PEER TELL APPLICATION

(A DISCUSSION POINT FOR SDSU CS DEPARTMENT)

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Veenu Prajapat
Spring 2016
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Veenu Prajapat:

Peer Tell Application

(A Discussion Point for SDSU CS Department)

__________________________
Carl Eckberg, Chair
Department of Computer Science

__________________________
Wei Wang
Department of Computer Science

__________________________
J. Carmelo Interlando
Department of Mathematics and Statistics

April 6, 2016
Approval Date
Copyright © 2016
by
Veenu Prajapat
All Rights Reserved
ABSTRACT OF THE THESIS

Peer Tell Application
(A Discussion Point for SDSU CS Department)

by

Veenu Prajapat

Master of Science in Computer Science
San Diego State University, 2016

Throughout the ages, knowledge has always been commercialized. We pay a cobbler for his knowledge on how to repair a shoe; an electrician makes a living by trading his expertise on how to fix electrical appliances and so on. Human race as a unit is full of knowledge and yet we accomplish little with it. This is changing with the emergence of online knowledge marketplaces. These knowledge markets provide a mechanism for distributing knowledge resources. Some of the popular ones are Quora, Answers, StackOverflow. They provide a place where people can post their queries; which gets resolved by a community of knowledgeable and experienced regular users. Questions and answers together with a capable search engine enables a community member to be a knowledge-giver or a knowledge-seeker. This phenomenon of shared intelligence that emerges from the collaboration is called collective intelligence.

Peer Tell, an android based mobile application that provides a platform to bring students together based on a field of study to discuss their issues/findings and suggestions on various topics. This thesis is based on research on information discovery, knowledge market and how Q&A Expert Systems are built. The application includes all broad categories of subjects taught in the Computer Science Department. Students and faculties of the department can respond to the questions posted by peers. All discussions are tagged appropriately to help in future topic searches. Therefore, the app serves as a common floor for the students where all their discussions can take place.
7 FUTURE ENHANCEMENTS .................................................................54
  7.1 Enhanced Search .......................................................................54
  7.2 More Powers to Admin ...............................................................54
  7.3 Reply to an Answer ....................................................................54
  7.4 Editing of a Question/Answer ....................................................55
  7.5 Parse ..........................................................................................55
  7.6 For iOS and Tablets ....................................................................55
  7.7 Finger Print Login ......................................................................55
  7.8 Voice Responses .........................................................................56
  7.9 Voice to Text Conversion ...........................................................56
REFERENCES ......................................................................................57
LIST OF TABLES

Table 4.1. UserData table captures profile information of all PTA’s users..........................31
Table 4.2. UserNotification table captures activities for which a user has to be notified.................................................................31
Table 4.3. UserSubscription table captures users’ subscriptions.................................32
Table 4.4. Category table provides the list of discussion categories of PTA .................32
Table 4.5. Tags table captures keywords that help in search........................................32
Table 4.6. QuestionTags tells which tag is married to which question.........................32
Table 4.7. SecurityQues holds all security questions which are asked to an admin user ....32
Table 4.8. Questions table captures all questions created using the PTA......................33
Table 4.9. Answers table captures all answers posted to a question............................33
Table 4.10. AnswerRating captures users’ reactions to an answer..............................33
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Data, Information, Knowledge and Wisdom</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Image attached with user’s question</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>Image attached with Answer 2</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Android Architecture</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Android Application Framework</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Android Versions</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>PTA’s Application class</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>Manifest file of Peer Tell Application</td>
<td>12</td>
</tr>
<tr>
<td>2.6</td>
<td>Navigation from Login screen to ‘Secret’ screen or ‘Main’ screen</td>
<td>13</td>
</tr>
<tr>
<td>2.7</td>
<td>Button created by drag and drop</td>
<td>14</td>
</tr>
<tr>
<td>2.8</td>
<td>Creating button programmatically</td>
<td>14</td>
</tr>
<tr>
<td>2.9</td>
<td>R.java having constants for each resource under ‘layout’ folder</td>
<td>14</td>
</tr>
<tr>
<td>2.10</td>
<td>Peer Tell Application’s skeleton</td>
<td>15</td>
</tr>
<tr>
<td>2.11</td>
<td>Accessing UI elements of ‘activity_sign_up.xml’ of PTA programmatically</td>
<td>16</td>
</tr>
<tr>
<td>2.12</td>
<td>Category object on PTA side</td>
<td>18</td>
</tr>
<tr>
<td>2.13</td>
<td>Category Table on Parse server</td>
<td>18</td>
</tr>
<tr>
<td>2.14</td>
<td>Code to show how PTA sends images to Parse server for storage</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>Agile Development Model</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Check for Admin Login</td>
<td>23</td>
</tr>
<tr>
<td>3.3</td>
<td>Admin Powers</td>
<td>24</td>
</tr>
<tr>
<td>3.4</td>
<td>Posting a question</td>
<td>25</td>
</tr>
<tr>
<td>3.5</td>
<td>Deleting a question</td>
<td>26</td>
</tr>
<tr>
<td>3.6</td>
<td>Posting an answer</td>
<td>27</td>
</tr>
<tr>
<td>3.7</td>
<td>Deleting an answer</td>
<td>28</td>
</tr>
<tr>
<td>3.8</td>
<td>Search</td>
<td>29</td>
</tr>
</tbody>
</table>
Figure 3.9. Web Search........................................................................................................30
Figure 4.1. Shows relationship between questions, answers and tags tables..................34
Figure 4.2. Shows relationship between users, questions, categories tables ..................35
Figure 5.1. When user logins ............................................................................................37
Figure 5.2. When user logs out .........................................................................................37
Figure 5.3. When it’s a returning user, skip ‘Login’ screen ............................................38
Figure 5.4. When it’s not a returning user, user has to login ...........................................39
Figure 5.5. Student Login .................................................................................................40
Figure 5.6. Admin Login ..................................................................................................40
Figure 5.7. Code to load list of discussion categories......................................................41
Figure 5.8. Add a question ...............................................................................................42
Figure 5.9. Before saving a question check if the tag exists in the ‘Tags’ table or not ......42
Figure 5.10. Question-Answers screen view when admin logins .....................................43
Figure 5.11. Question deletion logic .................................................................................44
Figure 5.12. Answer deletion logic ..................................................................................45
Figure 5.13. Notification Screen ......................................................................................46
Figure 5.14. Notification Screen Logic ............................................................................47
Figure 5.15. Pick image from gallery ...............................................................................49
Figure 5.16. Send mail notifications ................................................................................50
Figure 5.17. Notification Menu UI ..................................................................................51
Figure 6.1. Shows the application flow from Home screen till Category Screen ............52
Figure 6.2. Shows the application flow from Category Screen till end ............................53
LIST OF ACRONYMS

PTA  Peer Tell Application
UI   User Interface
API  Application Programming Interface
BaaS Backend as a Service
MBaaS Mobile Backend as a Service
SIS  Social Information Seeking
VM   Virtual Machine
ACKNOWLEDGEMENTS

I’d like to thank my committee members who were more than generous with their expertise and precious time. Thank you Professor Dr. Wei Wang and Professor Dr. Carmelo Interlando for agreeing to serve on my committee. I would like to acknowledge and thank my advisor and thesis chair Professor Dr. Carl Eckberg for his invaluable guidance and support.

I’d also like to thank my family and friends that assisted me with beta testing of the application. Their excitement and willingness to provide feedback made the completion of this research an enjoyable experience.
CHAPTER 1

INTRODUCTION

1.1 DATA, INFORMATION AND KNOWLEDGE

Data is the facts of the world. It is limitless, omnipresent and is the least abstract form. Human mind perceives these facts via senses, aggregates them and form information that helps an individual make decisions [1].

Information is the answer to a question of some kind. Information expands our knowledge beyond what our senses comprehend. Since information captures data at a single point, it is not always accurate. Information matched with incorrect dataset might result in erroneous conclusions. For example: If I am 20 years old in year 2016 (data), then information captured timely (i.e. in the same year-2016) will tell my birth year to be 1996 but if captured in year 2026 then it calculates my birth year as 2006. A piece of information is considered valuable only if it helps us make decisions and affect behavior/outcome of an event [1].

“Knowledge is a familiarity, awareness or understanding of someone or something, such as facts, information, descriptions, or skills, which is acquired through experience or education by perceiving, discovering, or learning. Knowledge is what we know. Knowledge can refer to a theoretical or practical understanding of a subject and is derived from extensive amounts of experience dealing with the information on that subject. It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject); it can be more or less formal or systematic” [2].
Figure 1.1. Data, Information, Knowledge and Wisdom. Source: [3]

1.2 KNOWLEDGE MARKET

A knowledge market is an instrument for distributing knowledge resources. It exists in two forms based on monetization of knowledge. In commercial knowledge market, as the name suggests there is a price for knowledge. In non-commercial version, knowledge is free to access and motivation behind distribution of knowledge is philanthropy.

Peer Tell application was envisioned as a non-commercial knowledge market for SDSU students. It provides a platform where information and knowledge are shared freely and are available on demand. It offers knowledge seekers, the freedom to investigate at their own pace, fostering a non-threatening environment where mistakes are accepted and learned from.

1.3 NEED FOR INFORMATION

Curiosity is an innate quality of human beings. Our curious nature helps us see new worlds and possibilities, which are usually obscure to us. In order to explore new things or to see current things differently, we need to gain information about them; which will in turn satisfy our conscious or unconscious need to broaden our knowledge.
“An information need is a recognition that our knowledge is inadequate to satisfy a goal that we have” [4]. We need information to increase our understanding and decrease uncertainty. The more uncertain an event is the more information is needed to resolve it.

Types of information need are as follows [5]:

- **Problem information** gives us an overview of the problem. For example, to design a car, we must first know how many people it can seat, what is the expected acceleration and what is the typical environment in which the car is supposed to operate. Answers to these questions will constitute problem information.

- **Domain information** consists of known facts, concepts, laws and theories in the domain of the problem. Taking the previous example of car design, domain information will typically include knowledge of relationship between engine sizes and torque.

- **Problem-solving information** focuses on how to approach the problem. It describes how a problem should be seen, formulated and broken in logical sub-units. It then uses problem and domain information pertaining to each sub-part to solve it. This type of information comes from experience. Moving further in our car design example, knowledge of impact of various engine configurations on maximum power, torque and fuel efficiency; pros and cons of tire sizes and pattern, will be part of problem-solving information.

1.4 **What is Information Seeking?**

“Information seeking is a conscious effort to acquire information in response to a need or gap in your knowledge. But human information seeking is not so simple. It is neither straightforward nor typically complete; it is more like a series of interruptions, punctuated by other interruptions” [4].

It is a convoluted process of acquiring knowledge through thought, experience and senses. It stems from a deep desire to explore the unknown and enriching one’s experience and knowledge. Simply put, information seeking involves the search, retrieval, recognition, and application of meaningful content [6].

1.4.1 **Social Information Seeking/Social Search**

“The term Social information seeking (SIS) or Social search refers broadly to the use of social mechanisms to find information online. SIS can involve the use of search engines if the engine indexes social media (e.g., public Twitter posts) or uses community member
actions to rank results” [7]. It is a specific field of information seeking where the seekers look to gain knowledge from social sites/application, which are driven by a community.

In present day online non-commercial knowledge markets are a huge success and command a colossal following. The number of questions answered on QA sites far exceeds the number of questions answered by library reference services. For example, Stack overflow has 1.3 million users and Quora has about 265 thousand users. Popularity of these portal can be gauged by their content size. Stack overflow has 3.45 million questions and over 6.86 million answers while Quora has 437 thousand questions with 979 thousand responses. The very first QA website was Korean Naver Knowledge iN which was launched in 2002. While the first English website ‘Answerbag’ soon followed and was launched in April 2003 [8].

1.5 Steps for Information Seeking

a) Initiation
   Initiation is the first step of information seeking. It is an effort to define the problem statement by understanding what a user needs to know. It consists of brainstorming and discussing the topic with others and researching further.

b) Selection
   In selection, an individual seeks to answer the question of ‘what to investigate’. He needs to understand what is known and what is unknown so as to focus his efforts to delve into the unknown.

c) Exploration
   In this stage seeker enhances his understanding on the problem and associated topics. He explores basic concepts and analyzes the information on the topic of interest. In this stage he might come across new information which is not consistent with his original knowledge which might lead to anxiety.

d) Formulation
   In formulation, the seeker starts evaluating the information gathered. He has to connect the dots between the new information and the existing one in order to get a clearer picture of what he needs to know. New perspectives emerge in this stage and hence further questions arise from more than just logical deductions.

e) Collection
   As the name suggests by ‘Collection’ stage the seeker should have already collected and organized the information so that he knows how any new incoming idea fits in the developing solution. He should have by now processed enough information to develop a good understanding of underlying concepts relevant to the problem.

f) Presentation
   This is the last stage of information seeking and hence is also referred as search
closure. The newly acquired knowledge may or may not conform with existing beliefs and hence can be a tool for resistance or assimilation [5].

1.6 How Peer Tell Application is Addressing the Problem Statement

Everyone desires knowledge. In order to gain knowledge, one has to go through information seeking and information retrieval phases. The Peer Tell application acts as a bridge between the students/faculties of the CS department who have knowledge and the students who need it (i.e. connects seekers to the knowledge-givers). It also acts as a platform where people with different perspectives come together to help and empower each other. The application’s focus is to resolve academic problems faced by students. It is accomplished by enabling students having doubts to reach their knowledgeable peers and professors. To make the experience lucid other features like a search engine and notifying a person when something of his interest is posted are implemented.

Let’s take a look at a concrete example where the application is serving as a knowledge market:

Assume a student wishes to develop a back-end server, but he is uncertain of what kind of back-end technologies and frameworks to use and whether to use a BaaS. He explores all the options by brainstorming and talking to people. This is the ‘Initiation’ phase of his information seeking process.

After going through all the available options, he decides to develop his own back-end server using Node.js and MongoDB (‘Selection’ phase). Now he decides to find a way for the front end to interact with the back-end and he uses Peer Tell application to post his doubts.

Following is an example of one of his doubts that can be posted to PTA:

**Question** "How to build restful APIs for the attached JSON image with node.js?"

With the question he attaches the Figure 1.2 image.
This is the 'Exploration' phase. At this point, other users of the application may review the question and post their replies (with or without an image).

A few examples of the replies to the above posted question can be:

**Answer 1** The JSON structure in the image suggests there can be 6 APIs for your application to implement.

- API to get all the locations, with its departments list
- API to add/drop a location
- API to add/drop a department
- API to get details of a particular location

**Answer 2** Sample code snippet is attached. It will give you an idea on how to work with your JSON structure. You can modify the table structure to include departments (Image in Figure 1.3 is attached to this answer).
Figure 1.3. Image attached with Answer 2

**Answer 3** Why don't you use back-end as a service. BaaS provides APIs to connect to your database.

**Answer 4** I don't think the Rest API has this ability. As far as I know you'll have to choose a different way, for example you can use the Soap API.

The app apprises the student who has posted the question about all new responses. Using the Peer Tell application, a user can also rate these replies to indicate the credibility of a response. Every effort is made in the app to weed out false and incomplete information by granting admin rights to dismiss solutions.

Now comes the 'Formulation' phase where the student evaluates every answer posted to his/her question using the Peer Tell application. In this phase, he decides what information is relevant to his question and what is not, and adds the relevant ones to his information bucket. Next comes the 'Collection' phase where the student fits the new information into the developing solution. He organizes everything to get the complete picture. There will be less uncertainty in seeker’s mind than in the earlier stages and this helps the seeker formulate his unique perspective, a vision that will guide his efforts to fruition. At this stage, the information gained becomes his/her personal knowledge.

At this point the student has understood the ways to make the front-end interact with back-end of the application using APIs. In the final phase of 'Presentation', the user puts his knowledge to use and his information seeking process comes to an end.
CHAPTER 2

TECHNOLOGIES

Several technological tools were used to implement this thesis application. Each is summarized in the sections that follow.

2.1 ANDROID

2.1.1 History

“Android is a mobile-based operating system that is based on a modified version of Linux. It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team). Google wanted Android to be open and free; hence, most of the Android code was released under the open source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code” [9]. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

2.1.2 Main Features

Features provided by Android includes but are not limited to [10]:

- Storage: Uses SQLite, a lightweight relational database for data storage.
- Connectivity: Supports GSM/EDGE, CDMA, Bluetooth, Wi-Fi, LTE etc.
- Messaging: Supports both SMS and MMS.
- Media support: Like MP3, MP4, WAV, JPEG, PNG, GIF etc.
- Hardware support: Like Camera, Digital Compass, GPS, various sensors etc.
- Supports multi-touch screens.
- Supports multi-tasking applications.
- Supports multiple languages.
2.1.3 Architecture

Android OS is divided in 5 sections (Figure 2.1) [9]:

- **Linux Kernel**: It’s the root of the Android architecture. It contains all the low-level device drivers for the hardware components of a device.

- **Libraries**: Contains all the code that provides Android OS’ main features. Examples of some native libraries are:
  - Webkit: Responsible for web support
  - SQLite: Responsible for database
  - Media: Responsible for playing audio and video contents

- **Android Runtime**: It includes Dalvik Virtual Machine that allows every application to run in its own process with its own instance of Dalvik VM.

- **Application Framework**: It exposes the capabilities of Android OS to the developers. Developers can use them to enhance their applications. Figure 2.2 gives brief description of some of the modules under this layer.

- **Application**: All applications that a developer creates are at this layer.

Figure 2.1. Android Architecture. Source: [11]
2.1.4 Android Versions Released till Oct 2015

Code-names for android versions range from A to M. The version names starting from the alphabet ‘C’ are based on a confectionery theme and also released in alphabetic order (Figure 2.3). The initial 2 releases were named after fictional robots "Astro Boy’, “Bender” and were released internally [13].

<table>
<thead>
<tr>
<th>Code name</th>
<th>Version number</th>
<th>Initial release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cupcake</td>
<td>1.5</td>
<td>April 27, 2009</td>
</tr>
<tr>
<td>Donut</td>
<td>1.6</td>
<td>September 15, 2009</td>
</tr>
<tr>
<td>Eclair</td>
<td>2.0–2.1</td>
<td>October 26, 2009</td>
</tr>
<tr>
<td>Froyo</td>
<td>2.2–2.2.3</td>
<td>May 20, 2010</td>
</tr>
<tr>
<td>Gingerbread</td>
<td>2.3–2.3.7</td>
<td>December 6, 2010</td>
</tr>
<tr>
<td>Honeycomb[a]</td>
<td>3.0–3.2.6</td>
<td>February 22, 2011</td>
</tr>
<tr>
<td>Ice Cream Sandwich</td>
<td>4.0–4.0.4</td>
<td>October 18, 2011</td>
</tr>
<tr>
<td>Jelly Bean</td>
<td>4.1–4.3.1</td>
<td>July 9, 2012</td>
</tr>
<tr>
<td>KitKat</td>
<td>4.4–4.4.4, 4.4W–4.4W.2</td>
<td>October 31, 2013</td>
</tr>
<tr>
<td>Lollipop</td>
<td>5.0–5.1.1</td>
<td>November 12, 2014</td>
</tr>
<tr>
<td>Marshmallow</td>
<td>6.0–6.0.1</td>
<td>October 5, 2015</td>
</tr>
</tbody>
</table>

Figure 2.3. Android Versions. Source: [13]

2.1.5 Application Fundamentals

“The Android SDK tools compile an application’s code along with any data and resource files into an APK: an Android package, which is an archive file with an ‘.apk’
extension. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app” [14]. Discussed below are some of the important aspects of any Android application development.

a) **Application Class:** It is the highest layer (as it preserves global state across the application) and Android creates an instance of this class when the application is started. It extends Android’s ‘android.app.Application’ class and is visible to all the modules within an application. Parse is initialized in this class (Figure 2.4). The application-id (that uniquely identifies an application on Parse server) and client key were provided when the database for PTA was created on Parse server.

```java
public class AppLaunch extends Application {
    private static final String APPLICATION_ID = "XH8tYPjRcnoQ4tRBXrI0reDQ23JJQ0B0yMsefi";
    private static final String CLIENT_KEY = "3jh0oy9B2Wc1f84IPSEpEAA4Uc2oSFJ1I20x";
    @Override
    public void onCreate() {
        // This is called before application starts to use the Parse library.
        Parse.initialize(this, APPLICATION_ID, CLIENT_KEY);
        super.onCreate();
    }
}
```

**Figure 2.4. PTA’s Application class**

b) **Manifest file:** Before Android OS can start an application, it needs to know all the components of the application. This information can be retrieved from the manifest file (called as ‘AndroidManifest.xml’). Figure 2.5 shows the content of the manifest file for PTA.
Figure 2.5. Manifest file of Peer Tell Application. The highlighted text in the figure tells which activity is the first screen a user sees when he opens the application.

This file also provides additional information like version details, API libraries linked, user permissions required (Ex: Internet access for the application).

c) **Back Stack:** It is created whenever an application starts. It keeps a history of the activities opened by the user in the order they are opened. The order is maintained and it is never re-arranged (only push and pop operations can be performed on it). Push happens when an activity is started and pop happens when the user leaves an activity.

d) **Application Components:** There are 4 major application components:
• **Services**: Services do not provide UI and are used for long background processes.

• **Content Providers**: A content provider is used to have a central repository for an application. It is primarily used by other applications.

• **Broadcast Receivers**: A broadcast receiver is a component that responds to system-wide broadcast announcements. Example: When a device’s battery is low then a broadcast message is sent out to all the open applications of the device.

• **Activities**: An activity represents a single screen with a UI. It handles all the user interactions. When a developer creates an activity, the following files are created:
  - Java class file (.java): Where all the actions are written.
  - Layout file (.xml): UI part of the activity, which is visible to the users.
  - Menu file (.xml): Where all menu buttons of the UI are placed.

Peer Tell Application has a total of 15 activities. *Intents* were used to navigate from one screen to another. Example: Upon successful login, the user is navigated to either ‘Secret’ activity (if admin tries to login) or to ‘Main’ activity of the application (Figure 2.6).

```java
Intent goToNext = new Intent();
if (session.isAdminLoggedIn()) {
    goToNext = new Intent(getApplicationContext(), SecretActivity.class);
} else {
    goToNext = new Intent(getApplicationContext(), MainActivity.class);
}
goToNext.putExtra("username", username.getText().toString());
goToNext.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(goToNext);
finish();
```

**Figure 2.6. Navigation from Login screen to ‘Secret’ screen or ‘Main’ screen**

• **Additional components**:
  - **Layouts**: Popular ones are the relative and linear layouts. Both were used in the Peer Tell application development.
  - **Views**: UI elements are drawn on screens which are visible to app users. For Peer Tell app `TextView` (readable text), `EditText` (editable text), `Button`, `WebView`, `ImageButton`, `Spinner` etc. widgets were used. Some were created by drag and drop action from the widgets pane (Figure 2.5). While others were created programmatically (Figure 2.6).
2.1.6 Android Anatomy of Peer Tell Application

An Application consists of different types of resources (like images, UI screens, menus). All of them are placed under the ‘res’ directory (Figure 2.10). For each type of resource, there is an R subclass present inside a resource class called ‘R.java’. When an application is compiled, ‘aapt’ generates the R class, which contains resource IDs for all the resources in application’s ‘res’ directory [10]. Every application has only one resource class present under ‘gen’ (auto-generated) directory.

For example, there is a constant for ‘layout’ type of resource and all files included in that folder have a static integer associated with it in the ‘R’ file (Figure 2.9). This integer is the resource ID that is used to retrieve the resource.

```java
public static final class layout {
    public static final int action_bar_notification_icon=0x7f033000;
    public static final int activity_about_app=0x7f030001;
    public static final int activity_add_ques=0x7f030002;
    public static final int activity_admin=0x7f030003;
    public static final int activity_discussions=0x7f030004;
    public static final int activity_home=0x7f030005;
    public static final int activity_image_view=0x7f030006;
    public static final int activity_im_menu=0x7f030007;
    public static final int activity_login=0x7f030008;
    public static final int activity_main=0x7f030009;
    public static final int activity_notification=0x7f03000a;
    public static final int activity_profile=0x7f03000b;
    public static final int activity_ques_view=0x7f03000c;
    public static final int activity_searchable=0x7f03000d;
    public static final int activity_secret=0x7f03000e;
    public static final int activity_sign_up=0x7f03000f;
    public static final int ques_ans_list=0x7f030010;
    public static final int spinner_item=0x7f030011;
}
```

Figure 2.9. R.java having constants for each resource under ‘layout’ folder
Figure 2.10. Peer Tell Application’s skeleton. Shows how all resources were distributed

PTA accesses the resources in figure 2.8 programmatically as follows:
2.2 Parse Framework

In today’s world there is a huge market for mobile applications and it is getting bigger with time. So it’s common for applications to require a web backend, which will serve as a central database. By consolidating data on cloud enables it to be shared among the users of the application. In order to develop the backend server, the developer needs to be well versed with technologies such as Ruby on Rails or PHP.

However, there are several challenges with developing backend on our own [15]:

- **Heavy Time Investment**: Developers invest humongous amount of time to develop front-end of the application. Devoting project resources towards backend exacts a heavy toll on project deadlines.

- **Heavy Skill Investment**: An experienced front-end iOS/Android developer doesn’t necessarily have skills required for developing back-ends. In most instances spending time and manpower to master requisite skills is not worth the investment.

- **Scalability Issues**: A developer cannot foresee the usage pattern of his application. If he develops backend on his own, he might err assuming specific traffic. It might cause issues like bloated prices or choked bandwidth. BaaS scales efficiently with usage and needs minimal developer intervention.

- **Security**: Developer might compromise security of user content by trying to develop his own backend. Most BaaS systems have industry grade security measures to protect against unauthorized accesses.

Luckily, these days several companies are now offering services called ‘Backend as a Service’ (or BaaS/MBaaS) to save developers’ time and money by providing backend cloud storage along with APIs to interact with it. Features provided by BaaS includes but are not limited to:

- User management
- Push notifications
- Integration with social networking services
Handy libraries to make the integration nice and easy.

BaaS forms a bridge to connect the front-end of an application and cloud-based back-end via a unified API and SDK.

### 2.2.1 Why Parse?

The Parse platform provides a complete backend solution for mobile and web applications. Its goal is to totally eliminate the need for writing server code or maintaining servers. Parse SDK is ready to use out of the box with minimal configuration on developer’s part [16].

The documentation provided by parse is straightforward and the service is one of the easiest to work with, especially for beginners. It is one of the fastest backend services; uploading and retrieving objects are quick. Developers can deploy custom ‘UIViews’ or refer to sample projects provided by Parse. It massively simplifies addition of features such as Facebook and Twitter login views in an application. There is a large collection of 3rd party libraries which can be used with the service. Parse also has multi-platform support and therefore application users based of different platform can interact and share data. Parse allows storing of different data formats like audio, video or image files. Hence Parse mitigates the timeline risks and decreases resources required [15].

How Parse features have and can benefit PTA [16]:

- **Custom objects:** The most important feature BaaS provides is the ability to store an application’s information into a database. Every object in PTA forms a row in one of the tables of the centralized database. For example: each discussion group object in the application (Figure 2.12) is stored as an entry in the ‘Category’ table (Figure 2.13).

  When an admin tries to add a new category, a new object is formed behind the scenes on the application side which is then saved as an entry on the Parse database.
Push notifications: Parse provides the ability to send push notifications to all or a subset of users. This feature will be helpful in implementing future enhancements like:

- Upon creating/removing a discussion group, an admin can notify everyone about the change.
- Admin can conduct an event/conference/class and notify a group of students.

File storage: In addition to storing plain old data, Parse allows developers to store files like images, large documents etc. In PTA, a user’s profile picture, images attached to questions/answers are stored in ‘Parse’. Below example (Figure 2.14) shows a snippet of code from the Peer Tell application where a user’s profile picture is compressed and sent to the Parse server. File saving can potentially take a lot of time, therefore it is done in a background thread and meanwhile a progress dialog is shown to the user.
// If user has picked an Image for profile
// Then picked image is compressed and converted to Byte Array
ByteArrayOutputStream stream = new ByteArrayOutputStream();
bitmap.compress(Bitmap.CompressFormat.PNG, 100, stream);
byte[] data = stream.toByteArray();
final ParseFile imgFile = new ParseFile(imgName, data);

// Saving ParseFile created from Byte-Array in Background.
imgFile.saveInBackground(new SaveCallback()
{
    public void done(ParseException ex)
    {
        if (ex == null) // If Image File saving is successful
        {
            currentUser.put("profilePic", imgFile);
            // Then save all other details of player in Parse
            currentUser.saveInBackground(new SaveCallback()
            {
                @Override
                public void done(ParseException arg0)
                {
                    // If progress-dialog is showing, dismiss it
                    if (pDialog.isShowing())
                        pDialog.dismiss();

                    moveToNextScreen();
                }
            });
        }
        else // If Image File saving is not successful
        {
            errorMessage.setText(ex.getLocalizedMessage());
        }
    }
});

Figure 2.14. Code to show how PTA sends images to Parse sever for storage
CHAPTER 3

DESIGN

3.1 SOFTWARE DEVELOPMENT MODEL FOR PTA

For a successful application, it’s important to adopt a software development life cycle that breaks down an application’s development into stages. For Peer Tell application, ‘Agile’ software development model was used. Agile Model is a type of incremental model as typical development of an application occurs in incremental and rapid cycles. In each cycle, a portion of the application was developed and tested to ensure software quality was not compromised. In the beginning of each sprint, planning was done to setup the targets and at the end of it, all new developments of the application were reviewed and tested by my thesis advisor, Professor Dr. Eckberg.

“Agile development promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change” [17]. Developers can continuously push improvements on ad hoc basis, which ultimately reduces the risk of encountering problems in later cycles of development. Resolving lots of accumulated bugs at the end of developing cycle is rather convoluted, and hence compliance with Agile adds value.

![Agile Development Model](image)

Figure 3.1. Agile Development Model.
Source: [18]
In each sprint, all the software development phases are executed. Planning for the current sprint is done in the design phase. All the features to be implemented in this sprint are discussed and some sort of screen layouts, pseudo code, design flows etc. are drawn for better understanding of the requirements.

Next is the development phase, in which input is taken from the design phase and developers start implementing code to accomplish functionality of a feature. Then that implemented feature is tested for bugs. If bugs are found, they’re resolved immediately. Upon successfully passing the test, the new feature/functionality is incorporated in the application and next sprint begins.

3.2 OVERVIEW OF THE PEER-TELL APPLICATION

The Peer Tell mobile based application provides a platform for students of SDSU-CS department to discuss their issues/findings and suggestions on various topics. The application includes all broad categories of subjects taught in the Computer Science department. Only the admin user is authorized to make any kind of update to these discussion categories. Additionally, it has the following features:

- Students can subscribe to a particular discussion group, which will keep them updated on every ongoing discussion in that group.
- Pictures can be attached to a problem.
- Students posting questions are suggested keywords to be tagged for better search.
- Users of the application can maintain their profile, subscriptions and decide whether if they want to be notified by E-mail for an event of their interest.
- Users can also E-Mail a discussion to other people.

3.3 LOGIN

In order to login, users have to first register themselves with the app. Following are the mandatory fields for signing up:

- Student-Id: The Red-Id given to students when they enroll to SDSU.
- Password: Password needed to login after the account is created.
- Email-Id: Mail-Id to notify users when there is a new activity on a question they have posted or a group they’re subscribed to updates.

After successful registration, students can login to the app using those credentials. Chapter 6 shows how the application chooses different path based on the type of user.
3.3.1 User Login

When a student/faculty provides valid combination of Red-ID and password on login, they land at the ‘Category’ screen which shows different groups of discussions. Following features are available to every user:

- Subscribe/Unsubscribe to a group
- Post a question with or without an image
- Post a reply to an available question
- Remove a question/answer they’ve posted
- Rate an answer
- Maintain profile
- Search web or SDSU CS’s important links
- Notification center
- Getting notifications via email
- E-mail a discussion

3.3.2 Admin Login

When an authorized entity (such as a professor/dean of the CS department) tries to login, there is an additional security checkpoint. An admin must answer one of the security questions in order to access his admin rights. Upon successfully answering the security question, the main activity (i.e. ‘Category’ screen) shows up with the additional feature of adding/removing a category from a list of categories (Read further in Section 3.5). Admin is also authorized to delete any question/answer that is inappropriate in his judgment.
3.4 Subscription

Users can subscribe to discussion groups of their interest. Once subscribed they receive notifications about all the new questions added to that group. A user is by default subscribed to a question he/she has posted. Notifications from a subscribed group or a subscribed question can be viewed from ‘Notification Center’. Users can unsubscribe from a discussion group by hitting ‘Unsubscribe’ button in ‘Profile’ screen or ‘Questions’ screen.

3.5 Notification Center

The notification center is a medium through which students are informed about all the activities related to questions they have posted, so that they can actively discuss the issues they’re facing. The notification center is also helpful as it notifies to professors or students who have expertise in the groups they’re subscribed to. They can pitch in to provide important insights and directions. If users have subscribed for email notification, then all events of their interest are emailed to them in real-time.
3.6 Category List Update (Admin Only)

The main activity or the ‘Category’ screen shows all the discussion groups by name. It is a dynamic list, so the list is updated every time the admin adds/drops a category from the list. The screen is refreshed on every update to the list. Whenever a category is deleted, all the dependent objects (like all the corresponding questions and answers in that discussion category) are also dropped too. One must be careful with this procedure, as the deleted content is not recoverable. Figure 3.3 shows application’s view for add/drop feature.

3.7 Profile

Users can access their account details through ‘Profile’ view. They can update their profile information like profile picture, group subscriptions, password and email id from
there. Once they’re done editing, they must click the ‘Update’ button on the screen in order to save the updated details to the server. Only information that is not permitted to be changed is the student id. Profile screen also provides ‘Delete’ option to the users; in case they have graduated from college and are no longer using their account.

3.8 Questions Under a Category

3.8.1 Post a Question

A student determines which category best represents his query and then proceeds to post it there. Following attributes are associated with each question:

- Subject: Brief description of the problem
- Tag: Keyword that assists in searching for a problem
- Attachment: An image can be attached to a question (optional)

Users have a choice to pick an existing tag from a drop-down or create a new tag while posting a question. Figure 3.4 describes the process flow of posting a question.

![Diagram](image-url)

**Figure 3.4. Posting a question**
3.8.2 Remove a Question

A question has a delete option enabled only if the student who is trying to delete has posted the question or if he/she is an admin. On deleting a question, all of its associated notifications, tags and answers are deleted as well. That is why this activity should be thoughtful and utmost care is recommended. Figure 3.5 describes the logical decisions made on the deletion path of a question.

Figure 3.5. Deleting a question
3.9 Answers to a Question

3.9.1 Post an Answer

When a student or a professor can assist on a question, he/she can do so by posting a reply. While posting a reply he/she can attach an image if that helps in resolving the problem or to reach the audience better. The answer posted can be liked/dis-liked by the user community, which can help students to assess the reliability and usefulness of a particular answer.

Figure 3.6. Posting an answer

3.9.2 Remove an Answer

Students have the flexibility of deleting erroneous or duplicate responses posted by them. Additionally, admin is also authorized to delete answers that is inappropriate in his/her discretion.
Allowing a user to delete an answer helps them cover their accidental mistakes. This feature also helps an admin to combat redundancy and maintain decorum in the forum.

### 3.10 Search

#### 3.10.1 Search Questions

The application allows a user to search for questions. The searched text is matched against all the question tags. This will help users to find what exactly they’re looking for. Search capability is very important to reduce a user’s frustration and give him a better app experience. He can save a lot of time by looking into the questions related to his interest rather than to go over the whole history, which given a certain size can be a formidable task.

Figure 3.8 shows how PTA navigates the user from the ‘Category’ activity to ‘Search’ activity on execution of a search operation.
3.10.2 Search Web

Additionally, to search for a resolution to a question, the application also gives users access to the web. Menu item ‘Links’, allows the user to access SDSU’s Computer Science department web pages so that the user doesn’t have to venture outside the app.

Figure 3.9 shows the tabs: ‘New’, ‘Current’ and ‘Faculty’ available on application, which allows a user to access SDSU CS department’s important links. It also provides a text-box to visit any web page of interest.
Figure 3.9. Web Search
CHAPTER 4

DATABASE DESIGN

1. **UserData**: This table stores details of all registered users of PTA.

Table 4.1. UserData table captures profile information of all PTA’s users

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UserID/RedID</td>
<td>Unique ID for Student/Faculty</td>
<td>Number</td>
<td>Primary</td>
</tr>
<tr>
<td>2</td>
<td>Password</td>
<td>Password for login</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Email</td>
<td>User’s Email address</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ProfilePic</td>
<td>User’s Picture</td>
<td>File</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SendMail</td>
<td>Should a user receive notification via Email</td>
<td>Boolean</td>
<td></td>
</tr>
</tbody>
</table>

2. **UserNotification**: This table holds notifications to be sent out in real-time to the users about new activities in groups of their interest.

Table 4.2. UserNotification table captures activities for which a user has to be notified

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UserID/RedID</td>
<td>Unique ID for Student/Faculty</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>2</td>
<td>QuesID</td>
<td>Unique Question ID</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>3</td>
<td>CategoryID</td>
<td>Category to which question is posted</td>
<td>Number</td>
<td>Foreign</td>
</tr>
</tbody>
</table>
| 4     | Ques_Ans    | True = Notification about the subscribed group activity 
False = Notification about the user posted question | Boolean |         |
| 5     | Read        | Once a user views the new notifications, they become read and they are no longer notified about them. | Boolean | T = read
F = unread |

3. **UserSubscription**: Tracks a user’s subscriptions. A user’s profile has details about his/her subscriptions, which is populated from this table’s data.
Table 4.3. UserSubscription table captures users’ subscriptions

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UserID/RedID</td>
<td>Unique ID for Student/Faculty</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>2</td>
<td>CategoryID</td>
<td>Subscribed discussion group</td>
<td>Number</td>
<td>Foreign</td>
</tr>
</tbody>
</table>

4. **Category**: This table populates ‘Category’ screen of the application. Lists all the broad categories of courses taught in the CS department.

Table 4.4. Category table provides the list of discussion categories of PTA

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CategoryID</td>
<td>Unique Category of Discussion</td>
<td>Number</td>
<td>Primary</td>
</tr>
<tr>
<td>2</td>
<td>CategoryName</td>
<td>Category Name</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

5. **Tags**: Records all tags created using PTA.

Table 4.5. Tags table captures keywords that help in search

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TagID</td>
<td>Unique Tag ID</td>
<td>Number</td>
<td>Primary</td>
</tr>
<tr>
<td>2</td>
<td>TagName</td>
<td>Tag</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

6. **QuestionTags**: It stores association of a question with a tag. It aids in searching questions using tags.

Table 4.6. QuestionTags tells which tag is married to which question

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QuesID</td>
<td>Question ID</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>2</td>
<td>TagID</td>
<td>Tag ID</td>
<td>Number</td>
<td>Foreign</td>
</tr>
</tbody>
</table>

7. **SecurityQues**: This table stores all the security questions and answers which are asked when an admin tries to login. An admin has to answer one of the security questions successfully in order to use his/her admin powers.

Table 4.7. SecurityQues holds all security questions which are asked to an admin user

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SecurityQuesID</td>
<td>Question ID</td>
<td>Number</td>
<td>Primary</td>
</tr>
<tr>
<td>2</td>
<td>Question</td>
<td>Security Question</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Answer</td>
<td>Security Answer</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

8. **Questions**: Keeps track of all the questions posted using the app.
Table 4.8. Questions table captures all questions created using the PTA

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CategoryID</td>
<td>Category ID</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>2</td>
<td>QuestionID</td>
<td>Question ID</td>
<td>String</td>
<td>Primary</td>
</tr>
<tr>
<td>3</td>
<td>Subject</td>
<td>Brief description of the question</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Topic</td>
<td>Actual Question</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Attachment</td>
<td>Image file attached to the question</td>
<td>File</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>UserID/RedID</td>
<td>User posting the question</td>
<td>String</td>
<td>Foreign</td>
</tr>
<tr>
<td>7</td>
<td>Timestamp</td>
<td>Timestamp of the question</td>
<td>Date</td>
<td></td>
</tr>
</tbody>
</table>

9. **Answers**: Stores all the replies sent to a question using the app.

Table 4.9. Answers table captures all answers posted to a question

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CategoryID</td>
<td>Category ID</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>2</td>
<td>QuestionID</td>
<td>Question ID</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>3</td>
<td>AnswerID</td>
<td>Unique Answer ID to question</td>
<td>Number</td>
<td>Primary</td>
</tr>
<tr>
<td>4</td>
<td>Answer</td>
<td>Actual answer</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>UserID/RedID</td>
<td>User responding</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>6</td>
<td>Timestamp</td>
<td>Timestamp of the answer</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Attachment</td>
<td>Image file attached to an answer</td>
<td>File</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Rating</td>
<td>Total likes of an answer</td>
<td>Number</td>
<td></td>
</tr>
</tbody>
</table>

10. **AnswerRatings**: Tracks user reactions towards a response.

Table 4.10. AnswerRating captures users’ reactions to an answer

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Field Name</th>
<th>Purpose</th>
<th>Type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AnswerId</td>
<td>Answer to which user is reacting</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>2</td>
<td>UserID</td>
<td>User doing the action</td>
<td>Number</td>
<td>Foreign</td>
</tr>
<tr>
<td>3</td>
<td>LikeCount</td>
<td>1= like, 0=neutral, -1=dislike</td>
<td>Number</td>
<td></td>
</tr>
</tbody>
</table>
4.1 ER Diagram

Figure 4.1. Shows relationship between questions, answers and tags tables.
Figure 4.2. Shows relationship between users, questions, categories tables
CHAPTER 5

IMPLEMENTATION

5.1 MAINTAINING A SESSION

When a user tries to login to the application, a file is created which maintains the current user’s login details. This file is accessed whenever the app is launched, to check if it is a returning user (a user who logged in but never logged out). A returning user doesn’t need to put in his/her login credentials again as session already holds that information (Read further in Section 5.2).

“Shared preferences are files on our file system that we read and edit using the ‘SharedPreferences’ class. An instance of ‘SharedPreferences’ acts like a key-value store, much like Bundle, except that it is backed by persistent storage. The keys are strings, and the values are atomic data types. If we look at them we will see that the files are simple XML, but shared preferences make it easy to ignore that implementation detail” [19]. Figure 5.1 shows how shared preferences were used in PTA to maintain a user’s session.
Whenever a user logs out from the application, the shared preferences are cleared (Figure 5.2) and he/she is navigated to the 'Home' (first screen) of the application.

```java
private SharedPreferences pref;
protected Editor editor;
private Context context;

// Shared preferences file name
private int PRIVATE_MODE = 0;
private static final String PREF_NAME = "LastLoginDetails";
private static final String IS_LOGGEDIN = "isloggedin";
public static final String KEY_USERNAME = "username";
public static final String KEY_PASSWORD = "password";
public static final String KEY_MAIL = "mail";
public static final String KEY_SEND_MAIL = "sendMail";
public static final String ADMIN = "10:000:011";

// Constructor
public SessionManager(Context context)
{
    this.context = context;
    this.pref = context.getSharedPreferences(PREF_NAME, PRIVATE_MODE);
    this.editor = pref.edit();
}

/*@ Method Name: createLoginSession
Arguments: String, String, String, boolean
Description: Creates session with user credential and users mail related details. */
public void createLoginSession(String name, String password, String mail, boolean sendMail)
{
    // Storing login value as TRUE & storing user details
    editor.putBoolean(IS_LOGGIN, true);
    editor.putString(KEY_USERNAME, name);
    editor.putString(KEY_PASSWORD, password);
    editor.putString(KEY_SEND_MAIL, sendMail);
    editor.putString(KEY_MAIL, mail);

    // commit changes
    editor.commit();
}

Figure 5.2. When user logs out

Whenever a user logs out from the application, the shared preferences are cleared (Figure 5.2) and he/she is navigated to the 'Home' (first screen) of the application.
5.2 **Sign-Up & Log-In**

In order to use the app, users have to create an account with their Red-ID and email. Once registered, they can login to the application with those credentials. The following validation checks are done at the time of registration and log in process:

- Red-ID / Password is missing
- Red-ID should be between 99999999 and 1000000000 (should be 9 digits)

When a user opens the application, the session details are fetched first and checked if it is a returning user (Figure 5.3).

```java
/*
 Method Name: checkLogin()
 Arguments: None
 Description: Called to check if user is logged-in or Not.
 If user is logged in previously, then redirect him to Main Activity
 with user-name passed via Bundle
 */

public void checkLogin()
{
    // Check login status
    if(this.isLoggedin())
    {
        Intent goToNext;
        // Check is Admin trying to login
        if(pref.getString(KEY_USERNAME, null).equalsIgnoreCase(ADMIN)) {
            goToNext = new Intent(_context, SecretActivity.class);
        } else {
            goToNext = new Intent(_context, MainActivity.class);
        }
        goToNext.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        goToNext.putExtra("userInput", pref.getString(KEY_USERNAME, null));
        _context.startActivity(goToNext);
    }
}
```

**Figure 5.3. When it’s a returning user, skip ‘Login’ screen**

If a user is not a returning user then the function ‘checkLogin()’ in figure 5.3 does nothing and the user has to login again. Upon successful login, session is set with current user’s information (Figure 5.4).
Figure 5.4. When it’s not a returning user, user has to login

5.3 LOADING DISCUSSION CATEGORIES

After successfully logging in, the landing page (‘Category’ screen) shows a list of discussion categories that are generated from the “Category” table. Fetching discussion group-names from the database allows the admin to control addition and removal of a category as required.

The landing page UI of the application varies for a ‘User’ (Figure 5.5) and ‘Admin’ (Figure 5.6) login. Although they have lots of common features like:

- **Links:** To access web and to visit SDSU CS web pages
- **Search:** To search for questions by a tag
- **Profile:** For a user to view his/her account details
- **Notifications:** To get all the notifications related to activities on questions posted by them or activities on subscribed groups.

The common features can be seen in both the figures 5.5 & 5.6. Additional menu options available for the admin are: ‘Add’ and ‘Drop’ a category (Figure 5.6). Both features affect the ‘Category’ Table.
The following function ‘getDataFromParseAndBuildView()’ (Figure 5.7) is called on the load of ‘Category’ screen to wipe out all the previous views and generate a new view for each ‘categoryName’ value (in ‘Category’ table) mapped to a button on UI.
private void getDataFromParseAndBuildView()
{
    categories.clear();
    // Clearing out activity's layout
    final LinearLayout layout = (LinearLayout) findViewById(R.id.mainLayout);
    layout.removeAllViews();
    final LayoutInflater lp = new LayoutInflater(this);
    layout.setOrientation(LinearLayout.VERTICAL);

    // Querying Parse Table 'Category'
    ParseQuery<ParseObject> query = ParseQuery.getQuery("Category");
    query.orderByAscending("categoryName");

    // Showing ProgressDialog until fetch from Parse is complete
    final ProgressDialog pDialog = new ProgressDialog(MainActivity.this);
    pDialog.setMessage("Loading Categories");
    pDialog.setCancelable(false);
    pDialog.show();

    query.findInBackground(new FindCallback<ParseObject>() {
        @Override
        public void done(List<ParseObject> categoryList, ParseException e) {
            // Hide ProgressDialog if showing
            if (pDialog.isShowing())
                pDialog.dismiss();

            if (e == null) // If there is no exception raised
            {
                for (ParseObject category : categoryList) {
                    // For each category, create a button
                    // * which allows access to discussions of that category
                    int categoryId = category.getInt("categoryId");
                    String categoryName = category.getString("categoryName");
                    Button btn = new Button(getApplicationContext());
                    btn.setText(categoryName);
                    btn.setTextColor(Color.BLACK);
                    btn.setID(categoryId);
                    btn.setOnClickListener(new OnClickListener()
                        {
                            public void onClick(View v)
                            {
                                int position = categoryList.size() -
                                    1 - categoryList.indexOf(category);
                                getDiscussionViewOfCategory(btn);
                                layout.addView(btn, lp);
                                categories.put(btnName, btnId);
                            }
                        });
                }
            }
        }
    });
}

Figure 5.7. Code to load list of discussion categories

5.4 LOADING DISCUSSIONS OF A CATEGORY

Selecting a category loads a list of discussions in that category. In order to know
which category’s discussions are to be retrieved from the database, the category-id is passed
from one activity to another via ‘Bundle’ object. On the receiving activity, this id is used to
query ‘Questions’ table to get the list of discussions. To add a new discussion to the list, the
user can hit “+” icon on the ‘Discussions’ screen (Figure 5.8) and it will navigate him/her to
‘AddQuestion’ screen.

Additionally, ‘Discussions’ screen allows a user to subscribe to this category by
providing “SUBSCRIBE” menu item.
At the time of framing a question (Figure 5.8), the user is given a choice of picking an existing tag or to add a new tag to the question. If both are provided, then the new tag is given higher priority. Any new tag request is checked against the entries in the ‘Tags’ table of the database (Figure 5.9). If not found, then it is added to the table and the new tag-id is associated with the new question. If the tag already exists, then the existing tag-id is associated with the new question in ‘QuestionTags’ table.

Every question is associated with exactly one tag. If not provided, then the user is given an error and he/she cannot further proceed until they fix it.

```java
private String checkIfTagExists() {
    if(existingTagFound) {
        String tag = tagExisting.getSelectedItem().toString();
        return tags.get(tag);
    }

    ParseQuery<ParseObject> query = ParseQuery.getQuery("Tags");
    query.whereEqualTo("tagName", tag.getText().toString().toLowerCase());
    try {
        ParseObject tag = query.getFirst();
        return tag.getObjectId();
    } catch (ParseException e) {
        // Create tag if not exists
        final ParseObject addTag = new ParseObject("Tags");
        addTag.put("tagName", tag.getText().toString().toLowerCase());
        try {
            addTag.save();
        } catch (ParseException e1) {
            return null;
        }
        return addTag.getObjectId();
    }
}
```

Figure 5.9. Before saving a question check if the tag exists in the ‘Tags’ table or not.
5.5 **Loading a Discussion (Question-Answers Screen)**

![Question-Answers Screen](image)

**Figure 5.10. Question-Answers screen view when admin logins**

To view the complete discussion (a question & its answers), one has to click on the corresponding right arrow of a discussion [➡️]. To answer a question, one can attach an image to better explain the answer (Figure 5.10 shows the ‘attachment’ and ‘send message’ buttons at the bottom which assists in answering a question). The question or answer without an image has a default image displayed in their respective boxes to have a consistent look on the screen. The attached image can be enlarged by clicking on the image. Additional features on this screen are:

- Email the discussion to other people
- Sort the answers by time of creation or by likes (in descending order)
5.5.1 Removing a Question

In figure 5.10, the text in yellow color box represents a question. To delete a question, a menu item called ‘Delete’ is available on top of the screen. Only the admin and the owner of the question are authorized to delete it. Other actions which go hand-in-hand with a question deletion process are:

- Deletion of all its answers and notifications
- Association of question and its tag is broken

These actions are executed in a background thread (Figure 5.11).

```
private void deleteQuest() {
    final ProgressDialog pDialog = new ProgressDialog(QuesViewActivity.this);
    pDialog.setMessage("Deleting");
    pDialog.setCancelable(false);
    pDialog.show();

    ParseQuery<ParseObject> query = ParseQuery.getQuery("Questions");
    if(!session.isAdminLoggedIn())
        query.whereEqualTo("userName", session.getUserName());
    query.whereEqualTo("questionId", quesId);
    query.getFirstInBackground(new GetCallback<ParseObject>() {
        @Override
        public void done(ParseObject questionObj, com.parse.ParseException arg0) {
            if(questionObj != null) {
                questionObj.deleteInBackground(new DeleteCallback() {
                    @Override
                    public void done(ParseException e) {
                        if(e == null) {
                            ParseQuery<ParseObject> ansQuery = ParseQuery.getQuery("Answers");
                            ansQuery.whereEqualTo("questionId", quesId);
                            ansQuery.findInBackground(new FindCallback<ParseObject>() {
                                public void done(List<ParseObject> answers, ParseException e) {
                                    if(e == null) {
                                        try {
                                            ParseObject.deleteAll(answers);
                                            deleteNotifications(quesId);
                                            deleteQuestionTags(quesId);
                                            if(pDialog.isShowing())
                                                pDialog.dismiss();
                                            finishC();
                                        } catch(ParseException e1) {
                                            showError(ERROR_DELETED_QUES);
                                        }
                                    }
                                }
                            });
                        }
                    }
                });
            }
        }
    });
}
```

Figure 5.11. Question deletion logic

The delete operation affects the following tables which can be seen in figure 5.11:

- Questions
5.5.2 Removing an Answer

In figure 5.10, each of the grey boxes represents an answer, which has buttons to like or dislike an answer. This functionality of rating an answer decides how credible/correct it is. Each answer box also has a red-cross button [✓] available to the user who has posted the answer to help them delete it.

A dialog box appears when the original poster of the answer tries to delete it (Figure 5.12). This measure is taken to eliminate accidental deletes. Once an answer is deleted, the screen is refreshed to show the updated list of answers. A deleted answer along with its text and image cannot be recovered.

```java
private void deleteReply(final RowItem currentItem) {
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    builder.setMessage("Are you sure you want to delete this answer? ");
    builder.setPositiveButton("YES", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            ParseQuery<ParseObject> query = ParseQuery.getQuery("Answers");
            query.whereEqualTo("objectId", currentItem.getAnswerId());
            try {
                ParseObject ans = query.getFirst();
                ans.delete();
                itemlist.remove(currentItem);
                adapter.notifyDataSetChanged();
            } catch (ParseException e) {
                e.printStackTrace();
            }
            dialog.dismiss();
        }
    });
    builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });
    AlertDialog alert = builder.create();
    alert.show();
}
```

Figure 5.12. Answer deletion logic

5.6 Notification Center

There are 2 ways a user can receive notifications from PTA:
• Get Email (Outside App)

• By clicking message icon on ‘Category’ screen [.Mail] (Inside App, Figure 5.13)

![Figure 5.13. Notification Screen](image)

The first way is an optional way to get notified. Users will be notified by emails only if they have agreed to receive mail notifications on their ‘Profile’ screen. This is useful when the user is away from his cellphone but can access his Email. The second way is always present for the user to see the latest updates in the subscribed groups (Figure 5.13).

For the second way, ‘UserNotification’ table is queried to fetch all the notifications in the last 3 months. The notifications are sorted with the latest notification as the first record. Then each notification from the list is evaluated for the following fields:

• Read: To know whether the user has already seen this notification or not.

• Ques_Ans: To know what kind of notification it is. If its value is true, then it signifies that a new question was posted to a subscribed discussion group else it signifies that an answer was posted to one of the user posted question.
For each notification entry retrieved from the ‘UserNotification’ table, a button is formed and then added to UI layout (Figure 5.14). Unread notifications are shown in yellow boxes, while grey boxes represent that the user has viewed these notifications. On click of these buttons, the user is navigated to ‘QuestionAnswers’ screen to view the complete discussion.

```java
for (ParseObject notification : notifications) {
    Button btn = new Button(getApplicationContext());
    int btnId = notification.getInt("quesId");
    boolean quesAns = notification.getBoolean("ques_ans");
    int catId = notification.getInt("categoryID");
    boolean read = notification.getBoolean("read");
    Date created = notification.getCreatedAt();
    quesCategory.put(btnId, catId);
    String btnName = "";
    // Find this notification belongs to which question
    ParseQuery<ParseObject> query = ParseQuery.getQuery("Questions");
    query.whereEqualTo("questionId", btnId);
    query.whereEqualTo("categoryID", catId);
    try {
        ParseObject ques = query.getFirst();
        String qusetTitle = ques.getString("subject");
        notification.put("read", true);
        if (quesAns) { // Decides which kind of notification it is
            btnName = "A new <i>QUESTION</i> with title: <b>" + qusetTitle
            + "</b> has been posted under category: <b>" + categoryName.get(catId);
        } else {
            int count = notification.getInt("count");
            btnName = count + " new <i>RESPONSE(S)</i> have been posted "
            + "for your question with title: <b>" + qusetTitle
            + "</b> under category: <b>" + categoryName.get(catId);  
        }
        notification.save();
    } catch (ParseException e1) {
        e1.printStackTrace();
    }

    if (read)
        btn.setBackgroundResource(R.drawable.custom_textview);
    else
        btn.setBackgroundResource(R.drawable.yellow_custom_textview);

    btnName += "<br><br> Posted on: </b>" + created;
    btn.setTextColor(Color.RED);
    btn.setText(Html.fromHtml(btnName));
    btn.setOnClickListener(getQues(btn));
    btn.setTextSize(12);
    btn.setCompoundDrawablesWithIntrinsicBounds(0, 0, R.drawable.right1, 0);
    btn.setGravity(Gravity.LEFT);
    layout.addView(btn, 1p);
    MarginLayoutParams params = (MarginLayoutParams) btn.getLayoutParams();
    params.bottomMargin = 10;
}
```

**Figure 5.14. Notification Screen Logic**
5.7 Admin Powers

The admin of PTA goes through an additional check. He has to provide an answer to one of the security questions asked to him at the time of login. Only after providing the valid answer, he is navigated to the main screen of the application (Chapter 6 demonstrates the flow of the application).

The admin of PTA is authorized to make changes to the list of discussion groups which normal users are not capable of. He is allowed to add a new group of discussion or delete an existing one. Questions and answers of all the discussion groups are visible to him along with the ‘delete’ option. Even if the question/answer is not posted by the admin, he still has the power to remove them.

5.8 Code Snippets of Some Useful Features of PTA

- **Picking an Image from a Device Gallery:** The code snippet in figure 5.15 is called whenever an image is attached to a question or an answer. On click of the image attachment button, a request is initiated to find applications in the user’s handset, which accepts action_type as ‘ACTION_PICK’ and has images stored to it. Once a user selects an image, the method ‘onActivityResult()’ is invoked and picked image in shown inside an ‘ImageView’ element on UI.
Figure 5.16 shows how a ‘Message’ object is created and an asynchronous task sends out that message in the background, to all the users who have subscribed for mail notifications.

**SEND MAIL NOTIFICATIONS:**
public class SendMailNotification {
    private final String USERNAME = "mock.mail4dsu";
    private final String PASSWORD = "mockmail";

    public void sendMail(String email, String subject, String messageBody) {
        try {
            Session session = createSessionObject();
            Message message = createMessage(email, subject, messageBody, session);
            new SendMailTask().execute(message);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private Message createMessage(String email, String subject, String messageBody, Session session) throws MessagingException, UnsupportedEncodingException {
        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress("pca_notification@gmail.com", "Peer Tell App Notification");
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(email, email));
        message.setSubject(subject);
        message.setText(messageBody);
        return message;
    }

    private Session createSessionObject() {
        Properties properties = new Properties();
        properties.put("mail.smtp.auth", "true");
        properties.put("mail.smtp.starttls.enable", "true");
        properties.put("mail.smtp.host", "smtp.gmail.com");
        properties.put("mail.smtp.port", "587");

        return Session.getInstance(properties, new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(USERNAME, PASSWORD);
            }
        });
    }

    private class SendMailTask extends AsyncTask<Message, Void, Void> {
        @Override
        protected void doInBackground(Message... messages) {
            try {
                Transport.send(messages[0]);
            } catch (MessagingException e) {
                e.printStackTrace();
            }
            return null;
        }
    }
}

Figure 5.16. Send mail notifications

- **NOTIFICATIONS ICON XML:** Figure 5.17 shows the xml file, which generates the UI view of the ‘Notifications’ menu on the ‘Category’ screen of the application. On screen it looks like this: 📨.
Figure 5.17. Notification Menu UI
CHAPTER 6

APPLICATION STORY

Figure 6.1 and 6.2 depict the flow of Peer Tell application.

Figure 6.1. Shows the application flow from Home screen till Category Screen
Figure 6.2. Shows the application flow from Category Screen till end
CHAPTER 7

FUTURE ENHANCEMENTS

7.1 ENHANCED SEARCH

The current search is limited to search for questions by a tag string. It can be improved to search for any string present in a question. A good search capability is a must for a complete user experience and increases the utility manifold. Future enhancements include but are not limited to:

- Able to read incomplete strings.
- Search inside the answers.
- Search for all posts/questions/answers by a user.
- Use different filters.

7.2 MORE POWERS TO ADMIN

At present an Admin can do the following:

- Add/Delete a Category.
- Delete any question or its replies.

An Admin must have more power to maintain decorum and refine the user experience. A few enhancements that should be a part of the utility are below.

- Allowing admin to create events and notify users
- Broadcast any updates to the application done by him/her
- Edit/Add/Delete tag of a question

7.3 REPLY TO AN ANSWER

Preserving the usage model of popular Q&A forums like Quora and StackOverflow, Peer Tell allows replies to only questions. However social networking sites like Facebook have demonstrated the effectiveness when a user can respond to replies to a question.
This is very handy and transparent if a user wants to address a certain reply. It will make the forum more readable since the intended reply is next to a response and not after ‘n’ number of posts.

7.4 Editing of a Question/Answer

Present implementation does not allow a user to edit his questions and replies. This results in addition of new questions/answers and duplications. The consequences span over increased reader frustration to increased delays in accessing the database because of the increase in size of data. In future versions this shortcoming will be addressed.

7.5 Parse

The Parse hosted service will be retired on January 28, 2017. The backend service is being discontinued but there is no need for code change on the front-end side of the application because the Parse API endpoint won’t change. At the moment the App does not have data that needs to be migrated. There are many alternatives to migrate to, like Firebase, built.io and the new Parse Server stack. The migration process includes 2 steps [20]:

- Set up a MongoDB database and let Parse send your data to it.
- Set up Parse Server (on Heroku, or equivalent) and connect it to the MongoDB database.

7.6 For iOS and Tablets

This version of the app supports only Android and was tested on Google Nexus cellphone platform for size (1280 X 720). It was identified as the most popular use case. For completeness, the application must also support iOS and larger tablet size. It will ensure better mass penetration and acceptance, and hence ensuring popularity and reachability.

7.7 Finger Print Login

With the advent of more and more smartphones, which can register fingerprints, the app can be made capable to use fingerprints to login instead of cumbersome text based login. This will help get rid of some frustration when a user forgets his login information. It will also be hassle free access and make the application safer.
7.8 Voice Responses

Sometimes the best way to reach the audience might be through voice. At present the only format of response supported is text and pictures. In future maybe it is desirable to include voice snippets along with text/picture help get one’s point across.

7.9 Voice to Text Conversion

A lot of modern keyboards support voice to text conversion. We might be able to use this feature to capture one’s thought in text and all he/she needs to do it speak into the app. Some users might not be comfortable posting their voice in a forum but might like the effortlessness of voice communication over typing. They might find this feature handy.
REFERENCES


