Ulysses

Software Requirements Specification

Version 1.0

November 5th, 2018

Anastasiia Roldugina: Project Manager / Team Lead

Tamara Saldina: User Interface Designer

Jordan Pike: User Experience Designer

Jerome Ching: Software Developer

Morgan Gill: Quality Assurance

Prepared for:

COMP3059 CAPSTONE PROJECT 1

Instructor: Professor Anjana Shah

Fall 2018

Contents:

[1.0 Introduction 3](#_Toc500004311)

[2.0 System Overview](#_Toc500004312) 3 - 5

[3.0 Functional Requirements](#_Toc500004318) 5 - 12

4.0 Non Functional Requirements……………………………………………………………………...13 - 14

5.0 Logical Database Requirements………………………………………………….……………………..15

# 1.0 Introduction

## Purpose

The purpose of this document is to present a detailed description of the Ulysses project web application. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to user input.

## Scope

The relative scope of Ulysses is to allow ease of uploading metrics for work orders by clients of the system for qualified personnel at InTech House Design to view and provide quotes for said clients.

# System Overview

This current chapter (chapter 2) of the document gives a general overview of the functionality of the Ulysses project. The third chapter looks to define the Functional Requirements that will need to be fulfilled in order for the project to be considered a success in addition to the data modeling and process modeling that will aid in the development of project Ulysses. Chapter four examines the Non-Functional requirements and how they will be viewed and experienced by the user base from multiple perspectives of the platform. Chapter five covers the Logical Database Requirements and why they’re structured in the way that they are in order to deliver the desired functionality of the project. The sixth chapter of this document goes into minor detail regarding any requirements that have not yet been addressed but are still ever-present. And the final chapter (Chapter 7) concludes with the signatures of the project contributors signifying that everything within this document is aligned in accordance with all of our contributors viewpoints towards this project.

## Project Perspective

The Ulysses Project is designed to be a self contained system that seeks to implement many currently used technologies to meet the needs of InTech House Design. The system has been requested by an individual under the pseudonym “Tyler Durden” to provide an easy and convenient way for the clients of InTech House Design to get in contact with qualified individuals to whom they supply various metrics for purposed construction projects. The metrics can be viewed by said qualified individuals who can then supply a quote for the project in a quick and timely manner. Our team aims to not only gather the logistical requirements necessary to properly identify the scope of the project, but also to program and design Ulysses from the ground up in order to provide a fully independent and operable platform for our primary stake holders to utilize throughout the operation of their business and that of their clients.

## System Context

The system aims to take advantage of four primary subsystems that will incorporate a variety of different types of software in order to realize it’s full functionality and potential. The web application that the majority of clients will use shall be heavily leveraging HTML, CSS, JavaScript and SASS in order to provide the user with a clean interface and responsive experience. The incorporation of an address prediction bot utilizing machine learning will ease some input from the user, allowing them to do more with less.

The desktop application exists to allow the qualified individuals (architects, engineers, etc…) to query the database that will be housing the majority of the of the usable data submitted by prospective clients. The desktop application will be running primarily on Java in order to ensure a high level of system compatibility.

The server shall be run largely on PHP incorporating “Laravel” in order to handle server requests, responses and database queries that may be sent by clients or passed to the database by users of the desktop application.

And finally, the database will be comprised of an SQL relational database following a strict schema following the conventions of “3rd Normal Form” data storage in order to reduce data redundancy while also increasing memory efficiency.

## General Constraints

Use of the AGILE project development method will impact the project by forcing incremental development on a week-by-week basis but ensuring that constant evaluation and re-evaluation is incorporated into the development of Ulysses. This will help promote a constant pace of developments while reducing scope creep, feature creep and crunch time. This methodology will also ensure that features are able to be tested for quality and functionality shortly after development in manageable “chunks” or increments rather than rushing project testing close to the end of the development window. AGILE also aims to include the primary stake holder, “Tyler Durden,” in the development process so as they can witness and evaluate the development on a week by week basis in order to stay maximally informed of the growth of Ulysses over time.

Use of a relational database with a strict schema as opposed to a non-relational database will greatly reduce the frequency of non-conforming or illegitimate data being committed to the database while also providing strict criteria through which to query valid or usable data.

Us of an external third party dedicated hosting service for the server and database will provide a steady and stable “always online” platform for which Ulysses to run, guaranteeing 24 hour service but incurring a regular monthly cost.

## Assumptions and Dependencies

Involvement of our primary stake holder (Tyler Durden) will likely be infrequent due to his work related responsibilities. Tyler Durden maintains a busy work life outside of this project as well as attending to other exterior duties. As a result, contact with him will not be able to be maintained on a weekly basis in synchronous with our stand-up meetings.

Acquiring an external domain for our online application will have to be factored into our costs as none of the stakeholders currently possesses the hardware necessary to maintain it. In addition a data base will also have to be supported and maintained as an always online service, further increasing costs.

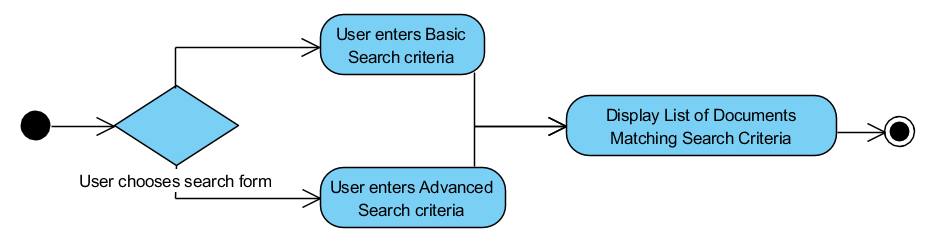
Time dedicated to the project will not be consistent from week to week due to exterior academic factors including assignments from other courses and regular scheduled testing of different knowledge sets and disciplines.

Finally, different academic time tables split across our project contributors will disrupt group member attendance of regular meetings. This will likely lead to communication gaps that will need to be filled on a regular basis.

## 3.0 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.1.0 Search Query

**Diagram**

**Trigger**

The end-user selects criteria

**Precondition**

End-user has permission to access project details and documents

**Basic Path**

1. Along with keywords, end-user specifies how to search through documents.

2. The system will generate a list of documents using keywords and other criteria presented in the search request and display them.

**Alternative Path**

In step 1, if the end-user chooses to perform an Advanced Search, the system will redirect the user to a page with an advanced search form

2. The system will generate a list of documents that match the search criteria and present them.

**Post Condition**

The system displays a webpage containing a list of zero or more documents that match the search criteria.

**Exception Paths**

The search query may be abandoned at any moment.

**Other**

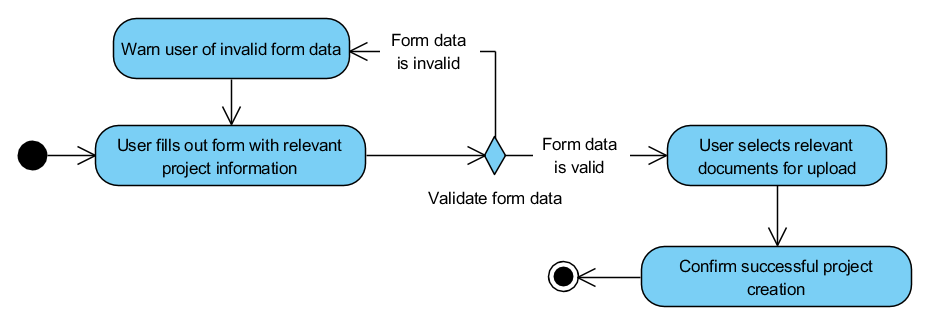
The list of documents will be generated based off the database entries made during document upload.

Keywords associated with the documents will be predefined and published during upload.

The end-user will only be able to view relevant information of projects that the end-user is a part of.

### 3.1.1 Create Project

**Diagram**



**Trigger**

The end-user clicks a Create Project link.

**Precondition**

End-user is logged in and has sufficient permissions

.

**Basic Path**

1. The system presents the end-user with a form, requesting in plain text, snippets of information regarding to project

2. End-user fills the presented fields that are required, then clicks submit.

3. After Validation, the system will redirect the end-user to an Upload Documents page.

4. End-user uploads zero or more relevant document then clicks Complete.

5. The system generates the appropriate file space and presents the end-user with a confirmation page.

**Alternative Path**

In step 3, if the system does not deem the provided information valid

3. The system redirects the end-user back to previous form with indicators for invalid fields.

4. End-user corrects information, then reattempts form submission.

5. On failure, repeat alternative step 3, otherwise return to main path step 4.

**Post Condition**

If the submitted information passes preliminary validation, the project will be created and assigns the project a proposal status.

**Exception Paths**

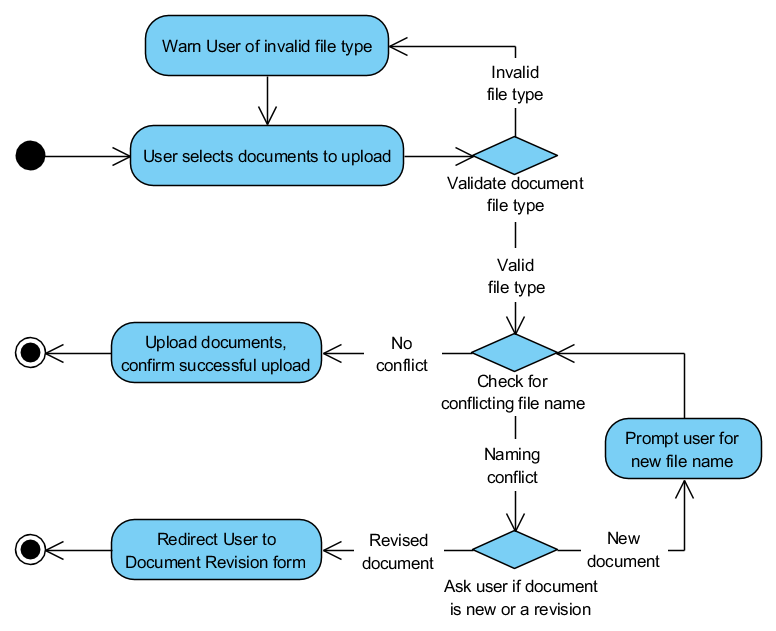
This process may be abandoned at any time. Any entered information or uploaded document will be discarded.

**Other**

The project proposal will require the name, address, type, description, and any other notable comments or remarks.

### 3.1.2 Upload Document

**Diagram**



**Trigger**

The end-user selects a document to upload

**Precondition**

End-user has permission to access project details and upload documents

**Basic Path**

1. The system presents the end-user with inputs for uploading documents.

2. The end-user selects documents from their file system for upload.

3. The system will check if selected file is of an accepted file type.

4. On success, the system will download the files from the client machine, then redirect the end-user to a page that displays confirmation for documents that were successfully uploaded

**Alternative Path**

In step 3, if the file is not of an acceptable file type, the system will present the user with a warning notification and will not allow the file to be uploaded.

In step 3, if the file has the same name as an existing file,

4. The system will prompt the end-user of the naming conflict, and request if the document is a revision, or a new file.

5. If it is a revision, go to 3.0.4 Revise Document use case, otherwise end-user enters in a new name and submits.

6. Go back to basic path step 3.

**Post Condition**

The system displays a webpage containing a list of zero or more documents that have been successfully uploaded to the server and the user is provided redirection back to their dashboard or project page.

**Exception Paths**

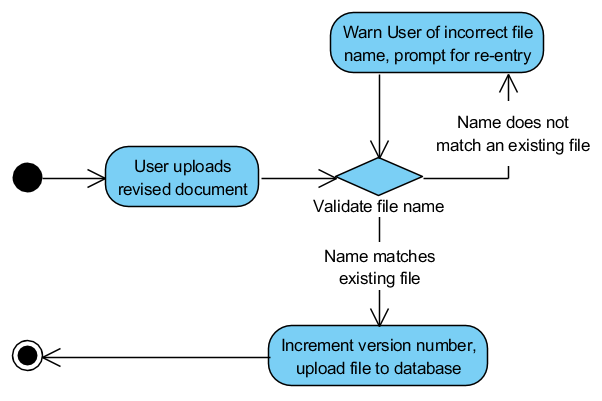
The user may exit the upload process at anytime and the system will discard any files that had been uploaded during that session.

Any file conflicts will be resolved

**Other**

Documents will be uploaded to a unique directory that is dedicated to the project.

### 3.1.3 Revise Document



**Diagram**

**Trigger**

The end-user uploads a revised document either from project summary or attempted to upload a document that already exists.

**Precondition**

User has permission to access project details and documents

**Basic Path**

1. End-user selects Upload Revised Document or attempts use case 3.0.3 Upload Document

2. The system checks the database for documents of the same name and obtains the last version number.

3. The system renames the uploaded document by appending an incremented version number and adds a new entry to the database.

**Alternative Path**

None

**Post Condition**

Document will be uploaded and accessible. Database will be updated with a new document entry.

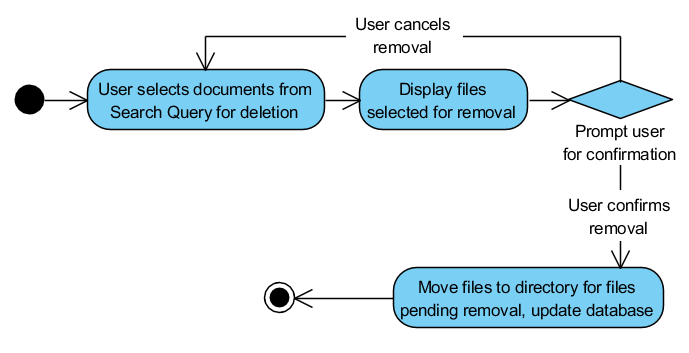
**Exception Paths**

Process can be abandoned at anytime. Any information uploaded or changed prior to the last confirmation will be discarded

**Other**

Older documents will still exist for document rollbacks or reviews.

### 3.1.3 Remove Document

**Diagram**

**Trigger**

The end-user selects a document for removal.

**Precondition**

User has permission to access project details and documents

**Basic Path**

1. The end-user selects one or more documents from a list using the 3.0.1 Search Query use case.

2. System response with a confirmation page populated with the files that had been selected.

3. End-user confirms removal.

4. System will move the files into a specific folder within the project directory that contains files that are pending removal.

**Alternative Path**

None

**Post Condition**

Files that have been selected for removal will be relocated to a designated folder that holds files that are pending removal. The system will update the database’s record of the document with the relevant status so that it will reflect in future search queries.

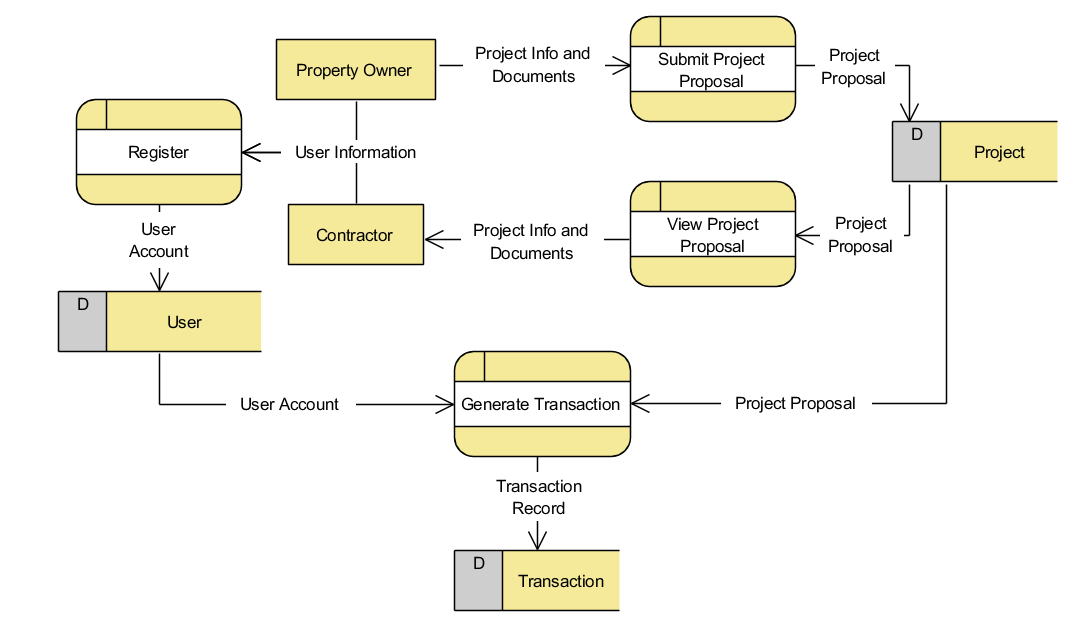
**Exception Paths**

The process may be abandoned at any time; Any changes made will be discarded.

**Other**

Documents will persist after the requested removal. Access to the removed file will be limited to users with permissions.

**3.2 Process Modelling**

**Data Flow Diagram**

## 4.0 Non-Functional Requirements

**Usability**

The interface is easy enough for user to finish the task of registering, uploading files, and submitting them.

**Security**

Only registered employees with access can review submitted client documents

**Reliability**

Neither website or desktop application will not crush if user inputs invalid data.

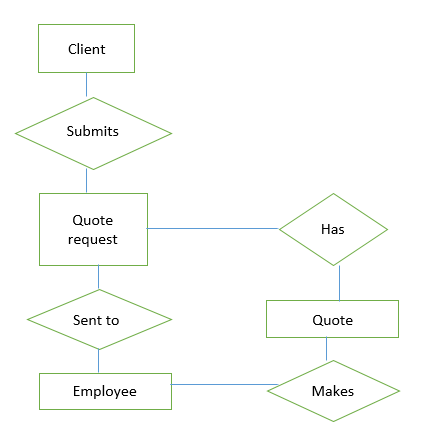
**Performance**

Front page and desktop application must be loaded under 3 seconds

**Scalability**

The website user load limit should support 100 users at a time

## 4.1 Logical Structure Diagrams

User (Web Application)

Assumptions:

1)Client is registered/logged in

2) All required documents are present

## 

## Administrator Web Application

## Employee (Desktop Application)

## 5.0 Logical Database Requirements

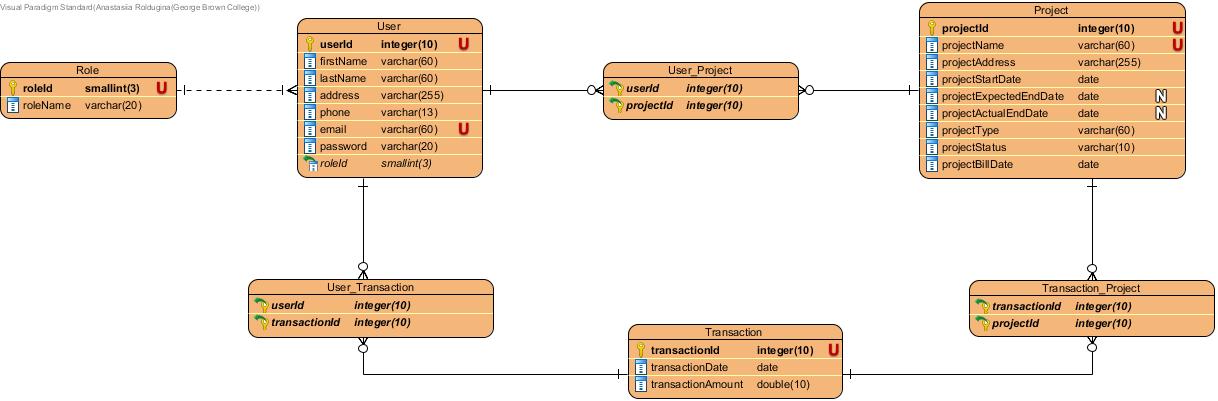
### The diagram shows the layout of database system entity relationship model.

### The system will use relational database for storing all the user account information as well as the project related data and transaction records in associated tables. All the data will be stored in a single SQL Data File. Entity integrity will be achieved by means of assigning a unique ID (integer) as a Primary Key to each user, project and transaction.

### All columns in User and Transaction tables will be non-nullable, whereas Expected End Date and Actual End Date columns in Project table can be null.

### Most of the data will be stored in string (varchar) format, with an exception of Ids (integer), Dates (date), User Role (small int) and Transaction Amount (double).

### \*\*Note: The Database format and the column data formats may change when the system is being developed.



## 

**7.0 Approval**

The signatures below indicate their approval of the contents of this document.

|  |  |  |  |
| --- | --- | --- | --- |
| Project Role | Name | Signature | Date |
| Project Manager / Team Lead | Anastasiia Roldugina | *Anastasiia Roldugina* | Nov 5th, 2018 |
| User Interface Designer | Tamara Saldina | *Tamara Saldina* | Nov 5th, 2018 |
| User Experience Designer | Jordan Pike | *Jordan Pike* | Nov 5th, 2018 |
| Software Developer | Jerome Ching | *Jerome Ching* | Nov 5th, 2018 |
| Quality Assurance | Morgan Gill | *Morgan Gill* | Nov 5th, 2018 |