TravelMatch

Software Transfer Document

Version 1.0

D.J. van den Brand (0772180)
S. He (0810831)
J.M.A.P. van Hoof (0778486)
G.C. Linders (0815449)
M.J.M. Muijsers (0805654)
G.H. Santegoeds (0890429)
L.D. Stooker (0819041)
J.W.J.H. Visser (0828234)

22nd June, 2015
Abstract

This is the Software Transfer Document (STD) for the TravelMatch application. This document is based on the ESA standard for software development. This document complies with the ADD from the Software Engineering Standard, as specified by the European Space Agency (ESA) [1].
Contents

Document Status Sheet

General .................................... 3
Document history .............................. 3

Document Change Records

General .................................... 4
Changes ..................................... 4

1 Introduction

1.1 Purpose .................................. 5
1.2 Scope .................................... 5
1.3 Definitions and abbreviations
   1.3.1 Definitions ......................... 5
   1.3.2 List of abbreviations .............. 6
1.4 References ............................... 6
1.5 Overview ................................ 8

2 Build Procedure

2.1 Configuring environment ............... 9
2.2 Building the application
   2.2.1 Prerequisites ...................... 10
   2.2.2 Build process .................... 10
2.3 Server
   2.3.1 Prerequisites ...................... 11
   2.3.2 Building server ................. 11
2.4 Testing
   2.4.1 Front end ......................... 12
   2.4.2 Back end .......................... 12

3 Installation Procedure

3.1 Node and NPM
   3.1.1 Windows .......................... 13
   3.1.2 Linux ............................. 13
   3.1.3 Mac ............................... 13
3.2 Git/GitHub
   3.2.1 Windows .......................... 13
   3.2.2 Linux ............................. 13
   3.2.3 Mac ............................... 13
3.3 Java Development Kit
   3.3.1 Windows .......................... 14
   3.3.2 Linux ............................. 14
3.4 Android SDK
   3.4.1 Windows .......................... 14
   3.4.2 Linux ............................. 14
3.5 Xcode
   3.5.1 Mac ............................... 14
Document Status Sheet

General

Document title: Software Transfer Document
Document identifier: TravelMatch.Doc.STD/1.0
Authors: D.J. van den Brand (0772180)
S. He (0810831)
J.M.A.P. van Hoof (0778486)
G.C. Linders (0815449)
M.J.M. Muijsers (0805654)
G.H. Santegoeds (0890429)
L.D. Stooker (0819041)
J.W.J.H. Visser (0828234)

Document status: Final document

Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author(s)</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>09-06-2015</td>
<td>J.W.J.H. Visser</td>
<td>Creating initial setup of the document.</td>
</tr>
<tr>
<td>0.2</td>
<td>20-06-2015</td>
<td>J.W.J.H. Visser</td>
<td>Adding new content, overview and implementing feedback.</td>
</tr>
<tr>
<td>0.3</td>
<td>21-06-2015</td>
<td>J.W.J.H. Visser</td>
<td>Adding extendability chapter, finish configuration item-list, adding software problems, better AT summary, making last two chapters not applicable.</td>
</tr>
</tbody>
</table>
Document Change Records

General

- Date: 22nd June, 2015
- Document title: Software Transfer Document
- Document identifier: TravelMatch.Doc.STD/1.0

Changes
Chapter 1
Introduction

1.1 Purpose
This document gives all information concerning the transfer of the product from the TravelMatch group to the customer. It describes the procedures for building setting up and building the TravelMatch application. Furthermore, it describes the items to be transferred and how the product has been tested prior to the transfer. Finally, it describes the state of these items compared to the requirements from the URD.

1.2 Scope
TravelMatch is an application designed for smartphones and tablets, conceived by iLysian B.V. and developed by the TravelMatch development team. The purpose of the application is to assist users in planning a vacation by showing them images from various destinations and hotels or other places to stay. The application employs machine learning to build a profile of the user in order to suggest the ideal trip.

1.3 Definitions and abbreviations
1.3.1 Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affiliate Network</td>
<td>A network that enables you to receive money from customer redirection [18]</td>
</tr>
<tr>
<td>Analytics Data</td>
<td>The log of analytics events that is recorded and stored on the database.</td>
</tr>
<tr>
<td>Android</td>
<td>A popular open-source operating system for embedded devices, including</td>
</tr>
<tr>
<td></td>
<td>smartphones and tablets, created by Google.</td>
</tr>
<tr>
<td>Angular JS</td>
<td>An open-source web application framework maintained by Google.</td>
</tr>
<tr>
<td>Cosine similarity</td>
<td>A measure of similarity between two vectors of an inner product space that</td>
</tr>
<tr>
<td></td>
<td>measures the cosine of the angle between them.</td>
</tr>
<tr>
<td>Destination advice</td>
<td>The city, and selection of hotels, that is advised to a user after performing</td>
</tr>
<tr>
<td></td>
<td>one or more interest analyses.</td>
</tr>
<tr>
<td>DNA attribute or tags</td>
<td>Each destination will have one or more destination attributes with an</td>
</tr>
<tr>
<td></td>
<td>associated numerical relative value, those attributes cover the same</td>
</tr>
<tr>
<td></td>
<td>preferences as the DNA attribute.</td>
</tr>
<tr>
<td>DNA attribute or tags</td>
<td>These are the attributes that the client wants to use to compose the DNA</td>
</tr>
<tr>
<td></td>
<td>of. In the beginning 10 attributes are chosen and each image shall have a</td>
</tr>
<tr>
<td></td>
<td>relative numerical value on one or more of the attributes. Attributes can be</td>
</tr>
<tr>
<td></td>
<td>added or removed later for new and existing images and DNA.</td>
</tr>
<tr>
<td>Google Play Store</td>
<td>A public repository of free and paid apps for Android, managed by Google.</td>
</tr>
<tr>
<td>Guest user</td>
<td>An user that does not provide login details but still uses the TravelMatch</td>
</tr>
<tr>
<td></td>
<td>app.</td>
</tr>
</tbody>
</table>
Hotelstars rating  A hotel classification with common criteria and procedures in participating countries to rate a hotel’s quality. See [21].

iLysian  Short for iLysian B.V., a software engineering company situated in Eindhoven, Netherlands. The client for the TravelMatch project.

Interest analysis  The action the user will do of judging the images.

iOS  A popular closed-source operating system for smartphones and tablets created by Apple.

iOS App Store  A public repository of free and paid apps for iOS, managed by Apple.

JWT  JSON Web Token: a compact URL-safe means of representing claims to be transferred between two parties, and used in TravelMatch as authentication token, since it is self-validating.

Relational database management system (RDBMS)  A database management system (a piece of computer software that interacts with users, other applications and a database to capture and analyze data) based on the relational model (commonly based on the relational database model)

TCP/IP  A computer networking model and set of communication protocols used on the internet and similar computer networks, including the Transmission Control Protocol (TCP) and the Internet Protocol (IP)

Tinder  A popular dating application for smartphones and tablets featuring a swipe based interface, where a swipe to the left indicates a dislike and a swipe to the right indicates a like.

Travel DNA  A collection of information about vacation preferences of a specific user or, more specifically, one vacation of that user. This information is stored on the server in a table with values representing the respective gain per attribute for each image the user has swiped.

TravelMatch  An application for smartphones and tablets that assists users in planning a vacation. The subject of this project.

TravelMatch team  A team of Computer Science students at Eindhoven University of Technology who will design and implement the TravelMatch application.

User  The user of the app.

Waverunner  Waverunner Search Service by Pyton Communication Services; a search service that provides vacation offers and prices of participating travel agencies.

1.3.2 List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPM</td>
<td>Node Package Manager</td>
</tr>
<tr>
<td>CMS</td>
<td>Content Management System</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</td>
</tr>
</tbody>
</table>

1.4 References

[1] ESA PSS-05-0 Issue 2, Software requirements and architecture engineering process, February 1991


1.5 Overview

The remainder of the document consists of all information regarding the transfer of the product. Chapters 2 and 3 go in-depth about the build and installation procedure of the software. Chapter 4 contains ways of extending the contents of TravelMatch by adding a language file for different languages. Chapter 5 concerns an overview of all deliverables. Chapter 6 contains a summary of the Acceptance Test. Finally, chapter 7 states all issues that are currently residing in the software. Chapters 8 and 9, which concerns the change requests and modification reports, are not applicable for this project.
Chapter 2
Build Procedure

2.1 Configuring environment

This section focusses on the prerequisites for the build procedure. To retrieve, build and deploy the TravelMatch application on a mobile phone, there needs to be an installation of Node and NPM, Git and either Android SDK or Xcode. Android SDK is used for building on Android phones; this section will describe the installation for Windows and Linux. Xcode is used for building on iPhones, and a laptop or computer with OS X is necessary.

1. Copy everything from the CD into a local folder.

2. Install Node and NPM
   (a) Windows
      • Head over to [https://nodejs.org/](https://nodejs.org/) and download the latest version.
      • Alternatively, use the node installer inside the install/windows folder.
   (b) Linux
      • In the terminal, execute the following commands to install Node with NPM:
        – sudo apt-get update
        – sudo apt-get install nodejs
        – sudo apt-get install npm
   (c) Mac
      • Head over to [https://nodejs.org/](https://nodejs.org/) and download the latest version.
      • Alternatively, use the node installer inside the install/mac folder.

3. Install Git or GitHub
   (a) Windows
      • Head over to [https://windows.github.com/](https://windows.github.com/) and download the latest version of GitHub.
      • Alternatively, use the github installer inside the install/windows folder.
   (b) Linux
      • In the terminal, execute the following commands to install Git:
        – sudo apt-get install git
   (c) Mac
      • Head over to [https://mac.github.com/](https://mac.github.com/) and install the latest version of GitHub.
      • Alternatively, use the GitHub installer inside the install/mac folder.

4. Install Java Development Kit
   (a) Windows
      • Head over to [http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-188.html](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-188.html) and download the latest Java 7 SE Development Kit for your platform.
      • Alternatively, use the Java installer inside the install/windows folder.
5. Install Android SDK

(a) Windows

- Head over to http://developer.android.com/sdk and download the latest version of the Android SDK.
- Alternatively, use the Android SDK installer inside the install/windows folder.

(b) Linux

- Head over to http://developer.android.com/sdk and download the latest version of the Android SDK.
- Alternatively, use the Android SDK installer inside the install/linux folder.

6. Install Xcode

(a) Mac

- Install the latest version of Xcode inside the Mac App Store.

2.2 Building the application

The TravelMatch app can be built and deployed by following the below procedure. The prerequisites should have already been fulfilled if the environment has been configured as described in the previous section.

2.2.1 Prerequisites

1. The build PC is prepared for building on Android or iOS.
2. NPM is installed.
3. Git is installed.

2.2.2 Build process

1. Clone the TravelMatch Git repository.
2. Open a console window with admin/superuser privileges and go to the src folder:

   - cd src

3. Create the output directory:

   - mkdir www

4. Use NPM to install gulp, Bower, Cordova and Ionic:

   - npm install gulp bower cordova ionic

5. Install karma-cli globally.

   - npm install karma-cli -g

6. Install all development dependencies:

   - npm install
7. Add Android and/or iOS as Cordova platforms. Note that adding iOS is only possible on PC running OS X.
   • cordova platform add android
   • cordova platform add ios

8. Install all app dependencies:
   • gulp cook

9. Building the app for either Android or iOS:
   • gulp android
   • gulp ios

2.3 Server

The TravelMatch server can be built and deployed by following the below procedure.

2.3.1 Prerequisites
1. Ubuntu 14.04 LTS or a compatible version is running on the server.
2. Python 2.7.6 or a compatible version is installed on the server.

2.3.2 Building server
1. Install Django via pip with following command:
   • sudo python get-pip.py
   • git clone git://github.com/django/django.git django-trunk
   • sudo pip install -e django-trunk/
   • sudo pip install djangorestframework
2. Install Mailgun.
   • sudo pip install -e git://github.com/mailgun/mailgun.py.git#egg=pymailgun
3. Install related Django packages.
   • sudo pip install django.facebook
4. Make migrations for the database in the ~/TravelMatch/server/travelmatch folder.
   • python manage.py make migrations
5. Set up the database. The database has the SQLite engine as its default configuration. This configuration can be changed in the settings.py file. The DATABASES variable must be set according to the Django tutorial. An example of a MySQL configuration may be found below.
   DATABASES = {
       'default': {
           'ENGINE': 'django.db.backends.mysql',
           'NAME': 'DB_NAME',
           'USER': 'DB_USER',
           'PASSWORD': 'DB_PASSWORD',
           'HOST': 'localhost',
       },
6. Initialize the database.
   • python manage.py migrate
7. Start the server.
   • python manage.py runserver 0.0.0.0:80

2.4 Testing

2.4.1 Front end
To test the TravelMatch application, go to ~/TravelMatch/src and execute the following command in the command line after the building procedure of the application:
   • karma start

2.4.2 Back end
To test the back end of the project, go to the ~/TravelMatch/server/travelmatch folder and execute the following command:
   • python manage.py test
Chapter 3

Installation Procedure

This section contains additional notes regarding the environment configuration. We assume that the user follows the build procedure in chapter 2.1, as it contains links to all installers and commands required to install the software.

3.1  Node and NPM

3.1.1  Windows

The basic configuration of Node should not be changed, therefore it is sufficient to click Next until the installer has finished.

3.1.2  Linux

An internet connection is required to download and install the dependencies. The commands required to do so are stated in Chapter 2.1 If the terminal asks to continue (y/N), type ‘y’ and press Enter to accept and continue.

3.1.3  Mac

The basic configuration of Node should not be changed, therefore it is sufficient to click Next until the installer has finished.

3.2  Git/GitHub

3.2.1  Windows

The basic configuration of GitHub should not be changed, therefore it is sufficient to click Next until the installer has finished.

3.2.2  Linux

An internet connection is required to download and install the dependencies. The commands required to do so are stated in Chapter 2.1 If the terminal asks to continue (y/N), type ‘y’ and press Enter to accept and continue.

3.2.3  Mac

The basic configuration of GitHub should not be changed, therefore it is sufficient to click Next until the installer has finished.
3.3 Java Development Kit

3.3.1 Windows
The basic configuration of JDK should not be changed, therefore it is sufficient to click Next until the installer has finished.

3.3.2 Linux
An internet connection is required to download and install the dependencies. The commands required to do so are stated in Chapter 2.1 If the terminal asks to continue (y/N), type ‘y’ and press Enter to accept and continue.

3.4 Android SDK

3.4.1 Windows
Make sure to select Android 4.4.2 (API 19), the rest can be done as default during the installation.

3.4.2 Linux
Make sure to select Android 4.4.2 (API 19), the rest can be done as default during the installation.

3.5 Xcode

3.5.1 Mac
It is sufficient to install everything as default for the installation of Xcode.
Chapter 4

Extendibility

This sections focusses on the different ways of extending the current application by using the provided software. Information about adding content such as images or feeds for the application is done within the CMS, which is elaborated upon in the user manual [6].

4.1 Language file

There is no button to change the language from within the app yet, but this could be added in future development. However, it is possible to change the language of the TravelMatch application. This can be done by adding a language file. A language file can be added by:

- Copying an existing file in the app/scripts/lang/ directory
- Renaming the translation name (e.g. 'en') on line 2
- Modifying the text strings to the preferred language.

The translation provider can be specified in the file app/scripts/app.js to use the language file by adapting $translateProvider.preferredLanguage('en') to use the required translation. Thereafter, the language file should be loaded in app/index.html as followed: <script src="scripts/lang/en.js"></script>, where scripts/lang/en.js points to the path of the required translation file.
Chapter 5
Configuration Itemlist

This section gives an overview of the configuration items transferred to the customer. All documentation are delivered in PDF format. The documentation directory contains the latest versions available for the documents.

5.1 Product items

The following product items will be delivered to the customer:

- User Requirements Document (URD) [2]
- Software Requirements Document (SRD) [3]
- Detailed Design Document (DDD) [5]
- Software User Manual (SUM) [6]
- Software Transfer Document (STD) [7]
- Unit Test Plan (UTP) [8]
- System Test Plan (STP) [7]
- Integration Test Plan (ITP) [9]
- Acceptance Test Plan (ATP) [10]

All code residing in the GitHub repository will be delivered. This represents a fully working system.

5.2 Process items

These are the process items that will be given to the customer. This includes all project plans used within the project.

- Software Configuration Management Plan (SCMP) [11]
- Software Project Management Plan (SPMP) [12]
- Software Quality Assurance Plan (SQAP) [13]
- Software Validation and Verification Plan (SVVP) [14]
Chapter 6

Acceptance Test Report Summary

A detailed description of the acceptance tests can be found in the ATP [10]. These acceptance tests cover the user requirements in the URD [2], except for the user requirements that have not been implemented. These are listed in Table 6.1. The requirements UCR156...UCR189 are constraint requirements, these are not individually tested but used as prerequisites for the tests.

6.1 First acceptance test

The first acceptance test was done in the office of our client, on the 16th of June 2015 at 12:00 PM. Present were the client: Menno, our manager: Tim, Guido, Dennis and Lieuwe. During the testing, there were only slight procedural problems in tests AT3 and AT4. The tests could still be completed, but some parts of the tests did not work properly: in both tests the back button did not work properly on one of the tested devices and during AT3 the 'calculating image' was displayed in the wrong dimensions. This last issue caused requirement UCR63 to fail. Other than what was in the requirements there some design preferences such as a different loading animation that the client would like us to include before handing over the software. We agreed with the client to work on these things as well before the second test.

6.2 Second acceptance test

The second acceptance test was done in the office of our client, on the 19th of June 2015 at 2:30 PM. Present were the client: Menno, our manager: Tim, Guido and Lieuwe. All requirements passed in all tests so the second test was a success. Additionally, the small changes that the client asked for had all been implemented to his satisfaction.

<table>
<thead>
<tr>
<th>UCR2a</th>
<th>UCR12</th>
<th>UCR14</th>
<th>UCR15</th>
<th>UCR16</th>
<th>UCR19</th>
<th>UCR20</th>
<th>UCR26</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCR29</td>
<td>UCR31</td>
<td>UCR32</td>
<td>UCR33</td>
<td>UCR34</td>
<td>UCR38</td>
<td>UCR39</td>
<td>UCR40</td>
</tr>
<tr>
<td>UCR56</td>
<td>UCR57a</td>
<td>UCR57b</td>
<td>UCR57c</td>
<td>UCR57d</td>
<td>UCR57e</td>
<td>UCR83</td>
<td>UCR86</td>
</tr>
<tr>
<td>UCR88</td>
<td>UCR90</td>
<td>UCR91</td>
<td>UCR92</td>
<td>UCR93</td>
<td>UCR94</td>
<td>UCR95</td>
<td>UCR103</td>
</tr>
<tr>
<td>UCR108</td>
<td>UCR109</td>
<td>UCR109a</td>
<td>UCR115</td>
<td>UCR118</td>
<td>UCR123</td>
<td>UCR130</td>
<td>UCR134</td>
</tr>
<tr>
<td>UCR135</td>
<td>UCR139</td>
<td>UCR140</td>
<td>UCR141</td>
<td>UCR142</td>
<td>UCR143</td>
<td>UCR144</td>
<td>UCR145</td>
</tr>
<tr>
<td>UCR146</td>
<td>UCR147</td>
<td>UCR148</td>
<td>UCR149</td>
<td>UCR150</td>
<td>UCR151</td>
<td>UCR152</td>
<td>UCR153</td>
</tr>
<tr>
<td>UCR154</td>
<td>UCR155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: All user requirements that have not been implemented.
Chapter 7

Software Problem Reports

There are still some software problems and features that are not fully implemented. A list of these can be found Table 6.1. This section focusses on the larger problems still within the software.

7.1 Guest Mode

The Guest Mode has not been fully implemented yet, which allows users to use the app with no account. This regards requirement **UCR2a**, which is a user requirement with a priority of should have. The back end and the API implementations have already been finished for this. To finish this feature, there needs to be some implementation in the front end.

7.2 Accounts

There are still some requirements that have not been implemented yet regarding accounts. An example are the requirements **UCR38** and **UCR39** with priority should have, which states that accounts can be deleted within the user details screen. This is already possible from within the CMS, but not from within the application. There needs to be a button to do so, as well as the functionality that it removes all accounts from the database.

7.3 Analytics

The analytics of the system are not fully functional yet. The data is recorded, but the client chose to use a different analytics provider, therefore the data has not been provided to this unknown analytics provider yet. This is why the analytics could not be tested within the Acceptance Test, which concerns requirements **UCR144-UCR155**, ranging from won’t have to must have priorities.
Chapter 8

Software Change Requests

Not applicable.
Chapter 9

Software Modification Reports

Not applicable.