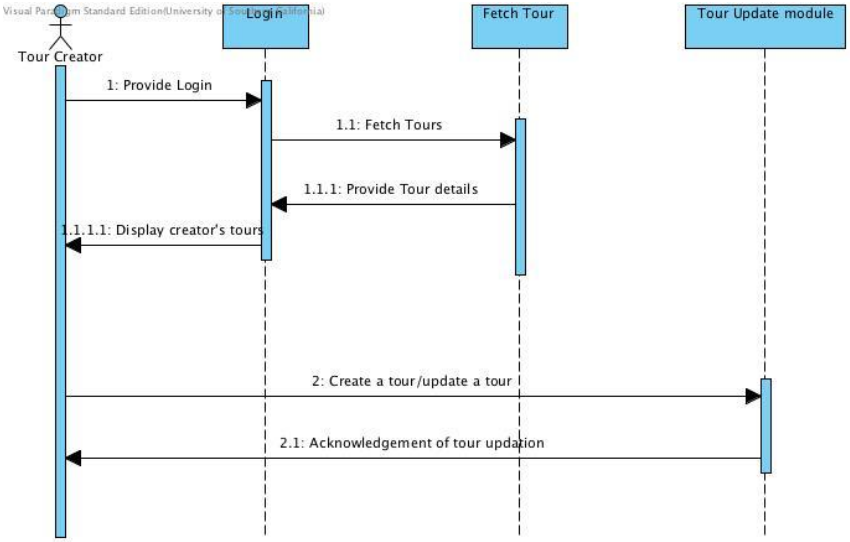



**Figure 8b: Process Realization Diagram**

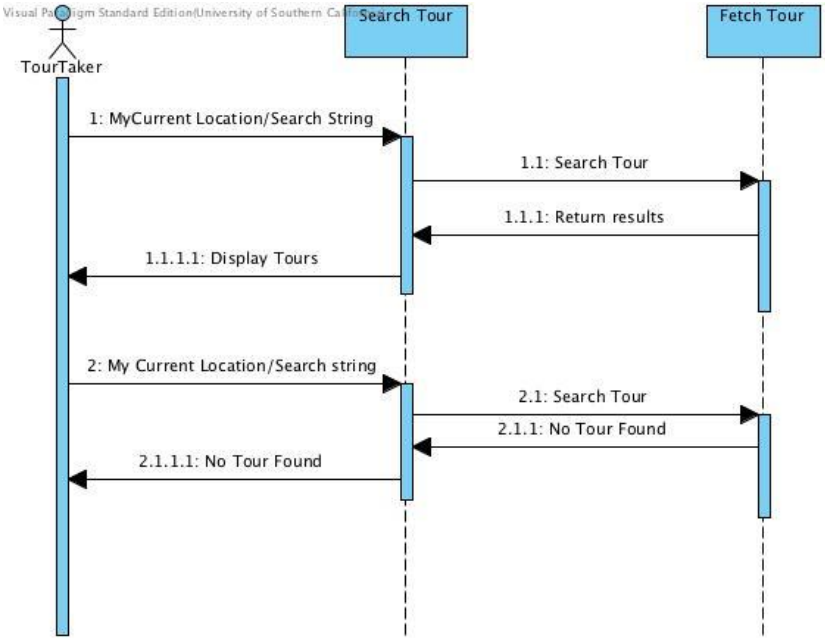



**Figure 8c: Process Realization Diagram**

# 3.2 Design Rationale

From the specification of the project, we needed a centralized server that listens to both the website requests (from where tours are uploaded/deleted) and the mobile application requests (from where the requests are sent for searching and viewing tours).
Hence, we designed a product with a centralized server having an attached storage, such as the database. The tour conductor website would manage the server and the data stored in the server. And hence, we implemented a simple authentication mechanism to gain access to the website core modules, via a login authentication controller.
The Tour taker will use a device to view the tours. Hence, we had to design a application (android application – as discussed with the client) that would request the server with the parameters; either the location coordinates or the search string for the tours and later, the results are displayed as received from the webserver.

# 4. Technology-Specific System Design

## 4.1 Design Overview

### 4.1.1 System Structure

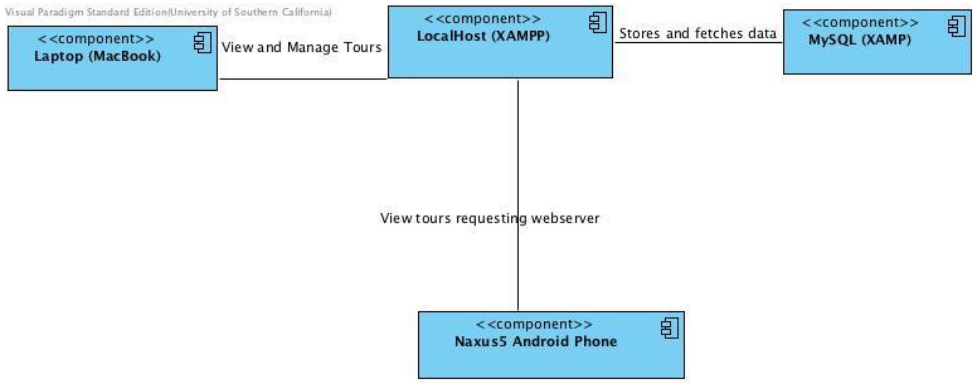



**Figure 9: Hardware Component Class Diagram**

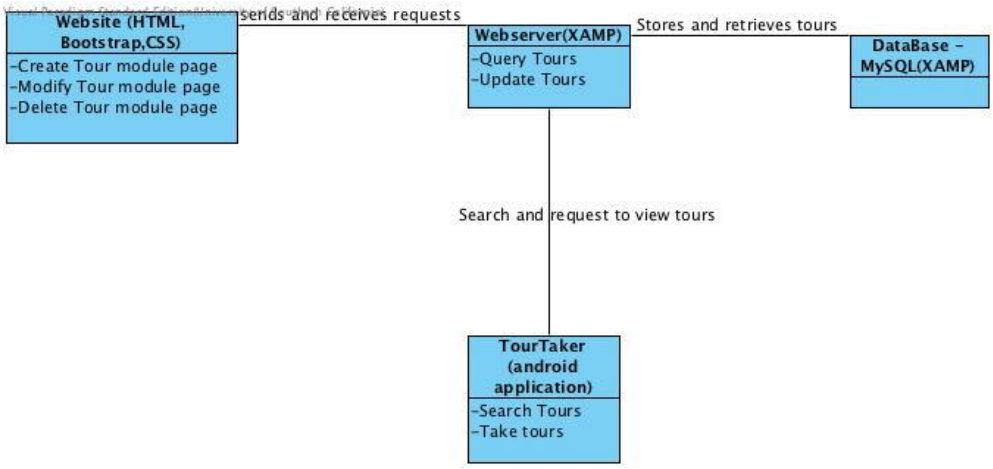



**Figure 10: Software Component Class Diagram**

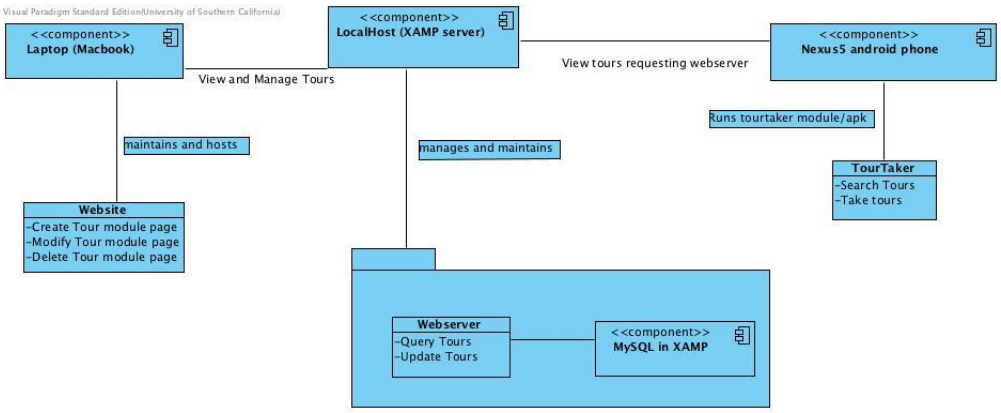



**Figure 11: Deployment Diagram**

**Table 9: Hardware Component Description**

| Hardware Component | Description |
|---|---|
| LocalHost (XAMP server) | This would be out webserver compute node. LocalHost (or a laptop running XAMP server) is the one that listens to the requests made from Tour Conductor Website and Tour Conductor mobile application |
| Any Laptop/desktop | This is our management node. Any laptop can connect to the hosted website and manage the tours and our webserver. If the webserver is on localhost, then the only condition is that the system hosting the server and the requestor needs to be in same sub network |
| MySQL (XAMP) | As we are using XAMP server, MySQL database is provided by XAMP and we are using the same database to store the contents |
| Nexus5 Android phone | This is our viewing node or the device the tour taker uses to search and view the tours. Any device which can support android applications (where an android application could be installed and executed successfully) can be used to install the tour conductor mobile application |

**Table 10: Software Component Description**

| Software Component | Description |
|---|---|
| Website (HTML, CSS, Javascript) | The tour conductor website component is built using HTML, CSS and javascript modules along with integrated google map API calls. |
| WebServer(XAMP) | The software component of webserver involves php scripts that listen to the requests, processes them and then returns the results |

| | |
|---|---|
| | as a JSON object. |
| MySQL (XAMP) | This software component holds javascripts and php scripts that generate and process Data Manipulation Language statements like insert, delete, alter and select queries to modify and fetch data. |
| TourTaker(android application) | Our application supports android versions from APK 16 and above (Icecream sandwitch version and above). So, any android device with these android versions should be able to install and execute our application successfully. |

## 4.1.2 Design Classes

### 4.1.2.1 <Classes n>

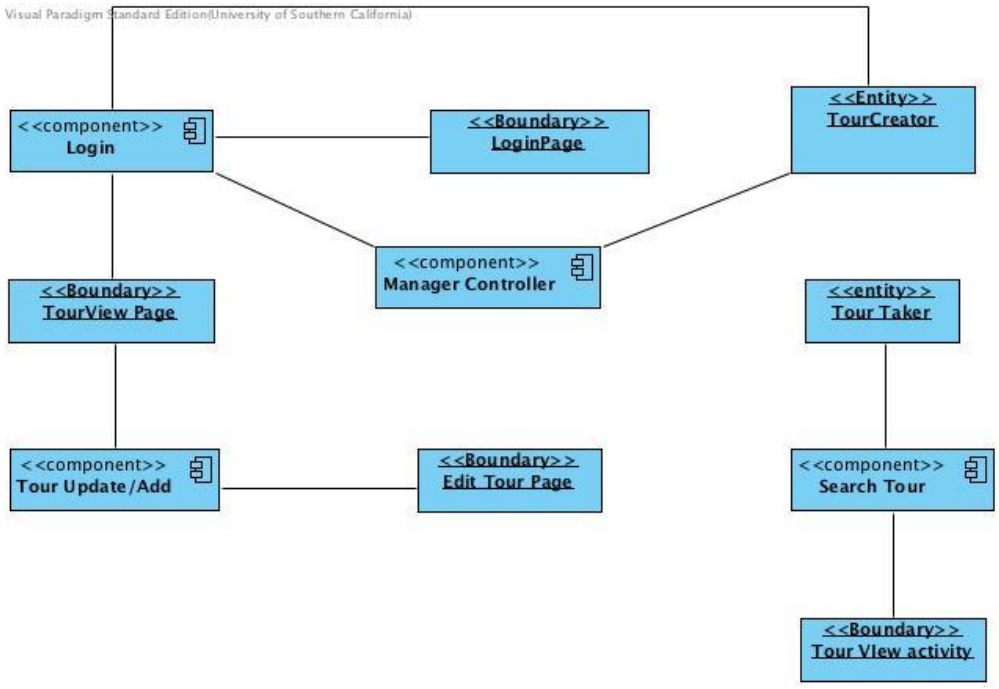



**Figure 12: Design Class Diagram**

**Table 11: Design Class Description**

| Class | Type | Description |
|---|---|---|
| Login Page | Boundary | The Tour Creators use this page to provide their credentials to login to the system. |
| Login | Controlling component | This component accepts Tour Creators' credentials for accessing the website and validates the user. If the validation fails, the Tour Creator is redirected back to this module. If the validation is successful, the management controlling module is called, with which the Tour Creator is redirected to a page where he can view or update the tours. |
| Tour View page | Boundary | The Tour Creators use this page to view the tours they have created. |
| Tour Update/Add | Controlling component | This component takes in the login Id of the tour creator and redirects him/her to a page that allows him/her to add or update a tour. |
| Edit Tour Page | Boundary | The Tour creators use this page to edit/update/add the tours they have created or want to create. |
| Search Tour | Controlling component | This component is used to search the tours either by accepting tour taker's current location or the search string. |
| Tour View Activity | Boundary | This activity displays the tours as requested by the Tour taker and also redirects to google maps activity(COTS) if Tour taker wants detailed instructions on how to reach from one stop to other. |
| Tour Creator | Entity | This would store the tour creator's id which would be appended to all the tours the tour creator creates/modifies and to track the tour creator activities |
| Tour Taker | Entity | This would accept the requests from the user on whether the user wants to search tours based on search string or location. |

### 4.1.3 Process Realization

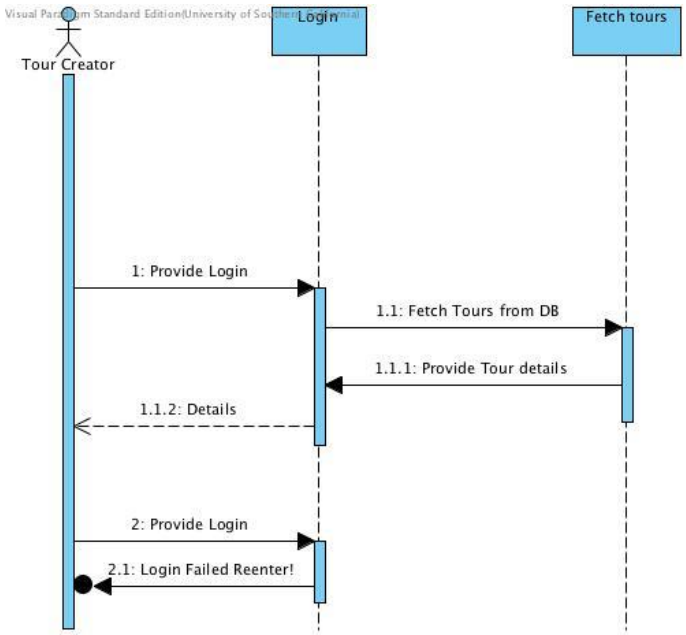



**Figure 13a: Process Realization Diagram**

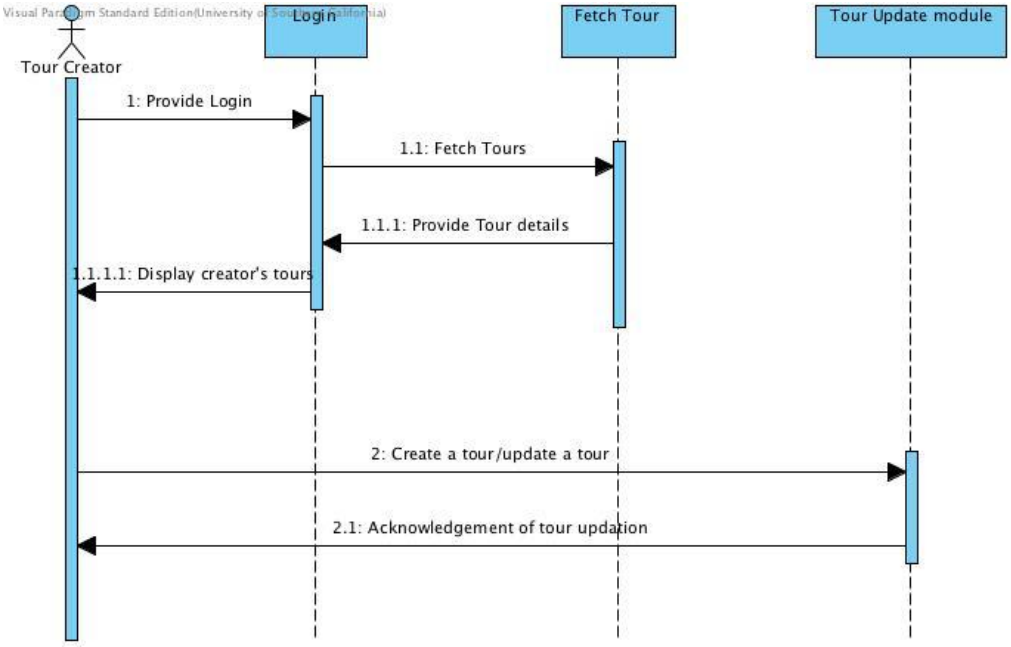



**Figure 13b: Process Realization Diagram**

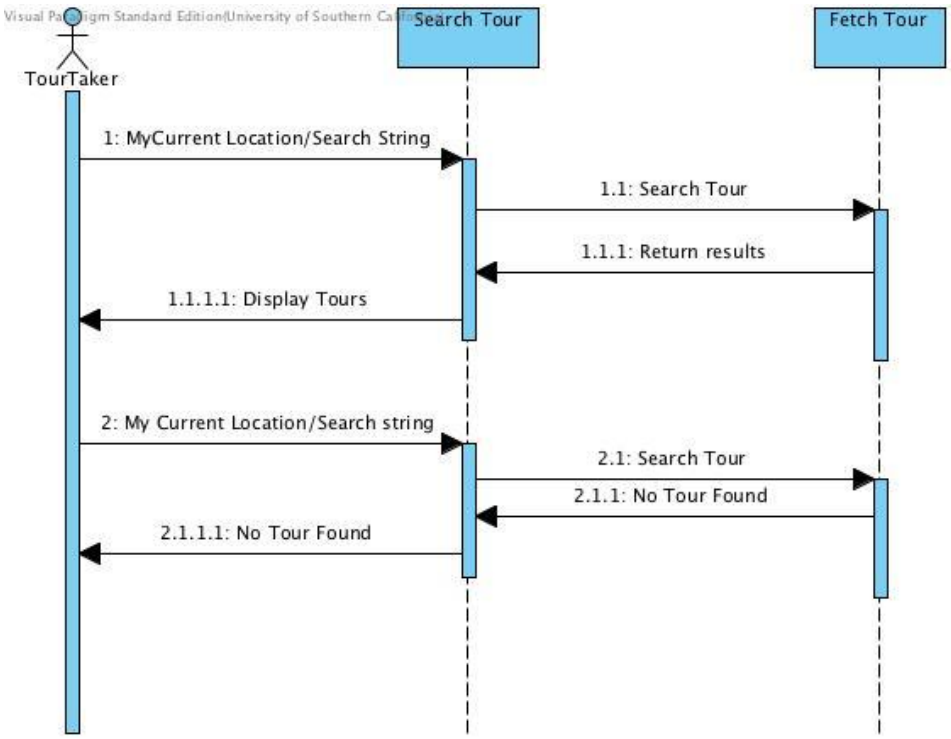



**Figure 13c: Process Realization Diagram**

# 4.2 Design Rationale

As we discussed earlier, we had to maintain a centralized server with a storage attached to it to host the database. As our client and the college was not ready to fund for a server on cloud services such as Amazon, Google App Engine, IBM Blue cloud or Microsoft Azure. Hence, the option we chose to host the server modules was on localhost (our own workstations/computers) using a software – XAMP server. With XAMP installed and started, our laptops can act as web servers and start listening to the requests made to it. XAMP in turn hosts a MySQL database, and hence, we used the same database, after discussing with the client, as this would eliminate the additional configuration details required to tweak any external database imparted to XAMP server. Another advantage was that the database was also hosted in the same system. Starting XAMP server, would start webservers and initializes the database connections at once.
We have set the minimum android operating system version for APK 16, since the android phones in market these days have much higher version of operating system. The tour taker will use the android phone to install the tour conductor mobile application, and view the tours by using a search string and if the tour taker's phone has GPS enabled, tours can be searched via current location too and display's the tours received as a JSON object.

# 5. Architectural Styles, Patterns and Frameworks

**Table 12: Architectural Styles, Patterns, and Frameworks**

| Name | Description | Benefits, Costs, and Limitations |
|---|---|---|
| Client-server | We chose client server model as we have a Tour conductor mobile application that sends the request and there would be APIs running in the server that listen to these requests, processes them and sends the results back to the application. Hence, client server style suits our application development. | Easy to design the project based on this architectural style as the requirements suited the style. Not much modification were needed and hence less cost. The only limitation is relying on the communication protocols between client and server. The server and client assumes that the communication protocol delivers messages as intended |
| Façade pattern | We chose Façade pattern because Tour conductor application uses many disjoint interfaces being it google map APIs, material designs interfaces and hence, this pattern would suit our development | Benefits were we were able to easily bind different core modules without much efforts. This also reduced a great deal of cost since we need not redundantly work on similar components. The only limitation is too much relying on the parser scripts that read information from these different modules. If one of the core module change, then the parser scripts need to be changed as well. |
| Software architectural pattern: MVC | We chose model-view-controller, since we had to isolate our database, the UI views and business logic from each other. Thus, changing one would not affect other | As we mentioned previously, it was easy to update/modify the components once we had decoupled them. |