# Table of Contents

1. **Introduction** ................................................................. 4  
   1.1 Background Information .................................................. 4  
   1.2 Project Title ..................................................................... 4  
   1.3 Project Purpose ............................................................... 4  
   1.4 Project Scope .................................................................... 4  
   1.5 Team .................................................................................. 5  
   1.5.1 Structure ....................................................................... 5  
   1.5.2 Member Roles ............................................................... 5  
   1.6 Process Model .................................................................... 5  
2. **Research** ............................................................................ 6  
   2.1 Market Research ............................................................... 6  
   2.1.1 Last.fm iPhone ............................................................. 6  
   2.1.2 TuneWiki .................................................................... 7  
   2.1.3 Blastfm ....................................................................... 7  
   2.2 Technology Research ......................................................... 8  
   2.2.1 Platforms of Development ............................................. 8  
   2.2.2 P2P Networking ............................................................. 9  
   2.2.3 Last.fm Integration ...................................................... 10  
   2.2.4 Location Based Services ............................................... 11  
   2.2.5 Programming Languages ............................................. 12  
   2.2.6 Database Management ............................................... 14  
3. **Requirement Analysis** .......................................................... 15  
   3.1 System Requirements ......................................................... 15  
   3.1.1 Hardware Requirements ............................................... 15  
   3.1.2 Software Requirements ............................................... 15  
   3.1.3 Development Environment Requirements .................... 16  
   3.2 Functional Requirements ................................................... 16  
4. **Modeling** .......................................................................... 16  
   4.1 Functional Modeling ......................................................... 16  
   4.1.1 Data Flow Diagrams .................................................... 16  
   4.1.2 Data Dictionary of Data Flow Diagrams ....................... 18  

4.2 Use Case Diagram

4.3 Use Case Scenarios

  4.3.1 Use Case 1: Chatting With Online Buddies
  4.3.2 Use Case 2: Organizing Events via Google Map
  4.3.3 Use Case 3: Playing Last.fm on Android
  4.3.4 Use Case 4: Accessing Album
  4.3.5 Use Case 5: Organizing Store Information

5 Project Scheduling

  5.1 Gantt chart

6 Appendix
1 Introduction

1.1 Background Information
In accordance with the development of the technology in the direction of mobile platforms, smart phones are introduced to users which enables people use their mobile phones for too many applications. Most of these applications are based on photography, music and internet. Taking in account these fields, we decided to combine music and internet in order to target users’ preferences.

After some research, we learned that Last.fm is the most popular social network based on music. We aimed to take advantage of popularity of such a hit social network and people’s desire to use their smart phones in every time of the day. So we wanted to make possible that people could be able to use Last.fm with more features on their smart phones.

1.2 Project Title
Our project is named as Finger Tune.

1.3 Project Purpose
Finger Tune is a mobile application which enables Android users to

- Listen to music or radio
- Chat with friends
- Create and share events
- Search for shopping centers which sells required album

via a musical social network which builds a detailed profile of each user's musical taste by recording details of all the songs the user listens to, either on the streamed radio stations or on the user's computer or some portable music devices. And this social network is Last.fm from which Finger Tune will inherit most of its features.

1.4 Project Scope
Finger Tune will be an Android Application and to develop it, Turkcell API, Last.fm API, Google Map API and Android SDK will be used.

Finger Tune will enable users to use Last.fm on their smart phones with more features. But before listing our applications features, we want to mention about what Last.fm provides to the users.

- Last.fm builds a detailed profile of each user's musical taste by recording details of all the songs the user listens to, either on the streamed radio stations or on the user's computer or some portable music devices.
- This information is transferred to Last.fm's database ("scrobbled") via a plug-in installed into the user's music player.
Profile data is displayed on the user’s profile page. Last.fm recommends new music to the listeners according to their music profile stored on the database. It presents the opportunity of learning about concerts, parties, etc via a social network.

Finger Tune will enable users play Last.fm on Android so Finger Tune users could be able to use features listed above. Moreover, Finger Tune users could have chance to chat with people who are online and chosen as friend before. Furthermore, one using Finger Tune could be able to create events like concerts, parties and share this information with his/her friends. Beside on these features, if one wants to see nearest shop selling the albums of artist recently listened, she/he can see the place of the shop from the Google Map view.

The user interfaces which we plan to design are listed in Appendix part.

1.5 Team

1.5.1 Structure
Our team has “Democratic Decentralized (DD) “structure .We do not have a permanent team leader. Every decision is made by team consensus.

Ground rules of our team are:

- Every Monday a meeting must be held and every one must be on time.
- If somebody can not come to the meeting, she must have an excuse.
- The given responsibilities in ex-meeting date should be completed until the next meeting date.
- Every team member should check her mail everyday.
- Emergency meetings are done according to team consensus.

1.5.2 Member Roles
- Derya Gocer Time Keeper
- F.Gulsah Kandemir Initiator, Optimist
- Gozde Kaymaz Devil’s Advocate
- Betul Kuruoglu Time Keeper, Devil’s Advocate

1.6 Process Model
Our team is supposed to progress through analysis, initial design, detailed design, release of prototype, implementation, testing and maintenance phases in a limited time. Although we are going to try our best not to encounter problems during the project, there is always a possibility of not being able to complete the project owing to the existing risks.
Therefore, the first goal of our team is being successful in the release stage of the prototype.

As process model of our project, we prefer to use ‘Spiral Model’. This model couples the iterative nature of prototyping with the controlled and the systematic aspects of linear sequential model. The Spiral Model provides us with the chance of developing software in a series of incremental releases. As a start, we design the project only on paper and develop it phase by phase. By the linear sequential model property of Spiral Model, at every phase, we study on our task regions and analyze the software design again according to feedback from our assistant Umut Erogul. By this methodology, we better understand risks at each development step of large-scale software. Technical risks are considered at every stage of the project and can be reduced before become problematic for us.

2 Research
We have conducted a wide research on two main fields. Firstly we have conducted a market research on the mobile applications similar to our product, and also social networks that we can get inspire or ideas. Second part of our research was technological research which helped us to decide on which platform, technologies, APIs and libraries we will use.

2.1 Market Research

2.1.1 Last.fm iPhone
This application is available for iPhone and iPod Touch. Some of its features are:\n
- Ability to tag songs
- Personal tag radio
- Calendar based events view
- Common artists when viewing a user profile

Our product Finger Tune is similar to iPhone version of Last.fm in main approach. And most of the features of Last.fm iPhone above will be included in Finger Tune, because the main idea of “mobile Last.fm” is same. While Finger Tune will have common features with Last.fm iPhone, there are some differences between these two.

For example Finger Tune will have chat option between friends and music neighbors which Last.fm iPhone does not have. And also Finger Tune will have advanced features in event handling, such as creating new events on Last.fm, inviting friends and keeping track of friends who attends or does not attend those events. But we don’t think Finger Tune will have calendar based events view.

---

1 [http://mashable.com/2008/07/13/lastfm-iphone/]
2.1.2 Tunewiki
Tunewiki is an audio and video player for Android which displays synchronized lyrics or translated lyrics while playing the music. Some of its features are:

- Allowing for a music based Social Network where users share music information with friends
- Displaying music maps that show songs playing in the same GPS location
- Creating music charts that shows top songs in GPS location

Our Finger Tune is similar to Tunewiki conceptually in main approach that they both focus on music. But Finger Tune has additional features on Tunewiki such as:

- Creating and sharing musical events
- Showing user the most available and near music stores (GPS location) where he/she can buy preferred albums
- Chatting with friends or people having similar music taste

2.1.3 Blastfm
Blastfm is an Audioscrobbler that works on Nokia N95 and S60. It actually is not an application but a patch to the mobile operation systems audio player.

So there is a big difference between our Finger Tune and Blastfm. For example;

- There is no event handling
- No chat option
- No location based events
- No store information

As can be seen, Blastfm is only a scrobbler which sends information of played songs to the Last.fm database.

---

2 http://code.google.com/android/adc_gallery/app.html?id=48

3 http://build.last.fm/item/368
2.2 Technology Research

2.2.1 Platforms of Development
Before we had decided on our project, we had to conduct a field research about the platforms that we would develop on. After some amount of research we finally come up with two alternatives:

1. Android SDK
2. iPhone SDK

Both of these platforms have pros and cons in some aspects. And as already known we have come up with the Android decision. Now we would like to tell the reasons why we have chosen Android.

Main differences between Android and iPhone are⁴:

1. iPhone is Closed System, where Android is Open System (Open source)
2. Android is totally free of charge, where iPhone is not
3. iPhone is only developed by and for Apple, where Android can be used in Motorola, HTC and Samsung in the future.
4. iPhone apps are developed with Xcode, where Android is pure Java.

The above technical facts have been very effective while taking our decision. But also there are some marketing issues we need to mention:

- iPhone is in the market for a long time, so has lots of applications including social networking and internet radio playing
- Android is brand new world wide and also in Turkish market, and there are a lot to do improve Android’s application gallery
- Android does not have Last.fm player and scrobble yet, where iPhone already has

These above reasons led us to the decision of Android, and to sum up most important reasons of our decision are as follows:

Android is;

1. Totally open source and free of charge
2. Java programming
3. Brand new in market and lack of Last.fm application
4. Adaptable in variety of hardware systems

⁴http://blog.radioactiveyak.com/2008/03/android-v-iphone-sdk-showdown.html
http://www.deviation.de/?p=20
But to mention; Android also has some drawbacks compared to iPhone. Such as:

- iPhone is more popular worldwide
- iPhone’s multitouch feature is not supported.

2.2.2 P2P Networking

As we mentioned in our project definition, Finger Tune will be capable of enabling users chat each other. For this reason we conducted a research on current chat providers suitable for Android and our application. We come upon many chat applications which are developed by Android’s developing group. Some of them are not open source; some of them are free now but will not be in the feature. At the end, we decided to analyze **Google Talk** and **Jchat** whether they are suitable or not for our project.

2.2.2.1 Jchat

Android Jchat is a Location Based Chat application that provides a good example of peer-to-peer application aimed at proving the power of adding JADE\(^5\) agent capabilities to the Android Platform.

JADE introduces the concept of agent oriented development and this application demonstrates the flexibility of this approach, that it is easily applied to a common p2p environment as in a chat application.

Some features of Jchat are listed below\(^6\):

- Using contacts from phone contact database
- Chatting with contacts by sending text messages
- Sending SMS to contacts
- Calling contacts by phone
- Seeing contacts’ position on the map in realtime
- Dynamically switching between different ongoing conversations

2.2.2.2 Google Talk

Google Talk is an instant messaging platform that Google builds. There are many different ways to use Google Talk, which are\(^7\):

- Via PC
- Via web
- Via other IM clients

\(^5\) [http://jade.tilab.com](http://jade.tilab.com)
\(^6\) [http://code.google.com/p/jchat4android/](http://code.google.com/p/jchat4android/)
\(^7\) [http://www.google.com/talk/about.html](http://www.google.com/talk/about.html)
Taking these functionalities into account, while searching for alternative chat options in Android, we thought that if Android is a Google product, then we can integrate Google Talk, another Google product, to our project easily.

But after some research we can list advantages and disadvantages of Google Talk and Jchat below.

**Advantages of Jchat:**
- Already designed for and usable in Android
- Usage of contacts from phone contact list (This may be changed according to Finger Tune’s specifications)
- Dynamically switching between conservations

**Disadvantages of Jchat:**
- May not be so easy to integrate Jchat with our Last.fm application
- May be so complicated to contribute in the source code

**Advantages of Google Talk:**
- Being a product of Google
- Available and usable in many platforms

**Disadvantages of Google Talk:**
- Is not integratable to a smaller mobile application
- Only usable between Gmail and Google talk friends
- May be expensive in performance(not light enough) compared to Jchat

For the above reasons, we decided to use Jchat in our product. We will contribute in the source code, change some features and implementations of Jchat if necessary.

### 2.2.3 Last.fm Integration
Since Finger Tune is actually a Last.fm player, and mobile network; Finger Tune should be able to gather all information about a user, his/her music taste, recommendations etc. We need to get all necessary information from Last.fm in order to enable our users to access their music profile, radios, friends and events.
The only way we get all these information is to use Last.fm API\(^8\) (Application Programming Interface) which is very convenient for the people who want to develop Last.fm applications.

The procedure of working on the API is sending methods or requests to the root URL of the API and the response to that request is an xml file.

Some of the API methods are:

- album.getInfo
- artist.getSimilar
- artist.getTopFans
- artist.search
- auth.getMobileSession
- event.getInfo
- event.attend
- playlist.fetch

All these methods and much more of them will be used in Finger Tune.

### 2.2.4 Location Based Services

Our application will be capable of showing the places of music stores which sells the albums of recent listened artist. Since some part of or project depends on location of user, we have to use location based services.

A location-based service\(^9\) (LBS) is an information and entertainment service, accessible with mobile devices through the mobile network and utilizing the ability to make use of the geographical position of the mobile device.

In our project, we will use Android’s map API and Google map API.

#### 2.2.4.1 Android Map API

The Android SDK includes two packages that provide Android's primary support for building location-based services: android.location and com.google.android.maps

In our project, android’s user place information will be handled by functions based on Android Map Api\(^10\).

---

\(^8\) [http://www.last.fm/api](http://www.last.fm/api)

\(^9\) [http://en.wikipedia.org/wiki/Location-based_service](http://en.wikipedia.org/wiki/Location-based_service)

android.location
This package contains several classes related to location services in the Android platform. Most importantly, it introduces the LocationManager service, which provides an API to determine location and bearing if the underlying device (if it supports the service). The LocationManager should not be instantiated directly; rather, a handle to it should be retrieved via getSystemService(Context.LOCATION_SERVICE).

com.google.android.maps
This package introduces a number of classes related to rendering, controlling, and overlaying customized information on your own Google Mapified Activity. The most important of which is the MapView class, which automatically draws you a basic Google Map when you add a MapView to your layout. Note that, if you want to do so, then your Activity that handles the MapView must extend MapActivity.

2.2.4.2 Google Maps API
The Google Maps API lets you embed Google Maps in your own web pages with JavaScript. The API provides a number of utilities for manipulating maps and adding content to the map through a variety of services, allowing you to create robust maps applications on your website.

In our project, since shopping center place will be placed on Google map, we will use Google Map API’s properties, methods, functions.

2.2.5 Programming Languages
2.2.5.1 Client-Side
When we look at all the possible technologies, libraries, platforms that we will use in our project; we have seen that the most convenient programming language is Java. Because

- Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language,
- Android applications are developed in Eclipse(Java) environment,
- Last.fm API has Java bindings, means we can send our requests using Java,
- Map application of Android is also in Java language.

So; all these reasons led us to use Java language as our applications default developing language. And also we will develop our mobile application Finger Tune in Eclipse environment.
2.2.5.2 Server-Side
As we mentioned before our project Finger Tune will have a simple web interface for Music Stores. Music stores will save and update their stock information from web. For this purpose we needed to search for current web developing languages and frameworks. At first glance we have the following options:

- PHP
- Java
- ASP

2.2.5.2.1 PHP
PHP is very common in today’s web market. Most of the web applications today are developed with PHP and its various frameworks.

Advantages:

- PHP is very popular along web developer
- Easy to code
- Easy to find code samples on web, because PHP is open source

Disadvantages:

- PHP seems to be popular among hackers and hobbyists, which is a bad reputation
- PHP has a confusing and inconsistent class/object system

2.2.5.2.2 Java
Java is another option for server-side development,

Advantages:

- Leads ‘enterprise’ tools and in general has better support for web services and other connectivity mechanisms.
- Maturated and advanced in last 3 years
- Easy to detect errors while coding
- Open source

Disadvantages:

- Edit / compile / loop timeframe in Java tends to be slower
- Java is complex
- Big apps eat huge amounts of memory

http://raibledesigns.com/rd/entry/php_vs_java_which_is
2.2.5.2.3  ASP\textsuperscript{12}

Microsoft’s Active Server Pages is widely supported in many web applications as well. Below advantages and disadvantages of ASP can be seen.

**Advantages:**
- Easy to program
- There are many web sites and books about ASP with lots of free code
- Well integrated components

**Disadvantages:**
- Too much dependency on Microsoft
- Poor performance in big projects
- Connection to database is expensive as MS-SQL is a product of Microsoft that needs to be purchase

As a result, after the research we have conducted on server-side coding; we decided on using Java, because it seems to be more convenient for our project.

2.2.6  Database Management

2.2.6.1  Oracle

**Advantages:**
- Runs on many platform.
- For very complicated store procedure, it runs faster.
- Has many advanced functions such as Real Cluster, Flush back recovery.

**Disadvantages:**
- Oracle DBA and developer cost higher to hire and train.
- Not convenient for small applications/projects.

\textsuperscript{12} http://www.webpronews.com/expertarticles/2005/12/22/asp-vs-php
2.2.6.2 **Ms SQL**

**Advantages:**
- Faster for big projects
- More resistant to data corruption due to the binary log keeping

**Disadvantages:**
- In order to have full performance, powerful hardware is required.
- Only runs on Windows.
- Expensive

2.2.6.3 **MySQL**

**Advantages:**
- Is not always free but very cheap
- Has pure performance due to its default table format MyISAM
- Binding with too many programming languages

Listing all pros and cons of these 3 database servers, we decided to use MySQL due to its compatibility with Java and much more advantages over other servers.

3 Requirement Analysis

3.1 System Requirements

3.1.1 Hardware Requirements

Considering our project, strictly defining hardware requirements is not wise. By the time as our number of users increase, our hardware requirements will change. As a baseline, we need a single Pentium 4 computer with 1 GB of ram. Nevertheless, as the number of users increase some enhancements on hardware configurations must be considered:

- Moving to a multi-processor and vast memory (both ram and durable storage) server
- Creating a clustered server architecture

3.1.2 Software Requirements

- Windows or Linux operating system
- IIS or Apache (depending on operating system)

---

3.1.3 Development Environment Requirements
- Eclipse IDE
- Android SDK
- phpMyAdmin

3.2 Functional Requirements
Finger Tune user could be able to chat, listen to music, create and share events, see the nearest shopping center selling the albums of recent listened artist. Capabilities of Finger Tune, what it can do is listed in detail in 4.3 Use Case Scenarios part.

4 Modeling
4.1 Functional Modeling
4.1.1 Data Flow Diagrams

LEVEL 0 DFD : Finger Tune Project
LEVEL 1 DFD : Finger Tune Application

1. Play Last.fm on Android
   - music request
   - music response
   - chat request
   - chat response
   - related radio details

2. Organize Chat Process
   - online buddy list

3. Access Album
   - album access request
   - google map response
   - albums' details

4. Organize Store Information
   - store details
   - store address details
   - album details

5. Organize Events
   - event request
   - SMS/MMS notification
   - event response
   - created events
   - created event on google map
   - created event by the user

Last.fm

Google

The System
### 4.1.2 Data Dictionary of Data Flow Diagrams

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>music request</td>
<td>Radio listening request according to the selected genre or artist</td>
</tr>
<tr>
<td>Input to</td>
<td>1. Play Last.fm on Android</td>
</tr>
<tr>
<td>Output from</td>
<td>It is an input to the system</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>music respond</td>
<td>Radio stream</td>
</tr>
<tr>
<td>Input to</td>
<td>It is an output to the Finger Tune user</td>
</tr>
<tr>
<td>Output from</td>
<td>1. Play Last.fm on Android</td>
</tr>
<tr>
<td>related radio details</td>
<td>Fetching the songs that will be presented to the user</td>
</tr>
<tr>
<td>Input to</td>
<td>1. Play Last.fm on Android</td>
</tr>
<tr>
<td>Output from</td>
<td>Last.fm Database</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

| chat request          | Request to see the online buddies or chat with the selected one             |
| Input to              | 2. Organize Chat Process                                                    |
| Output from           | It is an input to the system                                                |
| chat response         | Online buddy list or chat screen with the selected buddy                    |
| Input to              | It is an output to the Finger Tune user                                     |
| online buddy list     |                                                                             |
| Input to              | 2. Organize Chat Process                                                    |
| Output from           | Last.fm Database                                                             |
| Description           |                                                                             |

<p>| album access request  | Request to see the nearest stores having related album                     |
| Input to              | 3. Access Album                                                             |
| Output from           | It is an input to the system                                                |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Input to</th>
<th>Output from</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google map response</td>
<td>It is an output to the Finger Tune user</td>
<td>3.Access Album</td>
<td>The map of the nearest stores</td>
</tr>
<tr>
<td>stores’ album details</td>
<td>3.Access Album</td>
<td>The System Database</td>
<td>The list of the stores having the related album</td>
</tr>
<tr>
<td>albums’ details</td>
<td>3.Access Album</td>
<td>Last.fm Database</td>
<td>The list of the albums of the artist that is being listened</td>
</tr>
<tr>
<td>stores on Google map</td>
<td>3.Access Album</td>
<td>Google Database</td>
<td>Map of the nearest place-marked stores having the related album</td>
</tr>
<tr>
<td>store details</td>
<td>4.Organize Store Information</td>
<td>It is an input to the system</td>
<td>Address and stock details of the music stores</td>
</tr>
<tr>
<td>album details</td>
<td>The System Database</td>
<td>4.Organize Store Information</td>
<td>Album details that will be updated in the System</td>
</tr>
<tr>
<td>store address details</td>
<td>Google Database</td>
<td>4.Organize Store Information</td>
<td>Store address details that will be place-marked on Google Map</td>
</tr>
<tr>
<td>event request</td>
<td>5.Organize Events</td>
<td>It is an input to the system</td>
<td>Event request to create a new event and share, or see the created ones</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Input to</td>
<td>Output from</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------</td>
<td>-------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>SMS/MMS notification</td>
<td>It is an output to the Finger Tune user</td>
<td>5.Organize Events</td>
</tr>
<tr>
<td>Description</td>
<td>Reminder notification of the events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>event response</td>
<td>It is an output to the Finger Tune user</td>
<td>5.Organize Events</td>
</tr>
<tr>
<td>Description</td>
<td>List or map of the created events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>created events</td>
<td>5.Organize Events</td>
<td>Last.fm Database</td>
</tr>
<tr>
<td>Description</td>
<td>List of the already created events on Last.fm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>created events on Google map</td>
<td>5.Organize Events</td>
<td>Google Database</td>
</tr>
<tr>
<td>Description</td>
<td>map of the places of the events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>created event</td>
<td>Google Database</td>
<td>5.Organize Events</td>
</tr>
<tr>
<td>Description</td>
<td>Address of the created event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>created event by the user</td>
<td>The System Database</td>
<td>5.Organize Events</td>
</tr>
<tr>
<td>Description</td>
<td>All information of created event</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.2 Use Case Diagram
4.3 Use Case Scenarios

4.3.1 Use Case 1: Chatting With Online Buddies
This use case is for chatting via cell phone with friends who are online on Last.fm at the same time, in other words friends who are using the Last.fm application of their cell phones at that time.

Actors: Android user who wants to chat

Pre-Condition: The user should be logged into the system.

Post-Condition: The chat connection is established between two or more Android users.

Basic Flow:
1. Android user launches to the chatting application on his/her Android.
2. Online buddy list of the user is retrieved from Last.fm database and is shown to the user.
3. If the user wants to chat with any of the buddies, he/she selects the buddy and thus connection is established.

4.3.2 Use Case 2: Organizing Events via Google Map
This use case is for creating and listing music related events and sharing them with the buddies. Apart from the reminder side of this use case, it also provides the user to see the locations of the events on Google Map.

Actors: Android user who wants to manage his/her music based events

Pre-Condition: The user should be logged into the system.

Post-Condition: Event creation is shared with the buddies or location of the events is shown to the actors.

Basic Flow:
1. Android user launches to the events application on his/her Android.
2. If the user wants to create a new event and share it with buddies;
   2.1 The information of the new event is moved directly to the System Database and formed as an event package which will be sent to the buddies.
   2.2 The location information of the new event is place-marked on Google Map.
   2.3 Created new events are shared with the buddies by sending invitations.
3. If the user wants to see the list of the events that are related with his/her genre;
   3.1 The list of the created events is retrieved from System Database and Last.fm Database.
3.2 User selects one of them and information and location of the event is displayed on the screen.

4. Apart from these, with the help of Turkcell SMS and MMS reminders of the events will be sent to the user if he/she plans to attend.

4.3.3 Use Case 3: Playing Last.fm on Android
This use case is for simulating the web based Last.fm radios to the Android mobile environment.

**Actors:** Android user who wants to perform Last.fm’s basic functionality on his/her mobile phone

**Pre-Condition:** The user should be logged into the system.

**Post-Condition:** Music service is established on Android environment.

**Basic flow:**

1. The user searches a radio according to a specific genre or artist. Besides he/she may start his/her own radio.
2. Selected radio service is started by retrieving related radio details from Last.fm database.

4.3.4 Use Case 4: Accessing Album
This use case is for accessing an album owned by an artist who is being listened and for finding the nearest music stores that sell that album via Google Map.

**Actors:** Android user who wants to access an album of an artist recently listened

**Pre-Condition:** The user should be logged into the system and should be listening to a radio.

**Post-Condition:** The nearest stores having a specific album are shown with Google Map to the user.

**Basic flow:**

1. If the user wants to see the albums of the recently listened artist, a list of albums is retrieved from Last.fm Database.
2. User selects one of the albums and sends a request to see the nearest stores.
3. All the stores having this album are retrieved from System Database.
4. The map containing nearest stores is shown to the user.

4.3.5 Use Case 5: Organizing Store Information
This use case is for keeping store information in System Database and storing address information on Google Map with place-mark.
Actors: Music Stores that want to share their albums stocks

Pre-Condition: The user should be logged into the system.

Post-Condition: Stores are place-marked on Google-Map.

Basic Flow:

1. Stores’ stock albums are added to the System Database.
2. Addresses of the stores are place-marked on Google Map.
## Project Scheduling

### 5.1 Gantt Chart

<table>
<thead>
<tr>
<th>Task</th>
<th>Q4 - 2008</th>
<th>Q1 - 2009</th>
<th>Q2 - 2009</th>
<th>Q3 - 2009</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>October</td>
<td>November</td>
<td>December</td>
<td>January</td>
</tr>
<tr>
<td>Project Proposal Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Research</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirement Analysis Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milestone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Design Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detailed Design Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Page Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile Application Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype Integration and Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milestone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IMPLEMENTATION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface Design and Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last.fm Player Implementation on Android</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milestone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event management Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chatting Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration of the So Far Implemented Components</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milestone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server Side Web Application Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Managing Stores and Albums</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration of the So Far Implemented Components</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milestone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Finalization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milestone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6 Appendix