Detailed Design

|  |  |
| --- | --- |
| Authors | Niv Ben-David & Michael Dimenstein |
| Version | 1.0 |
| Created | 28/1/2012 |
| Last Update | 20/3/2012 |
| Current Status | Final Version |

Contents

[1 Introduction 3](#_Toc315530661)

[1.1 Overview 3](#_Toc315530662)

[1.2 Design Goals and Non-goals 3](#_Toc315530663)

[1.2.1 Goals 3](#_Toc315530664)

[1.2.2 Non-goals 3](#_Toc315530665)

[1.3 Dependencies, Assumptions and Design Constraints 4](#_Toc315530666)

[1.3.1 Dependencies 4](#_Toc315530667)

[1.3.2 Assumptions 4](#_Toc315530668)

[1.3.3 Design Constraints 4](#_Toc315530669)

[1.4 Audience 4](#_Toc315530670)

[1.5 Issues 4](#_Toc315530671)

[1.6 To-Do List 4](#_Toc315530672)

# Introduction

## Overview

Hitch! is a socially-based transportation solution devised to connect between consumers of (one-time and/or repeating) rides and the providers of such rides such as, but not limited to, private car owners, taxi drivers/stations, car-pooling companies, etc.

Unlike other hitchhiking projects abundant on the net, Hitch! gives the user a large suite of different ways to get from point A to point B, everything in order to provide the user with the product which they require.

## Design Goals and Non-goals

### Goals

Hitch! will define a connection between two types of users: a passenger and a driver, as well as the ‘Currency’ which they trade – a ride. A ride can be either one-time or repeating.

The application will support the following features:

1. Create a new user profile.
2. Create a new one-time ride.
3. Request to join an existing ride.
4. Approve/reject a join request to a ride.
5. Search for a ride by a pre-defined set of conditions (time, origin, destination, etc.).

In addition, the application would be able identify a user by their Facebook profile and register them via the data provided by the Facebook application APIs (Open Graph or FQL). Users will be able to see other users with whom they have mutual friends, interests, workplaces, etc.

The application will have a suggestion functionality that will enable users to see rides happening near predefined places of interest.

### Non-goals

The application may not, at the first stage, support secondary features such as:

1. Deleting an existing user.
2. Editing an existing user account.
3. Deleting an existing ride.
4. Canceling a join request.

Also, the application would not be able to connect to other data providers (like Twitter or Google+), though the option to extend the database to those providers will be considered during the design and implementation processes.

## Dependencies, Assumptions and Design Constraints

### Dependencies

In order to connect between the smartphone end-point and the server an internet connection is required. The application may also depend on a location provider (such as a GPS) system in the smartphone and 3rd -party map providers (such as the stock map application of the Windows Phone or Google Maps).

As a consumer of social network data, the application will depend on the data provided by the Facebook developer APIs (Open Graph or FQL) and the OAuth 2.0 protocol access that data.

### Assumptions

Since the primary users of the application in its current state can be safely assumed to be only those involved in its production and members of the Tel-Aviv University's Department of Computer Science faculty, input from the user will be treated as safe and not intended to attack or harm the application or server; only basic safety checks will be performed on user input on the site or application.

### Design Constraints

Though the application will support a requested/proposed fee field for rides, the exchange system itself will not be implemented and will be assumed to work without interference from the application.

Also, as the application cannot control users’ behavior, users will be assumed to behave accordingly, i.e. pay their passengers fee, arrive to their meeting points (either as drivers or passengers) on time, etc. No functionality to "report" misbehaving users will be implemented.

## Audience

The application is intended for users who are seeking either one-time rides to some destination or a repeating ride to a usual location as part of their routine (work, etc.) as well as for drivers who wish to add passengers to rides they already planned (either as their routine or occasionally) that will accompany them and share the ride’s expenses, or in the case of professional drivers such as bus or cab drivers, pay the ride's fee.

## Issues

There are no fundamental issues foreseen at the current stage.

## To-Do List

|  |  |
| --- | --- |
| Item | Status |

# Logical Architecture

## Application Context

Azure

Hitch! Server

Network Connection

Hitch! Application

Phone’s GPS

Social Networks’ APIs

Facebook

Google+

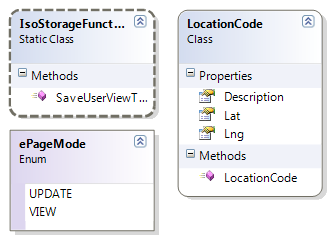
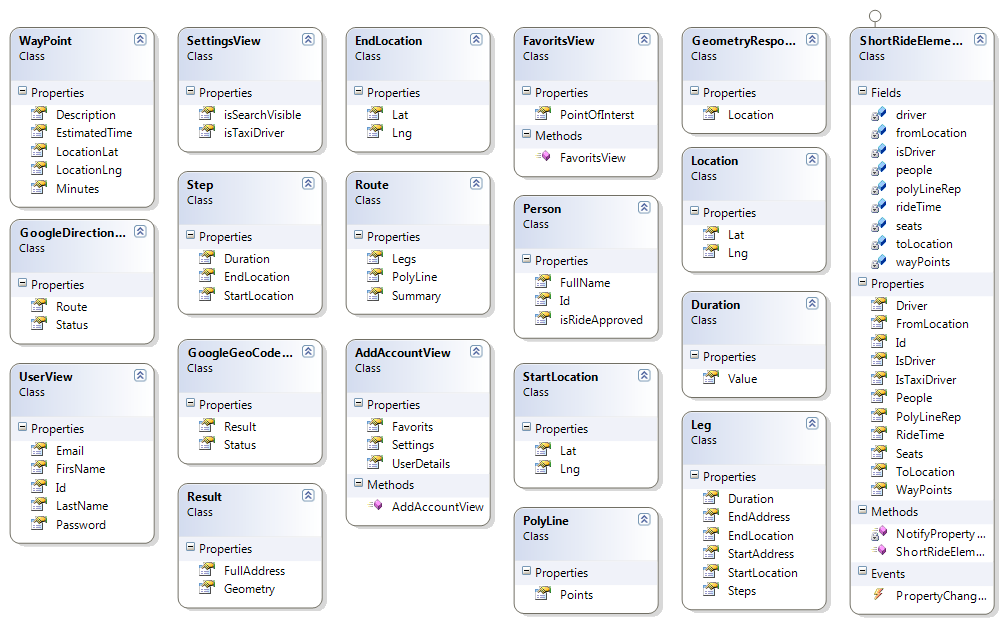
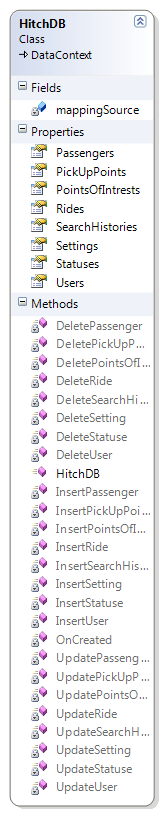
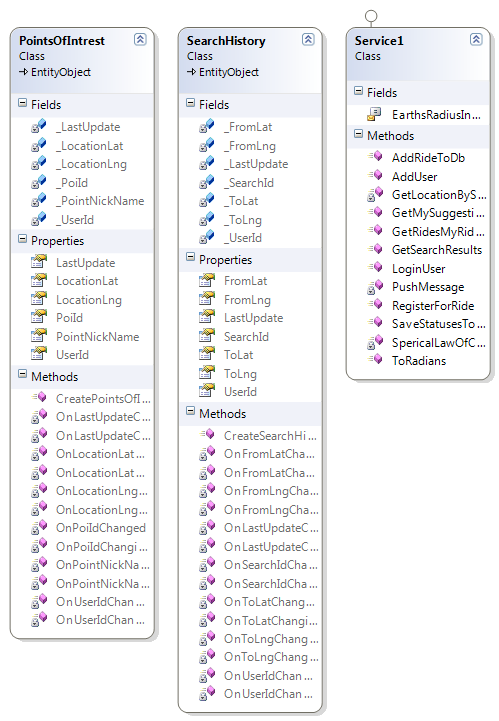
Twitter

Map Application

# Design

## Classes

### Diagram

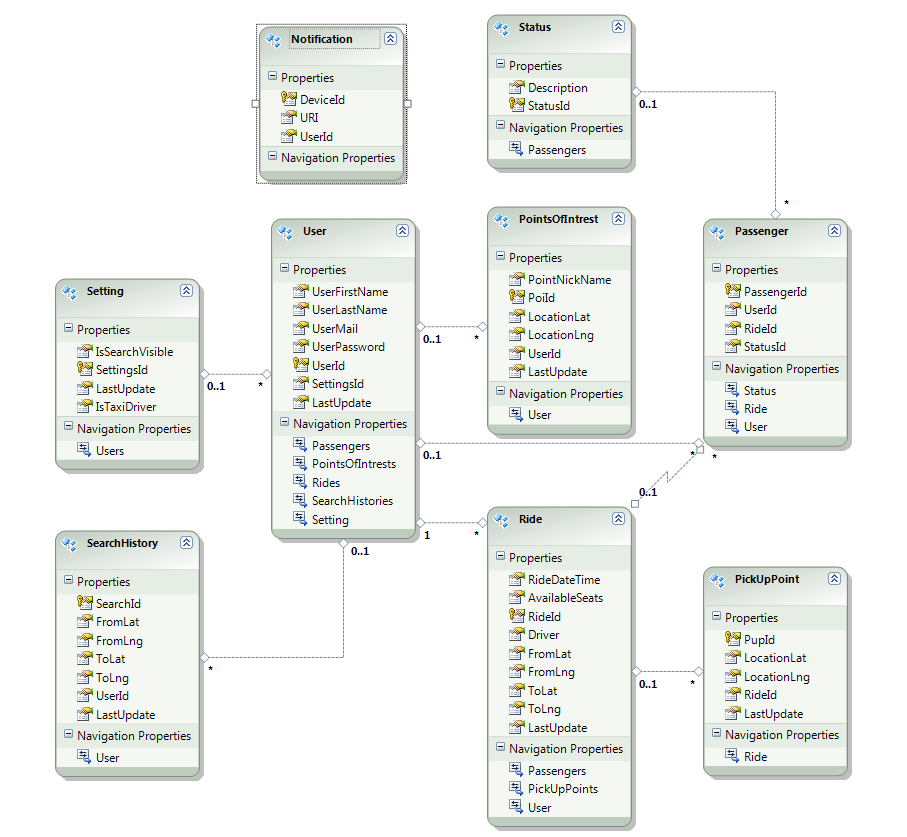
  
  
  


### Details

1. Hitch (namespace)
   1. Hitcher (class)  
      The basic entity used by both the server and application to handle the different users, rides, groups, etc. Each object holds a list of Aspect pointers that represent the various attributes dynamically associated with it (for example, an object with the Follower aspect can follow and be followed by other objects).
   2. Aspect (namespace)
      1. Base (abstract class)  
         The base class for all the various aspect classes, defining the interface with which the Hitcher objects interact with their composing aspects.
      2. Database (abstract class)  
         A base class for aspects linked with a database record, implementing utility functions for such classes.
      3. Group (class)  
         A Hitcher with the Group aspect represents an aggregation of other Hitcher objects, delegating event handling to its members.
      4. Follower (class)  
         An entity with the Follower aspect can follow and be followed by other objects
      5. Position (class)  
         A Hitcher’s positional data, stored in the server’s database.
      6. Auth (abstract class)  
         A base class for aspects holding a user’s authentication data in social networks, implementing utility functionality to gain access to the user’s social profiles.
      7. Facebook/GooglePlus/Twitter (class)  
         Implementation of the Auth abstract for each major social network.

## Database

### Diagram

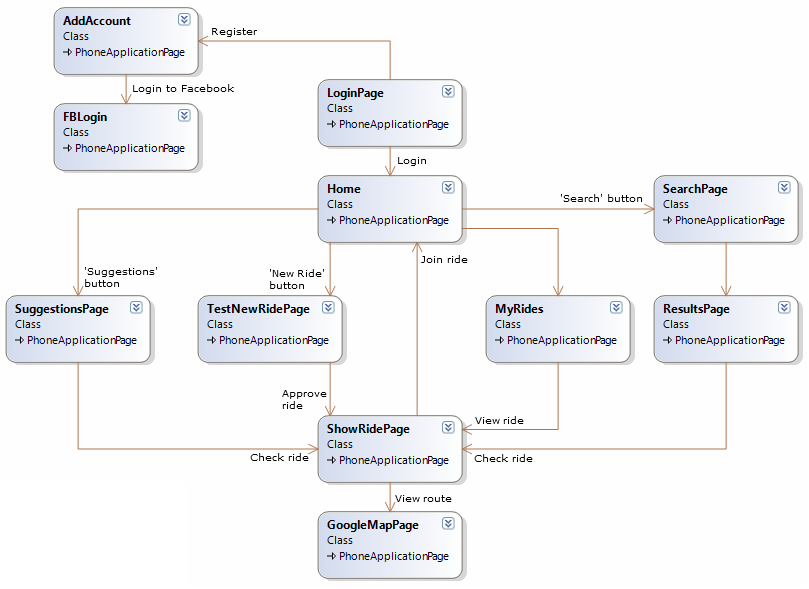


### Details

1. User  
   The application's users table, holding basic user data and defines a user entity's unique identifier.
2. PointsOfInterest  
   A bank of defined waypoints used to translate geo-data (latitude/longitude) to user's search text and vice-versa.
3. Ride  
   Each ride is stored with a uniquely generated ID and is mapped to the driver's unique ID. Route's beginning and end waypoints are stored for quick access.
4. Passenger  
   Each passenger request is stored as a ride-user couple, along with the request's status.
5. PickupPoints  
   Mid-points along a ride's route for passenger pick-ups. Driver-defined when creating/editing a ride.
6. SearchHistory  
   Search log, used to intelligently suggest contacting other users who've made similar searches to the user's.
7. Setting  
   Users' settings.
8. Status  
   Ride request status identifiers.
9. Notification  
   Device push-notifications.

## Flow

### Diagram



### Details

1. LoginPage  
   The application's login/register page. When not logged-in, every page leads back to this one.
2. Home
   1. SearchPage  
      Create a new search, leads to the Results page when done.
   2. MyRides  
      Shows a user's upcoming rides (both as a driver and a passenger).
   3. SuggestionsPage  
      Suggested rides according to a user's points of interest.
   4. TestNewRidePage  
      Create a new ride as a driver.
3. ResultsPage  
   A ride search results page, listing the query's contents in a list control. Each list item leads to the Ride View page. At the end of the list there's button leading to the Similar Searches page.
4. ShowRidePage  
   Displays a ride's details (from, to, when, etc.) for viewing only. The ride's driver's page displays a button leading the Ride Edit page.
5. GoogleMapPage  
   Displays a ride's route on the Google Maps control. Pickup Points can be edited from here.
6. AddAccount  
   A user's profile page.
7. FBLogin  
   A Facebook login page.
8. Settings  
   A user's settings page.
9. Application Bar  
   The application bar at the bottom of the screen always displays shortcuts to the user's profile, new ride and settings pages.

## Synchronization and Protection Mechanisms

No mechanisms are planned at the moment.

# Physical Architecture

There aren’t any physical devices needed aside a smartphone operating on Windows Phone 7.1.

# References

1. Google Maps API
2. Facebook Open Graph API
3. WP7 Silverlight toolkit

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Revision |
| 28/1/2012 | 0.0 | Niv Ben-David | Created the first document draft. |
| 28/1/2012 | 0.1 | Niv Ben-David | Added rudimentary class designs. |
| 30/1/2012 | 0.1 | Miki Dimenstein | Minor changes around the document. |
| 16/2/2012 | 0.3 | Miki Dimenstein | Added the database schema. |
| 18/2/2012 | 0.3 | Niv Ben-David | Added the Entity System class design. |
| 24/2/2012 | 0.4 | Niv Ben-David & Miki Dimenstein | Made a few changes to the database, decided to refer to waypoints in a separate table. |
| 25/2/2012 | 0.5 | Miki Dimenstein | Dropped the Entity System design for a more simple solution. |
| 29/2/2012 | 0.6 | Miki Dimenstein | Updated the pages flow diagram. |
| 1/3/2012 | 0.7 | Miki Dimenstein | Updated the Goals/No-Goals sections with an accurate description of the suggestions function. |
| 3/3/2012 | 1.0 | Niv Ben-David | Minor changes and document's final version. |
| 22/3/2012 | 1.0 | Niv Ben-David | Made sure diagrams are up-to-date with current code design. |