Contents

[1. INTRODUCTION 1](#_Toc498420542)

[A. Purpose 1](#_Toc498420543)

[B. Scope 1](#_Toc498420544)

[C. Definitions, Acronyms, Abbreviations 1](#_Toc498420545)

[D. Revision history 1](#_Toc498420546)

[E. Reference Documents 1](#_Toc498420547)

[F. Document Structure 1](#_Toc498420548)

[2. ARCHITECTURAL DESIGN 1](#_Toc498420549)

[A. Overview 1](#_Toc498420550)

[B. Component view 1](#_Toc498420551)

[C. Deployment view 1](#_Toc498420552)

[D. Runtime view 1](#_Toc498420553)

[E. Component interfaces 1](#_Toc498420554)

[F. Selected architectural styles and patterns 1](#_Toc498420555)

[G. Other design decisions 1](#_Toc498420556)

[3. ALGORITHM DESIGN 1](#_Toc498420557)

[4. USER INTERFACE DESIGN 1](#_Toc498420558)

[5. REQUIREMENTS TRACEABILITY 2](#_Toc498420559)

[7. IMPLEMENTATION, INTEGRATION AND TEST PLAN 2](#_Toc498420560)

[8. EFFORT SPENT 2](#_Toc498420561)

[9. REFERENCES 2](#_Toc498420562)

# 

# 1. INTRODUCTION

## A. Purpose

This document presents advanced technical details related to general design of Travelander+ application which is calendar application with recommender system for feasible mobility options. With this document, authors aim to explain and illustrate design views of the Travelander+ application by relating previously presented RASD. The main audiences for this document are the developers who aim to understand or implement:

* Components of proposed system
* Interfaces between external and internal components
* Overall architecture of system
* Employed design patterns (and also possible motivations)
* Behavior of system during runtime

## B. Scope

The proposed system ,Travelander+, aims to serve as a personal calendar and recommender system for mobility. In particular, the application provides a platform to its users that create feasible schedules with suggestions on mobility options by considering user preferences and constraints, time and location constraints, weather, traffic and public transformation information. As advanced features, the application allows to the user know about car/bike share applications and add customized event whose duration and time interval may be different and editable. Possibly, everyone who needs to plan the his/her personal calendar and mobility options may be the user of Travelendar+.

## C. Definitions, Acronyms, Abbreviations

### C.1 Definitions

* Event: Any appointment or customized break by the user
* Break: An event with a flexible duration assigned between the selected start and end time
* Periodic Event: An event that is repeated with a selected frequency. (e.g. break, gym)
* Activated/Deactivated: An activated mobility option is listed in the preference list with the given constraints and priority, whereas a deactivated mobility option is not listed in the preference list at all or only on the restricted times given by the user or based on the information received from APIs (e.g. walk and bike deactivated on rainy or snowy weather).
* Restricted Time Interval: A time interval set by the user for a certain mobility option to be temporarily deactivated.
* Distance Limit: A maximum distance set by the user to deactivate the certain mobility option for any larger distance.
* Preference List: A list constructed by the user with the Travlendar+’s given mobility options by giving them priorities to be selected or to activate-deactivate them based on the user’s abilities and preferences.
* Default List: Ready to use preference lists offered by the Travlendar+.
  + Minimize Carbon Footprint
  + Minimize Expenditure
* Reachable/Unreachable: An attribute of the location of the event to be added which shows if the event is actually reachable by any means of mobility based on the already scheduled events, and if it is unreachable forbids the selection of that location.
* Public Transportation Information Provider API: General name for the public transportation APIs. Its purpose is to provide official information about the public transportation of the current city.

### C.2 Acronyms

* RASD : Requirement Analysis and Specification Document
* PTIP: Public Transportation Information Provider
* API: Application Programming Interface
* JDBC : Java Database Connectivity
* AS: Application Server

## D. Revision history

## E. Reference Documents

## F. Document Structure

# 2. ARCHITECTURAL DESIGN

A. Overview :

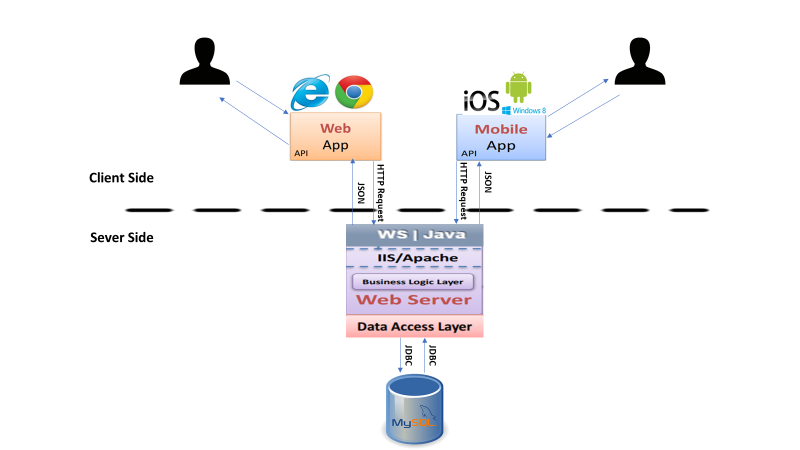
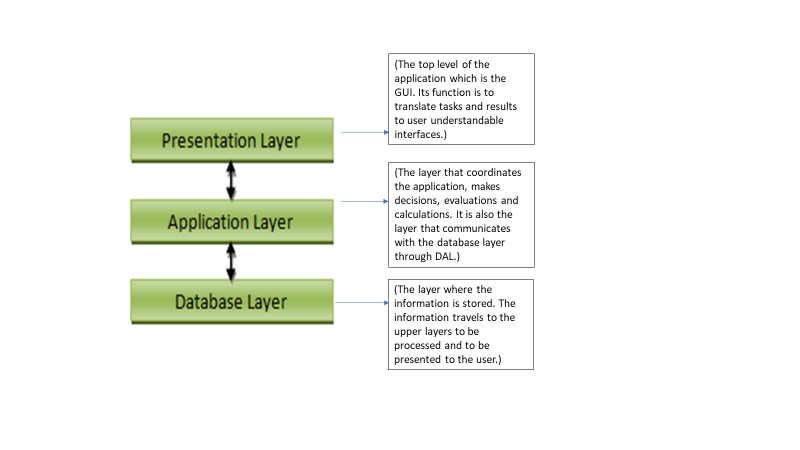


Figure 0: General Architecture

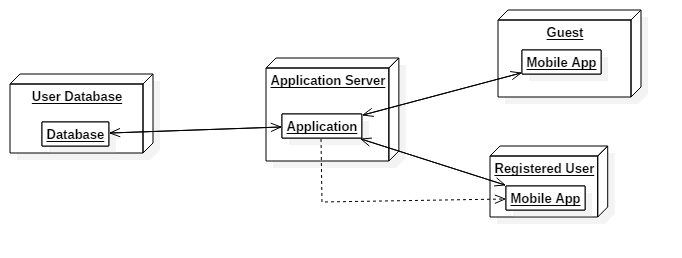
Travelander+ is built on three-tier architecture which is composed of presentation, application and data layer. These layers are charged by the following:



* Presentation Layer: This layer provides the visualization of the user content and interface for any usage of the application.
* Application Layer: This layer provides the necessary computation of algorithms and communication with database.
* Data Layer: This layer provides a database for the user contents.

High-level components and their interaction

Travelander+ has four higher level elements: Guest, registered user, application server and user database. Travelander+ fully operates with these components and their communication. The main component is application server (AS) which is charged for necessary computations of feasible schedules, mobility recommender system. AS communicates with both guest and registered user in different ways. The communication between registered user and application server initiates with synchronous message coming from user to AS (Application Server) which corresponds to user log-in. After this message, another synchronous message goes to user database which is consisted of all users content from AS, and queries the user credentials. In case that query is found and log in is successful, the mobile application allows to user add/edit/delete events from his/her schedule, changing on user preferences and travel constraints, visualization of personal calendar which generates synchronous messages from registered user through AS and thereafter user database. Also, AS is able to generate asynchronous messages for notifying the user about generated schedule, mobility options according to his/her input recorded in user database. Another messaging is done between AS and user database in synchronous way. During its computations, AS needs user content such as event time and locations, mobility preferences, travel constraints etc. Therefore, AS does queries to user database and after that database returns requested elements.

 Figure X. Higher Level Components

## 

## B. Component view

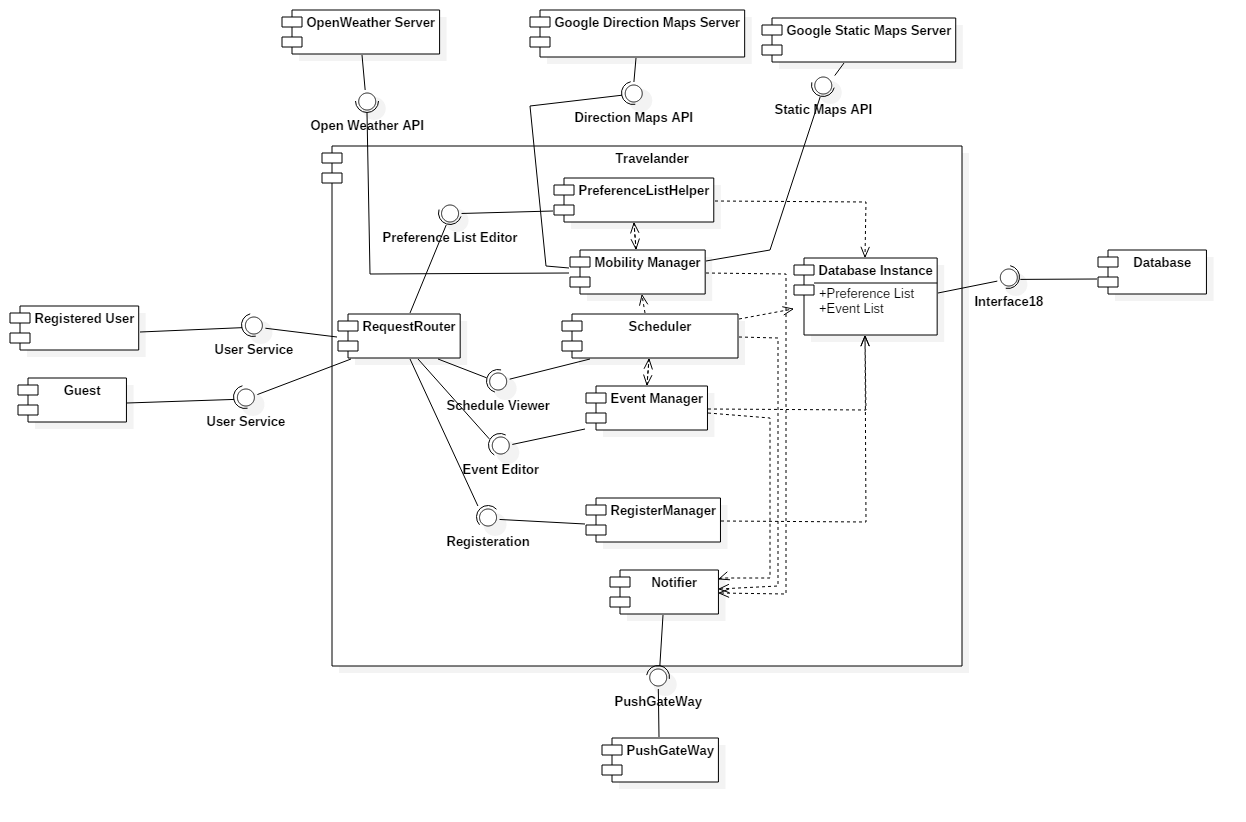


Figure Y. Component Diagram

Registered User: User who is previously registered and authenticates with correct credentials. All data related to registered user are recorded in the database.

Guest: User who is not currently registered or logged in.

Request Router: This component provides a general interface to user side for accessing all allowed services. It mainly leads user to requested services.

Event Manager: This component allows to user to add/delete/edit event operations through his/her schedule. It has dependency to scheduler component for informing user whether the requested operation is feasible or not.

Scheduler: It computes the feasible schedule for the user for currently added events and under mobility constraints (such as user preferences, weather, traffic, etc.)

Mobility Manager: Mobility manager choses feasible mobility option by running elimination algorithms.

PreferenceList Helper: This component provides an interface to user to input his/her mobility preference list.

Notifier: This component enables to application to send notifications to user by deploying push notifications.

Database Entry Instance: This is a model in the application for a database entry and it establishes an interface with the database.

## C. Deployment view

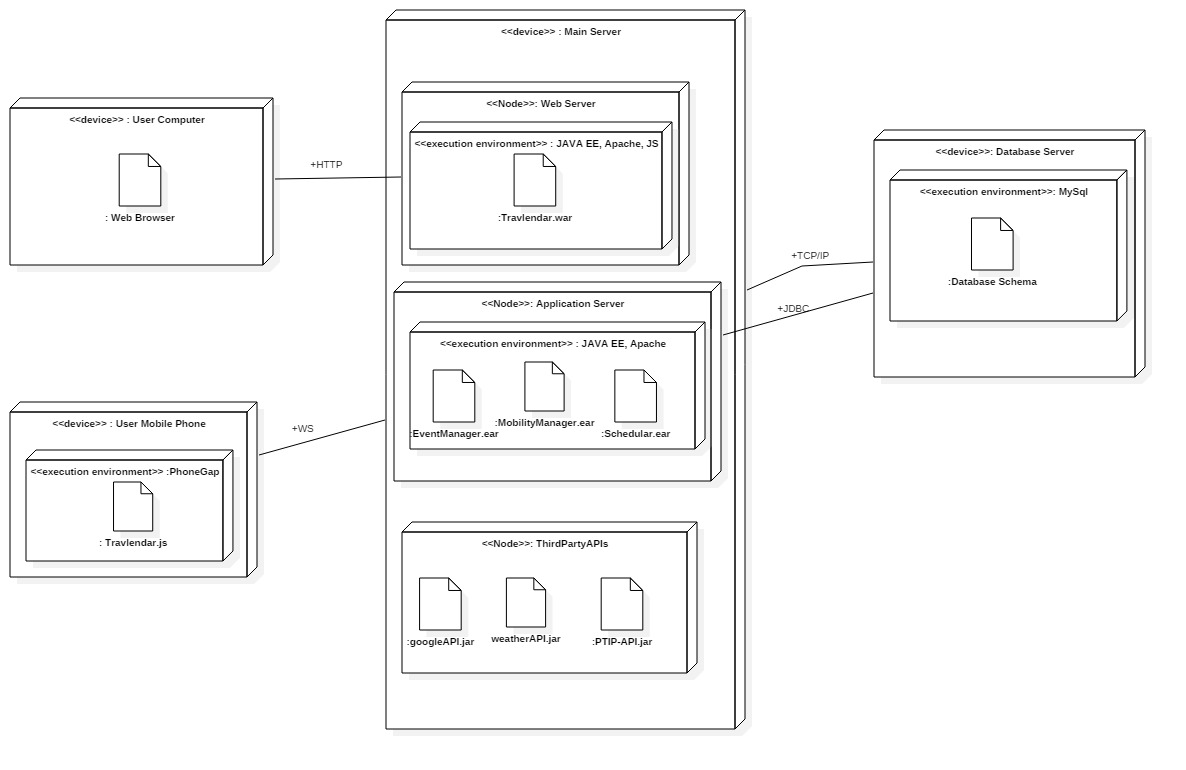


Figure Z: Deployment View

D. Runtime view You can use sequence diagrams to describe the way components

interact to accomplish specific tasks typically related to your use cases

## E. Component interfaces

F. Selected architectural styles and patterns: Please explain which styles/patterns you

used, why, and how

## G. Other design decisions

3. ALGORITHM DESIGN : Focus on the definition of the most relevant algorithmic part

4. USER INTERFACE DESIGN : Provide an overview on how the user interface(s) of your system

will look like; if you have included this part in the RASD, you can simply refer to what you have

already done, possibly, providing here some extensions if applicable.

5. REQUIREMENTS TRACEABILITY :Explain how the requirements you have defined in the RASD

map to the design elements that you have defined in this document.

7. IMPLEMENTATION, INTEGRATION AND TEST PLAN : Identify here the order in which you plan

to implement the subcomponents of your system and the order in which you plan to integrate

such subcomponents and test the integration.

8. EFFORT SPENT :In this section you will include information about the number of hours each

group member has worked for this document.

# 9. REFERENCES