MSQUAD COMPANY NAME

GT2

Garbage Tuck Tracker
Software Architecture Document

Version 1.0

J.K.M.M.Thilakarathne
130597L

Revision History

Date	Version	Description	Author
10/April/2016	1.0	First phase	MSquad

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

[This section defines the role or purpose of the **Software Architecture Document**, in the overall project documentation, and briefly describes the structure of the document. The specific audiences for the document is identified, with an indication of how they are expected to use the document.]

1.2 Scope

[A brief description of what the Software Architecture Document applies to; what is affected or influenced by this document.]

The scope of this document is to depict the architecture of the Garbage Truck Tracker GT_2 system.

It is not intended that this document be totally complete before development is completed and in fact it is expected to be updated and refined all through the development process as lessons are learnt the design developed, refactored, and finalized.

The aim of this document is to reflect what actually worked and was developed, not what we thought might work before the development was undertaken.

The above said, changes to the document that occur later are expected to be to the more fine grained details than to the high level areas such as Goals and Constraint, Use-cases, and Logical views.

1.3 Definitions, Acronyms, and Abbreviations

Acronym, abbreviation	Definition
GT_2	Garbage Truck Tracker
Hybrid mobile application	A mobile application that combines elements of both native and Web applications
Townspeople	The people who live in a town
Townsman	A person who lives in a town
SMS	Short Message Service
Real-time	Real-time or real time is a term often used to distinguish reporting, depicting, or reacting to events at the same rate and sometimes at the same time as they unfold
Truck Fleet	All the trucks
PC	Personal Computer
Info Window	Google maps information window popups when you click a map marker
route	The path which a truck will travel
PS = DC	Pradeshiya Saba , Divisional Council
MC	municipal council
UC	urban council

1.4 References

[1] Wiki, “4+1 architectural view model,” [Online]. Available: https://en.wikipedia.org/wiki/4%2B1_architectural_view_model.

[2] S. Chokkalingam, “4+1-view-model,” [Online]. Available: http://www.slideshare.net/shobanachokkalingam/41-view-model.

[3] Tutorialspoint.com, “UML - Standard Diagrams,” [Online]. Available: http://www.tutorialspoint.com/uml/uml_standard_diagrams.htm .

[4] “UML 2 Activity Diagram,” [Online]. Available: http://www.sparxsystems.com/resources/uml2_tutorial/uml2_activitydiagram.html.

1.5 Overview

The rest of this document is organized as below.

- Architectural Representation

This section describes what software architecture is for the current system, and how it is represented. Of the Use-Case, Logical, Deployment, and Implementation Views, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.

- Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture, for example, safety, security, privacy, and use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.

- Use-case View

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage - they exercise many architectural elements, or if they stress or illustrate a specific, delicate point of the architecture.

- Logical View

This section shows the architecturally significant components of the application, such as its main components and interfaces.

- Process View

This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes).

- Deployment View

This section describes one or more physical network (hardware) configurations on which the software is deployed and run. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software, and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the Process View onto the physical nodes.

- Implementation View

This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.

- Data view

A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.

- Size and Performance

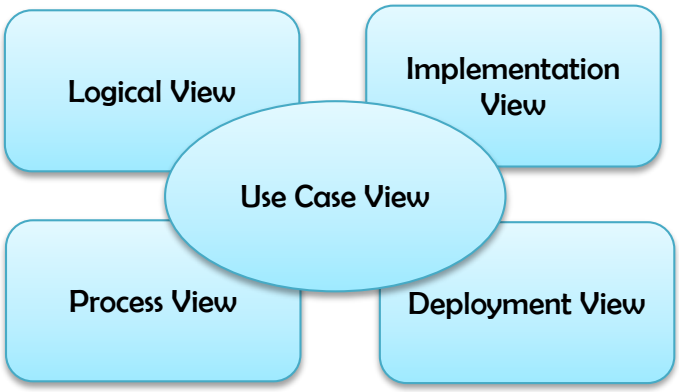
A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.

- Quality

A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, for example safety, security or privacy implications, they should be clearly delineated.

2. Architectural Representation

The "4+1" view model framework is used as the software architecture for GT_2. Logical, development, process, physical and use-case views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers and project managers.



4+1 view model

Logical View	Considers functional requirements. This is viewed by end-users. This view shows the components of the system as well as their interactions/relationships. ***Representation: UML Class diagram, Communication diagram, Sequence diagram.***
Process View	Considers non-functional requirements Viewed by integrators. Shows the processes/workflow rules of a system and how those processes communicate with each other. ***Representation: UML activity diagram***
Development View	Considers the software module organization. Viewed by programmers and software managers. It gives a building block view of the system ***Representation: UML component diagram, package diagram***
Physical View	Considers the non-functional requirements regarding to underling hardware (Topology, communication). Viewed by system engineers. Shows the systems execution environment. ***Representation: UML deployment diagram***

Use case View	Considers system consistency and validity Viewed by all users of other views and evaluators. Validation and illustration to show the design is complete. It is redundant with other views ***Representation: UML use case diagram, activity diagram***

3. Architectural Goals and Constraints

3.1 Goals

- **Usability**
  In most cases, this systems will be used by housewives. So it would take some time to learn the system. So this should be minimized by making a simple user experience.

- **Reliability**
  Errors should be handled well. Debug messages should not prompted to the user. The application will not be containing any “critical” bugs. For example, complete loss of data or a complete inability to use certain parts of the system’s functionality. This application will be able to download from the Google Play Store and ITunes anytime. Frequent updates will be supplied.

- **Performance**
  Meteor presents 'Data-on-the-wire' with latency compensation and conflict resolution built-in. When data changes, updates propagate reliably to affected clients and users’ screens update via live query, full stack DB drivers and mini database sources.

- **Supportability**

  A complete documentation for the system will be generated. It will support for future maintenances and developments.

3.2 Constraints

The system is to deploy on Galaxy. So the system architecture should be able to get the best performance out of it.

As Meteor only supports, MongoDB the database view should be designed considering MongoDB characteristics.

The Internet connection is also a constraint for the application. Since the application fetches data from the database over the internet, it is crucial that there is an internet connection for the application to function.

The location service is the other constrain for mobile application. To work some major functionalities, it is a necessary requirement.

4. Use-Case View

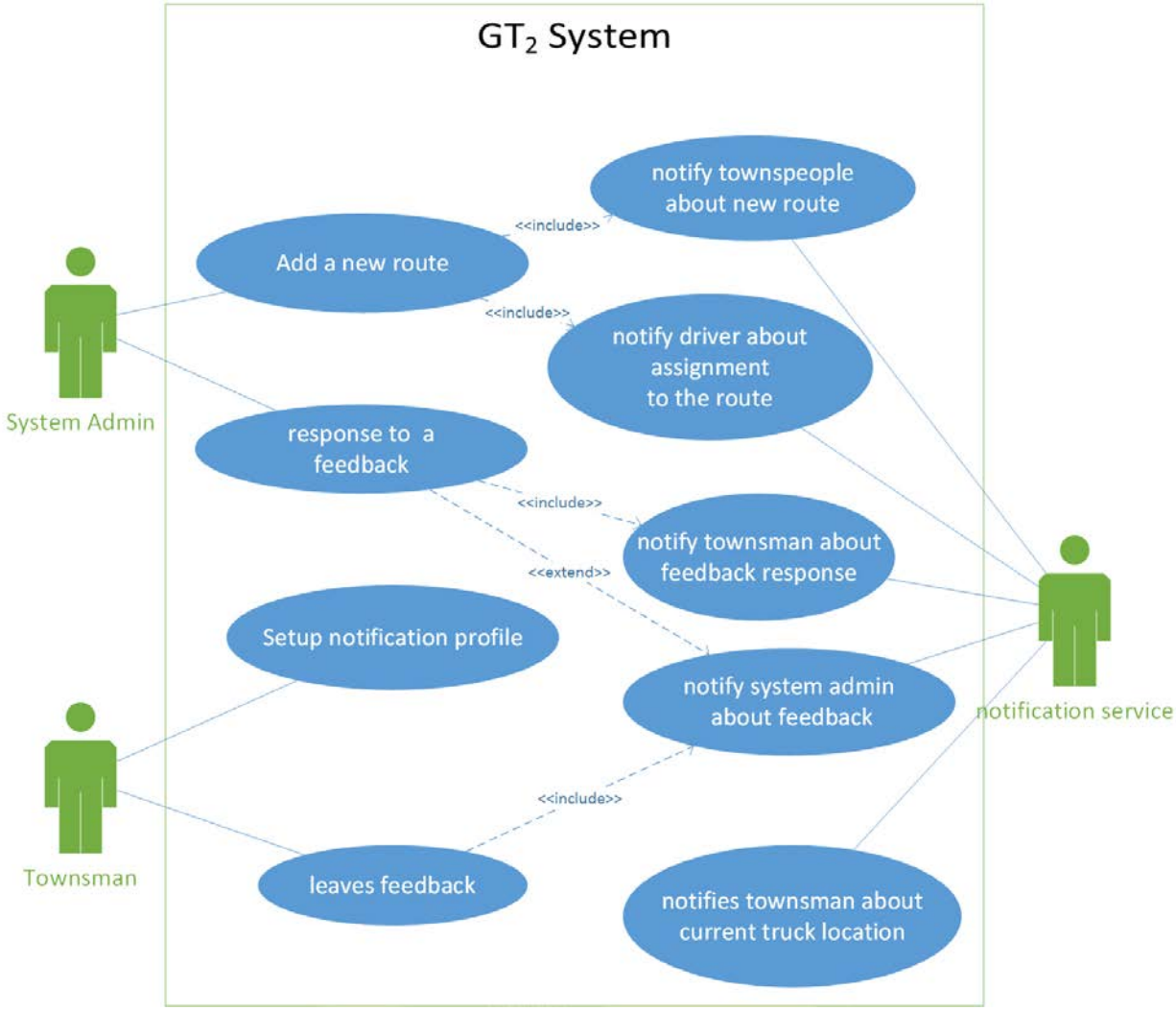
Following use cases and scenarios from the use-case mode represent some significant, central functionality of the Content Aggregator and they have a large architectural coverage - they exercise many architectural elements and they stress and illustrate a specific, delicate point of the architecture.

- System administrator adds a new route
- System administrator views feedback
- Townsman signs up in the GT2 system

- Townsman setups his notification profile
- Townsman leaves feedback
- GT_2 notifies townspeople about current truck location

4.1 Use-Case Realizations

4.1.1 Use Case Diagram



Use Case Diagram

4.1.2 Use Case Descriptions

Use case name: System administrator adds a new route	
Scenario:	System administrator adds a new route via web application
Triggering event:	Management requests to register a new route
Brief description:	Management requests to register a new route, if pre conditions are there, system displays the new route registration form. System administrator fill it and submit. Then the new route is updated in the system. Finally relevant notifications are sent.
Actor:	System Administrator

Related use cases:	Notify townspeople about new route Notify driver about route assignment	
Pre-condition:	The GT_2 has identified and authenticated the System Administrator. The new route is validated practically by the council. There exists an available truck There exists an available driver	
Post condition:	Route is stored in the database with a unique route number automatically generated by the GT_2. Notification is sent to townspeople Notification is sent to assigned driver	
Flow of events:	Actor 1. System Administrator sends a "add new route" request 2. Fill the form and submit	System 1.1 Display the "add new route" form 2.1 Save new route in the system 2.2 Display success message 2.3 Display summary
Exception Conditions:	2. If there are no available drivers or trucks for the selected time, admin has to change the time or cancel the registration.	

Use case name: System administrator views a feedback		
Scenario:	System administrator views a townsman's feedback via web application	
Triggering event:	GT_2 system got a feedback from a townsman.	
Brief description:	A notification is shown to the system administrator due to a townsman feedback. Then admin views the feedback. Then the notification count decreases by one.	
Actor:	System Administrator	
Related use cases:	N/A	
Pre-condition:	The GT_2 has identified and authenticated the System Administrator. The new route is validated practically by the council. There exists an available truck There exists an available driver	
Post condition:	Townspeople has given some feedback	
Flow of events:	Actor 1. System Administrator sends a "view feedback" request 2. View the feedback	System 1.1 Display the feedback with respond input field 2.1 update the notification count
Exception Conditions:	2.1 If system administrator responds to the feedback a notification is sent to the townsman	

Use case name: Townsman signs up in the GT_2 system		
Scenario:	Townsman signs up in the GT2 system via mobile application	
Triggering event:	New townsman registration	
Brief description:	Townsman request, get and fill the sign up form. System stores new user.	
Actor:	Townsman	
Related use cases:	N/A	
Pre-condition:	Location service of the mobile device should be turned on	
Post condition:	The new user details are stored in the database. Townsman is requested to setup his notification profile.	
Flow of events:	Actor	System
	1. System Administrator sends a “sign up” request 2. Fill the form and submit	1.1 Display the sign up form 2.4 Save new user in the system 2.5 Display success message 2.6 Display summary
Exception Conditions:	2. If the entered NIC or mobile number already exists in the system the user is warned.	

Use case name: Townsman setups his notification profile		
Scenario:	Townsman setups his notification profile	
Triggering event:	New townsman registration	
Brief description:	Townsman request, get and fill the notification setup form. System stores the details.	
Actor:	Townsman	
Related use cases:	N/A	
Pre-condition:	Connected to the internet	
Post condition:	The notification settings for the townsman is updated to the database	
Flow of events:	Actor	System
	1. System Administrator sends a “setup notification profile” request 2. Click on the plus signed button to add a new notification trigger 3. Select the amount of time in advance he wants to get the notification about the truck location 4. Repeat 4,5 until the satisfied amount of notifications are setup	1.1 Display the notification profile form 2.1 Save notifications in the system 2.2 Display success message 2.3 Display summary
Exception Conditions:	2. 1 the notification receive time cannot be between 10:00 pm and 5:00 am. If it is, warning is sent and requested to change the inputs	

Use case name: Townsman leaves feedback		
Scenario:	Townsman leaves feedback via mobile application	
Triggering event:	Leave new feedback	
Brief description:	Customer fill the necessary detail and give an feedback about the service	
Actor:	Townsman	
Related use cases:	GT_2 notifies system administrator about a townsman feedback	
Pre-condition:	N/A	
Post condition:	The notification settings for the townsman is updated to the database	
Flow of events:	Actor	System
	1. System Administrator sends a "leave feedback" request 2. Enter feedback title 3. Enter feedback 4. Check "leave feedback on a service from a specific truck" checkbox select route 5. Select location 6. Enter truck license number 7. Submit the form	1.1 Display the feedback form 4.1Shows the new input fields (route, location, truck license number) 7.1 Save feedback in the system 7.2 Display success message 7.3 Display summary
Exception Conditions:	7.1 If the truck licence number is not a valid one (not registered in the system, not match with the route) the feedback is rejected.	

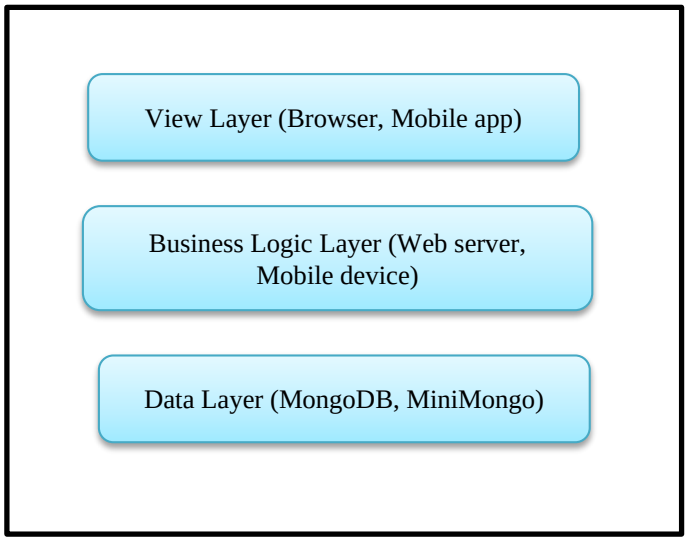
Use case name: GT2 notifies a townsman about current truck location		
Scenario:	GT2 notifies townspeople about current truck location while the truck is on an active route. Townsman can access the system via mobile phone application.	
Triggering event:	The advance time in the townsman's notification profile has arrived.	
Brief description:	When the time come, a notification with location data, garbage collection data, estimated time to arrive is sent to townsman's mobile phone	
Actor:	GT_2 System	
Related use cases:	N/A	
Pre-condition:	The notification profile for the townsman must be created. Truck should reach at the proper location to arrive at townsman around the advance time setup in the notification profile	
Post condition:	A "notification sent" flag is set in the townsman's notification profile	
Flow of events:	Actor	System
	1. Truck arrives at the proper location	1.1 system set the "notification sent" flag in the profile 1.2 sent the notification to the townsman
Exception Conditions:	N/A	

5. Logical View

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages.

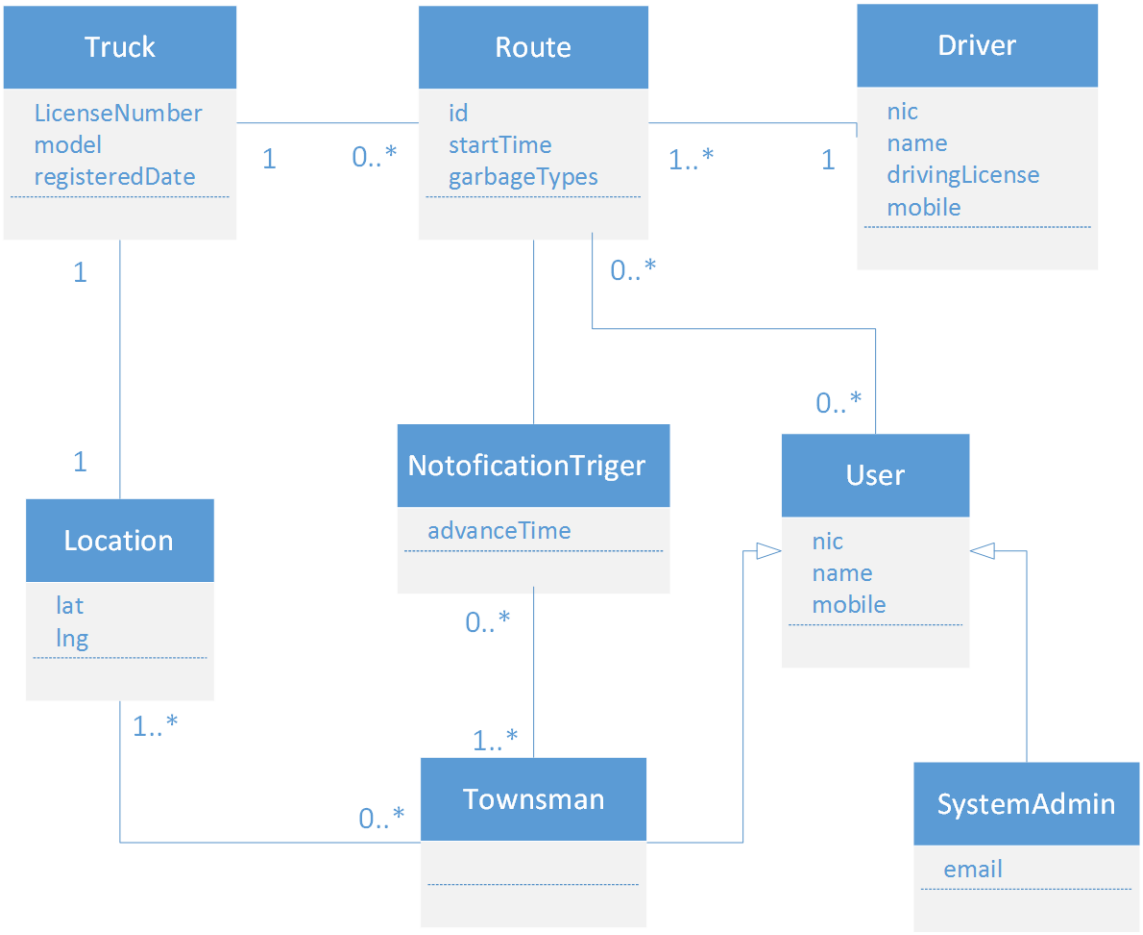
5.1 Overview

GT_2 system is divided in to following layers.



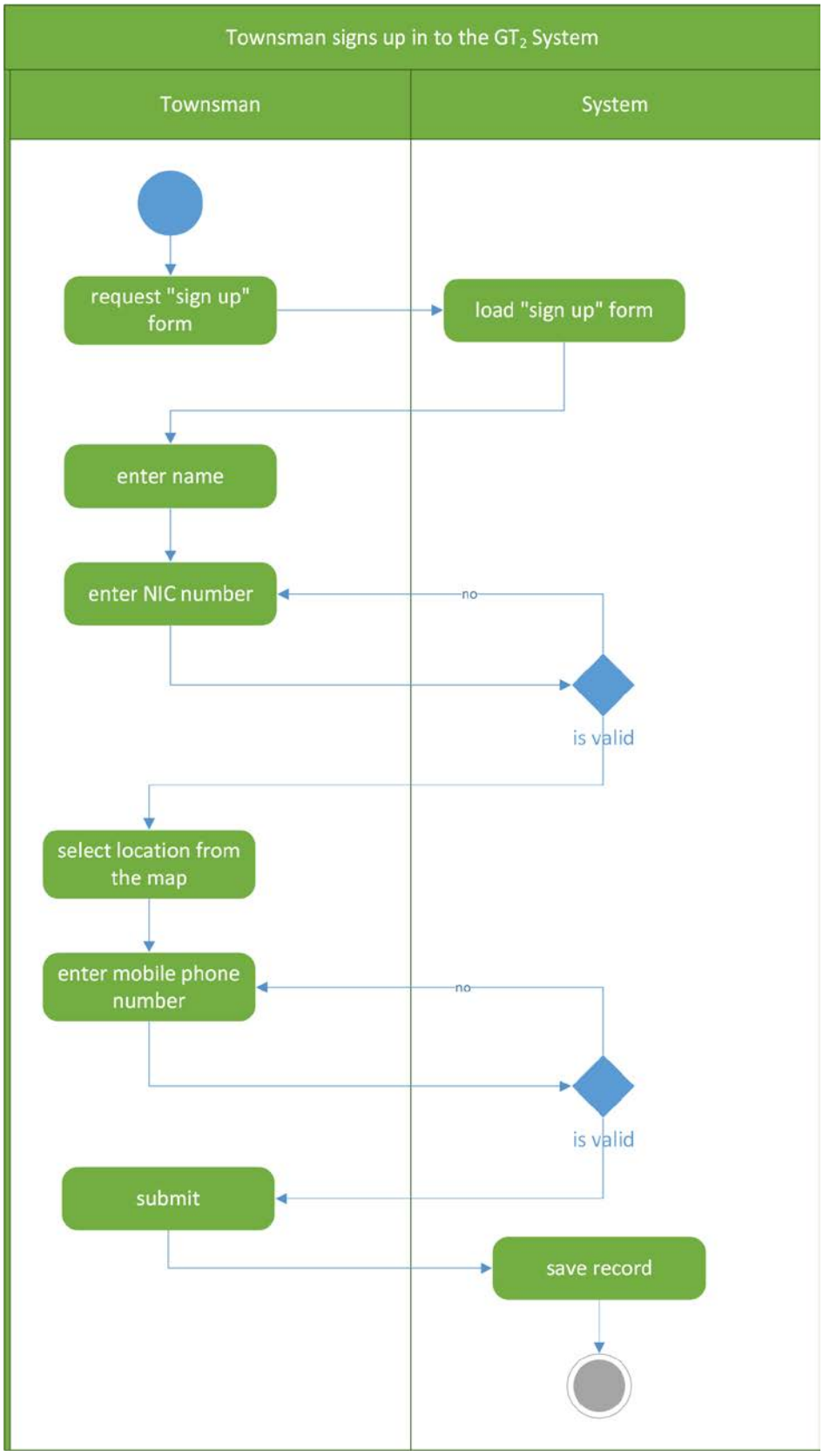
Block diagram 1

5.2 Architecturally Significant Design Packages

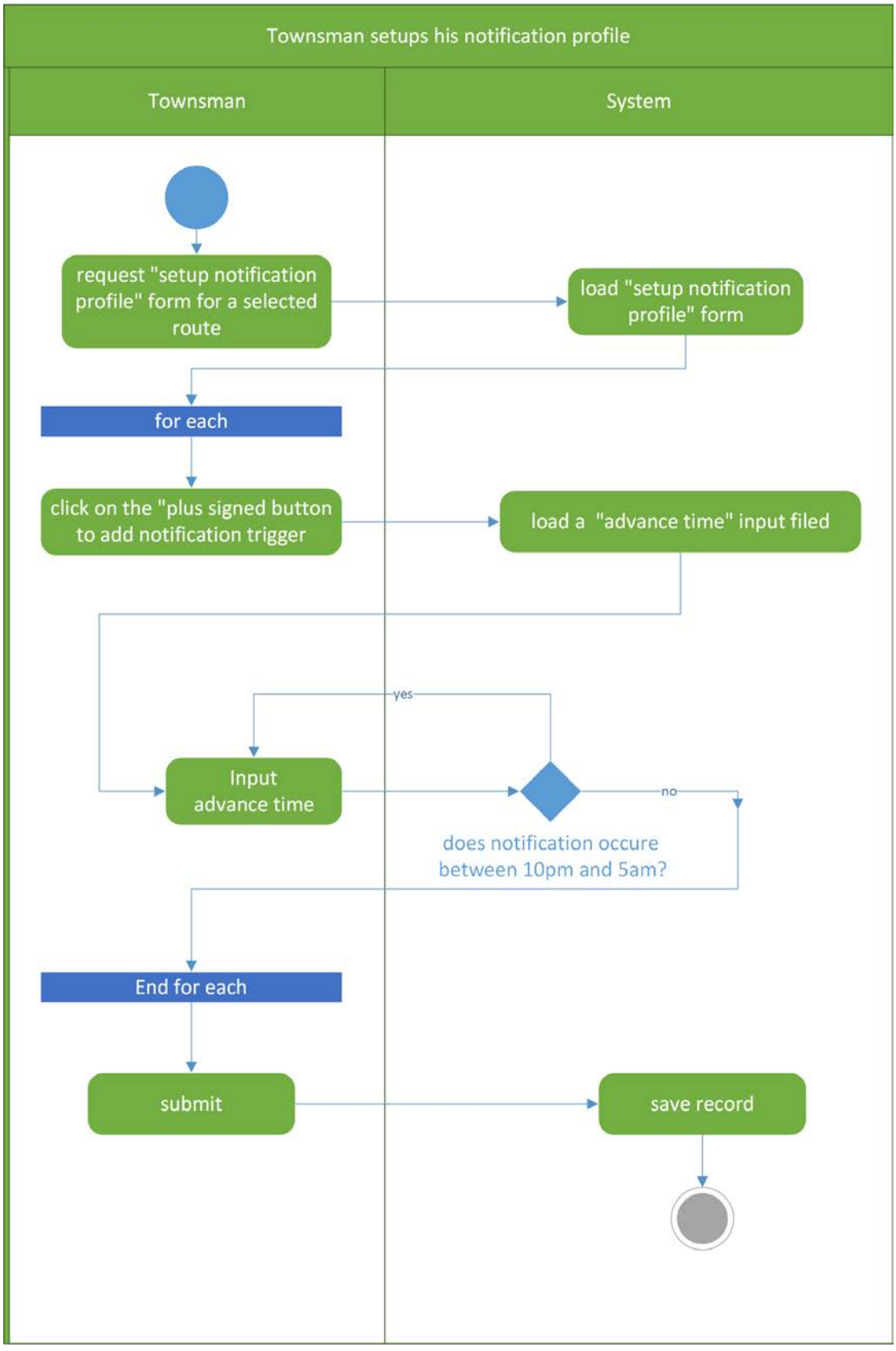


6. Process View

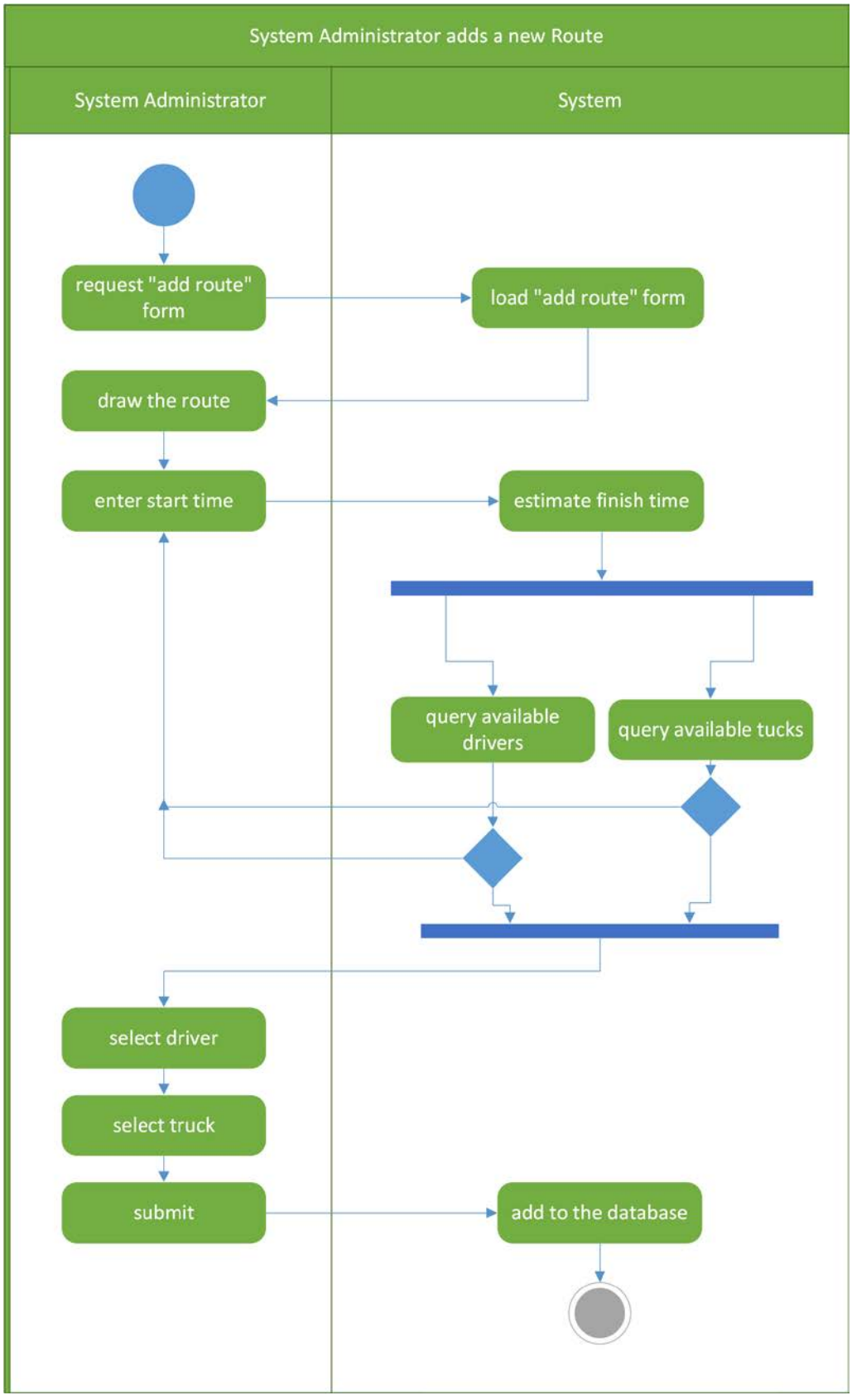
This section describes the system's decomposition into lightweight processes



Activity Diagram 1

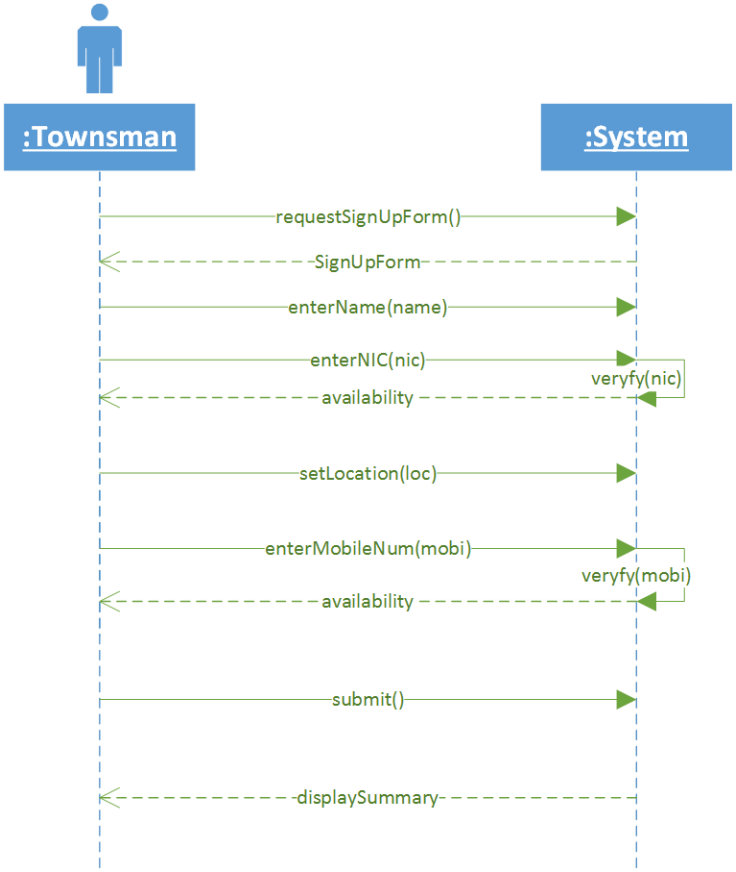


Activity Diagram 2

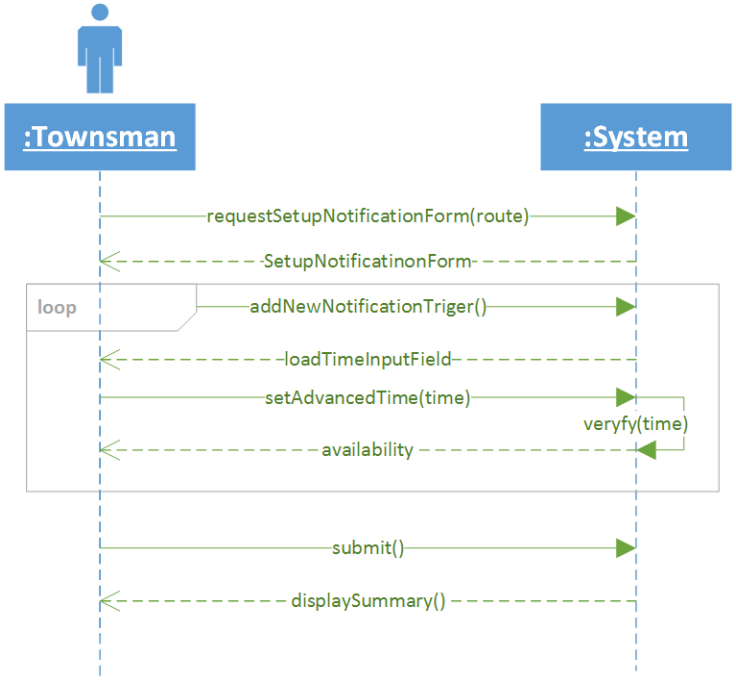


Activity Diagram 3

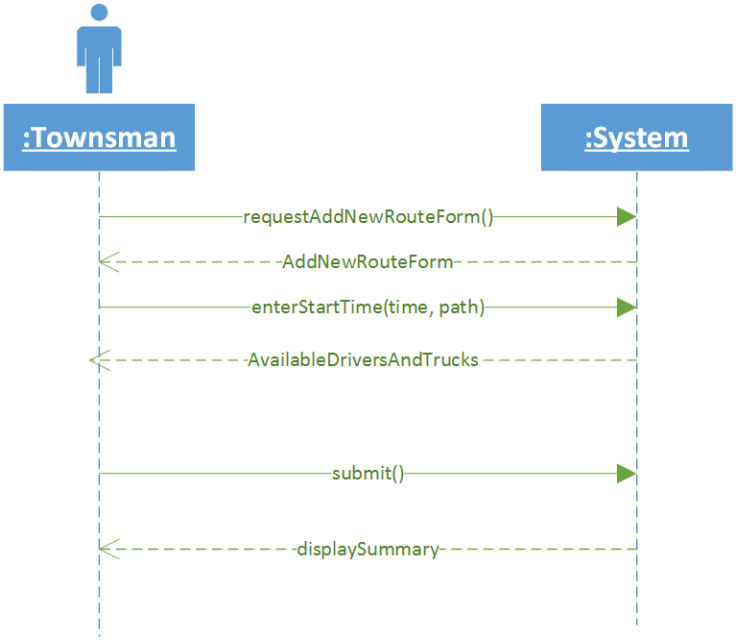
6.1 System Sequence Diagrams



System Sequence Diagram 1



System Sequence Diagram 2



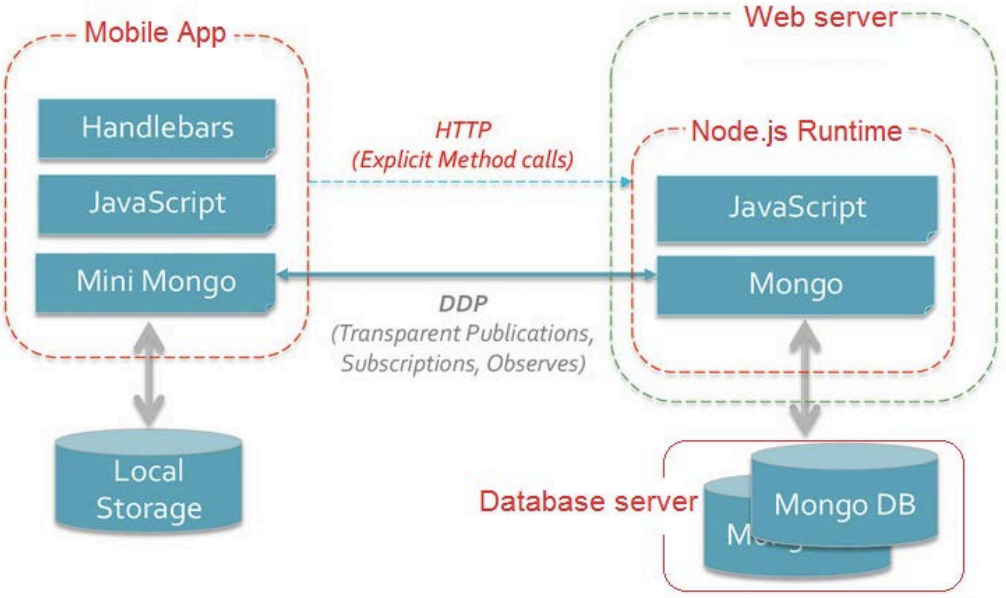
System Sequence Diagram 3

7. Deployment View

GT_2 maintains a local copy of data by implementing its own MiniMongo database in client machine's memory. All of the client-to-server data communication and synchronization is provided by Meteor. MiniMongo is a JavaScript implementation of the MongoDB API.

GT_2 uses JavaScript on both client and server for business logic.

GT_2 maintains a separate database server that runs a MongoDB deployment.



Deployment Diagram

8. Implementation View

The Implementation view describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components (directories and files, including source code, data, and executable files)

8.1 Overview

Package Diagram

8.2 Layers

8.2.1 View Layer

The view layer contains all the components needed to allow interactions with an end-user. This hands responsibilities corresponding to viewing map, managing feedback, managing profile and viewing notifications. Logically we can identify several sub packages in this layer.

8.2.2 Business Logic Layer

We can consider this layer as the layer which handles core functionalities of the system. Here we can identify core logical sub components as location handler, user identification handler

8.2.3 Data Access Layer

This layer is responsible for managing the database activities of the GT_2. It cannot directly access the database. GT_2 has to access database via the data access methods provided by web server.

9. Size and Performance

9.1 Size

The number of townspeople will vary with the UC, TC and MC.
Usually garbage trucks travel around the routes from 7am to 10am. So this will be the peak time period of system usage.

Total number of families per council is around
So estimated online users in the peak time: 350

- Web application users: 5%
- Mobile application users: 75%
- SMS notification service users: 20%

9.2 Performance

- Maximum time to start the web application: 4 seconds
- Maximum time to start the mobile application: 3seconds
- Maximum time to load the map from web application: 5seconds
- Maximum time to load the map from mobile application: 3seconds

10. Quality

Application is designed for releasing international Market. It will be distributed through following repositories.

- Android application: Google PlayStore
- IOS application: ITunes

So the application should full fill the standards requested from these repositories.

10.1 Scalability

This includes the system reaction when the user demands increase.
The web application will be hosted in Galaxy. It can increase the container number and scale to provide uninterrupted service to the clients. The database server will also be configured to scale up, when the client requests rises.

10.2 Availability

Galaxy has a special feature named “coordinated version updates”. So when we do a hot code push in Galaxy, it coordinates each user's transition from old code to new code, making sure each client is always connected to its proper backend version. All with zero downtime for the clients.

10.3 Portability

The same code runs from the client to the cloud, from packages to database APIs. The same code runs cross browsers and mobile devices via Meteor's unified isobuild system.