UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Requirement Specification

Time Tracking System (Mini-Project)

by:

COS 301 Students

for:

Jaco Kroon (Ultimate Linux Solutions)

V1.1

March 6, 2013

Change log/History

Date	Version	Description
3 March 2013	V0.1	Created skeleton of requirement specification
3 March 2013	V0.2	Added section 1-3
3 March 2013	V0.5	Added section 4
3 March 2013	V0.6	Added section 5-6
4 March 2013	V0.7	Added class diagram
5 March 2013	V0.8	Finalized non-functional requirements
5 March 2013	V0.8	Cleaned up document formatting
5 March 2013	V1.0	Finalized draft
6 March 2013	V1.1	Finalized requirement specification

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to formulate an agreement between the developers and the client Ultimate Linux Solutions (ULS) about the requirements of a Time Tracking solution for management and billing purposes. This ensures that the scope of the project is clear and understood by all parties so that the requirements set up in this document form the basis of further phases. Due to the nature of the software development process i.t.o. requirement finalisation, this document acts as an official contract between both parties.

1.2 Document conventions

- Documentation formulation: LaTeX
- Entity Relationship Diagram (ERD): Notation: Crows Foot

1.3 Project scope

The scope of the Time Tracking project can be summarised as a software solution that enables the client to have a billing system that performs billing based on:

1. Keeping track of time spent on tasks
2. Keeping track of mileage spent driving to clients

By formulating this foundation, the project can later be expanded to an Inventory Tracking system (for inventory tracking at client, not a full inventory tracking system), but is left for another project. Reporting should be provided to run billing from.

1.4 References

- Jaco Kroon, by means of requirements elicitation and formal meeting
- Requirement matrix document (Requirement Matrix.ods)

2 System description

The goal of the system is to provide a service to which the client can manage time frame in a task-oriented business environment. The system is intended to replace a job-card system

with a digital method by which time and resources expended on a task can easily be recorded and viewed.

There are thus 3 main parts to define before the system can be understood as a whole:

Tasks

Tasks can be defined as any service of job requested by either external client or internal staff (from now on, clients and internal staff will be referred to as affiliates). These tasks must be created in the system and must be assigned to a specific person (technician), this person is the responsible person for that task. Tasks may have subtasks. There is a responsible person for every task. This person must see to it that every task assigned to him as responsible person must be completed, and must manage subtasks, if they are not necessarily assigned to him/her. Task meta-data indicate the progress and status of a task.

Time stretches

Time stretches are defined as the time that a technician spends on completing a task. Time stretches may vary, and is logged by "starting time" and "completed by" fashion. Time stretches must be confirmed and signed off by a client in some way, such as smart phones. This will allow for "on the fly" logging and tracking. Must provide for online and off-line usage. Off-line only applicable to mobile system.

Billing

Billing can be defined as the system part that queries tasks and time stretches for some time span, so that clients can be billed accordingly. The billing part is not included in this scope, however reporting for the purpose of the billing system to query information must be provided.

The system must be modular, extendible and scalable.

3 Functional requirements

3.1 System features

Task API

Task creation
(Source: Jaco Kroon, Priority: High, Requirement: FRQ1)

- A task must be creatable.
- Created task must have meta-data to identify state, progress, etc.
- Dependencies can exist between task, i.o.w. tasks can have subtasks.
- Tasks must always be assigned to one responsible person.
- Each task must have an estimated time frame.
- A task is created with a location whether on-site or at client.
- Every task has the following meta-data:
    - A priority
    - Date created
    - Deadline
    - Estimated time required
    - State of task
    - Progress of task
    - Can optionally be associated with a client
- State changes must be able to be tracked in a query-able fashion

Task update
(Source: Jaco Kroon, Priority: High, Requirement: FRQ2)

- Change of meta-data such as:
    - Task status.
    - Task progress.
    - Task estimated time.

    - Resource management changes such as: changes to the responsible person.
- Client must be able to sign off a task

Task assignment
(Source: Jaco Kroon, Priority: High, Requirement: FRQ3)

- A user can be assigned a number of tasks.
- When a task is assigned to a person, that person is responsible for that task.

Task view
(Source: Jaco Kroon, Priority: Medium-High, Requirement: FRQ4)

- Different types of views of tasks must be provided depending on the nature of the request.

Time Tracking API

Time stretch creation
(Source: Jaco Kroon, Priority: High, Requirement: FRQ5)

- A time stretch is created with a time period.
- A time stretch is allocated to a task.
- Thus through the task there is a responsible person for that time stretch.

Locations
(Source: Jaco Kroon, Priority: Medium-High, Requirement: FRQ6)

- A location is linked to an affiliate whether external or internal.

Way points
(Source: Jaco Kroon, Priority: Medium-High, Requirement: FRQ7)

- A way point is defined at a location and links with the task location.
- Should store time arrived at, time left at.

Trips

(Source: Jaco Kroon, Priority: Medium-High, Requirement: FRQ8)

- A trip consist out of one or more way points.
- Must store vehicle registration number.
- Start and end way points form the starting location and ending location.

Billing API

Information pulling
(Source: Jaco Kroon, Priority: High, Requirement: FRQ9)

The Billing API will pull information, with specific regards to reporting information, from both the Task API and Time Tracking API. It is important to note that billing does not form an essential part of this project scope, but will at a later stage.

Other

Off-line usage
(Source: Jaco Kroon, Priority: High, Requirement: FRQ10)

- The system must be usable off-line when technicians are mobile, and must sync up with system once connected again.

Clients
(Source: Jaco Kroon, Priority: Medium, Requirement: FRQ11)

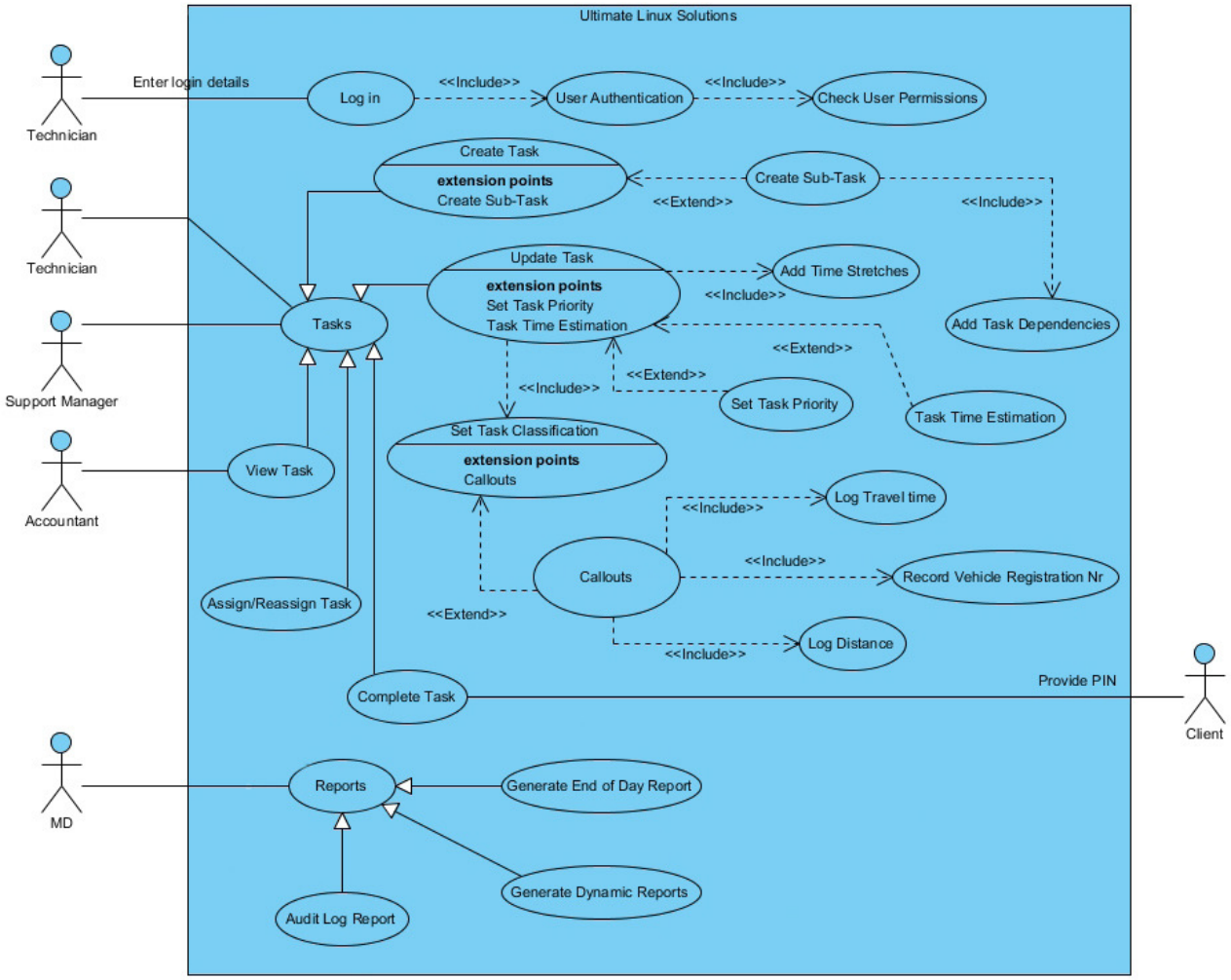
- A client must be stored

People/Resources
(Source: Jaco Kroon, Priority: Medium, Requirement: FRQ12)

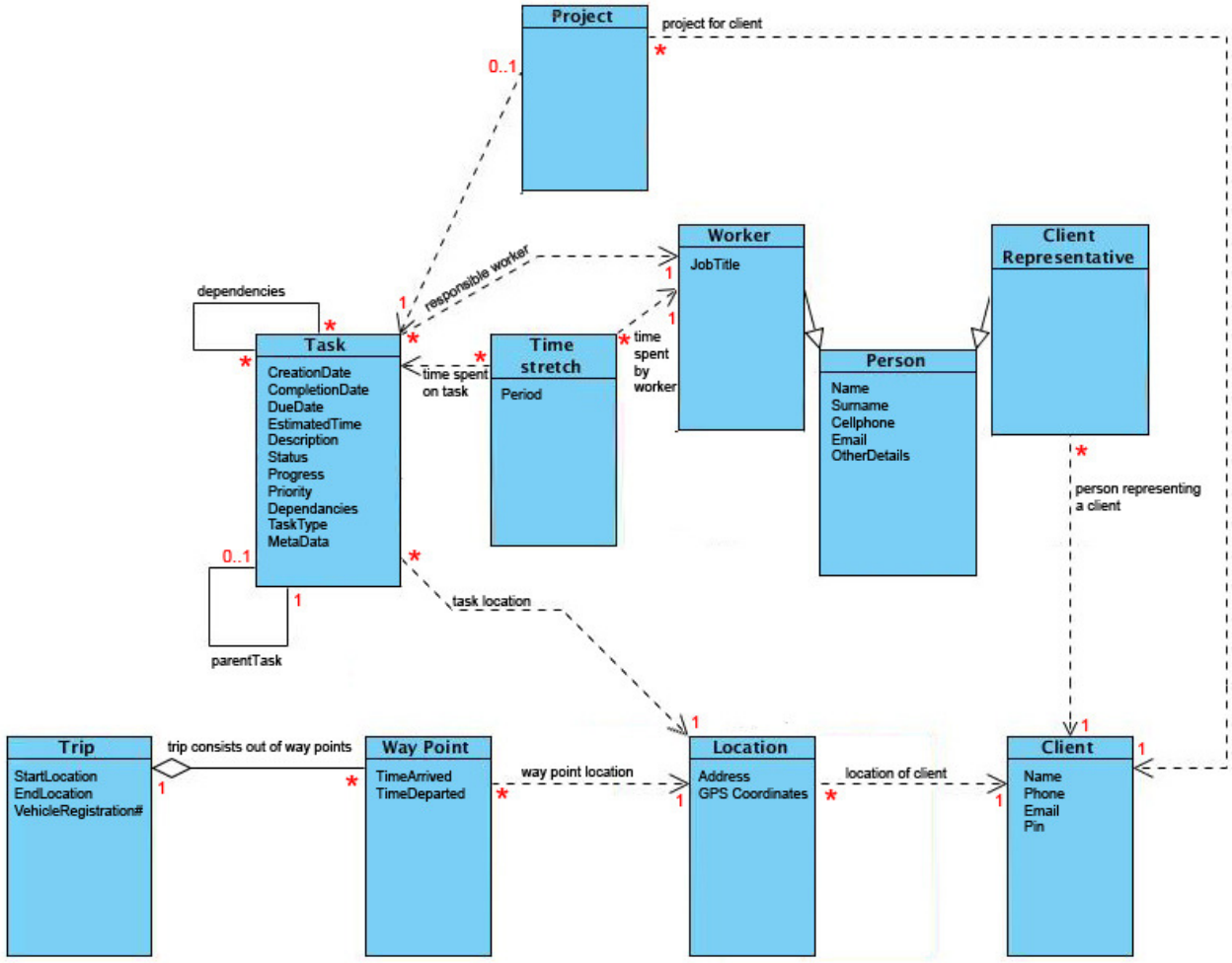
- A person/affiliate can be either one of the following:
  - Technician
  - Client representative
  - Manager
  - Accountant

- Other.. (Can later add to this list)

3.2 Use-cases



3.3 Class diagram



3.4 Data dictionary

Topic still open, to be completed as design is discussed

4 External interface requirements

The system consists of a number of interrelated sub systems which have to be able to communicate with each other. These subsystems communication takes place Sin a form of a public API. The external interface requirements are thus as follow:

- External API for communication between subsystems

- Internal API for cross-subsystem presentation and representation formatting
- External API should be built upon SOAP
- Internal API could be used for reformatting between subsystems from SOAP to:
    - CSV
    - XML
    - etc.
- The API should accept requests and provide responses to requests in a meaningful and modular way
- Information should thus be sent and received via API
- The nature of the request determines the nature of the response

5 Technical/Non-functional requirements

5.1 Non-functional requirements

Authentication
(Source: Jaco Kroon, Priority: Medium, Requirement: NFRQ1)

- There must be access control in the form of a login/logout system
- Only valid users, with valid authentication and presence on the system may gain access to the system
- Thus, authentication takes place in the presentation layer

Authorization
(Source: Jaco Kroon, Priority: Medium, Requirement: NFRQ2)

- Grant action based in model of privileges
- Thus, authorization takes place in the services layer

Scalability
(Source: Jaco Kroon, Priority: Medium, Requirement: NFRQ3)

- Should be able to handle or cater for up to 1000 users
- Performance and reliability should not be dependent on the amount of users on the system

Audit-ability
(Source: Jaco Kroon, Priority: Medium-High, Requirement: NFRQ4)

- Everything that happens on the system must be logged in some query-able fashion
- Logging includes any:
    - Additions
    - Updates
    - Changes/Deletions
    - Any executable action

5.2 Technical specifications

- Database technology: MySQL
- Should run over HTTPS
- Should be developed with specific regard to Gentoo Distribution of Linux OS
- Although PHP is a recommendation, it is not a requirement
- The following client platforms must be supported
    - Android
    - iOS - iPad/iPhone
    - Web browser (HTML 5 compliant)
    - Cross compatible OS support for Linux, Mac OS X and Windows
    - Communication between client and server subsystems must be encrypted
    - Any sensitive information stored must be stored in encrypted format

6 Requirement matrices

6.1 Requirement identifiers

Requirement Indentifiers (Functional Requirements)			
Functional Requirements			
Requirement	**Description**	**Source**	**Priority**
FRQ1	Task creation	Jaco Kroon	High
FRQ2	Task update	Jaco Kroon	High
FRQ3	Task assignment	Jaco Kroon	High
FRQ4	Task view	Jaco Kroon	Medium-High
FRQ5	Time stretch creation	Jaco Kroon	High
FRQ6	Locations	Jaco Kroon	Medium-High
FRQ7	Way points	Jaco Kroon	Medium-High
FRQ8	Trips	Jaco Kroon	Medium-High
FRQ9	Information pulling	Jaco Kroon	Medium-High
FRQ10	Off-line usage	Jaco Kroon	High
FRQ11	Clients	Jaco Kroon	Medium
FRQ12	People/Resources	Jaco Kroon	Medium
Non-Functional Requirements			
Requirement	**Description**	**Source**	**Priority**
NFRQ1	Authentication	Jaco Kroon	Medium
NFRQ2	Authorization	Jaco Kroon	Medium
NFRQ3	Scalability	Jaco Kroon	Medium
NFRQ4	Audit-ability	Jaco Kroon	Medium-High

6.2 Requirement traceability matrix

Requirement Traceability Matrix (Functional Requirements)

	FRQ1	FRQ2	FRQ3	FRQ4	FRQ5	FRQ6	FRQ7	FRQ8	FRQ9	FRQ10	FRQ11	FRQ12
Test cases												
Test case 1												
Test case 2												
Test case 3												
....												
Done												
Acceptance testing												
Acceptance test 1												
Acceptance test 2												
Acceptance test 3												
....												
Accepted												

Requirement Traceability Matrix (Non-Functional Requirements

	FRQ1	FRQ2	FRQ3	FRQ4
Test cases				
Test case 1				
Test case 2				
Test case 3				
....				
Done				
Acceptance testing				
Acceptance test 1				
Acceptance test 2				
Acceptance test 3				
....				
Accepted				

7 Open issues

· Theft/Loss of mobile devices

· Can not be responsible to client not responding to sign-off request.

· <Topic not closed; to be extended if other issues occur>

8 Glossary

· ULS - Ultimate Linux Solutions

· ERD - Entity Relationship Diagram

· Affiliate - Any external or internal person who has an influence on the task at hand

· API - Application programming interface

· Consumables - Any billable item used during the performance of a task

· Authentication - Granting access based on legitimate identifying details

· Authorization - Granting action based on nature of privileges after being authenticated