Software Design Document

For

Storyboarding on the Move

Prepared by:
NC Hareerah Rafaqat
NC Zainab Akhtar
NC Zaineb Niaz

Approved by:

____
Project Supervisor: Assistant Professor Bilal Rauf

____
Project Co-Supervisor: Lecturer Mobeena

Military College of Signals (MCS), National University of Science and Technology (NUST)

February 12, 2014

1. Introduction

1.1. Purpose

The purpose of the project is to develop an android based mobile application along with a website based on similar context that focuses on maintaining a storyboard of user's place to place, time to time activities. This paper typically documents all basic features of the android application as well as of the website.

Main focus of this application is to provide the user with an attractive interface that offers a single platform to organize typical gallery items (pictures, videos, audio recording), personal notes, location alarms and to do list. All these features are maintained on the website as well, along with consistent updating.

1.2. Project Background

The product will be an application of the developed storyboarding system. It will maintain user's time to time travel log by uploading the data on the website in form of a storyboard. It will be a new system focusing on facilitating the user to save his pictures, videos, personal notes, and other gallery features on the move, along with the coordinates from map of his current location using Wi-Fi to a secure server. If Wi-Fi is not available at some location then this application will buffer all the information and will upload this data after connecting to a Wi-Fi automatically. The product will be a new one of its kind.

Software system will operate on Android based mobile devices and a web server. The product will be hosted by a web server since half part of the product is web-based.

The application will be developed basically for mobile devices with limited memory and processing power. We have also considered the uploading time and slow speed of internet. For an effective Storyboarding application and Website, we need to limit the sizes of videos, audio and pictures etc. (gallery items) initially. For now we have limited the size of the pictures, videos or audio recordings to be not more than 10MB each. This way, we can manage to upload the content easily from the android device to the website.

In addition, due to limited time of the project, some of the features that have been kept optional include providing an online platform to the users to comment and discuss about their updates and the places they have already been to i.e. updating statuses and check-ins and creating a friend group such that only that group can view the shown content.

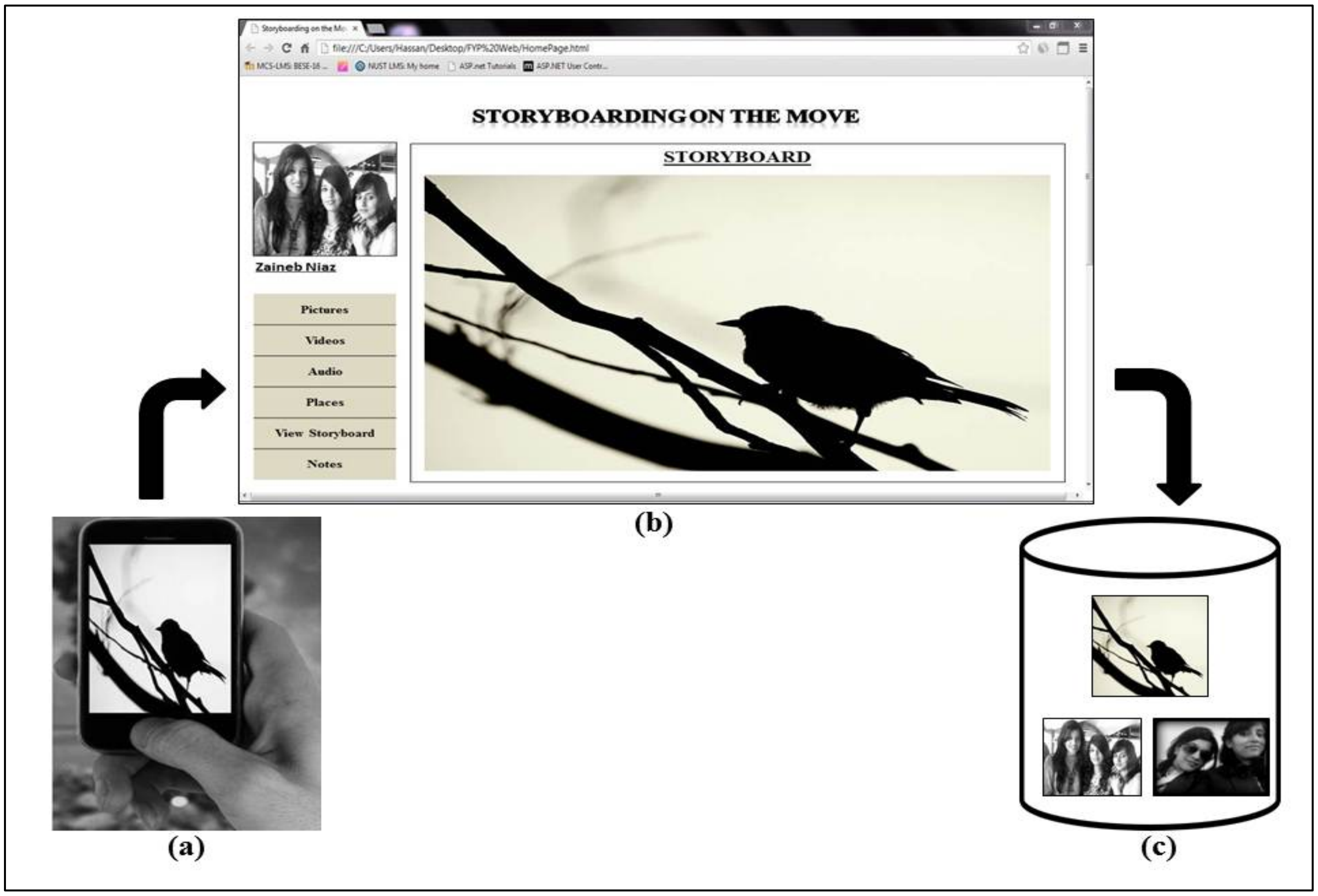


Figure 1: Storyboarding On the Move. (a) Android User taking a picture (b) Content uploaded on Website (c) Content saved in Database

1.3. Scope

Purpose of the system is to develop an Application for Smartphones and a Web Application that maintains Users Timeline automatically. Content can only be uploaded from mobile phone and not the website. The tasks include uploading of pictures, videos, audio recordings, personal notes, status and location updates to the web server through web services (automatically when Wi-Fi is available otherwise kept in buffer).
Optionally, the project might involve restricting the content to specific group of people added in the user's contact list.

1.4. Reference Material

1.4.1. Project Proposal

1.4.2. Project Defense Report

1.4.3. Literature Review

1.4.4. Software Requirements and Specification Document

1.5. Work Breakdown Structure

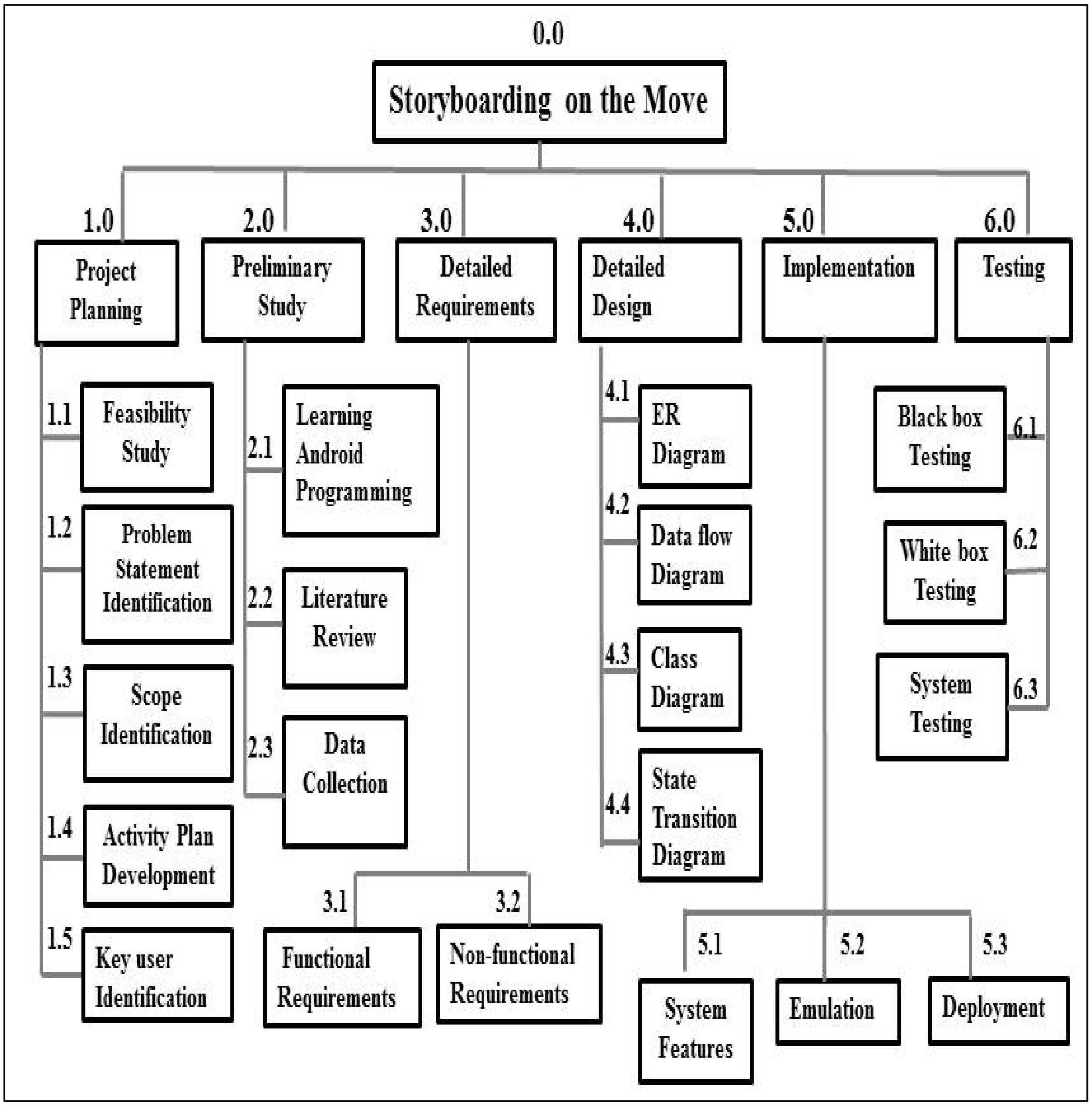


Figure 2: Work Breakdown Structure

2. Design Considerations

2.1. Assumptions and Dependencies

The project is based on the following assumptions:

1. An Android based mobile will be available for deployment of application.
2. The application would be developed using Android's OS architectural model.
3. The Android version on which the application will be created is Android 4.1 Jelly Bean.
4. Application would run on required mobile device without integration of any extra hardware.
5. The mobile device that will be used must have a Camera, GPS and WIFI hardware properly functional.
6. We may use some third party components, COTS, to specifically achieve functional or non-functional requirements, as is required.
7. We may require some initial training (i.e. workshop to familiarize us with the technology).
8. As now the project scope/core functionalities to achieve have been finalized and approved by the concerned committees, scope may not be further enhanced at any stage during the project construction.

2.2. General Constraints

The application will be developed basically for mobile devices with limited memory and processing power. We have also considered the uploading time and slow speed of internet. For an effective Storyboarding on the Move, we need to limit the sizes of videos, audio and pictures etc. (gallery items) initially. For now we have limited the size of the pictures, videos or audio recordings to be not more than 10MB each. This way, we can manage to upload the content easily from the android device to the website.

In addition, due to limited time of the project, some of the features that have been kept optional include providing an online platform to the users to comment and discuss about their updates and the places they have already been to i.e. updating statuses and check-ins and creating a friend group such that only that group can view the shown content.

2.3. Development Methods

The approach used for this software design is **V-Model method**. V-model means Verification and Validation model. The V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development. In figure 3, V-model has been shown that shows how at each phase testing has to be carried out.

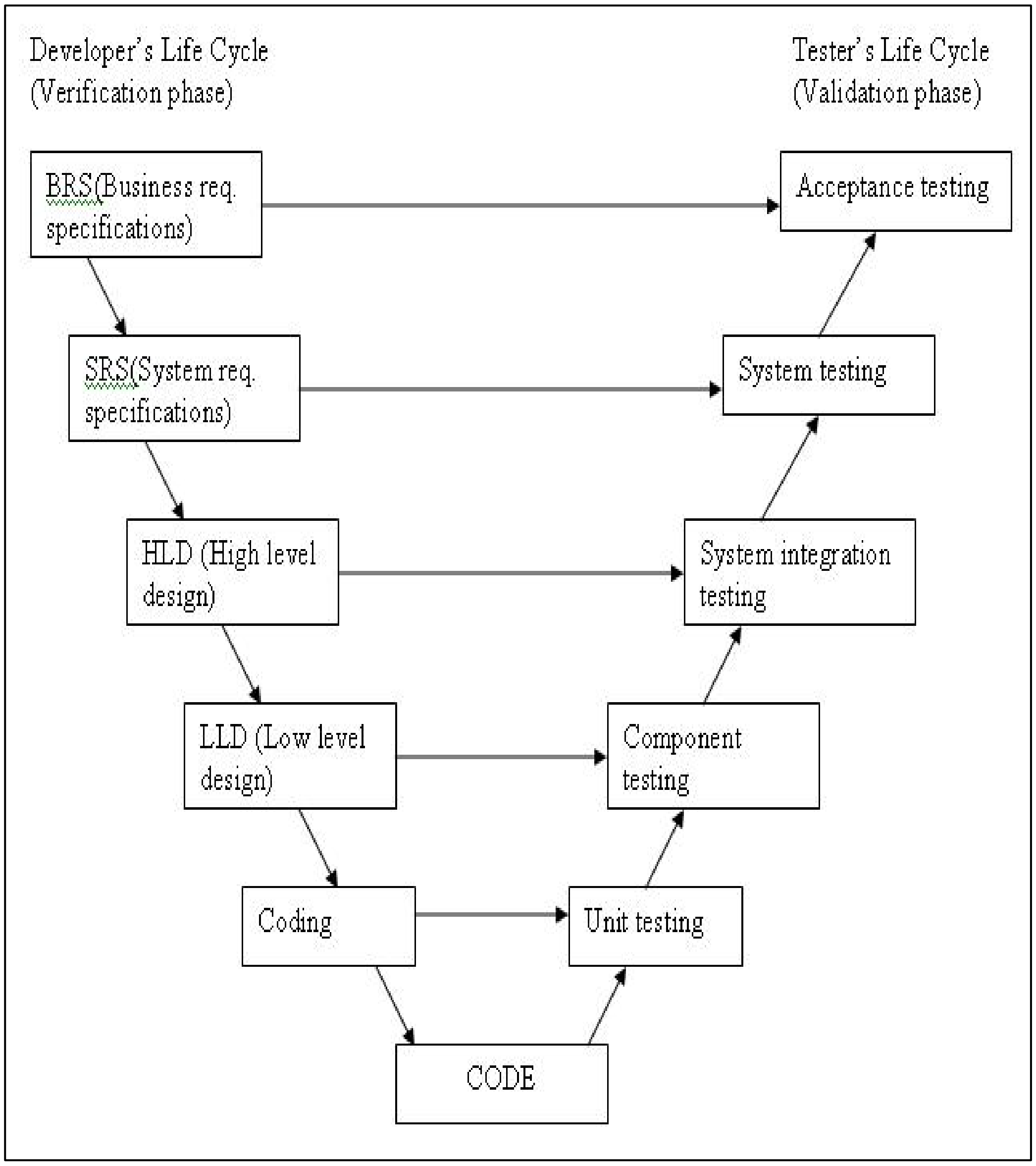

Developer's Life Cycle (Verification phase)
Tester's Life Cycle (Validation phase)
BRS(Business req. specifications)
Acceptance testing
SRS(System req. specifications)
System testing
HLD (High level design)
System integration testing
LLD (Low level design)
Component testing
Coding
Unit testing
CODE

Figure 3: V-Model

Requirements like BRS and SRS begin the life cycle model before development is started; a **system test plan** is created. The test plan focuses on meeting the functionality specified in the requirements gathering.

The high-level design (HLD) phase focuses on system architecture and design. It provides overview of solution, platform, system, product and service/process. **An integration test plan** is created in this phase as well in order to test the pieces of the software systems ability to work together.

The low-level design (LLD) phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. **Component tests** are created in this phase as well.

The implementation phase is, again, where all coding takes place. Once **coding** is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

Coding is at the bottom of the V-Shape model. Module design is converted into code by developers.

The **reason of selecting V-Model** as a development method for this project is because at each step testing was required to verify the perfection of the product. If at any step an error occurred and was ignored i.e. was not tested then it causes a bigger error in later stage. Also, because this model is Simple and easy to use and testing activities like planning, test designing etc. happens well before coding, which saves a lot of time, hence higher chance of success over the waterfall model. V-Model supports Proactive defect tracking i.e. defects are found at early stage. It avoids the downward flow of the defects and works well for small projects where requirements are easily understood.

3. Architectural Design

3.1.System Block Diagram

Block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. It is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

In Figure 3, system block diagram of Storyboarding on the Move clearly shows the relationship between the User, Android Device, Website, Server and the Database. The user can interact with the Android Device and Website. The android device uploads data to the server and the server then sends the data to the website and the database.

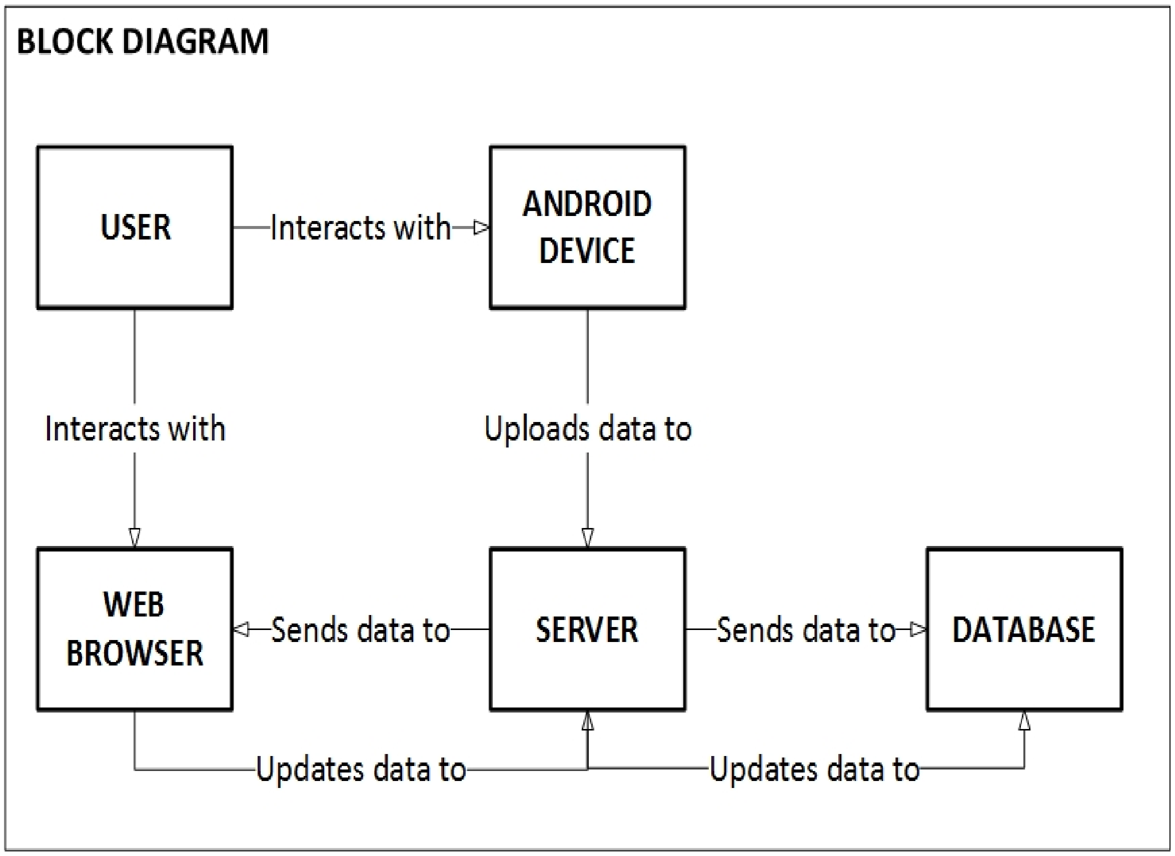


Figure 4: System Block Diagram

3.2.System Context Diagram

A System Context Diagram (SCD) is a diagram that defines the boundary between the system, or part of a system, and its environment, showing the entities that interact with it. This diagram is a high level view of a system. It is similar to a block diagram.

In Figure 5, the context diagram of Storyboarding on the Move shows how the user interacts with the website and the android device. The admin interacts with the website and the database. Android device sends the data to the web server via web services, whereas the web server sends the data to the web site and the database via web services.

Before going into the detail of architectural design we here present logical division of project into modules/parts/subparts and different useful views.

3.3. Software Components

Our project contains following software components:

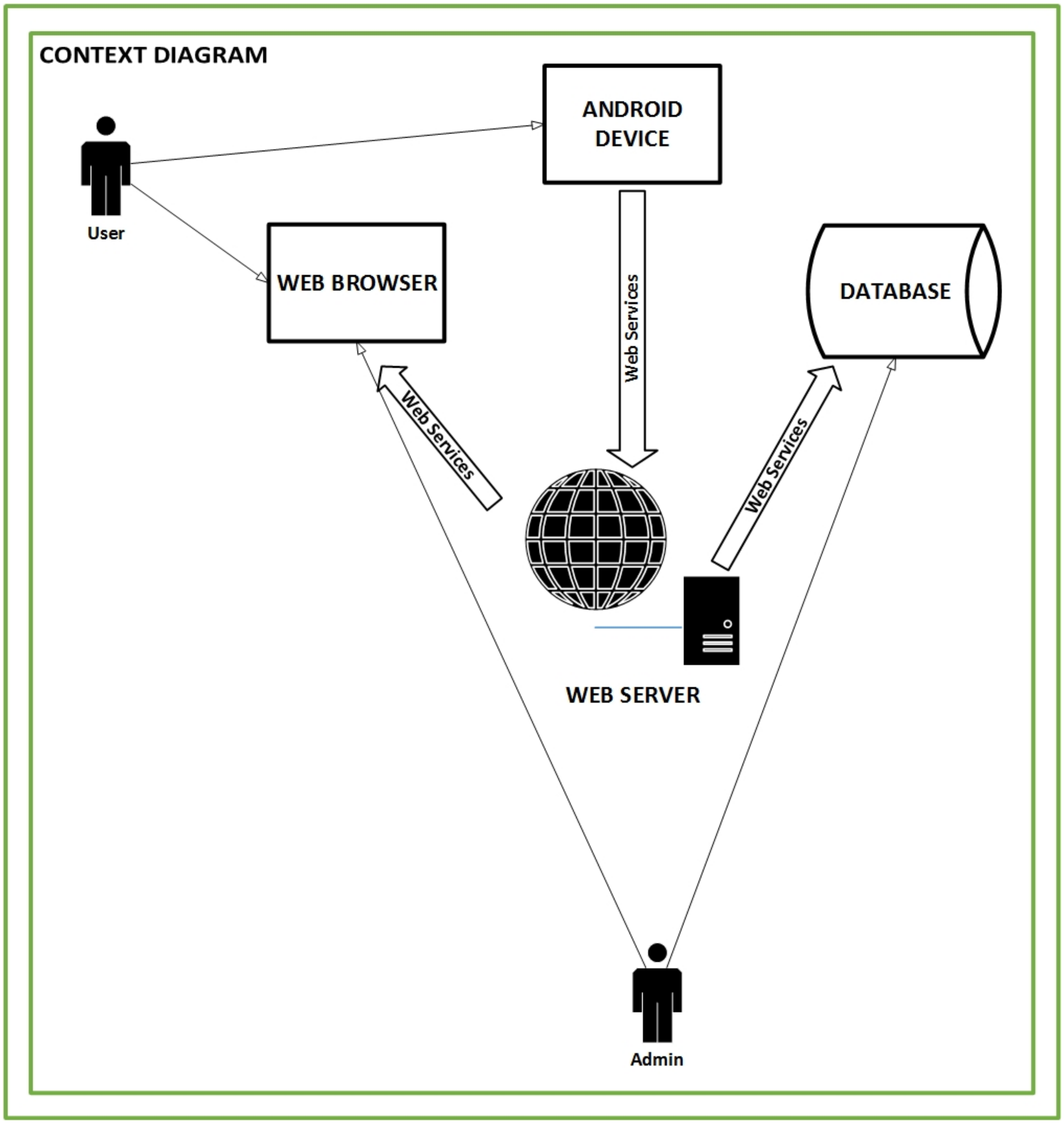


Figure 5: System Context Diagram

3.3.1. Operating Systems

3.3.1.1. Windows 7-8

3.3.1.2. Android OS (4.1.1 Jelly Bean)

3.3.2. Software Packages

3.3.2.1. Application

3.3.2.1.1. Android SDK (Software Development Kit)

3.3.2.1.2. Eclipse IDE (Integrated Development Environment)

3.3.2.1.3. Java SDK 1.6

3.3.2.2. Website

3.3.2.2.1. Wamp Server

3.3.2.2.2. Internet Browsers i.e. Google Chrome, Mozilla Fox etc.

3.3.2.3. Database

3.3.2.3.1. MySQL (Wamp Server)

3.4. Hardware Components

3.4.1. Personal Computer(s)

3.4.2. Android Device with proper functioning Camera, GPS, Wi-Fi etc.

3.4.3. Connecting Cables

3.5. Architectural Style

3.5.1. Application

Architecture of Storyboarding on the Move application can be modeled using **Model-View-Controller**.

3.5.1.1. Model

A model is an object representing data or even activity. Model has the application data. In the project's application the data will be saved in the devices' storage memory.

3.5.1.2. View

A view is told by the controller all the information it needs for generating an output representation to the user. It can also provide generic mechanisms to inform the controller of user input. User class which has the functions related to the GUI forms the view. Interface of the system is distinct from the application logic. Interface forms the View of the architecture. In the project's application all the .xml files are included in the view.

3.5.1.3. Controller

Controller in this system is Usecase handlers as the system functionality is clearly divided in modules. But for reusability of the system a façade controller will be a bad choice so use-case handler controllers are used. Controller controls the coordination between the view and the model. There is no direct communication between model and the view all the communication is directed using the controller.

The system is divided into modules as per the main functionality of the system. These functionalities can be used separately as off-the shelf components. Interface of this system has no application logic embedded in it so the system architecture can easily be made using Model View Controller approach.

For Example, pressing TakePicture_Button, RecordAudio_Button and ViewMap_Button are the functions that involve the processing of interface and no application logic is involved in it. So these will be included in **View**.

TakePictureHandler is used for the TakePicture module, RecordAudioHandler is used for the RecordAudio module and ViewMapHandler is used for the ViewMap module. These UsecaseHandlers only invoke the functions of their respective modules. **Controller** only invokes the functions but does no processing.

Model in this system is the TakePicture, RecordAudio, ViewMap class which is encapsulating the application logic. These classes are composing the other classes which have the distributed functionality. These classes process the requests made by the view through the controller. After processing, Model gives the retrieved results back to the controller.

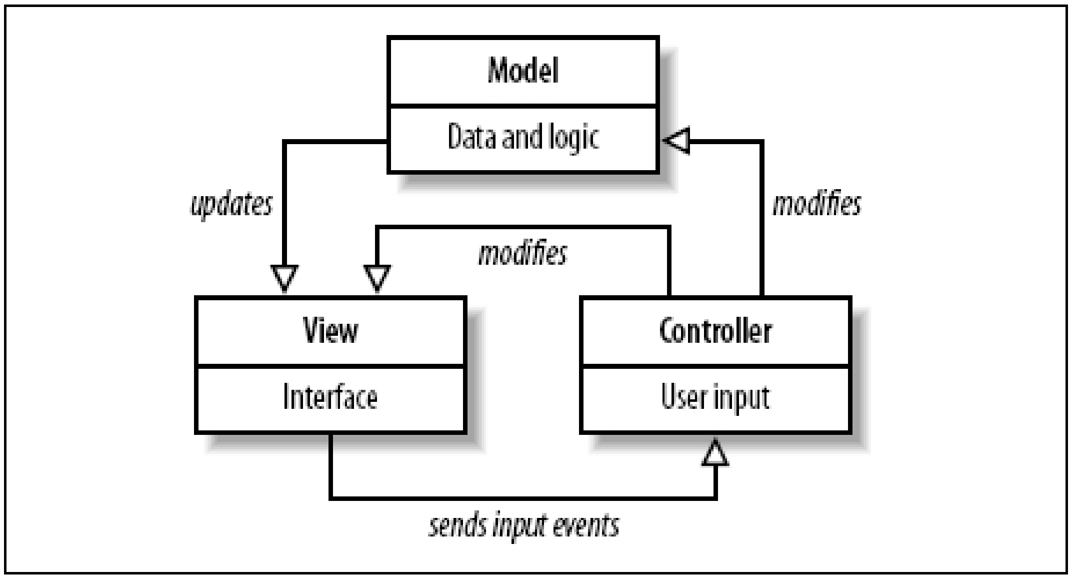


Figure 6: Model-View-Controller Diagram

3.5.2. Website

Architecture of Storyboarding on the Move website can be modeled using **Three-Tier Architecture**.

3-tier architecture is client–server architecture in which presentation, application processing, and data management functions are logically separated. It provides a model by which developers can create flexible and reusable applications. By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application. 3-tier architecture is typically composed of a presentation tier, a logic tier, and a data tier.

As more users access the system a three-tier solution is more scalable than the other solutions because you can add as many middle tiers (running on each own server) as needed to ensure good performance (N-tier or multiple-tier). Security is also the best in the three-tier architecture because the middle layer protects the database tier. There is one major drawback to the N-tier architecture and that is that the additional tiers increase the complexity and cost of the installation.

3.5.2.1. Presentation tier

This is the topmost level of the application. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. In simple terms it is a layer which users can access directly such as a web page, or an operating systems GUI.

In the project's website, the data displayed by the web pages in the web browsers comes under this tier.

3.5.2.2. Logic tier

The logical tier is pulled out from the presentation tier and, as its own layer; it controls an application's functionality by performing detailed processing. In this project, the PHP script will be in logic tier. The presentation layer contacts with the logic tier

3.5.2.3. Data tier

This tier consists of database servers. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance. The database is included in the data tier. If there is any SQL query it is asked from the data tier, which in return sends a table to the logic tier for further processing.

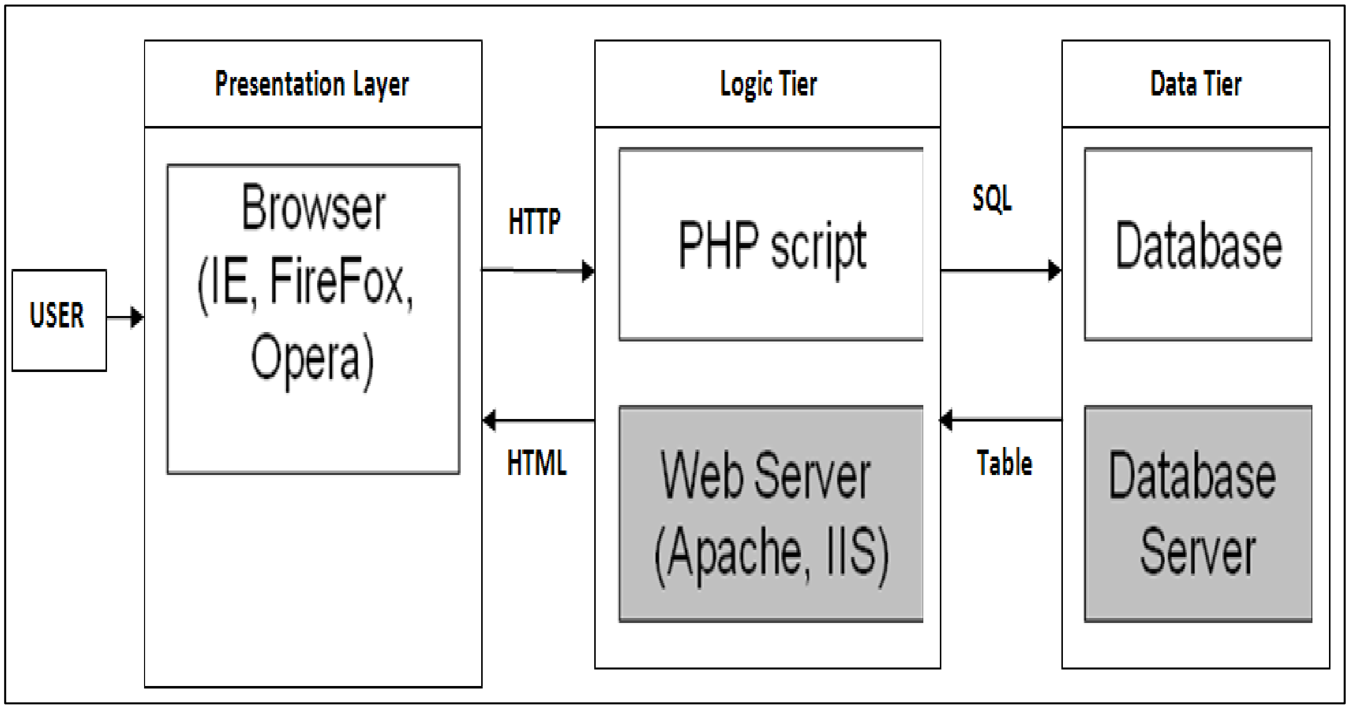


Figure 7: 3-Tier Architecture

4. Detailed Design

4.1. Database Diagram

4.1.1. Entity Relationship Diagram

ENTITIES	ATTRIBUTES
User	UserID, FullName, Gender, Year Of Birth, Profile Pic
Posts	PostID, UID, DateTime, Post(Picture, Video, Audio, Places, Personal Notes)
Comments	CommentId, Pid, Uid, Comments
Friends	Uid, FriendID
Accounts	UserId, UserName, Password

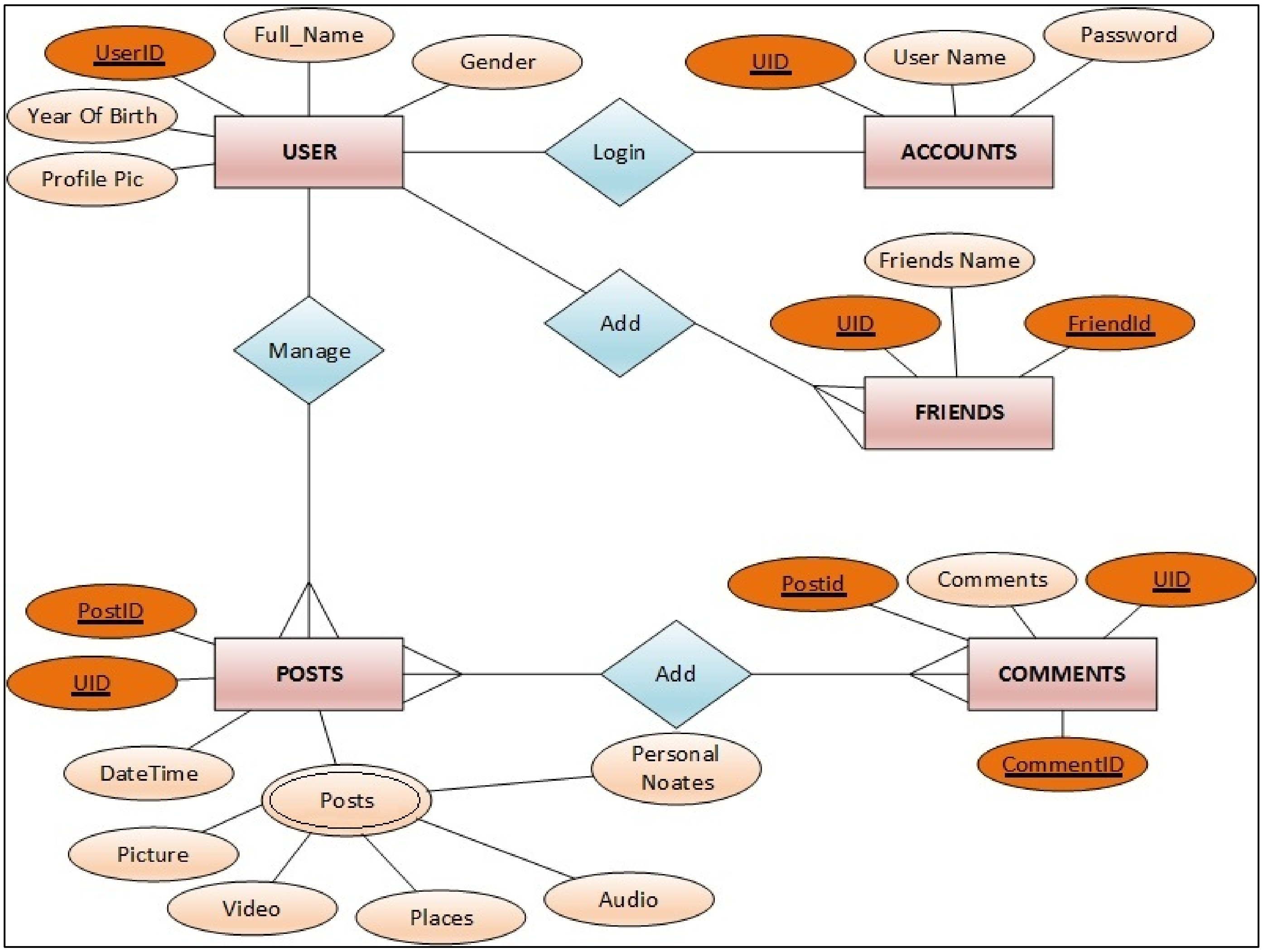


Figure 8: Entity-Relationship Diagram

4.2. Unified Modeling Language (UML) Diagrams

4.2.1. Use case Diagram

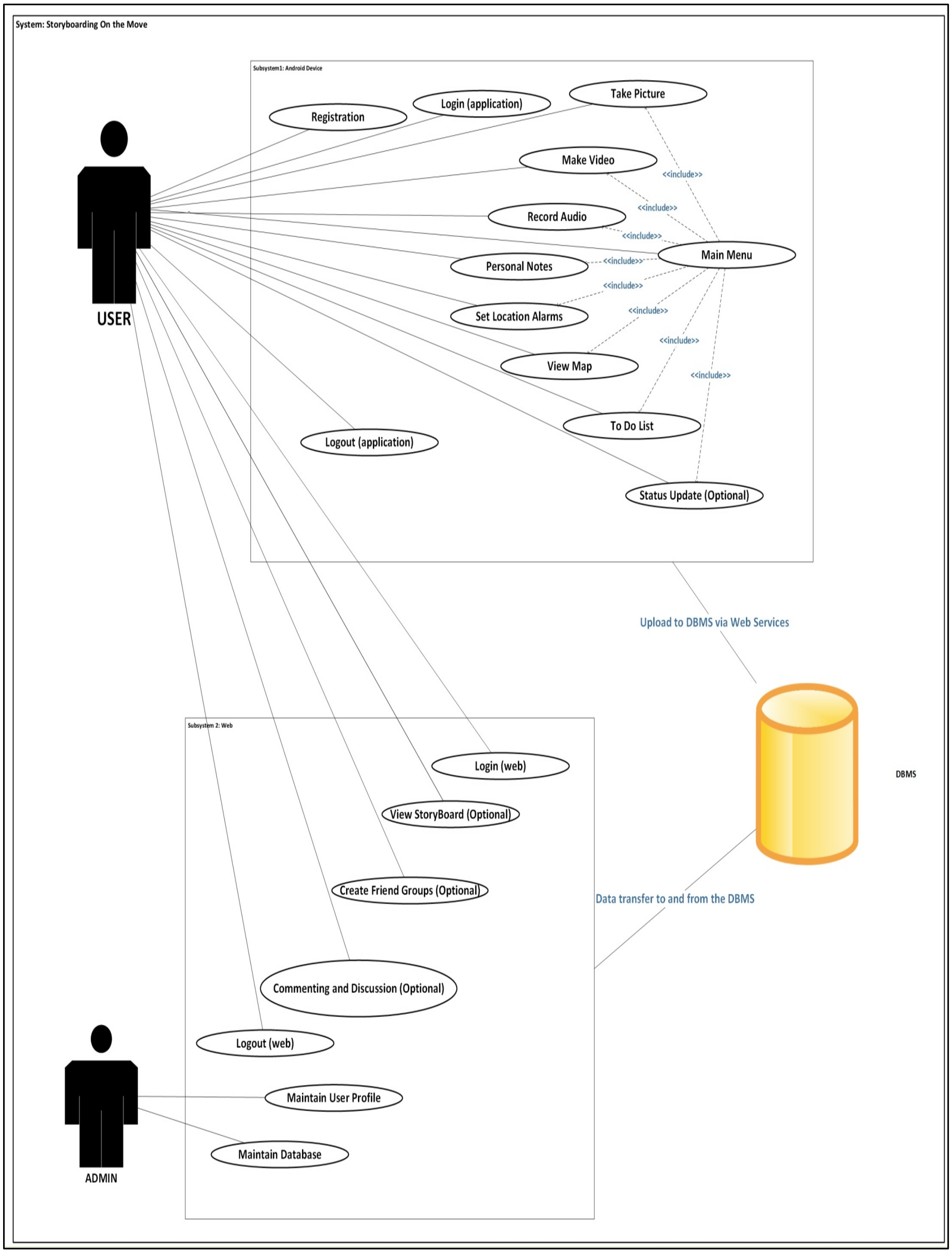


Figure 9: Use Case Diagram of System

4.2.1.1. Use case Specifications

Actors:

User

Admin

Use Cases:

Main Menu	To Do List	Maintain User Profile
Take Picture	Status Updates (Optional Feature)	Registration (application end)
Make Video	Maintain Database (on server end)	Commenting and Discussion (Web end – Optional Feature)
Record Audio	Login (for application)	Create Friend Groups (Web end – Optional Feature)
Personal Notes	Login (for website)	View Map
Set Location Alarms	Logout (for application)	Logout (for website)
View Storyboard		

Login (for website)

Use Case Specification

Use Case ID	1
Use Case Name	Login (for application)
Actor(s)	User
Description	The 'Login' feature on the web end will authorize the actor to enter and start with the website. After the actor's login (username and password) is validated, he is able to use all features of the website. This actor authentication will take the actor to user profile where the he can view all system features

	and his timeline.
Pre-Conditions	The Website must be linked with the application.
Post-Conditions	If the use case is successful, the actor will effectively login to the application. Otherwise, not.
Normal Flow (Primary Scenario)	The use case starts when the actor launches a browser: 1. Write website address in the URL ; i.e. https://www.storyboardingonthemove.com 2. The login screen will appear in case the actor is visiting the website for the first time or he has unchecked the “stay signed in” option previously. 3. Actor will enter the username and password if already registered.
Alternative Flow(s)	**First Alternative Flow** The use case starts when the actor launches a browser: 1. Write website address in the URL ; i.e. https://www.storyboardingonthemove.com 2. The login screen will appear in case the actor is visiting the website for the first time or he has unchecked the “stay signed in” option previously. 3. Actor will enter the username and password if already registered. 4. Otherwise, if the actor has checked the “stay signed in” option previously, he will be redirected to the homepage. If the user is not registered, however, he would have to click on register option on the screen.

Sequence Diagram

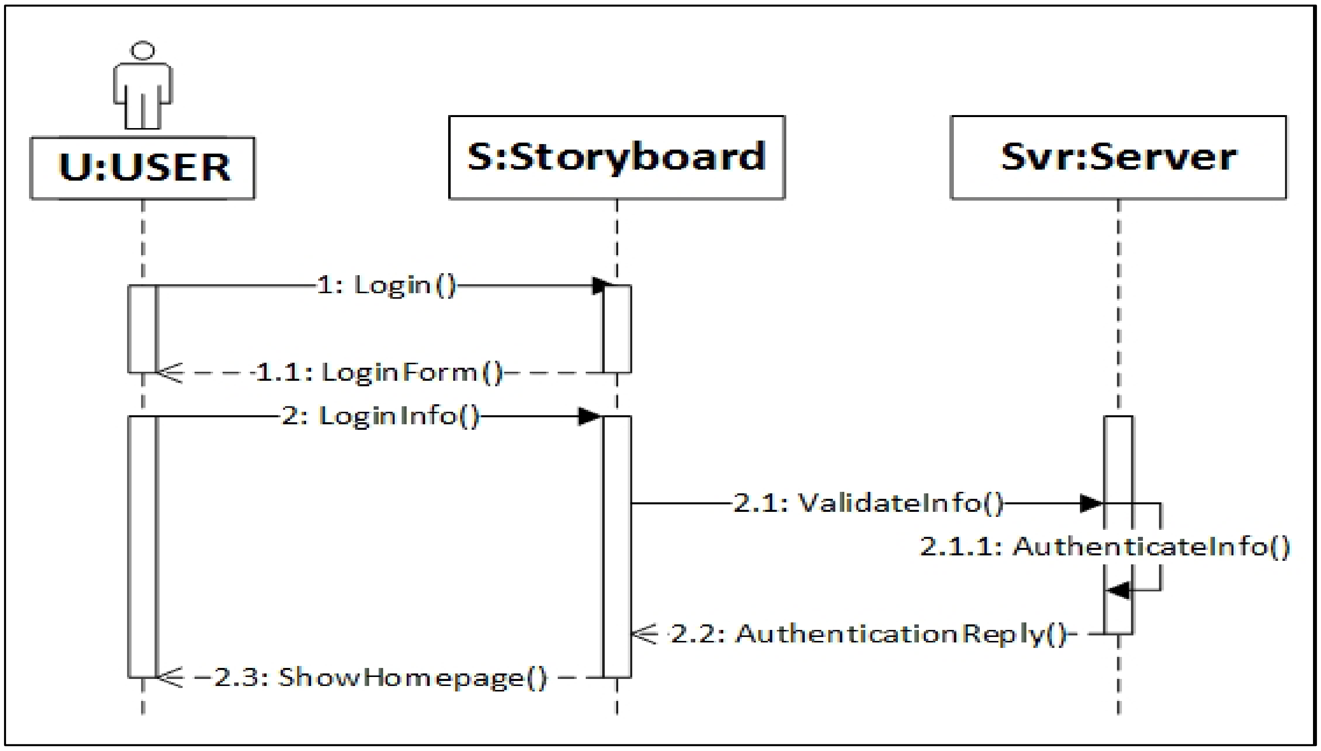


Figure 10: Sequence Diagram for Login

Collaboration Diagram

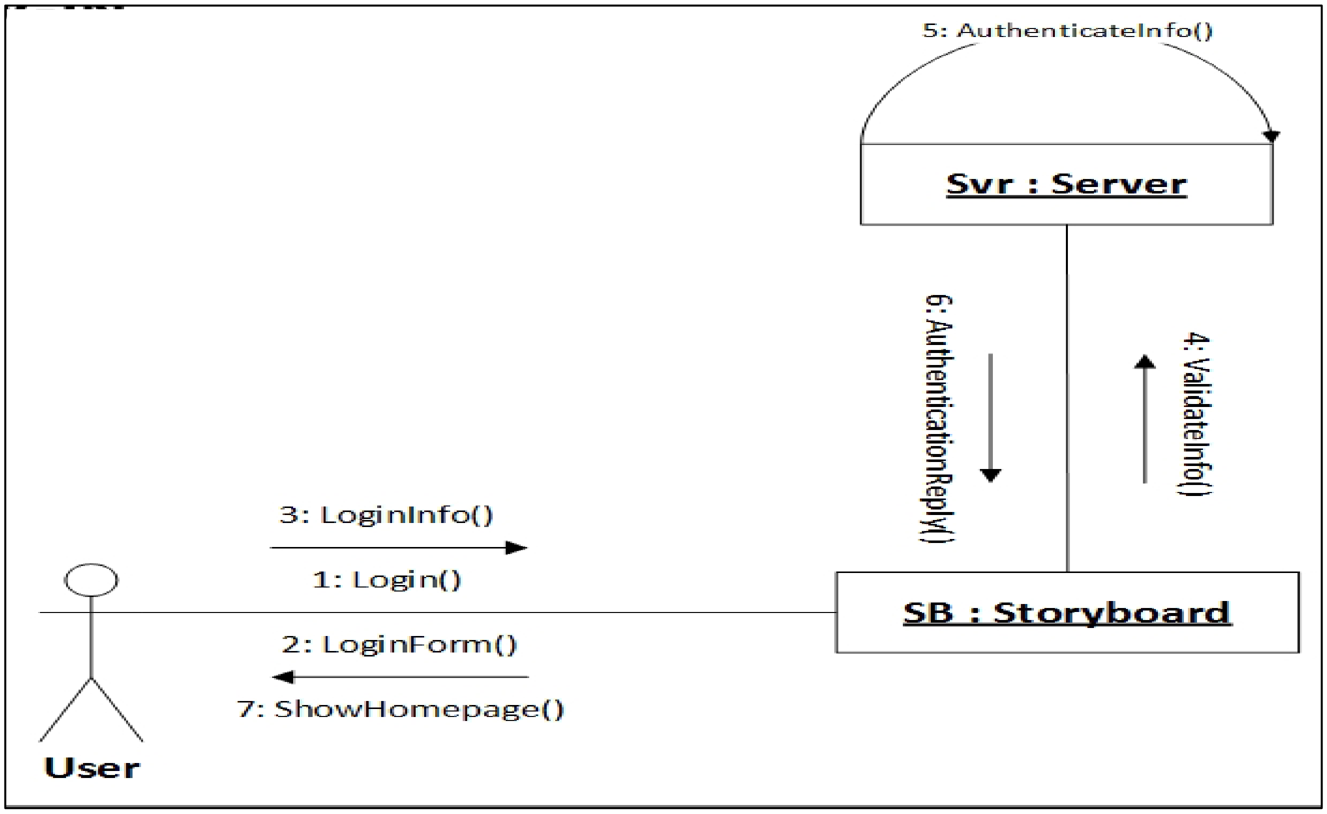


Figure 11: Collaboration Diagram for Login

Main Menu

Use Case Specification

Use Case ID	2
Use Case Name	Main Menu
Actor(s)	User
Description	Main menu allows the actor to select a particular feature among basic seven features of the application. These include: 1. Take picture 2. Make video 3. Record audio 4. Personal notes 5. Set Location Alarms 6. View Map 7. To do List 8. Status Update (Optional) Furthermore, the actor will also be able to access the pictures and videos from the main gallery. As for the audio recordings the actor will be able to access them from the application's 'Audio' folder. Also, there will be a 'Log Out' icon on the top right corner of this 'Main Menu' screen.
Pre-Conditions	The application must be properly installed on the device and actor should be authorized to use this application. The actor has to start the application first and log in to it. Also, the content related to the respective feature must be loaded in the application already and each feature must be linked with its metadata.
Post-Conditions	If the use case is successful the actor has the choice to enjoy any

	of the above mentioned features of application. Otherwise, he can exit from the application as and when desired.
Normal Flow (Primary Scenario)	1. Click on any of the desired icon of the features being displayed on the Main Menu screen. 2. A new screen will be displayed that will contain the content related to the feature selected in point 1.
Alternative Flow(s)	1. Click back icon on the android device’s screen. 2. Actor will be navigated back to the device’s menu where all other applications are seen in a grid view.

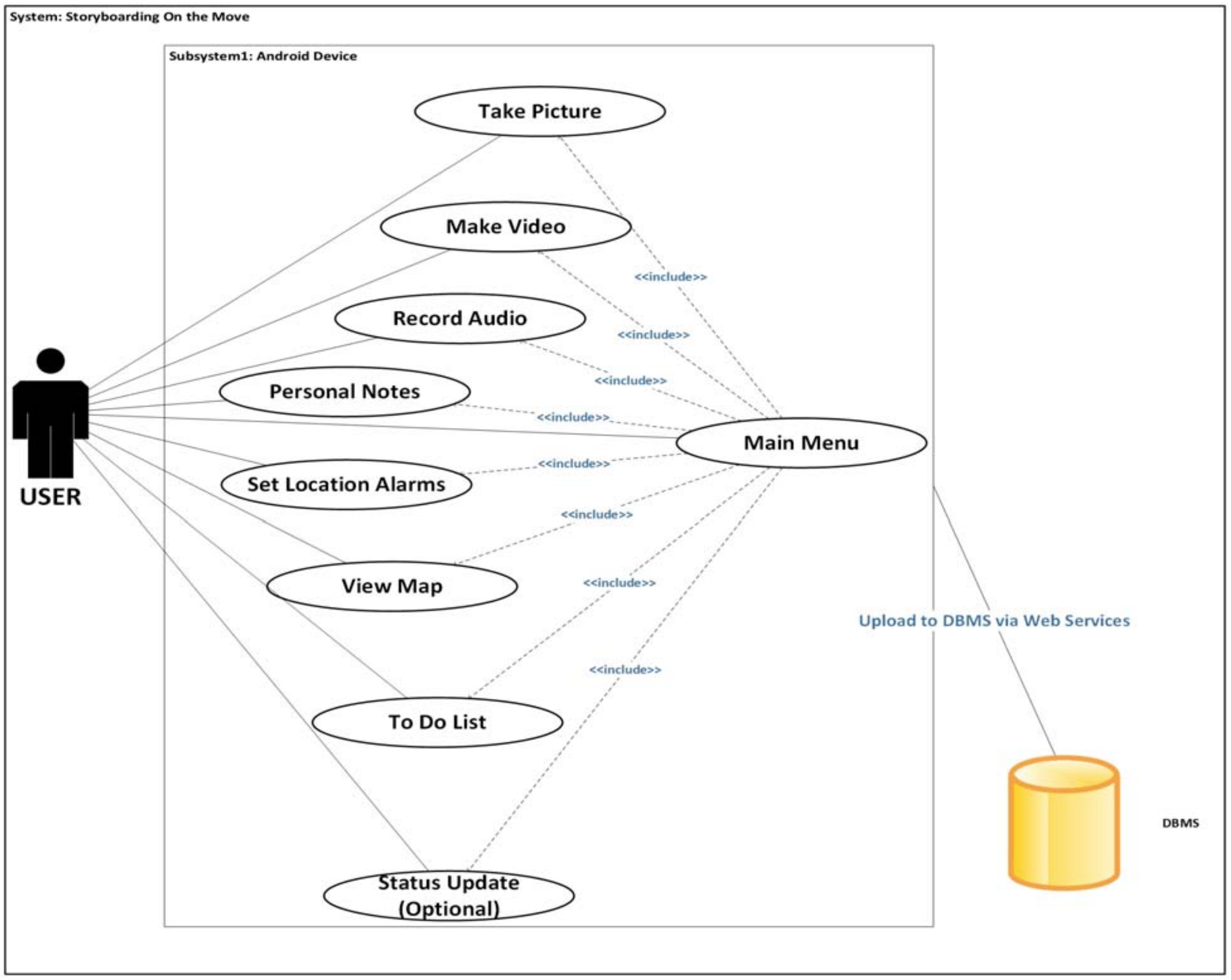


Figure 12: Use-case Diagram of Main Menu

Take Picture

Use Case Specification

Use Case ID	3
Use Case Name	Take Picture
Actor(s)	User
Description	The 'Take Picture' feature takes the actor to device's camera, allows him to take picture. The picture is then saved to device's gallery. Also, the actor will be given the option to accept or reject and retake a picture as per the desire. Accepting or rejecting the picture taken will generate an alert that is described in the following flow.
Pre-Conditions	The application must be properly installed on the device and actor should be authorized to use this application. Also, the device's camera must be loaded in due time.
Post-Conditions	If the use case is successful the picture is saved in gallery. Otherwise, back or cancel button would close the 'Take Picture' activity safely and bring back the 'Main Menu' screen.
Normal Flow (Primary Scenario)	1. Click on the capture icon of the camera. 2. An image will be captured and the actor will be having the options of accepting or rejecting the picture represented by a tick or cross respectively. 3. Clicking on the tick mark will (firstly save the picture into the gallery at the back end and then) prompt the actor if he wants to upload the picture privately or publicly.
Alternative Flow(s)	**First Alternative Flow** 1. Click on the capture icon of the camera. 2. An image will be captured and the actor will be having the options of accepting or rejecting the picture represented by a tick or cross respectively. 3. In case if the actor does not want to upload the picture at all,

	he can click the 'No' option on prompt that will take him back to camera screen. **<u>Second Alternative Flow</u>** 1. Click on the capture icon of the camera. 2. An image will be captured and the actor will be having the options of accepting or rejecting the picture represented by a tick or cross respectively. 3. If the actor wants to discard the picture altogether, he can click the cross sign, that will take him back to camera screen.

<u>Sequence Diagram</u>

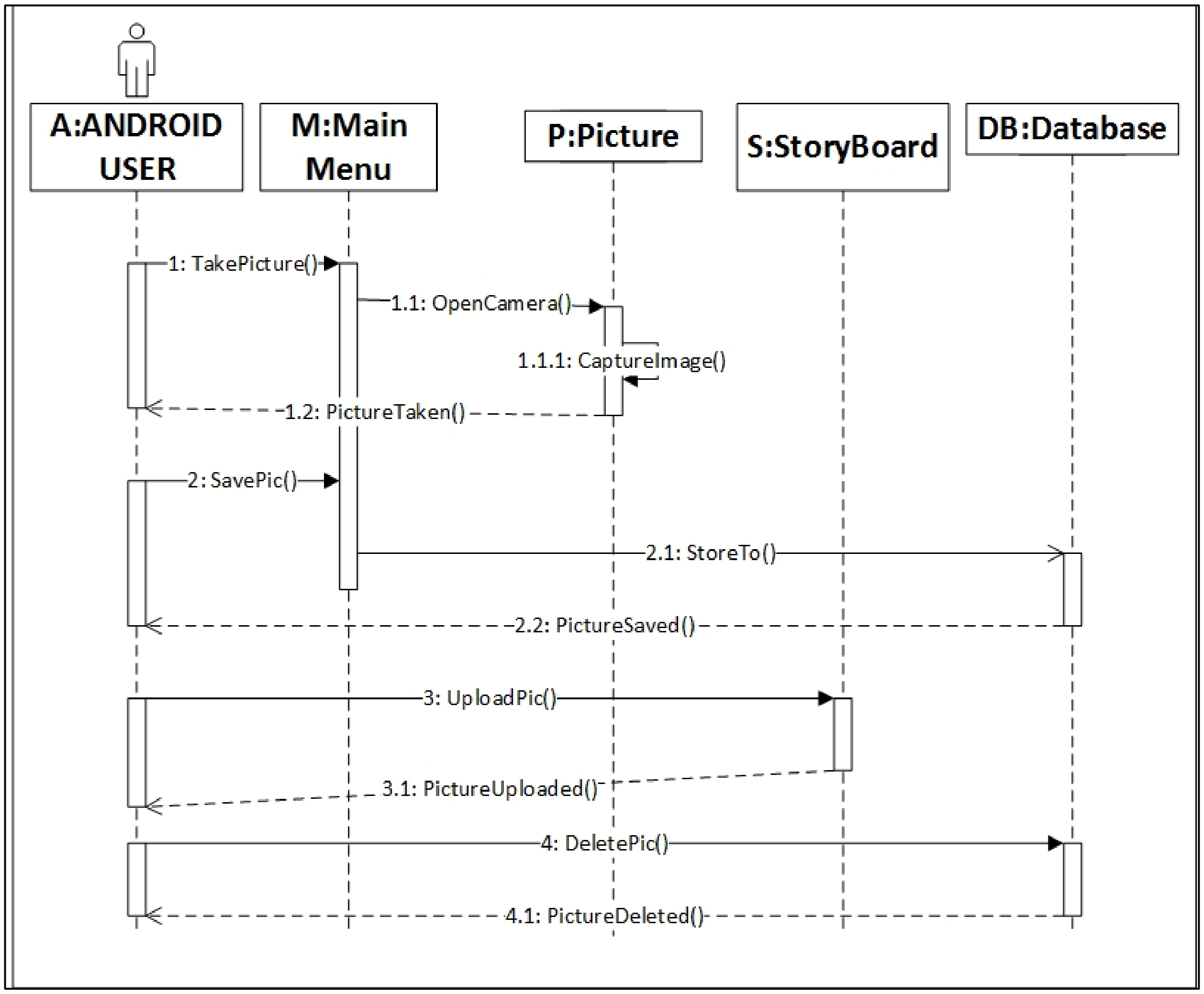


Figure 13: Sequence Diagram for Take Picture

Collaboration Diagram

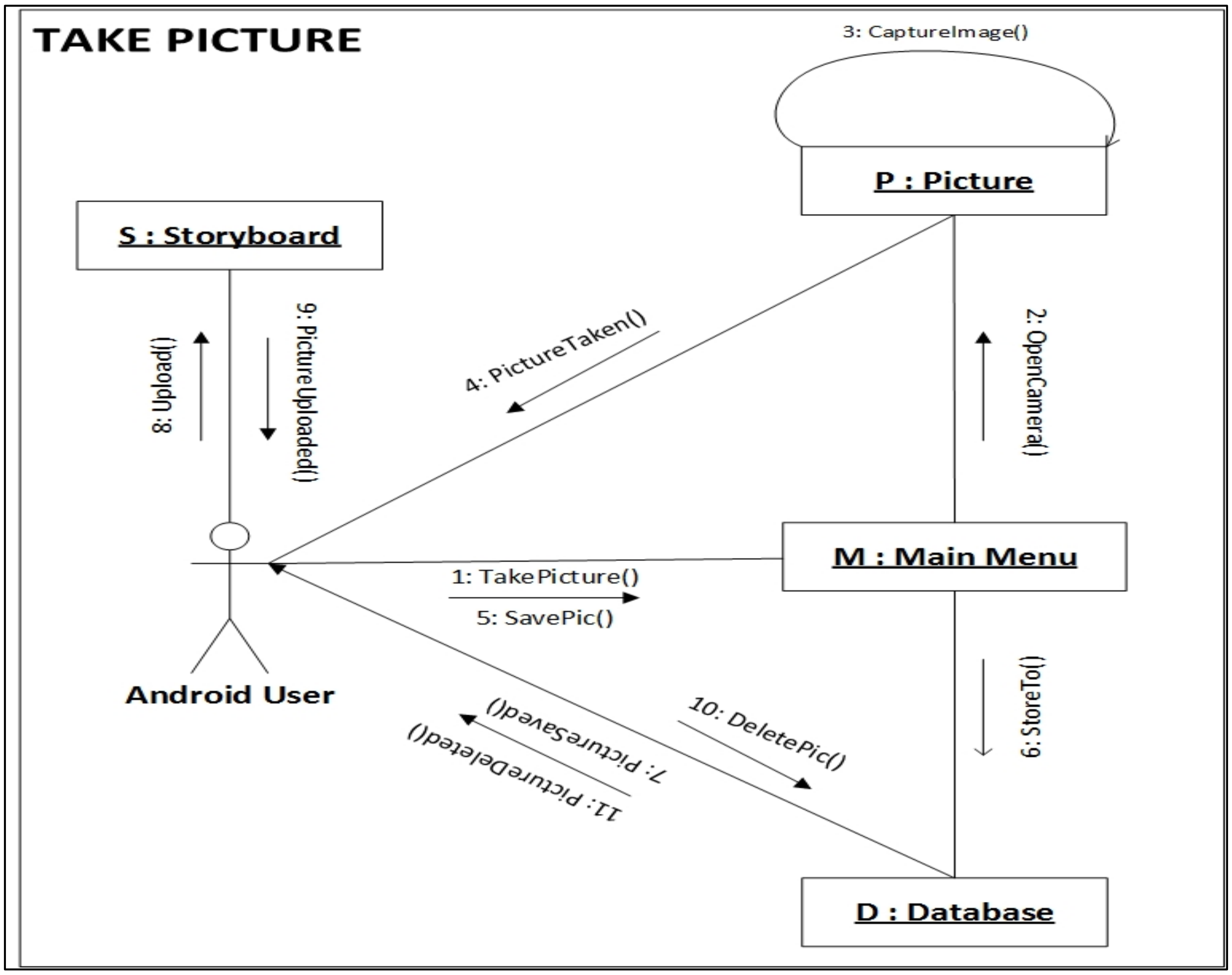


Figure 14: Collaboration Diagram for Take Picture

Set Location Alarms

Use Case Specification

Use Case ID	4
Use Case Name	Set Location Alarms
Actor(s)	User
Description	This feature will be interlinked with the 'To Do List' feature of the application. Based on the location of a particular listed in the 'To Do List' feature, the 'Location Alarm' feature will get the co-ordinates of that location. An alarm will be triggered when the actor would be about to approach the respective location.
Pre-Conditions	The application must be properly installed on the device and actor

	should be authorized to use this application. Information of 'To Do List' feature must be linked and maintained constantly with this feature. Also, GPS of the android device should be activated and turned on.
Post-Conditions	If the use case is successful, application will trigger the respective alarm as the actor approaches relevant place.
Normal Flow (Primary Scenario)	1. The actor needs to turn on the GPS while he's on the move. 2. This feature will automatically trigger any alarms that might be related to the place(s) which cross the actor's route at that time. 3. A pop up window will appear that the actor would need to click/tap. 4. Clicking this pop up will show all the details that the actor had already recorded while entering the task in 'To Do List' feature. 5. Clicking the 'OK' button will take the actor back to 'Main Menu'.
Alternative Flow(s)	**First Alternative Flow** 1. The actor needs to turn on the GPS while he's on the move. 2. This feature will automatically trigger any alarms that might be related to the place(s) which cross the actor's route at that time. 3. A pop up window will appear that the actor would need to click/tap. 4. Clicking this pop up will show all the details that the actor had already recorded while entering the task in 'To Do List' feature. 5. Clicking the 'Done' button will delete the alarm and mark out the respective task from the 'To Do List' feature.

Sequence Diagram

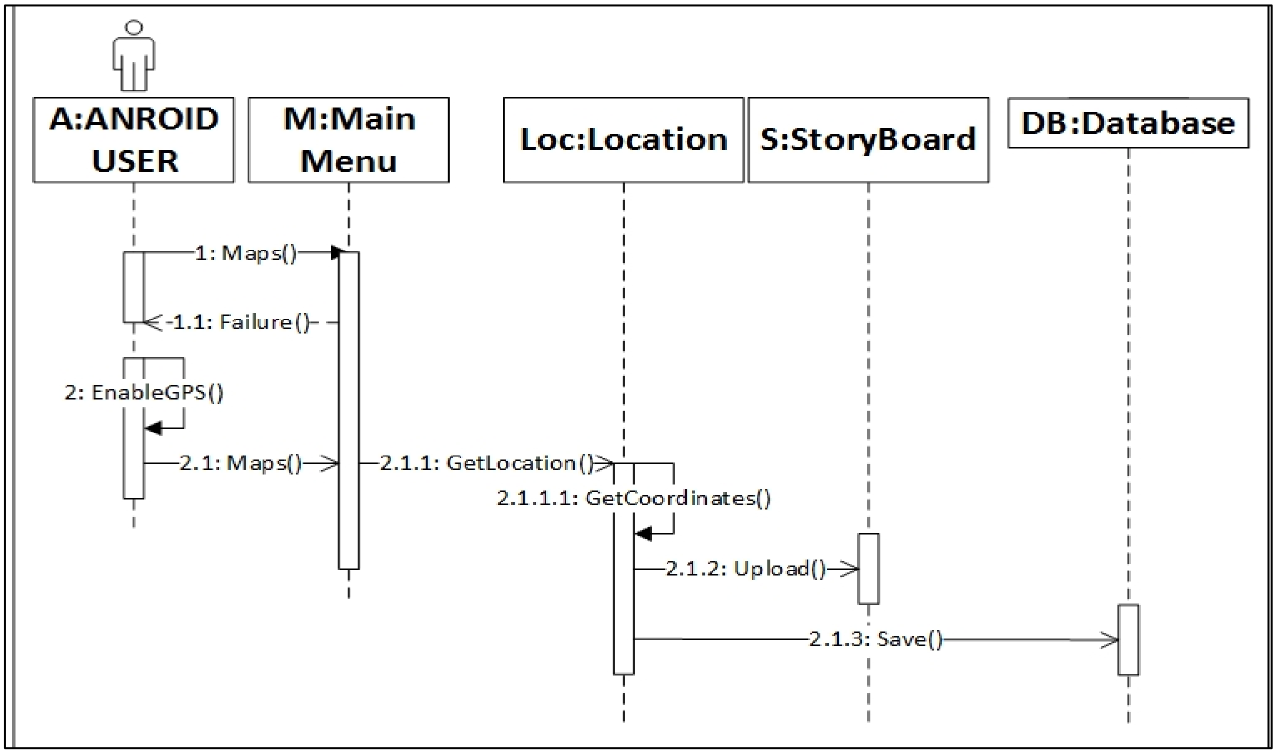


Figure 15: Sequence Diagram for Set Location Alarm

Collaboration Diagram

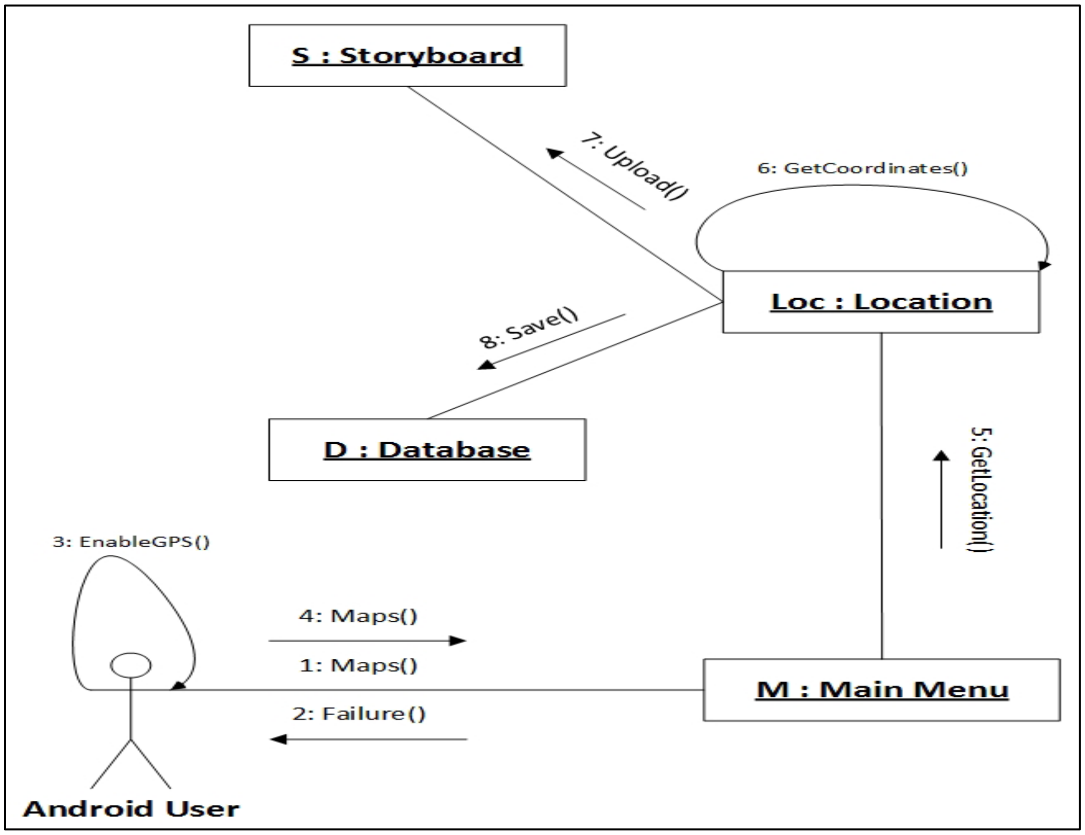


Figure 16: Collaboration Diagram for Set Location Alarm

View Storyboard

Use Case Specification

Use Case ID	5
Use Case Name	View Storyboard
Actor(s)	User
Description	It will facilitate the actor to view a friend’s or his own storyboard activities.
Pre-Conditions	The user must be logged in already. The Website must also be linked with the DBMS along with consistent updating of data from the DBMS.
Post-Conditions	If the use case is successful, the actor will have the option to set privacy on his activities by making them visible only to selective people.
Normal Flow (Primary Scenario)	1. Click on name of the actor that is seen just below his picture at top left corner. 2. The actor’s own storyboard will be seen.
Alternative Flow(s)	1. Navigate to the website’s ‘Homepage’. 2. Click on name of the actor that is seen just below his picture at top left corner. 3. The actor’s own storyboard will be seen. 4. In case the actor wants to view a friend’s storyboard, he searches the friend’s name in ‘Friends’ category on his own homepage, and clicks it. 5. His friend’s storyboard will be seen.

Sequence Diagram

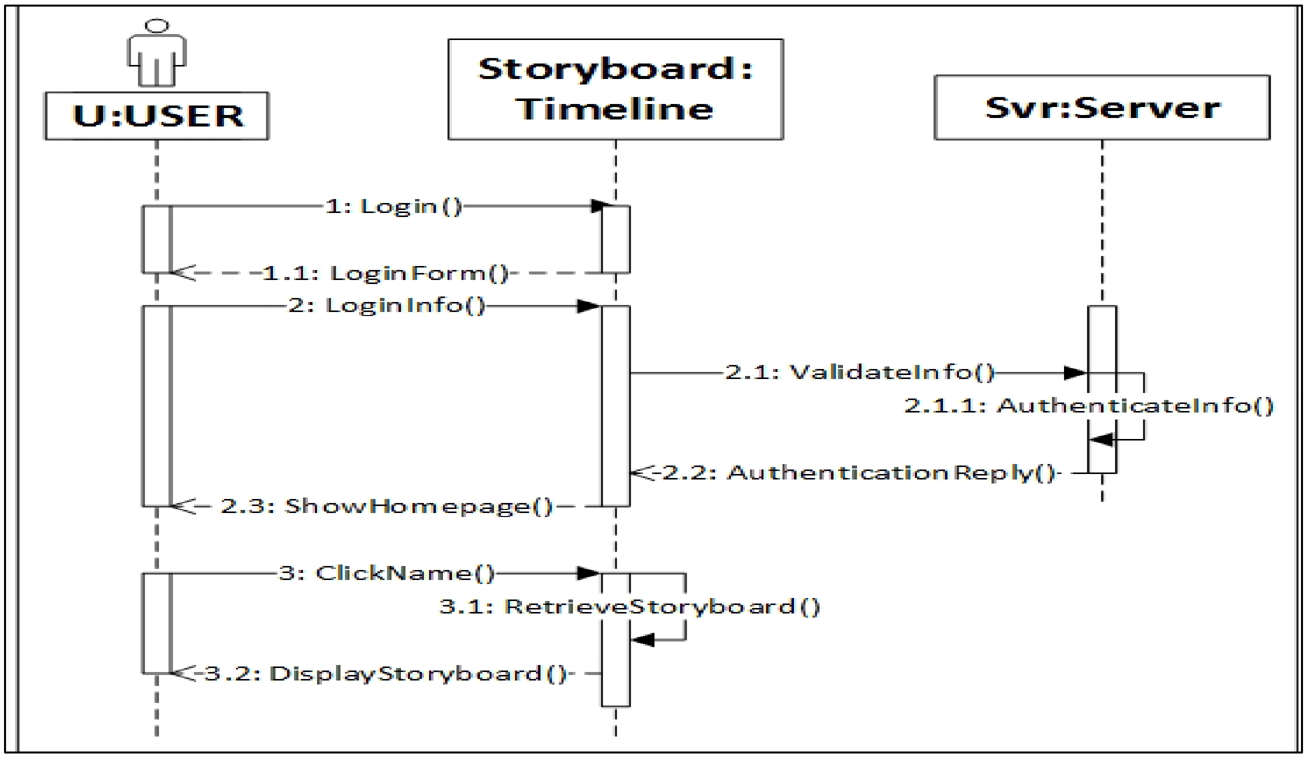


Figure 17: Sequence Diagram for View Storyboard

Collaboration Diagram

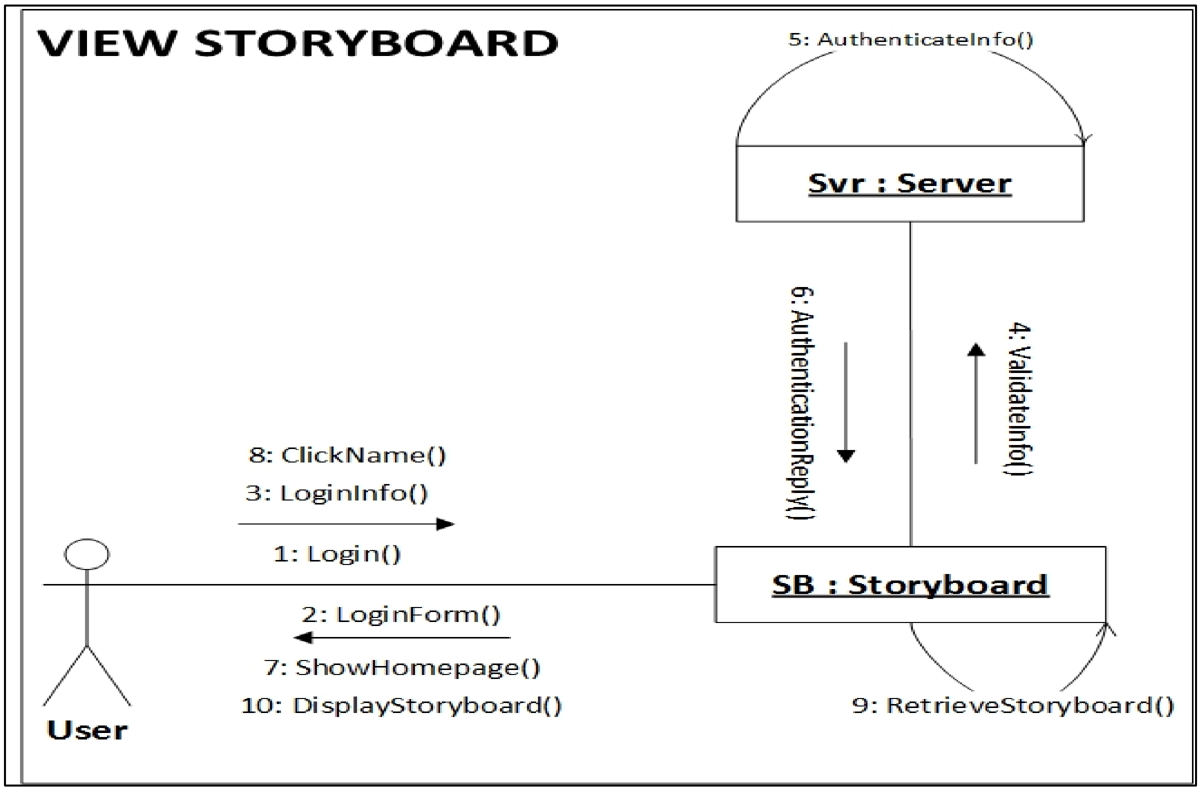


Figure 18: Collaboration Diagram for View Storyboard

4.2.2. **Logical View**

State Transition Diagram

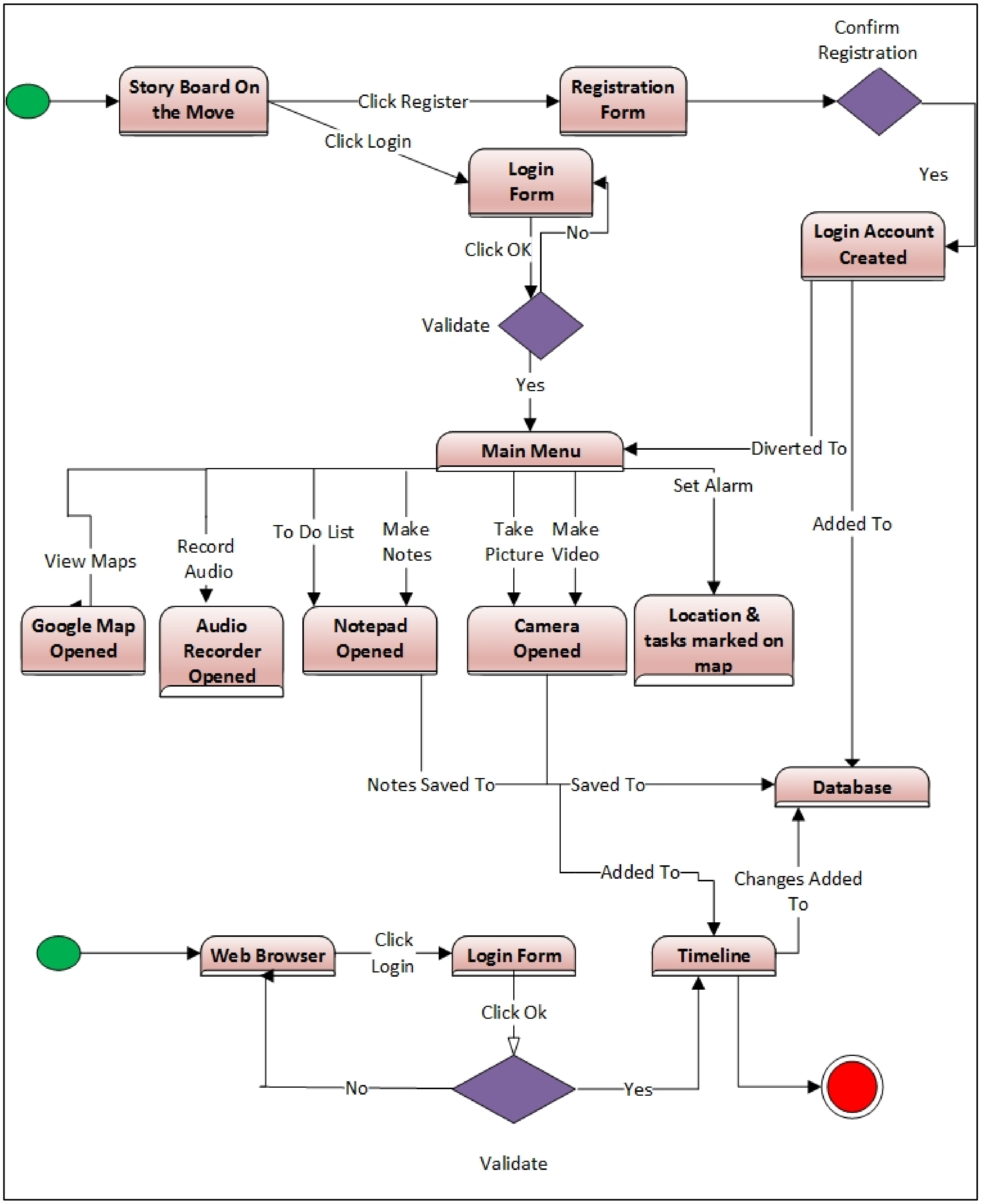


Figure 19: System State Transition Diagram

4.2.3. **Dynamic View**

Activity Diagram for Application

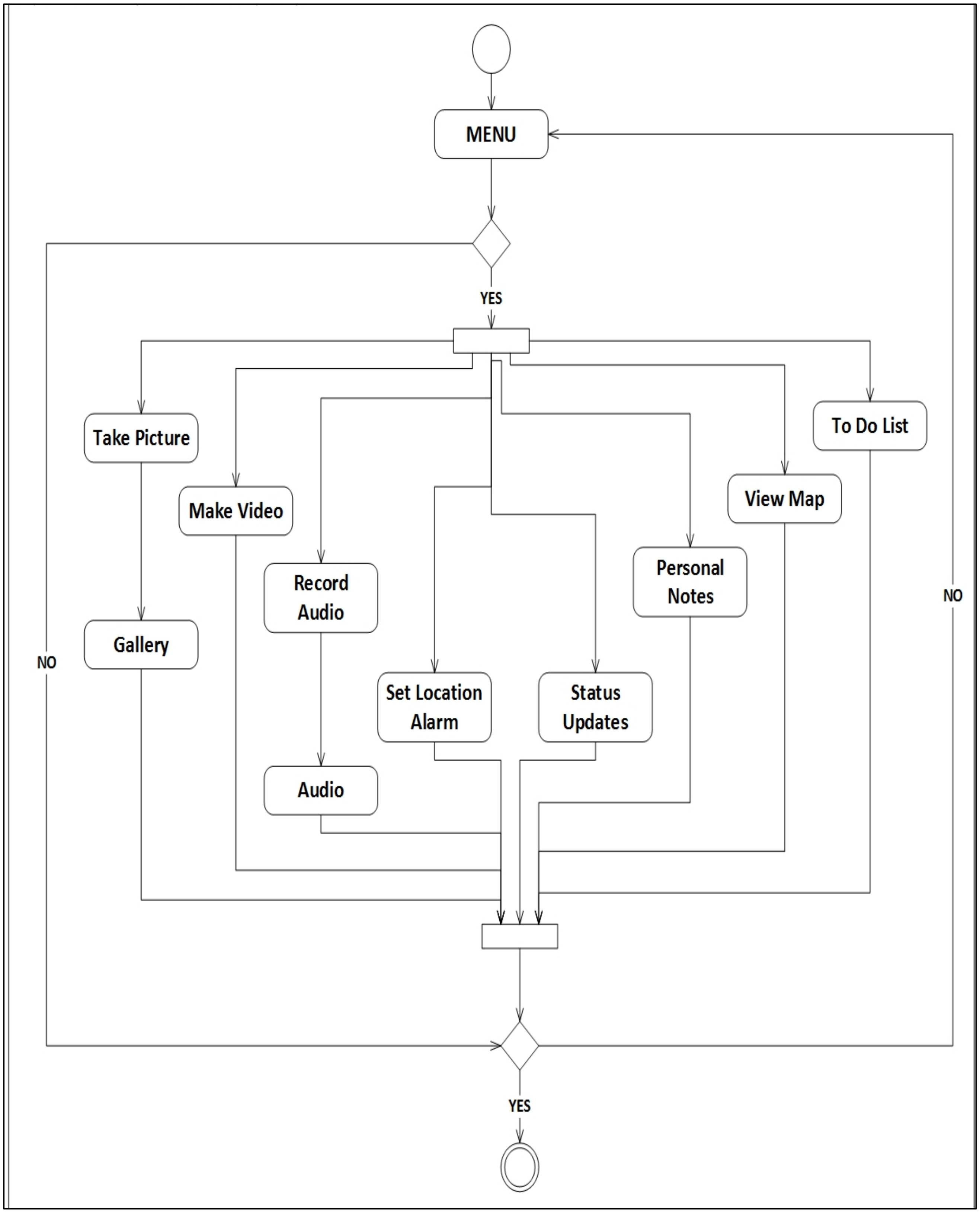


Figure 20: Activity Diagram - Android Subsystem

Activity Diagram for Website

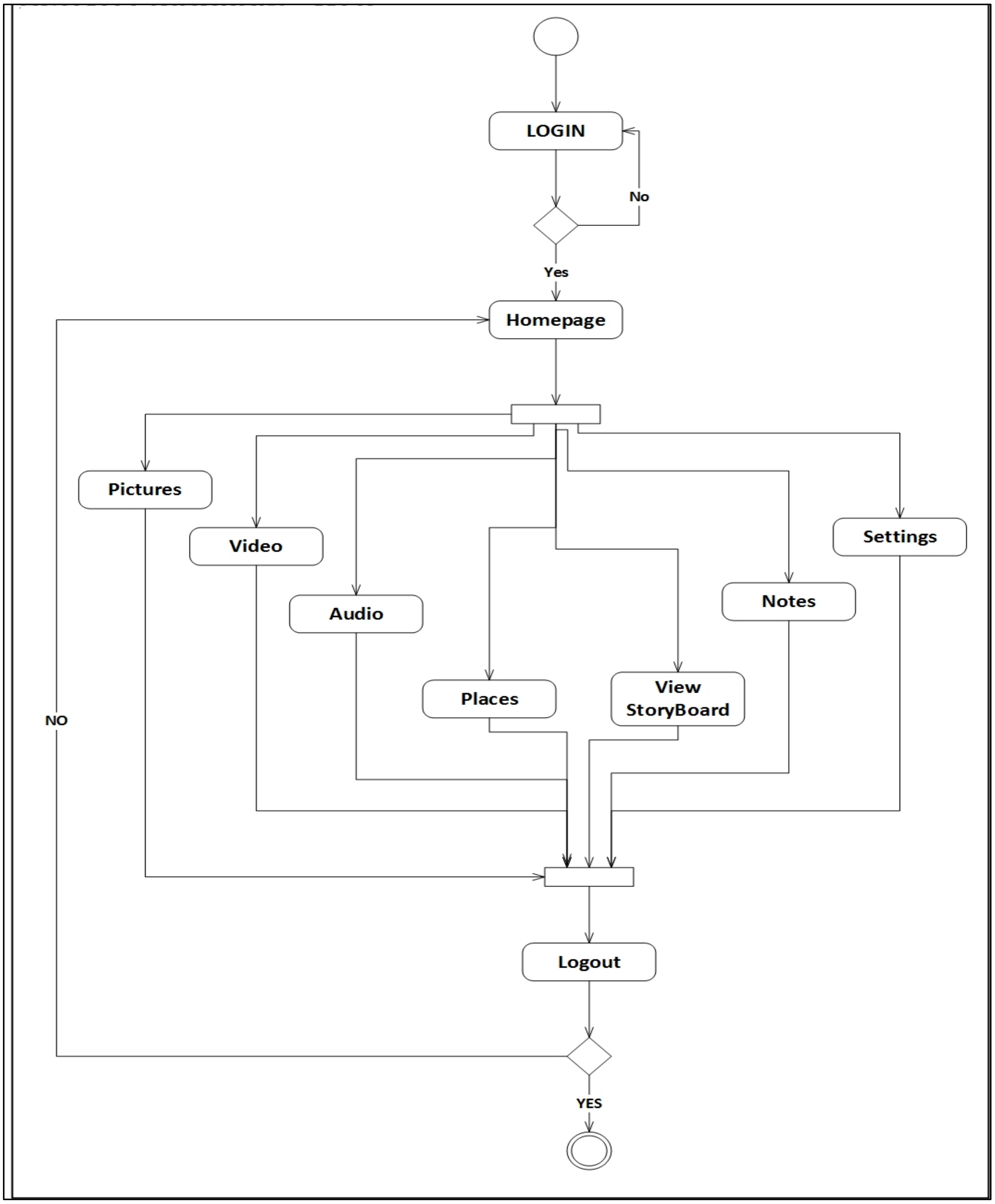


Figure 21: Activity Diagram - Web Subsystem

4.2.4. **Implementation View**

System Class Diagram

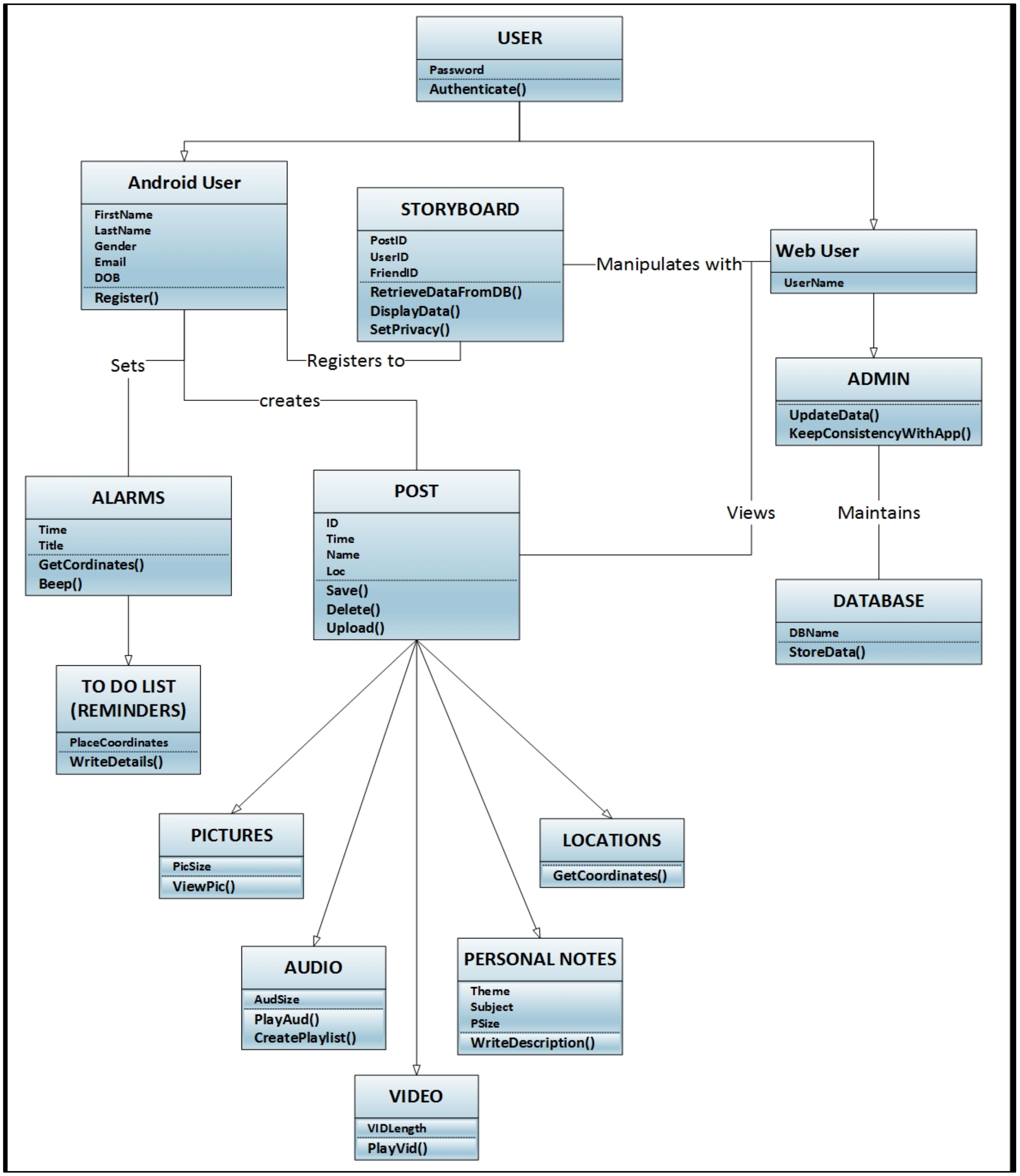


Figure 22: System Class Diagram

System Class Description

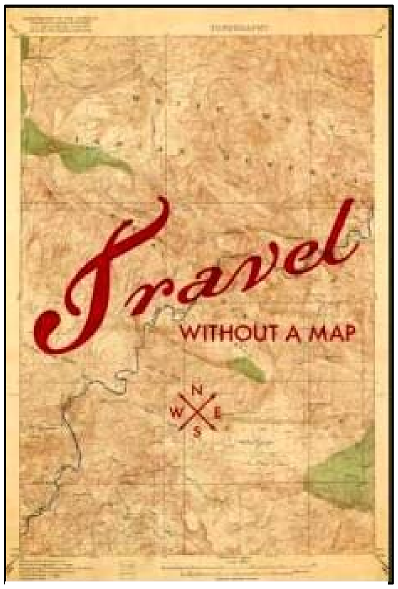
Class Name	**Description**
User	This is the base class of the system. The subclasses, Web User and Android User, inherit from this base class. Same user can be web user and android user also.
Web User	This class is the main class on the web side. Its sub classes include Admin class that maintains the database class. This class can manipulate the data uploaded and can view all the content that is pictures, video etc.
Android User	This class inherits from the base class User. It is the main class of the android application. It contains the following classes Picture, Video, Audio, Personal Notes, Locations, Alarms, To Do List. It leads to the other classes of the application. This class includes the function Register() that registers a new user.
Post	This is the main class for the posts the user posts on the webpage. It forms the base class for the following subclasses: Pictures, Video, Audio, Personal Notes and Locations. Post class has the following funtions: Save(), Upload() and Delete(). So the derived classes inherit the functions.
Picture	This class is responsible for openning camera activity, taking the picture and adjusting the camera features.This class is solely responsible for the capturing of image, saving it and uploading it as the user is asked whether he wants to upload the picture or not. It saves the image in the application's folder. This class inherits the functions Save(), Delete() and Upload() from class 'Post' and another method of its own, i.e; ViewPic().
Video	This class is responsible for openning camcorder activity, making the video and adjusting the camcorder features.This class is solely responsible for the recording of video, saving it and uploading it as the user is asked whether he wants to upload the video or not. It saves the image in the application's folder. This class inherits the functions Save(), Delete() and Upload() from class 'Post' and another method of its own, i.e; PlayVid().
Audio	This class records the audio tracks or voice notes.This class is solely

	responsible for recording audio, saving it and uploading it as the user is asked whether he wants to upload the recording or not. It saves the audio file in the application’s folder. This class inherits the functions Save(), Delete() and Upload() from class ‘Post’ and a couple of methods of its own, i.e; PlayAud() and CreatePlaylist().
Personal Notes	If the user chooses to save some notes for himself then this class handles it. The user is given the choice to whether upload the noted words or not. This class inherits the functions Save(), Delete() and Upload() from class ‘Post’ and another method of its own, i.e; WriteDescription().
Locations	Locations class using the Google Maps API to display the map and also returns the coordinates of the user’s current location. When the user changes his location this class updates the coordinates using location listener. This class inherits the functions Save(), Delete() and Upload() from class ‘Post’ and another method of its own, i.e; GetCoordinates().
Alarms	When the user has to set a reminder on a specific location, this class is responsible. It coordinates with the location class so that when the user passes that location, the alarm rings. It includes the functions; GetCoordinates() and Beep().
To Do List	This class takes the users tasks to be done on the locations and saves it until the location is reached. It then triggers an alarm. This class inherits the functions GetCoordinates() and Beep() from class ‘Alarms’ and another method of its own, i.e; WriteDetails().
Admin	This class is solely responsible for maintaining the database. As whatever the user does has to be saved in the database.
Database	This class maintains the data. It has all the data stored in it including the comments, posts, content etc.
Storyboard	This class is the main class on the user end. As it shows the data uploaded by the user. It contains the following funcitons; RetrieveDataFromDB(), DisplayData() and SetPrivacy().

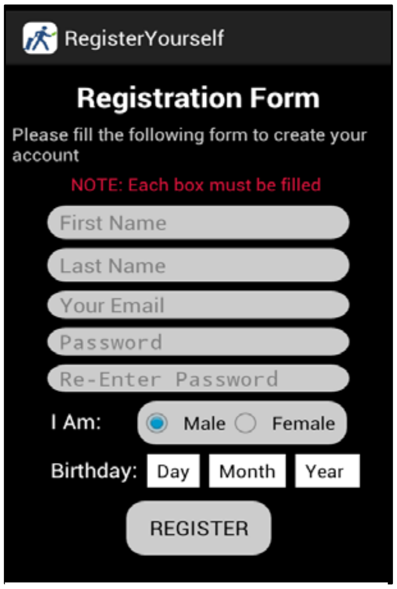
5. Human Interface Design

5.1. User Interface

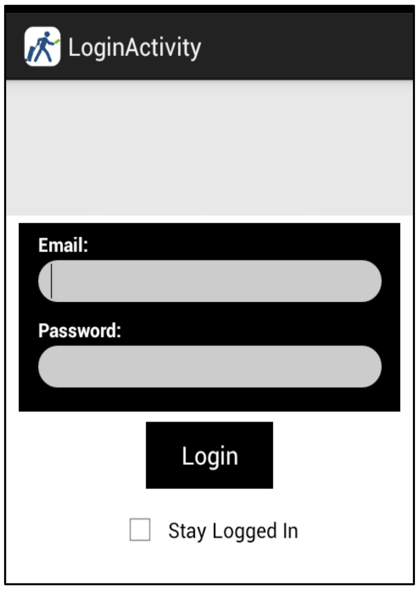
Storyboarding on the Move Application



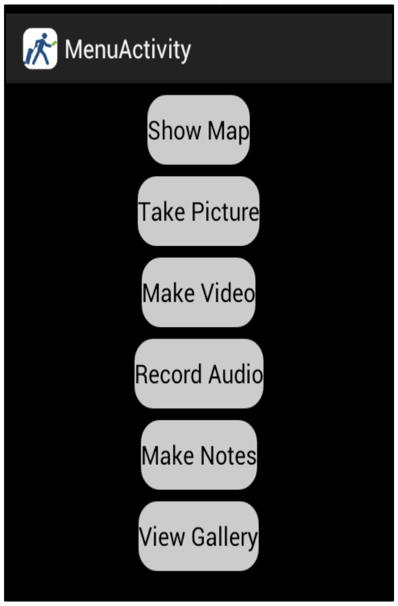
Splash Screen



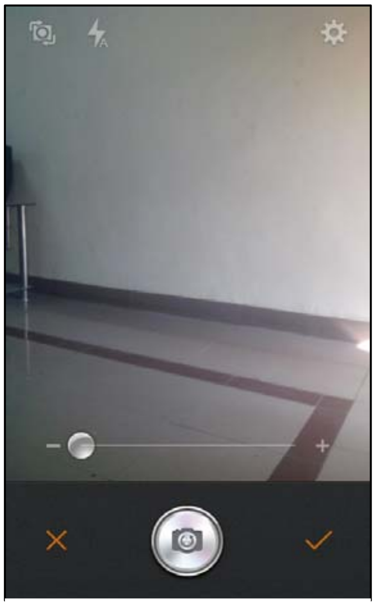
Registration Form



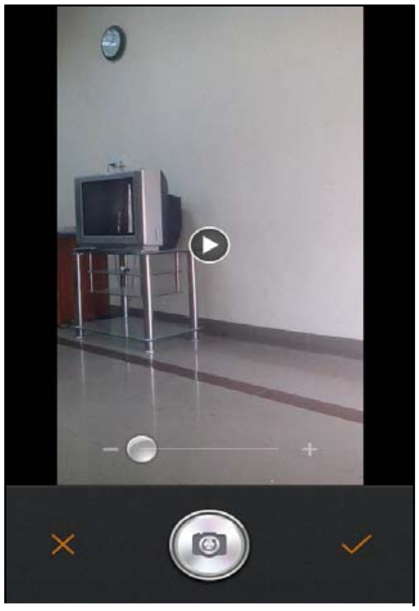
Login Screen



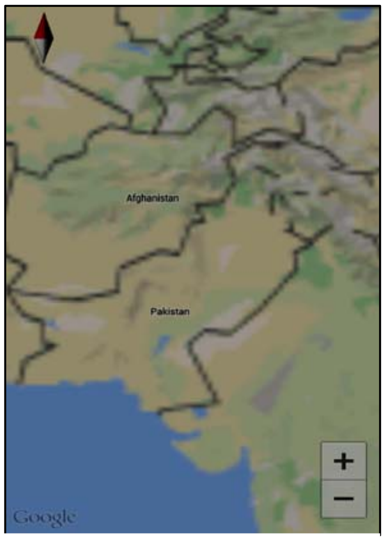
Menu Screen



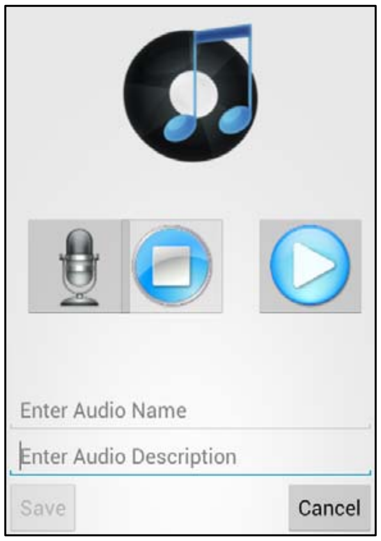
Camera Screen



Video Screen



Map View



Record Audio Screen

Storyboarding on the Move - Website

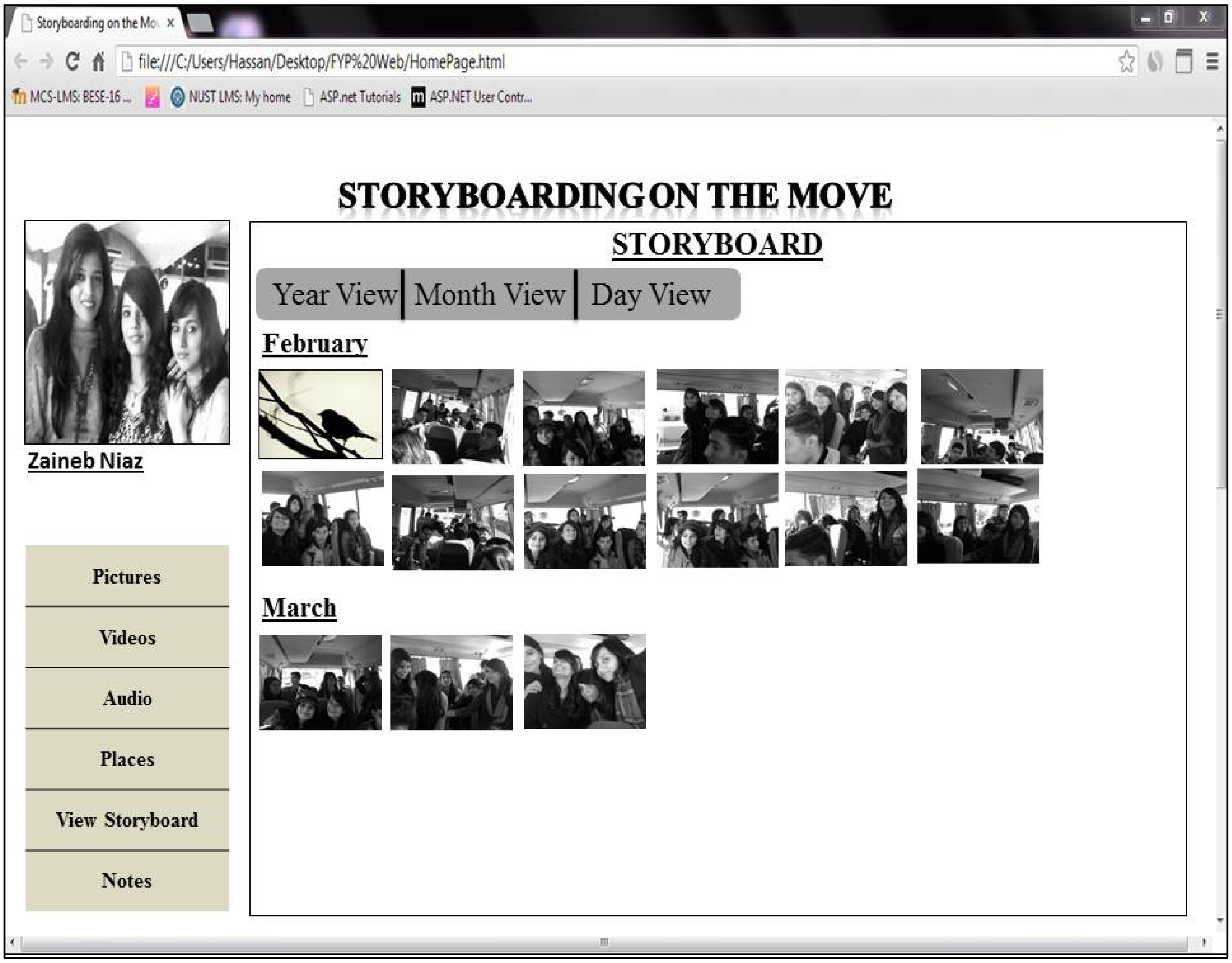
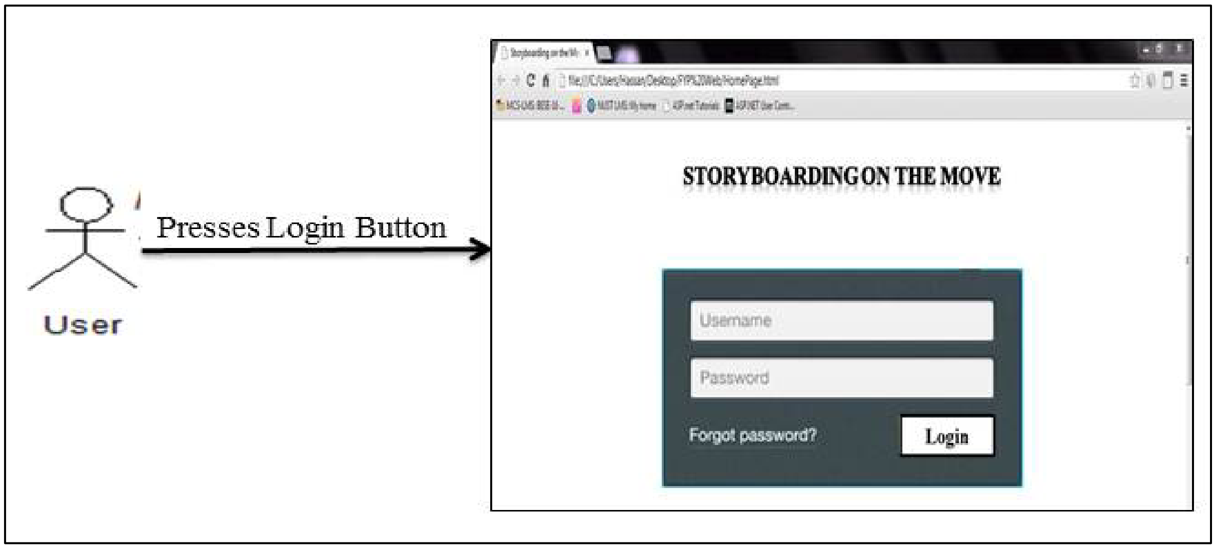


Figure 23: Storyboard Interface - Web

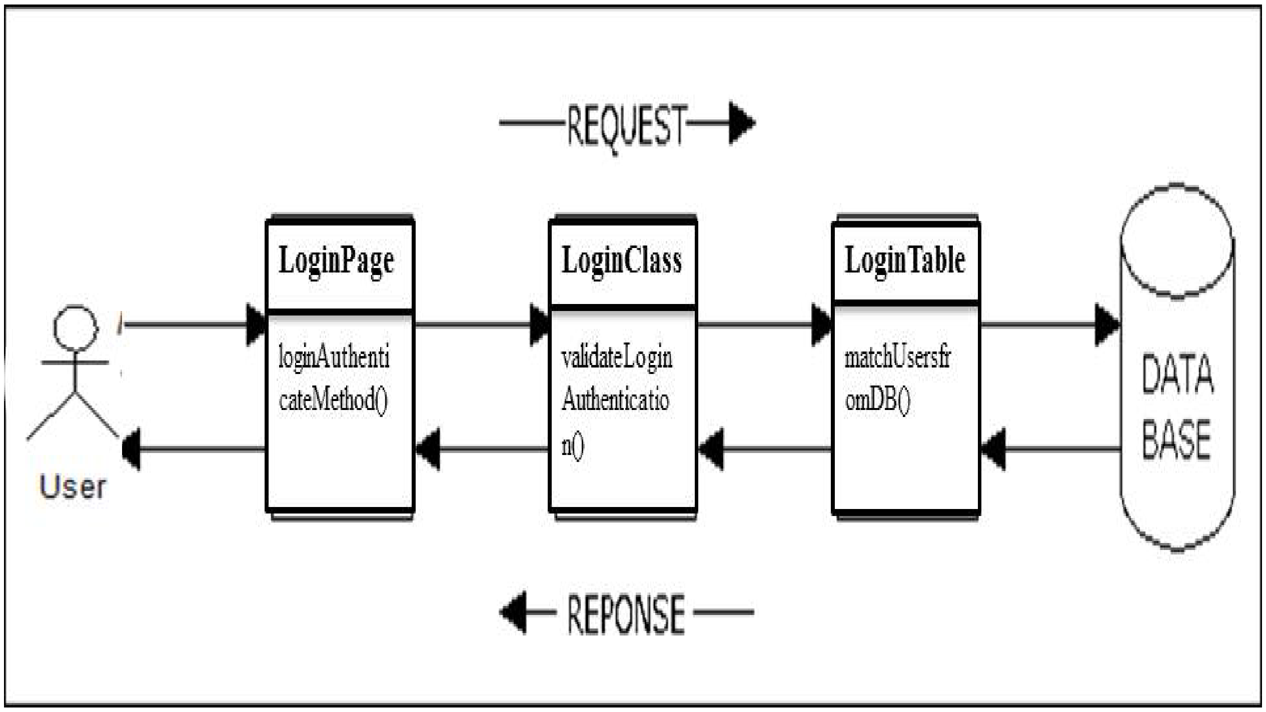
6. Design Patterns

6.1. Pattern Name: 3-Tier Pattern

Problem: What first object beyond the UI layer receives and coordinates a system operation (Login of Website)?



Solution:



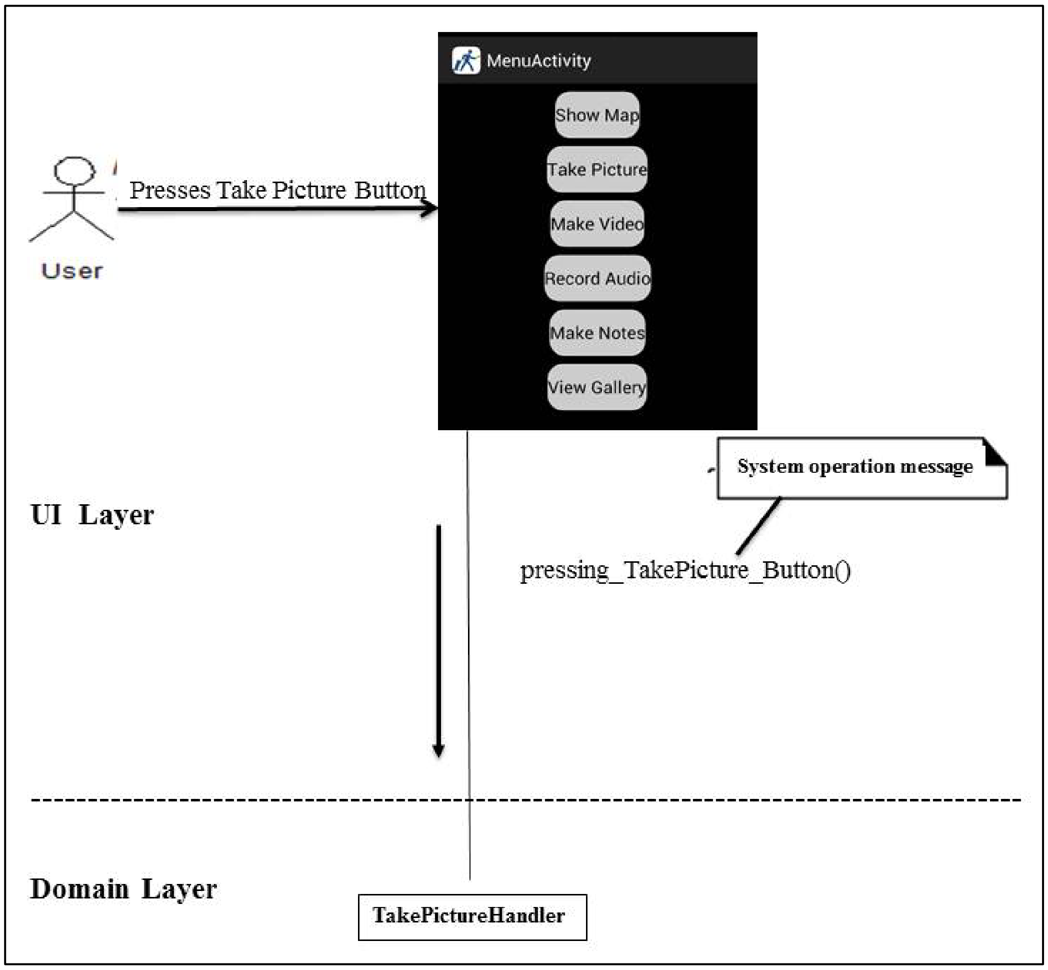
Whenever an input system event is generated by pressing the Login Button which is in the **Presentation Layer** of the system, request for the processing will be made to the **Business Layer** which in this scenario is Login class. This layer will then require the authenticated users from the database so the process will enter the **Data Access Layer** which in this case is the Login. This will use the database and retrieve the authenticated users from the database table. After processing, user will login if the user name is valid and the password matches the username.

By using 3-tier architectural design pattern, clear separation of user-interface-control and data presentation from application-logic is obtained. Change in business logic won't need change in other layers. Also, Dynamic load balancing can be done by use of multiple servers. Flexibility and scalability is achieved by running each layer on a different server.

6.2. <u>Pattern Name:</u> Controller Pattern

<u>Problem:</u> What first object beyond the UI layer receives and coordinates ("controls") a system operation (Take Picture)?

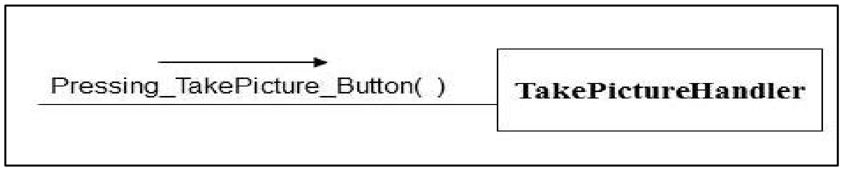
<u>Solution:</u>



Whenever an input system event is generated by pressing the TakePicture Button which is in the **View** of the system, request for the processing will be made to the **Controller** which in this scenario is TakePictureHandler - a usecase controller class. TakePictureHandler will then forward this request to the TakePicture class which is **Model** in this case, will then process the request using its functions and data. After processing, Model will return the result to view which in this case will be an image.

Processing in the Model includes opening of the camera feature of mobile, adjusting the image if required and then capturing it.

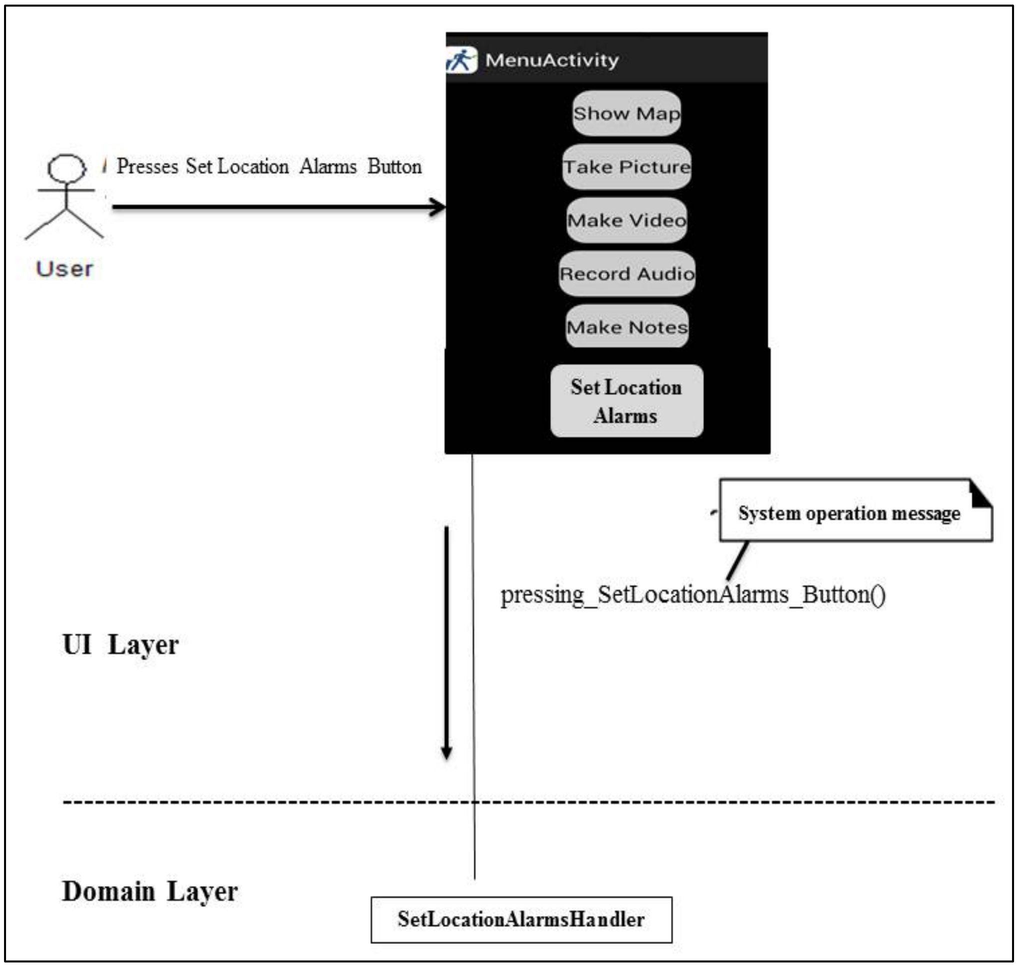
By using this Controller pattern, interface logic will be separated from the business logic .It will increase the potential for reuse and pluggable interfaces. It also provides an opportunity to reason about the state of the use case which is done by the controller which allows the state information to be saved in the usecase handler. It provides low coupling, high cohesion and high reusability.



6.3. **Pattern name:** Controller Pattern

Problem: What first object beyond the UI layer receives and coordinates ("controls") a system operation (Set Location Alarms)?

Solution:



Whenever an input system event is generated by pressing the SetLocationAlarms Button which is in the **View** of the system, request for the processing will be made to the **Controller** which in this scenario is SetLocationAlarmsHandler - a usecase controller class. SetLocationAlarmsHandler will then forward this request to the SetLocationAlarms class which is **Model** in this case, will then process the request using its functions and data. After processing, Model will return the result to view which in this case will be an image.

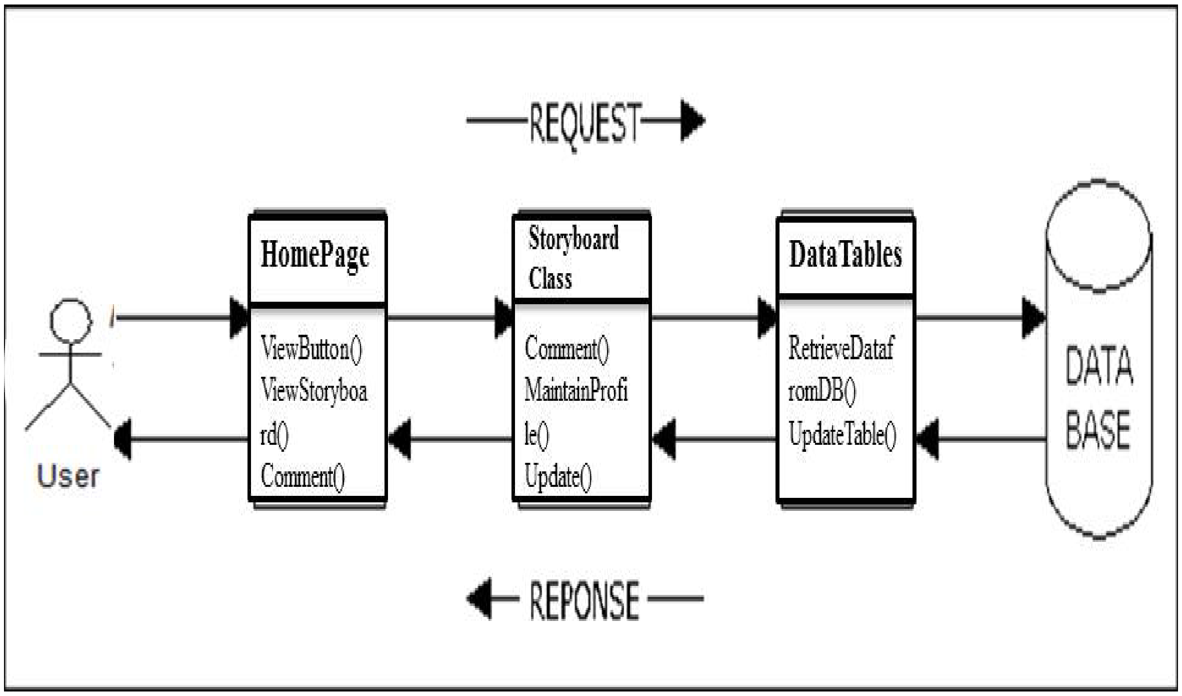
Processing in the Model includes taking a location from the user, locating it on the map, taking time for the alarm and setting it to beep at that time.

By using this Controller pattern, interface logic will be separated from the business logic .It will increase the potential for reuse and pluggable interfaces. It also provides an opportunity to reason about the state of the use case which is done by the controller which allows the state information to be saved in the usecase handler. It provides low coupling, high cohesion and high reusability.

6.4. <u>Pattern name:</u> 3-tier Pattern

<u>Problem:</u> What first object beyond the UI layer receives and coordinates a system operation (View Storyboard)?

<u>Solution:</u>



Whenever an input system event is generated by pressing the ViewStoryboard Button which is in the **Presentation Layer** of the system, request for the processing will be made to the **Business Layer** which in this scenario is Storyboard class. This layer will then require the authenticated users from the database so the process will enter the **Data Access Layer** which in this case is the tables in the database. This will use the database and retrieve all the data from the database tables.

By using 3-tier architectural design pattern, clear separation of user-interface-control and data presentation from application-logic is obtained. Change in business logic won't need change in other layers. Also, Dynamic load balancing can be done by use of multiple servers. Flexibility and scalability is achieved by running each layer on a different server.

7. References

[1] IEEE Standard SRS Template
[2] www.processimpact.com
[3] http://www.utd.edu/~chung/RE/NFR-18-4-on-1.pdf
[4] https://www.student.cs.uwaterloo.ca/~se463/Examples/SRS-Alex-Kalaidjian.pdf
[5] http://www.threesl.com/pages/webletterFebruary06/Non_Functional_Requirements.php
[6] https://play.google.com/store/apps/details?id=gaugler.backitude
[7] https://play.google.com/store/apps/details?id=se.bjuremo.hereiam.lite
[8] https://play.google.com/store/apps/details?id=silvertech.LocationAlarm&hl=en
[9] https://play.google.com/store/apps/details?id=igost.presents.locationalert&hl=en
[10] http://www.studymode.com/essays/Conclusion-For-Benifits-Of-Using-a-1093275.html
[11] https://sites.google.com/site/facebooktlc/what-is-facebook-1
[12] https://play.google.com/store/apps/details?id=gaugler.backitude&hl=en

Appendix A: Glossary

Abbreviation	**Complete**
SDS	Software Design Specification
Wi-Fi	Wireless Fidelity
DBMS	Database Management System
GPS	Global Positioning System
APP	Application
API	Application Programming Interface
HTTP	Hyper Text Transfer Protocol
OS	Operating System
COTS	Component off the shelf
APK	Android Package Kit
HTML	Hyper Text Markup Language