WikiWalks – Architecture

# Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation, such as as WikiWalks’ dependency on Google Maps, and a reliable back-end server. It also sets out to describe the reason certain platforms were chosen for front- and back-end development.

# Architectural Goals & Philosophies

The WikiWalks Application is being developed using a waterfall methodology. This was decided as it allows WikiWalks to plan all the features in advance, and helps to develop a better vision of what WikiWalks will be so that, when the development process reaches the implementation stage, the system can be more easily designed. The WikiWalks application will be designed with a focus on simplicity and usability for its users. Because WikiWalks uses community-driven data, the more people who use the app the more data the app is able to collect to create its PathMaps. Other key priorities of the app’s development include a focus on maintainability, as WikiWalks hopes to be able to update the application with ease. Recoverability is also another key priority of the WikiWalks application; this is due to WikiWalks being reliant on a database to store all its data. If this database was damaged, the app would no longer work. Lesser, but still important architectural goals include: capacity, as many people could add data to the server or request it at any time, and data integrity as the data going to the server will be available for everyone. WikiWalks needs to make sure that the data it’s provided is reliable and accurate, and ensure proper security and privacy measures are taken, as the application will have access to user’s location, which needs to stay private.

# Assumptions & Dependencies

When designing the WikiWalks application, it has been assumed that users will have had experience using basic mobile apps and that they will know how to operate a mobile application. WikiWalks will be designed to work on GPS enabled android phones running Android 4.1 (Jelly Bean) or higher, and assume that users will have access to an internet connection to download and upload map data from WikiWalks’ server. WikiWalks will be dependent on users having access to a phone that meets these requirements. The application will also be dependent on Google Maps to provide the map.

For developing the back-end, WikiWalks is dependent on a reliable internet connection. A server to store and process incoming path data and way of backing up data without causing disruptions to the server is required.

# Architecturally Significant Requirements

Significant architectural requirements include:

* Data backups being done regularly without disrupting user access
* Paths being able to be removed from the database
* Handling faulty data such as a path being created in the middle of the ocean
* No less than 10 minutes of server downtime a week
* Combining the data of multiple path recordings to more clearly show the track
* The ability to update the application without affecting the servers
* The ability to update the servers without affecting the application
* The ability for the WikiWalks server to handle multiple users uploading and requesting data at the same time

# Decisions, Constraints, & Justifications

Decisions:

* The WikiWalks team is developing our application for Android 4.1 (Jelly Bean). This was chosen as applications developed to run on at least 4.1 are able to be run on 99.6% of Android devices.
* WikiWalks has chosen to use Google Maps to provide its mapping data and to show user location. This was done as the Google Maps API is widely used and well-documented. Google also provides $200 worth of credit each month, which means that its implementation will likely come at no financial cost to the team.
* WikiWalks has chosen to implement a way for users to remove or request removal of a path. This was done so that paths that go over private property or are dangerous can be removed.
* WikiWalks will implement a feature to allow users to add points of interest to paths they have been on so that other users are able to navigate more easily, and set a destination to a POI they want to get to on their walk.
* The WikiWalks back-end will be developed using Python and the database where path data is stored will use SQL. Python and SQL were chosen due to the WikiWalks team having had experience in both languages.
* The WikiWalks front-end will be developed using Java in Android Development Studio. This was selected due to WikiWalks having experience in developing programs in Java, as well as having experience using Android Development Studio.
* WikiWalks version control will use GitHub. GitHub was selected as all team members of the WikiWalks app have had experience using GitHub.
* WikiWalks will be able to run in the background and use low power so as to not drain the user’s battery. This was decided as, if the application drains user’s power quickly, users would be less inclined to use it.
* Each track will be a different colour depending on how often it walked by users. This decision was made as users may want to see path popularity, and take paths that are travelled more often.
* WikiWalks will implement a “navigate to” feature that will allow users to navigate to a location using data it has already stored in the back-end database.
* WikiWalks will implement a group walk feature so that multiple users can meet up and walk a track together.

Constraints:

* WikiWalks has less than a year to be completed. Due to this time constraint, the development team needs to stay on schedule and not fall behind.
* WikiWalks is constrained by some Google Maps features costing money to use. To work around this, our team is developing the application to interact with Google Maps as infrequently as possible.

# Architectural Mechanisms

## Usability

WikiWalks must have a high level of usability for users of all skill levels. WikiWalks will also be designed so that it is able to be run in the background of phones without using a significant amount of power, allowing users to take longer walks.

## Relational Databases

WikiWalks contains a relational database to store PathMaps, reviews, points of interest and images. This is to increase usability and performance of the system as it will allow PathMaps to be shared and users to review tracks they take, further contributing to the crowdsourced data.

## Backups

Regular weekly backups of the WikiWalks database will be done. This to increase usability and recoverability. If the WikiWalks database is ever corrupted or has some other issue causing it to go down, all users will be affected with the PathMaps and other info stored on the database becoming inaccessible to the user. This also stops them being able to contribute more paths to the database. The backups aim to mitigate this by allowing quick recovery.

## Network Connection

Networks are used to allow users to submit data to the WikiWalks servers, request the PathMap for an area to store in the phones cache, and get maps from Google Maps. A connection must be made at some point after a new walk is completed to allow users to upload their path data.

## Removing Paths

The WikiWalks relational database will contain the ability to remove paths. This is to help increase security and usability as paths may no longer be accessible or may become dangerous to take.

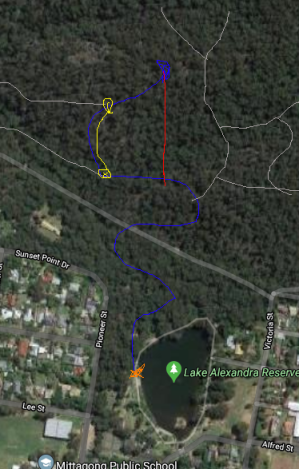
## Security

Verification is done on all incoming data to the WikiWalks servers. This is done to increase usability and reliability as data comes into WikiWalks. WikiWalks will check no points are in inaccessible areas, e.g. lakes, and that the data is consistent and reliable, so that the data of people who have driven while using WikiWalks or have changed the time while using the application can be filtered.

Additionally, as little user data as possible will be collected, and what is collected will be anonymised, meaning it’s more difficult to track users in the case of a data breach on the WikiWalks servers.

# Key Abstractions

|  |  |
| --- | --- |
| Key Abstraction | Description |
| User | Someone who uses the WikiWalks application |
| Track | A section of path that can make up a track, often an alternate route |
| Path | A combination of multiple tracks combined together, the complete route that a user takes on a walk |
| PathMap | A map of all tracks in an area combined |
| Front-end | What the user sees when using the WikiWalks application |
| Back-end | The part of the application that the user does not have access to, such as the application’s logic or WikiWalks’ server and database |
| GPS | Global Positioning System, a way of finding a user's exact location |
| Google Maps | A mapping service and API provided by Google |
| App | Abbreviation for application |

Key:

* Blue line is the path a user took to get to a destination
* The yellow line is a track for an alternate route on the path
* The grey lines are other paths on a PathMap

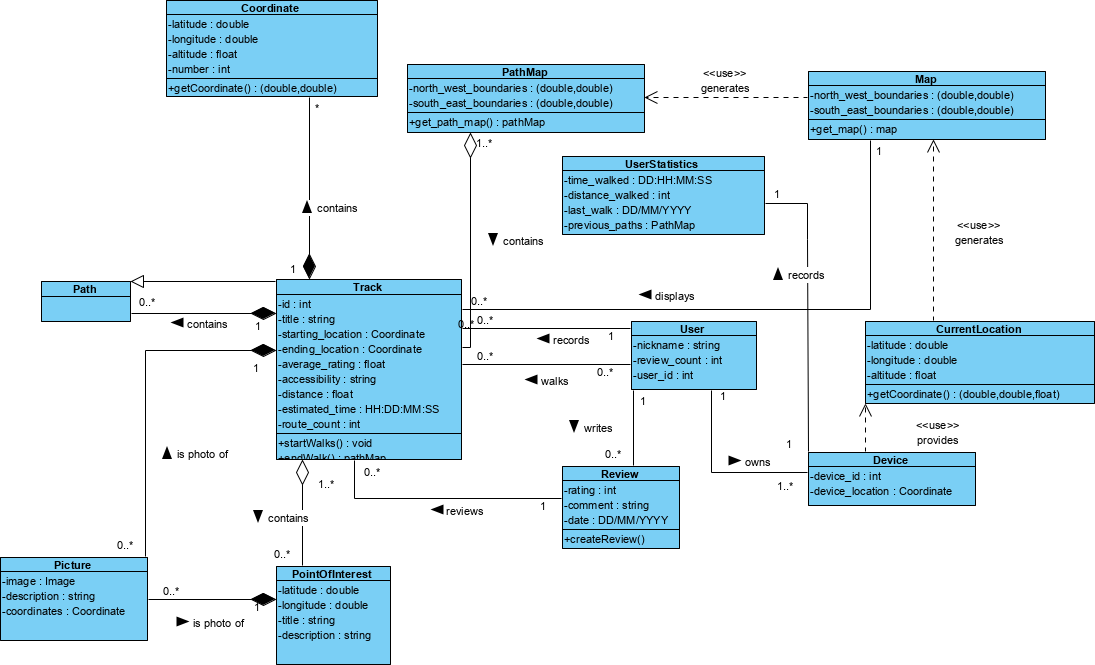
# Layers or Architectural Framework

WikiWalks is going to use a four-tier architecture that contains:

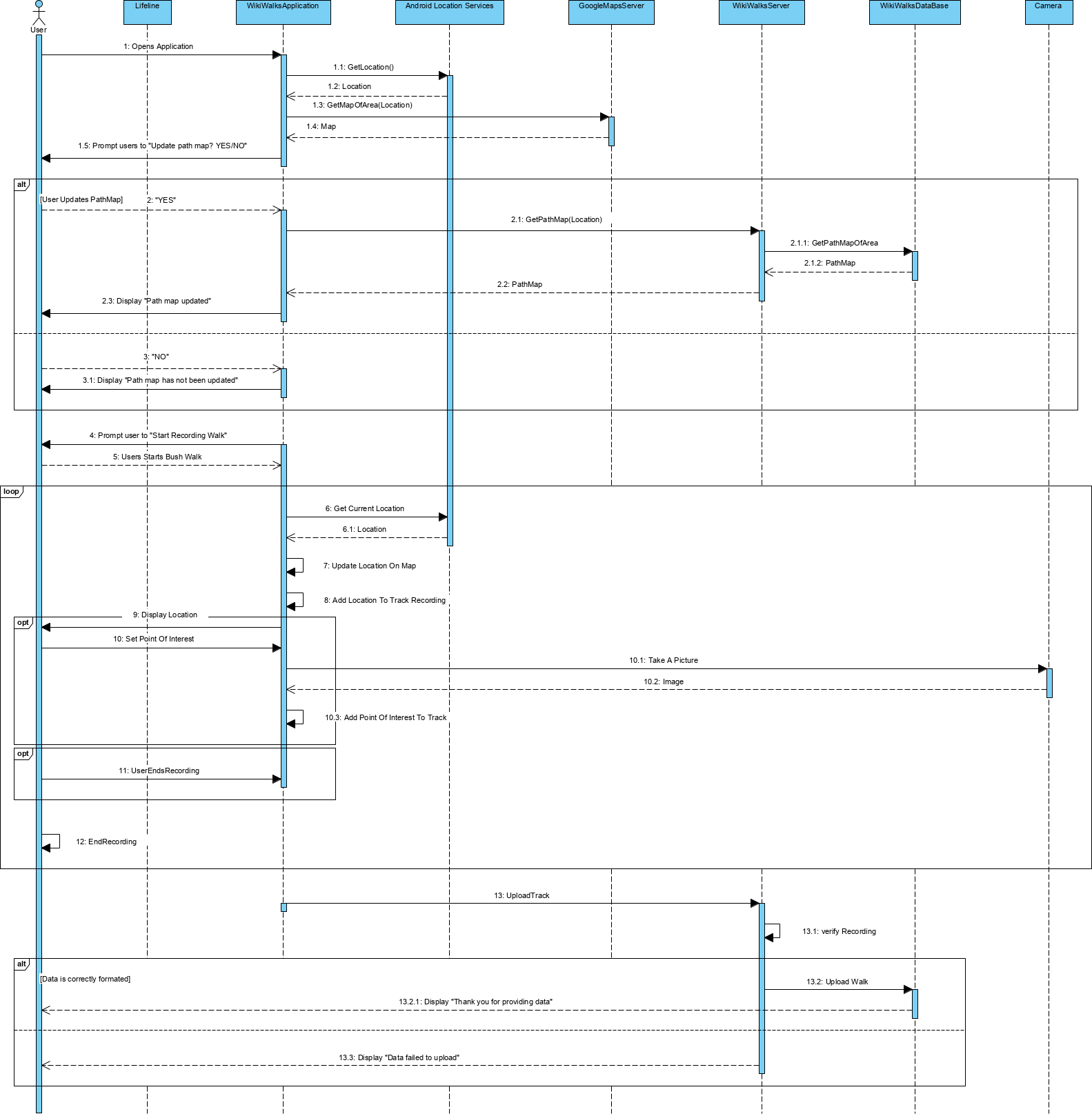
1. A presentation tier. This is the part of the WikiWalks application that the users see and interact with. It contains data stored on the phone itself such as a cache of the PathMap of an area. This tier interacts with the application logic tier, and also pulls maps from Google Maps.
2. An application logic tier. This is where processes for the WikiWalks application take place, such as recording paths or creating points of interests. The application logic tier interacts with the presentation tier and sends data to the data processing tier.
3. The data processing tier receives and validates data sent via the application, and aggregates it with data stored in the storage tier to create clearer tracks. This data is then uploaded to the storage tier. It also pulls and responds with data from the storage tier when requested.
4. The storage tier contains all paths and points of interest in a database. This tier provides data for the data processing tier.

# Architectural Views

## Logical View – Class Diagram

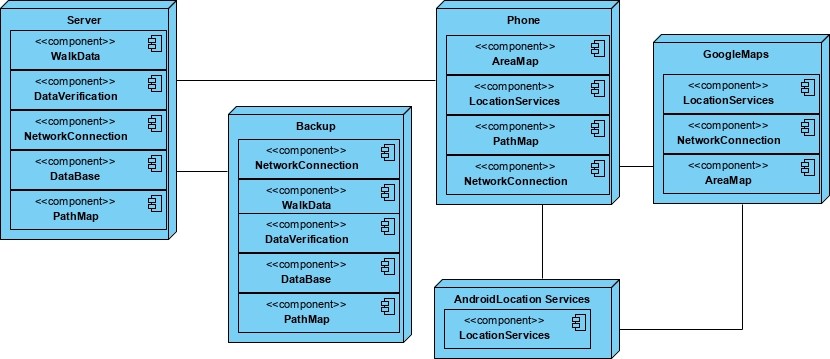


## Process View – Sequence Diagram



## Development View – Component Diagram

## Physical View – Deployment Diagram



## Scenarios – Use Case

