RateIT High Level System Design

<Draft>
Author: Angela Mariani
Date: 17-08-2014

Table of Contents

Introduction

RateIT allows transport customers to both rate and retrieve information about transport services in NSW. Information collected by the system provides data for both transport researchers and operators.
This system design document describes the client-server and database components.

High Level Overview

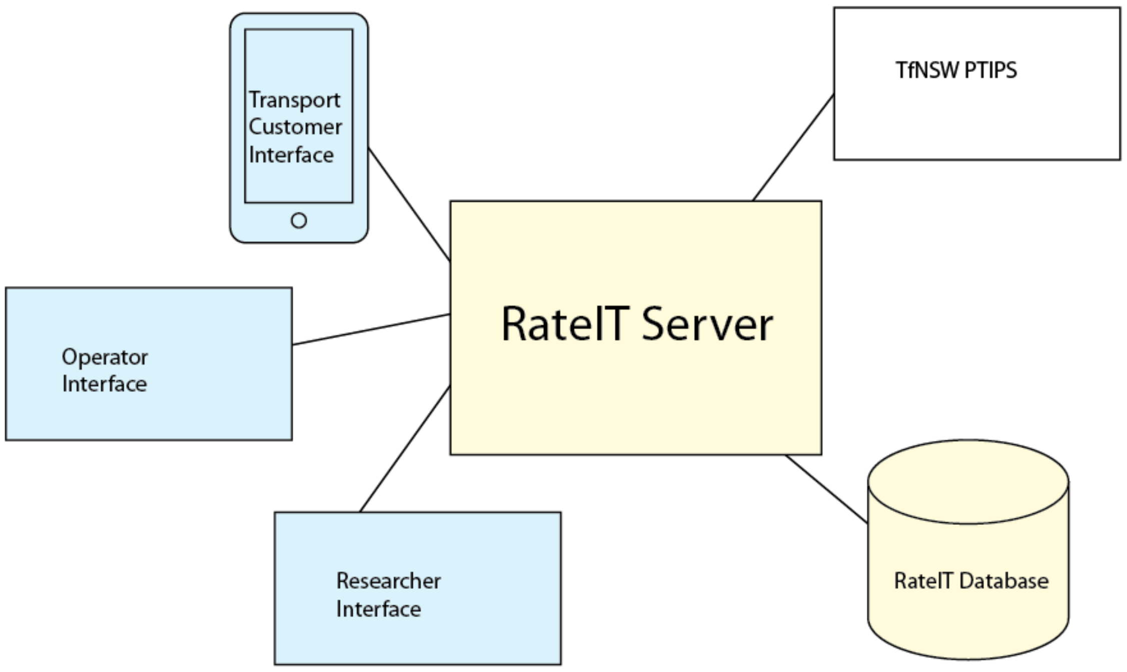

Diagram 1.

RateIT Server

Provides 3 client interfaces and communicates with TfNSW PTIPS and the RateIT database.

Requirements

- Availability?
- Load?
- Security
- Account verification?

Transport Customer Interface

Provides interface for customers to interact with RateIT application.

Requirements

- HTML on mobile browser. Preferably using AJAX technologies.
- Nice to have Android and iOS clients

Operator Interface

Provides interface for operators to interact with RateIT application and customers.

Requirements

- Simple HTML (AJAX not required)
- Security

Researcher Interface

Provides interface for researchers to interact with RateIT application and customers.

Requirements

- Simple HTML (AJAX not required)
- Security

RateIT Database

Stores data from interactions RateIT application.

Requirements

- SQL/NoSQL? (Note: NoSQL databases are less expensive and more scalable than SQL however, SQL databases do provide richer query facilities than NoSQL.)

TfNSW PTIPS

Public Transport Information and Priority System (PTIPS) gives real time bus information.

Features

Transport Customer Accounts

Transport customers need to be identified.

- Username/email (-regex?)
- Password
- Email verification
- optional: log in using gmail, facebook or other?

Researcher and Operator Accounts

These types of accounts are distinct from transport customer accounts as they have access to administrative options. Security requirements for these types of accounts are best met by using a third party accounts system.
The main distinction between accounts is that, researcher accounts should be able to query all data for all operators while operators should only be able to query their customer's data.

System Administration

- Backups
- System Activity Reports

Design Proposals

There are many options for the hosting of web applications. However, over the years Amazon, Google and Microsoft have been the most prevalent in developing cloud based hosting for web applications. It would seem prudent, to preserve the web development effort, to target one of these platforms for hosting RateIt. It is necessary to evaluate the needs of RateIT versus the features provided on the various systems.
Cloud services can be compartmentalised into two different types. The first being PaaS (Platform as a Service) versus virtualised servers. PaaS provides limited configurability however it is significantly less expensive than a virtualised server. Currently the only viable PaaS is Google's App Engine.

Google App Engine Proposal

Google App Engine is a platform as a service offering, with it's key features being:

1. Free tier for small to medium websites/Competitive pricing.
2. Minimal server administration requirements.
3. Auto scaling. System will automatically scale to load requirements.
4. Google Cloud Endpoints. Compatibility with AJAX, Android and iOS clients.
5. Free integrated datastore, NoSQL database. Cloud SQL also available.

App Engine supports 4 programming languages, of which Python and Java are compatible with the Endpoints api. Python is the more mature App Engine language because it was the first supported language under App Engine and there is a ride range of community support.

Virtualised Server Proposal, Amazon EC2/Google Compute Engine/Microsoft Azure

The key feature of virtualised servers is that there are no limits to the configurability meaning that virtually any software could be made to run on it. However, to make a truly scalable system requires careful system design. The major drawback is that system administration is a non trivial task compared to PaaS offerings.

Given the teams level of expertise with system administration and available resources, App Engine provides a compelling solution for hosting the RateIT project.

App Engine datastore or SQL?? Depends on complexity of transactions.
NoSQL is better for scalability and it's free.