1 Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Android Application that which is created to specifically target users who are looking to travel places in India. It will explain the features of the system, the interfaces of the system, what the system will do , the constraints under which it must operate and how the system will react to external stimuli. This document is intended for the users, evaluator and the developers of the system.

1.2 Scope of the Project

This Application will be a boon for those users who cannot make up their mind as to where to travel in India. It takes budget as an input from the user and suggests places including mode of transit, staying in hotels, sightseeing all within the budget.
Thereby, giving the end user several options to travel in India which fits in the budget.

More specifically, this system prepares several itineraries some below, some equal and some above the budget and the user has an option to select the itinerary which best suits him/her and plan their travel accordingly. A user is also given the option to add locations of their choice in the prepared itinerary, thereby adjusting the budget as per the location.

The Database updates itself by crawling the data from various travel sites like ixigo.com , trip advisor.com, goibibo.com etc, thereby all the fares of hotel, trains and buses remains updated.

1.3 Intended Audience and Document Overview

The next chapter, the overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written, primarily for the developers and describes in technical terms the details of the functionality of the product.

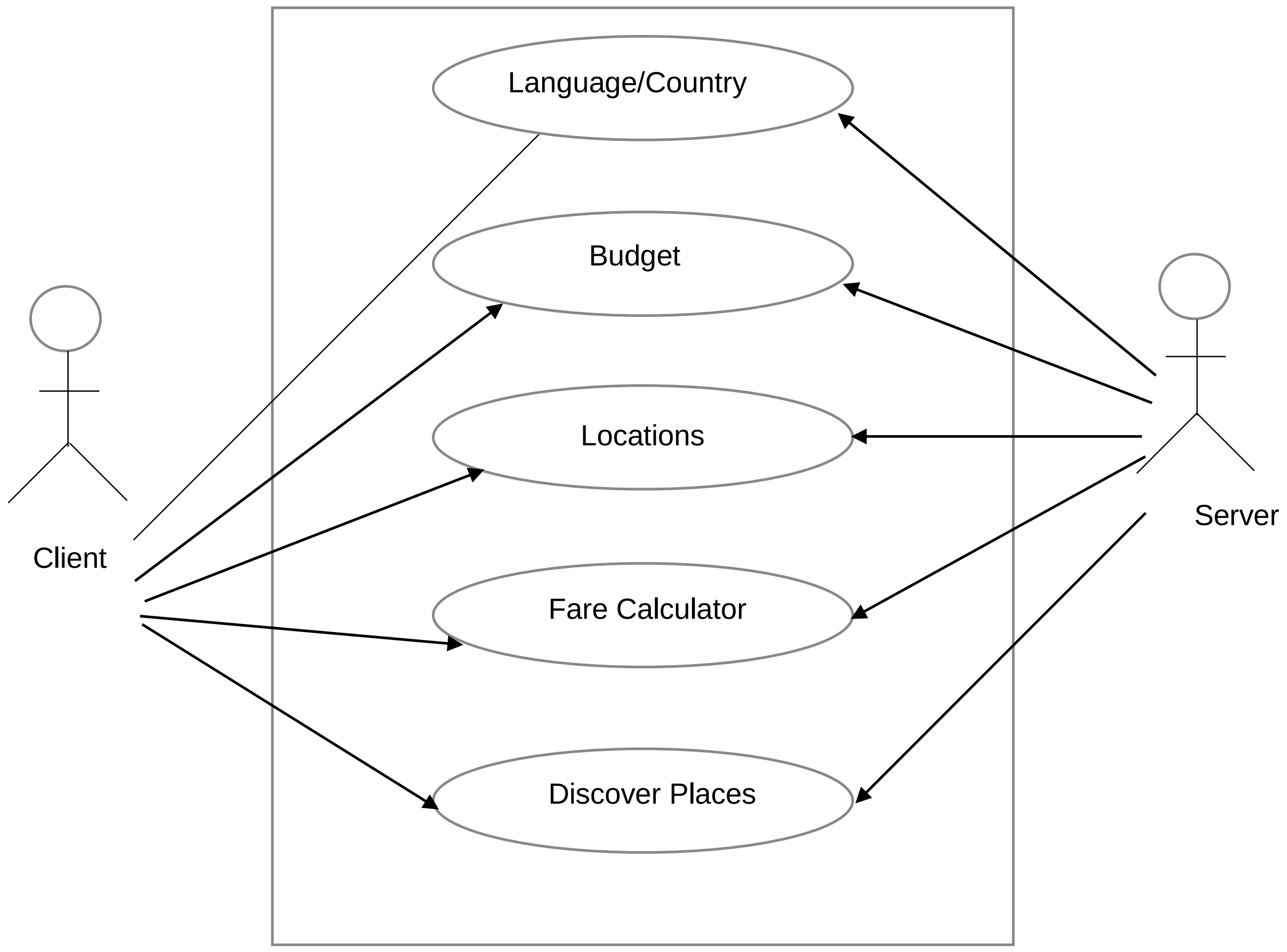
Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

1.4 Glossary

Term	**Definition**
Crawler	A crawler is a service that extracts information from a website according to the user preferences.
Database	Collection of all information monitored by the system.
Field	A cell within a form.
Internet Database	All the information that is superficial and that can be extracted from a website by the crawler.
User	The end users who uses the application.
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.
Developer	A person who develops standalone software and gets involved in all the phases of the development.

2 Overall Description

2.1 Product Perspective



The application works on the principle of Client server model. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A client need not share any of its resources with clients, but requests a server's content or service function. Clients therefore initiate communication sessions with the servers which await incoming requests. In our case the user requests various itinerary based on the type of holiday, budget and season.

2.2 Product Functionality

The product makes use of many open source applications under the GNU general public Licence. It is a mobile based system implementing the client-server model. The application provides a simple mechanism for users to plan their travel with the given budget.

The following are the main features that are included in the Android based Application :

- Number of users being supported by the system: Though the number is precisely not mentioned but the system is able to support a large number of online users at a time.

- Tags: Users can select the places to visit according to the tags suggested by the application, like spiritual, nature, wildlife, lifestyle etc .

- Search: Search is based upon the online search engine based on keywords.

- Dynamic Addition of Locations: Users can add their own locations that are not being suggested by the application. The application then adds the location and adjusts the budget accordingly.

- Fares: Displays the fares of hotels, trains, flights, buses of the destination.

2.3 Users and Characteristics

This application is meant for the users who are looking to spend their vacations travelling or holiday at some exotic locations in India with limited budget but are confused as to where to travel. According to the usage of the application the users can be classified into 3 types:

<u>A user who doesn't know where to travel :</u> The application has the feature called ‘Discover Places’ , which has the places sorted according to the ratings by the user. A user can then select the best place to travel and proceed further.

<u>A foreign Tourist :</u> The application adapts itself according to the country set by the user during the registration, thereby the language of the application changes according to the user preferences for ease of understanding.

<u>A user who has pre-requisite knowledge of the location :</u> A user who has the knowledge as to where to travel can set the budget and locations manually and can view various itineraries generated by the application.

Most frequent users would be an average adult with age spanning from 25-30 years, mostly working professionals looking for a nice holiday trip. It is assumed that the user is well aware of mobile technologies and knows the Android OS.

2.4 Operating Environment

The application is designed to work only on the Android Operation System and its supporting devices. The Android OS on the device should be a minimum of JellyBean (4.1.0) and a maximum of Lollipop (5.1.1) . Requirements for the minimum amount of RAM for device running Android 5.1 range from 512 MB of RAM for normal density screens, whereas the recommendation for Android 4.4 is to ave atleast 512MB of RAM, while for low RAM devices 340 MB is the required minimum amount that does not include memory dedicated to various hardware components such as baseband processor.
Android devices incorporate many optional hardware components including GPS, orientation sensors, dedicated proximity sensors, the application will use all the mentioned sensors if permitted by the user. The app will rely on several functionalities built into Android’s Application Programming Interface (API), so ensuring appropriate usage of API will be a major concern. Beyond that, the application is a self contained unit and will not rely on any other Android-related software components.

The application will, however, frequently interact with the dedicated online server hosted by parse.com . The server operates on a Linux CentOS platform with 8GB of RAM and 20GB of allocated storage space.

2.5 Design and Implementation Constraints

The primary design constraint is the mobile platform. Since the application is designated for mobile handsets, limited screen size and resolution will be a major design consideration. Creating a user interface which is both effective and easily navigable will pose a difficult challenge. Other constraints such as limited memory and processing power are also worth considering. The app is meant to be quick and responsive, even when dealing with large groups , so each feature must be designed and implemented with efficiency in mind.

2.6 Assumptions and Dependencies

Time Dependencies

The features of the our Application are basically divided into two groups :
Core Features and additional features. Core Features are crucial to the basic functionality of the application. These features must all be implemented in order for the application to be useful.

Optional features , however , are not critical to the function of the application.

Hardware Dependencies

Some of the additional features rely on hardware components present in Android handsets. For instance, the GPS will be used to track the current location of the user, and is entirely dependent upon the ability to access the camera's functionalities.

References : http://developer.android.com/guide/topics/location/index.html

External Dependencies

GOOGLE PLACES API

The software application may integrate the ability to interact with Google Places for marking the location of the user. Note that this API is not guaranteed to be perfectly functional, as it is currently in the beta phase of its lifecycle and is provided by an experimental branch of its host company, Google Labs. Using this API may require the use of the GPS hardware on the Android platform, where it is available.

References:*http://code.google.com/apis/maps/documentation/places/*

FACEBOOK API

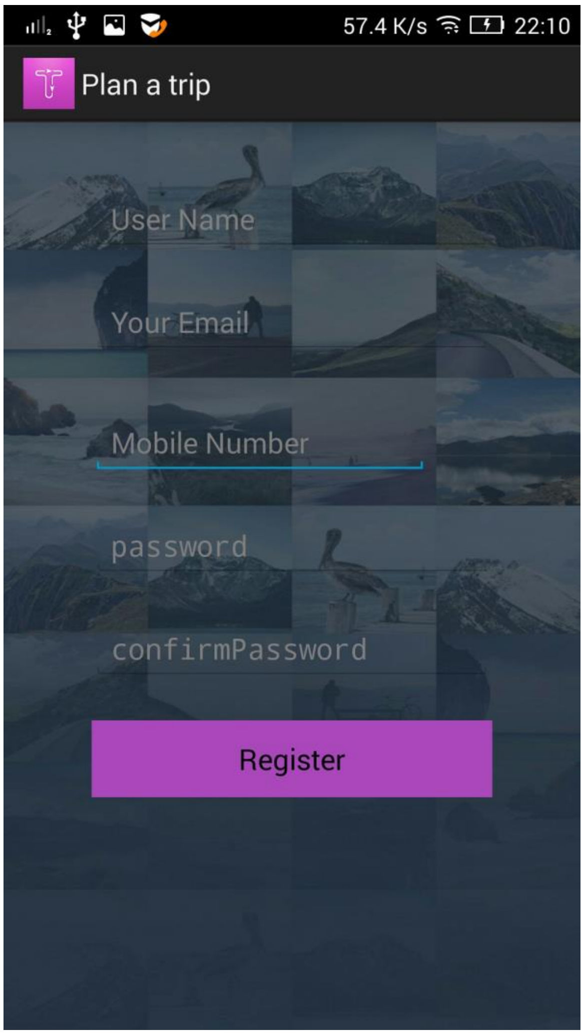
As large and large number of people are now active on Facebook, therefore instead of cumbersome registration process one ca easily login using their Facebook Account for quick and easy login.

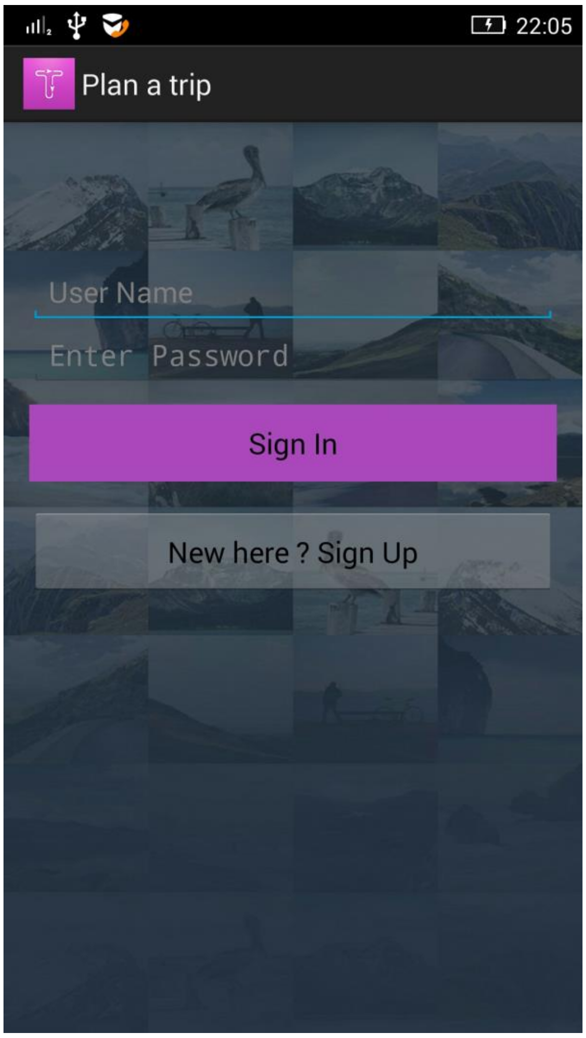
References : http://developer.facebook.com/

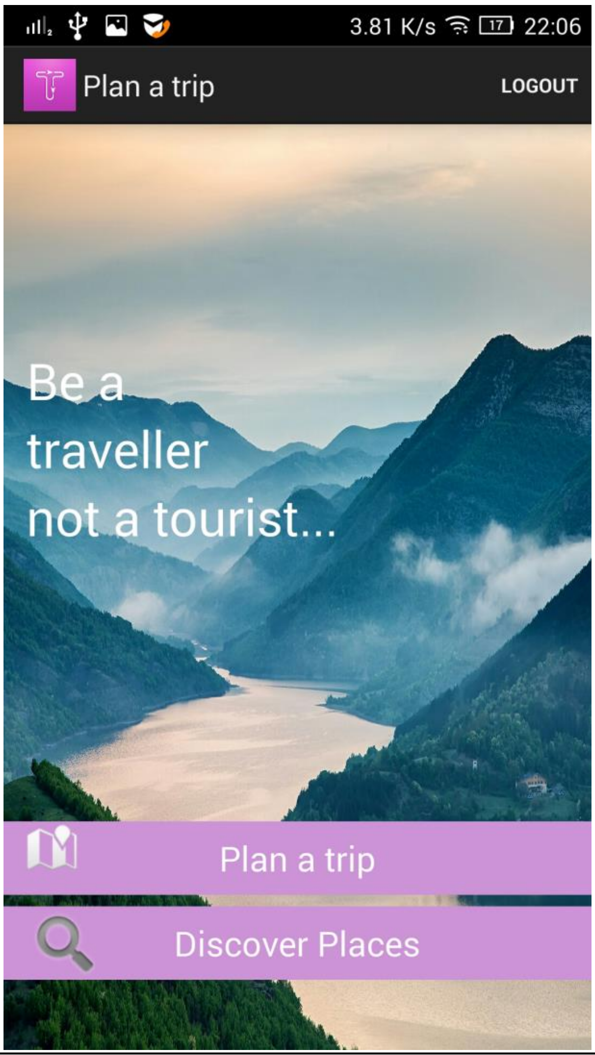
3 Specific Requirements

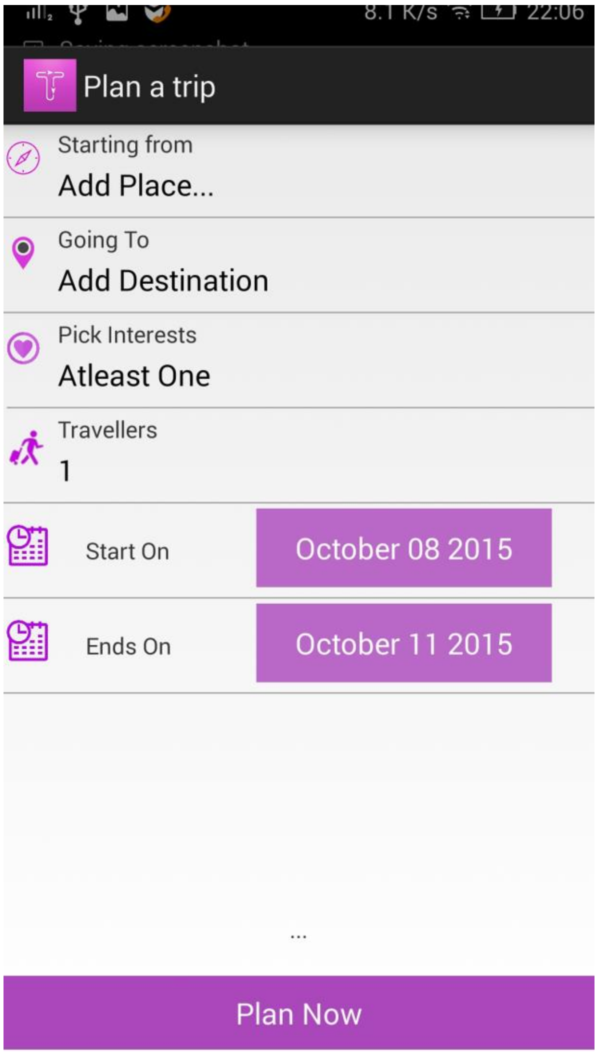
3.1 External Interface Requirements

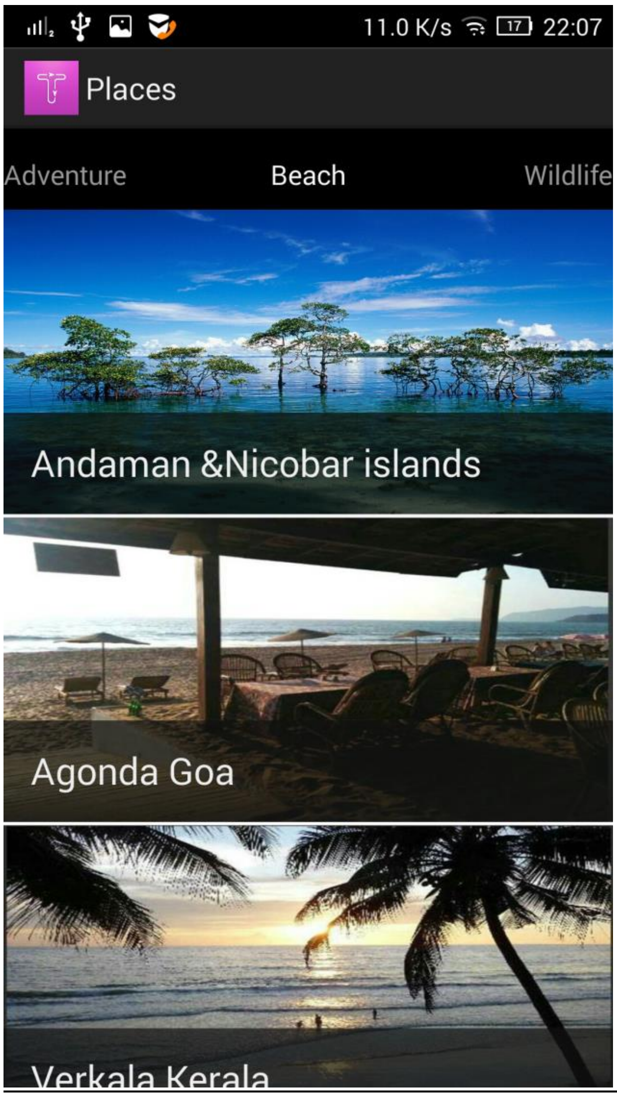
3.1.1 User Interfaces

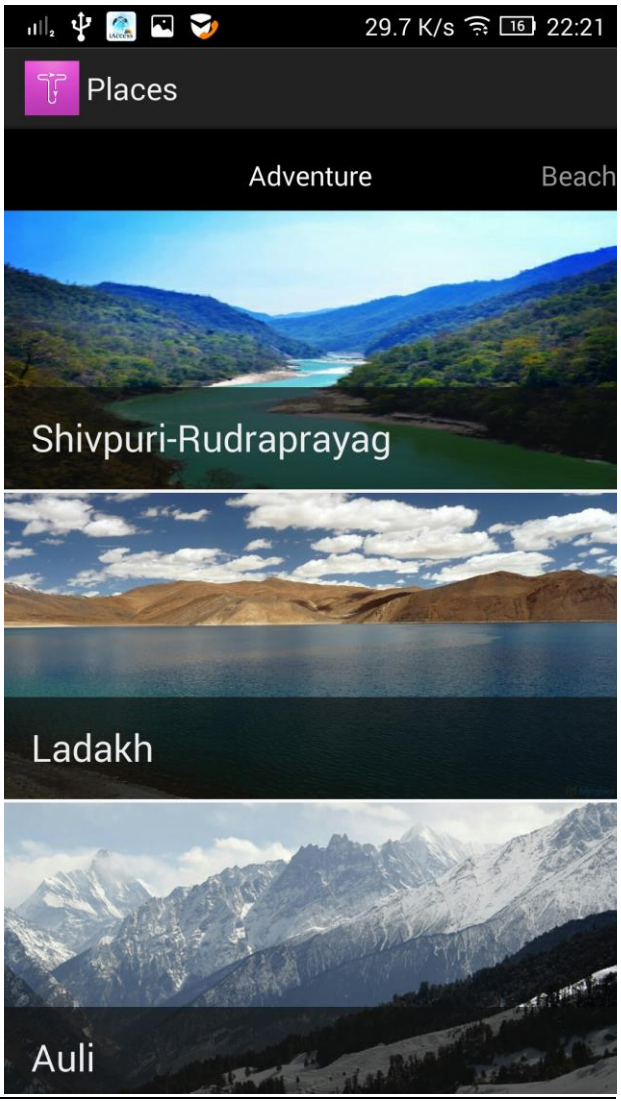





3.81 K/s 22:06
Plan a trip
LOGOUT
Be a traveller not a tourist...
Plan a trip
Discover Places


8.1 K/s 22:06
Plan a trip
Starting from
Add Place...
Going To
Add Destination
Pick Interests
Atleast One
Travellers
1
Start On
October 08 2015
Ends On
October 11 2015
...
Plan Now


11.0 K/s 22:07
Places
Adventure
Beach
Wildlife
Andaman &Nicobar islands
Agonda Goa
Verkala Kerala


29.7 K/s 22:21
Places
Adventure
Beach
Shivpuri-Rudraprayag
Ladakh
Auli

3.1.2 Hardware Interfaces

The Application is intended as a mobile application for the Android Devices and hence is solely supported on Android-powered devices. Updates and data exchanged between Android devices are transmitted to and handled by the host server. It basically replicates a client server model, where a user requests certain data from the user and the server replies with the most suitable results.

We are basically using the cloud services provided by the parse.com . Parse also lets you store data on Android device itself. We can use this for data that doesn't need to be saved to the cloud , but this is especially useful for temporarily storing data so that it can be synced later. Information will be sent using TCP/IP and the HTTP protocol using the port number 80.

The Android platform provides abstractions for all network communications interfaces and thus thee hardware as well.

3.1.3 Software Interfaces

The app is to be developed under the Android operating systems using the Java JDK (Java Development Kit) , the Android SDK (software development kit) tools and the Parse SDK.

Incoming and Outgoing Items

- Outgoing data consists of user preference that is coordinates of the current location, type of holiday, budget , destination, number of people to the server.

- Incoming data consists of various itineraries produced by the algorithm running in the backend , updates regarding the bug fixes and performance improvements will be communicated.

Services and Communications

- The application relies on server push and pull protocols to be fully functional
- Communication will occur in occasional, short bursts between the user’s phone and the server in the following situations:
  A. Whenever user request a itinerary.
  B. Whenever server updates its database.
  C. The application will notify the server when it successfully receives an update.

3.1.4 Communications Interfaces

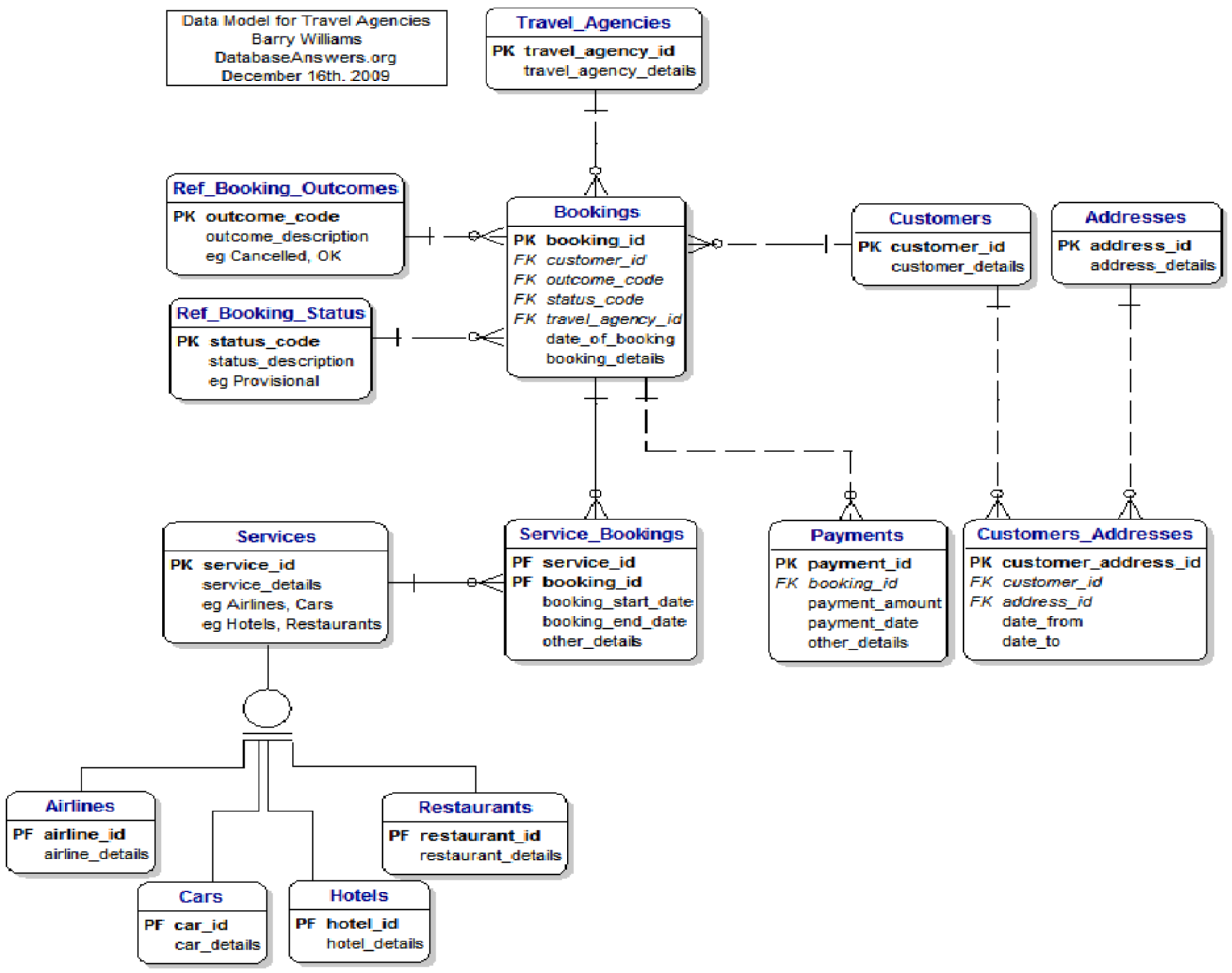
The application will have a cloud server that is web-based and created using the JAVA language. The server exists to retrieve information from the database and calculate the amount in itinerary . The product also calls for a database system that stores users information and preferences. The HTTP server will use a push protocol to push notifications of updates onto the Android phones. Furthermore, whenever a user opens the app from their phone, a pull protocol will be used to retrieve and sync the latest updates from the server.

3.2 Functional requirements

1) Login and Signup for the user
2) Post intelligence item
    1) required meta-data spatial: location/region , index timestamp
    2) optional meta-data such as source , category , team , privacy (partly defined . . application).
    3) optionally transform by security / group membership fields header
3) Filter / retrieve intelligence items
   By timespan / route / geometry
4) by security level / group membership.

3.3 Behaviour Requirements

3.3.1 Class Diagram



4. Non-functional Requirements

1) Response times - How long should your app take to load ? What about refresh or choreography
2) Processing time
Can I get a spinning beachball please ? How long is acceptable to perform key functions or export / import data?

3) Query and Reporting time

This could be covered off with general reporting times, but if you're providing an API you should probably consider acceptable query times too.

4) Throughput

Think about how many transactions your system needs to handle. A thousand a day? A million? When Amazon solved this for their needs, they decoupled systems and created a queue service that became the foundation of AWS.

5) Storage

How much data are you going to need to store to do the awesomeness you need it to do?

6) Hours of operation

When does your app need to be available? If you need to do a database upgrade or a system backup, can you take the system offline while you do it?