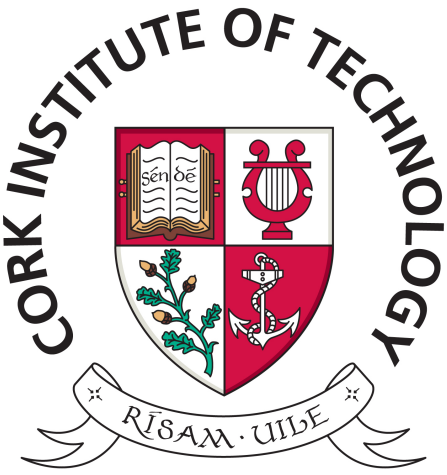


Localisation Program Management System

by

Adam Lloyd

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science in Web Development

in the
Faculty of Engineering and Science
Department of Computer Science

May 2017

Declaration of Authorship

I, Adam Lloyd, declare that this thesis titled, 'Localisation Program Management System' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Bachelor of Science

by Adam Lloyd

This program management system has been developed to satisfy the requirements of Intel Security (McAfee) for use in their software localisation department.

It features a responsive and highly usable interface with which all the relevant stakeholders can access project information. This includes financial, project, job and stakeholder details and serves as a central repository for all of this information.

All stakeholders now reference the same information resulting in work-flow improvements and a reduced error rate. This information can also be accessed and integrated with existing Intel systems via a robust API.

Acknowledgements

I would like to thank my family and friends, my supervisor Mary Davin and everyone at Intel Security (McAfee) for their support throughout the year.

Contents

List of Figures

List of Tables

Abbreviations

SQL	**S**tructured **Q**uery **L**anguage
QA	**Q**uality **A**ssurance
DEV	**D**evelopment
CSS	**C**ascading **S**tyle **S**heet
REST	**RE**presentational **S**tate **T**ransfer
UI	**U**ser **I**nterface

To Shiv - Part II - Electric Boogaloo

Chapter 1

Introduction

For my final year project I have been asked by my former employer, Intel Security whom I worked with during work placement, to develop a business management system for the localisation department to replace their current system which is unsuitable for their current requirements.

1.1 Vision Statement

For software localisation teams who want an integrated financial and project management system. This will be a bespoke management system that will streamline the process of software localisation without the need to use multiple applications and be extensible without third party support.

1.2 Executive Summary

The localisation department of Intel Security is responsible for localising all of the content that Intel Security owns. This not only includes software but sales, marketing and support documents. To ensure that all of this content is correctly localized into all of the target languages, the expertise and resources of translation vendors are leveraged.

This presents a problem such that, for large projects, thousands of words must be translated into many languages all with varying rates often involving multiple vendors to meet the language requirements of the project. With existing systems, the management of these projects is done manually resulting in time loss and increased risk of error.

To facilitate this process, an application was required to provide stakeholders with an interface to manage all project and financial details in one location. It must be customer facing to allow the translators themselves to submit the logs of their work once completed and integrate with existing localisation tools.

The resulting program management solution features a modern and highly usable interface with which all the stakeholders can access project information. This includes financial data, project planning details, project progress, details of project activities, stakeholder details and serves as a central repository for all of this information. All stakeholders now reference the same information resulting in work-flow improvements and a reduced error rate. This information can also be accessed and integrated with existing Intel systems via a robust API.

The solution has been developed from the ground up to allow for maximum potential for expanding its functionality while following the requirements spec laid out by Intel Security at the inception of the project. This was achieved by following agile development principles and adhering to a strict sprint schedule.

Chapter 2

Solution Approach

The application was developed to the requirements of Intel Security which were provided at the inception of the project. It was mandated to use technologies with which their development team was familiar and with which their other applications are developed. This ensures that the application can be maintained after it is completed by myself and that it is as compatible as possible with existing systems.

2.1 Mandated Technologies

As per the specification, the client side development was done using JavaScript and the AngularJS front-end framework. CoffeeScript was used to make the extensive JavaScript code base more manageable. SASS was used as the CSS pre-processor to integrate existing style components from other Intel applications and maintain a consistent style. The build tasks are managed by Grunt to parse the CoffeeScript/SASS and build the client for multiple deployment environments.

The server side development which will manage the data and expose an API to the client was developed using C# and the underlying database will be created in SQL Server.

2.2 Development Environment

The development of this application was integrated into Intels existing continuous integration process using Jenkins to pick up changes to the Git repository and build the application for Development and QA environments.

2.3 Project Management

The management of this project was done using the agile methodology, scrum in particular. This methodology values regular deliverable of valuable software and regular collaboration with stakeholders.

As the requirements specification was divided into sections focusing on stakeholder use cases, it was decided that the sprints would be setup in kind. 4 epics were created to manage the project at a higher level. These were the vendor, project, job and user management components.

The team opted for 3 week sprints with each sprint focusing on one of the epics listed above.

This was achieved using JIRA project management software which allowed visualisation of the sprints and updating their status conveniently. Sub tasks were created within stores and time estimates were input to ensure that there was sufficient time allocated to complete them.

Unfortunately, JIRA does not offer any export functionality so, to capture the complete overview of the stories, a spreadsheet was created, see 2.3.

FIGURE 2.1: JIRA Stories

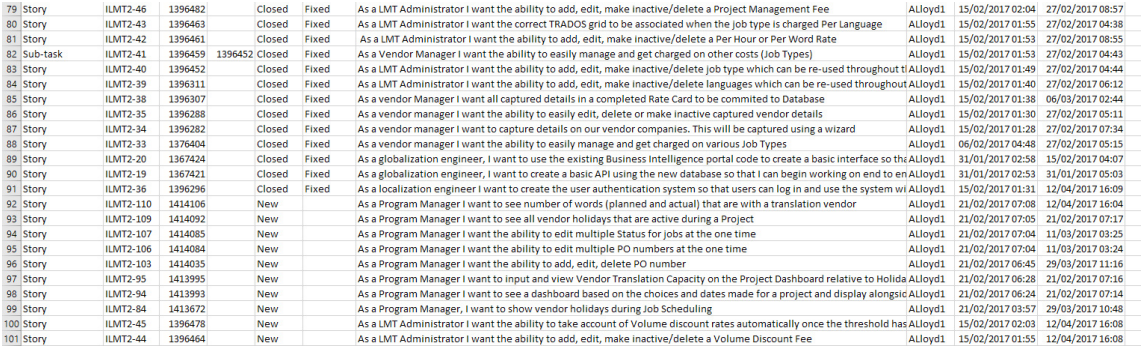
	Issue Type	Issue key	Issue id	Parent id	Status	Resolution	Summary	Assignee	Created	Updated
2	Technical Task	ILMT2-145	1474381	1474379	Closed	Fixed	Upload files during vendor edit	ALloyd1	15/04/2017 14:31	17/04/2017 18:24
3	Technical Task	ILMT2-144	1474380	1474379	Closed	Fixed	Upload files during vendor creation	ALloyd1	15/04/2017 14:31	17/04/2017 18:24
4	Story	ILMT2-143	1474379		Closed	Fixed	As a vendor manager, I want to upload SLA, MSA and SOW files and associate them wit ha vendor	ALloyd1	15/04/2017 14:29	17/04/2017 18:24
5	Technical Task	ILMT2-142	1474378	1396296	Closed	Fixed	Create dashboard system	ALloyd1	15/04/2017 13:44	15/04/2017 13:44
6	Story	ILMT2-141	1474377		Closed	Fixed	As a software engineer I want to refactor object detail view to extend parent controller so that the codebase is cleaner	ALloyd1	15/04/2017 13:41	15/04/2017 13:43
7	Technical Task	ILMT2-140	1474074	1396296	Closed	Fixed	Create the login screen	ALloyd1	12/04/2017 16:15	14/04/2017 14:16
8	Technical Task	ILMT2-139	1474073	1396296	Closed	Fixed	Create the authentication based modifications to existing pages based on the permission level of logged in user	ALloyd1	12/04/2017 16:15	17/04/2017 22:41
9	Technical Task	ILMT2-138	1474072	1396296	Closed	Fixed	Create the authentication service in the client	ALloyd1	12/04/2017 16:14	17/04/2017 22:40
10	Technical Task	ILMT2-137	1474071	1396296	Closed	Fixed	Create the authentication validation to protect sensitive routes in the API	ALloyd1	12/04/2017 16:13	17/04/2017 22:41
11	Technical Task	ILMT2-136	1474069	1396296	Closed	Fixed	Create authentication routes in the API	ALloyd1	12/04/2017 16:13	17/04/2017 22:40
12	Technical Task	ILMT2-135	1474067	1470948	Closed	Fixed	Update job	ALloyd1	12/04/2017 16:12	13/04/2017 12:41
13	Technical Task	ILMT2-134	1474064	1470948	Closed	Fixed	Delete job	ALloyd1	12/04/2017 16:11	13/04/2017 15:35
14	Story	ILMT2-133	1470953		Closed	Fixed	As a localization engineer, I want to implement the peer review feedback I receieved from Rouslan	ALloyd1	29/03/2017 11:00	12/04/2017 16:02
15	Story	ILMT2-132	1470951		Closed	Fixed	As a Program Manager I want to see that Default language assignment occurs for Products during Project Creation (cont	ALloyd1	29/03/2017 10:59	13/04/2017 12:41
16	Story	ILMT2-131	1470950		Closed	Fixed	As a Job Submitter I want to be able to view details of all Jobs in Metis	ALloyd1	29/03/2017 10:58	12/04/2017 16:02
17	Story	ILMT2-130	1470949		Closed	Fixed	As a Job Submitter I want to upload Logs to Metis	ALloyd1	29/03/2017 10:58	13/04/2017 12:41
18	Story	ILMT2-129	1470948		Closed	Fixed	As a job Submitter I want to be able to Update/delete Jobs in Metis	ALloyd1	29/03/2017 10:57	12/04/2017 16:04
19	Story	ILMT2-128	1470946		Closed	Fixed	As a Job Submitter I want to use a Job Management Wizard to create a job in Metis	ALloyd1	29/03/2017 10:56	12/04/2017 16:01
20	Technical Task	ILMT2-127	1458415	1413375	Closed	Fixed	Create group/cat/bu/product settings section	ALloyd1	13/03/2017 03:36	13/03/2017 03:38
21	Technical Task	ILMT2-126	1458261	1413375	Closed	Fixed	Create activity management section	ALloyd1	13/03/2017 03:04	13/03/2017 03:37
22	Sub-task	ILMT2-124	1456028	1413449	Closed	Fixed	Create first step of project creation wizard to persist basic data	ALloyd1	11/03/2017 02:30	11/03/2017 03:30
23	Sub-task	ILMT2-123	1441338	1413384	Closed	Fixed	Capture group/category/solution/product, start/return date and project name	ALloyd1	06/03/2017 02:46	11/03/2017 02:33
24	Story	ILMT2-121	1426601		Closed	Fixed	As an LMT Developer I want to remove PM Fee Details Page from Create Vendor Wizard	ALloyd1	27/02/2017 06:01	04/03/2017 04:15
25	Story	ILMT2-120	1426592		Closed	Fixed	As a Vendor Manager I want to be able to download the blank rate card from a browser	ALloyd1	27/02/2017 06:00	06/03/2017 02:43
26	Story	ILMT2-119	1426583		Closed	Fixed	As a vendor manager I want the ability to easily delete captured vendor details	ALloyd1	27/02/2017 05:58	06/03/2017 02:49
27	Sub-task	ILMT2-117	1424016	1396486	Closed	Fixed	Create summary screen	ALloyd1	25/02/2017 05:13	27/02/2017 05:14
28	Sub-task	ILMT2-116	1424015	1396482	Closed	Fixed	Add PM fee field to Vendor Creation wizard	ALloyd1	25/02/2017 05:12	27/02/2017 04:41
29	Sub-task	ILMT2-115	1424014	1396311	Closed	Fixed	Language CRUD fnctionality	ALloyd1	25/02/2017 05:11	27/02/2017 04:45
30	Sub-task	ILMT2-114	1424013	1396675	Closed	Fixed	Export rate card as XLSX file	ALloyd1	25/02/2017 05:10	27/02/2017 05:13
31	Sub-task	ILMT2-113	1424012	1396675	Closed	Fixed	Create rate card generation wizard	ALloyd1	25/02/2017 05:09	27/02/2017 05:08
32	Sub-task	ILMT2-112	1424011	1396288	Closed	Fixed	Add Error handling	ALloyd1	25/02/2017 05:04	27/02/2017 05:10
33	Story	ILMT2-111	1414108		Closed	Fixed	As a Program Manager I may need the ability to add attachments to a project	ALloyd1	21/02/2017 07:09	13/04/2017 12:41
34	Story	ILMT2-108	1414089		Closed	Fixed	As a Program Manager when I click on a job I want to see full details we store of the job	ALloyd1	21/02/2017 07:04	12/04/2017 15:59
35	Story	ILMT2-105	1414081		Closed	Fixed	As a Program Manager viewing the Dashboard, each Translator will need a separate line item in the common event a re	ALloyd1	21/02/2017 07:04	12/04/2017 16:01
36	Story	ILMT2-104	1414059		Closed	Fixed	As a Program Manager I want the ability to move job to a new status	ALloyd1	21/02/2017 06:56	13/04/2017 11:34
37	Story	ILMT2-102	1414031		Closed	Fixed	As a Program Manager I want to be able to see a translation request dashboard showing the following data	ALloyd1	21/02/2017 06:44	14/04/2017 14:17
38	Story	ILMT2-101	1414024		Closed	Fixed	As a Program Manager when I click on a project I want to see project specific data	ALloyd1	21/02/2017 06:43	12/04/2017 16:00
39	Story	ILMT2-100	1414022		Closed	Fixed	As a Program Manager I want to be able to search for specific projects	ALloyd1	21/02/2017 06:42	12/04/2017 16:00

FIGURE 2.2: JIRA Stories 2

40	Story	ILMT2-99	1414020		Closed	Fixed	As a Program Manager I want to be able to select number of projects shown on the dashboard	ALloyd1	21/02/2017 06:42	12/04/2017 16:08
41	Story	ILMT2-98	1414018		Closed	Fixed	As a Program Manager I want to be able to filter on column data in the dashboard	ALloyd1	21/02/2017 06:41	12/04/2017 16:00
42	Story	ILMT2-97	1414017		Closed	Fixed	As a Program Manager I want to see pre-defined filters on Projects:	ALloyd1	21/02/2017 06:38	11/03/2017 03:24
43	Sub-task	ILMT2-93	1413968	1413952	Closed	Fixed	The original release date should always be stored and be viewable	ALloyd1	21/02/2017 06:10	12/04/2017 15:59
44	Story	ILMT2-92	1413964		Closed	Fixed	As a Program Manager I want to see Project percentage complete on the Project Dashboard	ALloyd1	21/02/2017 06:09	12/04/2017 16:00
45	Story	ILMT2-91	1413960		Closed	Fixed	As a Program Manager I want to see the Health Status of my Project represented in Colours	ALloyd1	21/02/2017 06:08	13/04/2017 11:35
46	Story	ILMT2-90	1413952		Closed	Fixed	As a Program Manager I want to add/edit/review the Current Release date within the Program Management Dashboard	ALloyd1	21/02/2017 06:05	12/04/2017 15:59
47	Story	ILMT2-89	1413943		Closed	Fixed	As a Program Manager I want configurable views on stored Project data	ALloyd1	21/02/2017 06:03	12/04/2017 16:04
48	Story	ILMT2-88	1413923		Closed	Fixed	As a Program Manager I want the ability to see a dashboard on Projects	ALloyd1	21/02/2017 05:50	14/04/2017 14:17
49	Story	ILMT2-87	1413903		Closed	Fixed	As a Program Manager, within the Project Wizard, I want the ability to see a Project summary before submission	ALloyd1	21/02/2017 05:41	12/04/2017 15:59
50	Sub-task	ILMT2-86	1413739	1413732	Closed	Fixed	Program Management Activity Planning	ALloyd1	21/02/2017 04:25	29/03/2017 15:09
51	Story	ILMT2-85	1413732		Closed	Fixed	As a Program Manager I want to Request Schedule and Assign various Activities for planning purposes	ALloyd1	21/02/2017 04:22	29/03/2017 10:50
52	Sub-task	ILMT2-81	1413635	1413609	Closed	Fixed	Data to be captured during Project Creation continued	ALloyd1	21/02/2017 03:46	12/04/2017 15:58
53	Sub-task	ILMT2-80	1413627	1413609	Closed	Fixed	Data to be entered during Project Creation	ALloyd1	21/02/2017 03:45	12/04/2017 15:58
54	Story	ILMT2-79	1413609		Closed	Fixed	As a Program Manager, within the Project Wizard I want the ability to easily set up and, edit, make inactive or delete pr	ALloyd1	21/02/2017 03:39	12/04/2017 15:59
55	Story	ILMT2-78	1413449		Closed	Fixed	As a Program Manager, within the Project Wizard I will input Basic Project Information (as per current POR): Project Nan	ALloyd1	21/02/2017 02:24	27/03/2017 09:39
56	Story	ILMT2-77	1413421		Closed	Fixed	As a Program Manager, within the Project Wizard, if Assignee is YES, then a text field is displayed to add email address f	ALloyd1	21/02/2017 02:14	27/03/2017 09:38
57	Story	ILMT2-76	1413400		Closed	Fixed	As a Program Manager, within the Project Wizard, if Track by Time is YES, then a weekly time block is offered to select fr	ALloyd1	21/02/2017 02:09	27/03/2017 09:38
58	Story	ILMT2-75	1413391		Closed	Fixed	As a Program Manager, within the Project Wizard I want to add the same Project Activity several times	ALloyd1	21/02/2017 02:06	27/03/2017 09:37
59	Story	ILMT2-74	1413388		Closed	Fixed	As a Program Manager, within the Project Wizard I want to choose from a list of Project Activities	ALloyd1	21/02/2017 02:05	27/03/2017 09:37
60	Story	ILMT2-73	1413384		Closed	Fixed	As a Program Manager I want to view and use a Project Creation Wizard.	ALloyd1	21/02/2017 02:02	27/03/2017 09:36
61	Story	ILMT2-72	1413375		Closed	Fixed	As a Program Manager I want to view an Admin Section for Projects	ALloyd1	21/02/2017 02:00	27/03/2017 09:36
62	Story	ILMT2-71	1406362		Closed	Fixed	As a LMT Admin I want to have the ability to email blank rate card to vendor	ALloyd1	20/02/2017 03:06	06/03/2017 02:48
63	Sub-task	ILMT2-69	1399922	1396486	Closed	Fixed	Test Summary Screen	ALloyd1	16/02/2017 09:15	27/02/2017 05:15
64	Sub-task	ILMT2-68	1399920	1396482	Closed	Fixed	Test PM Fee	ALloyd1	16/02/2017 09:14	27/02/2017 04:40
65	Sub-task	ILMT2-67	1399919	1396461	Closed	Fixed	Test Per Word and Per hour Rate data	ALloyd1	16/02/2017 09:14	27/02/2017 04:42
66	Sub-task	ILMT2-66	1399917	1396452	Closed	Fixed	Testing Job Types	ALloyd1	16/02/2017 09:13	27/02/2017 04:44
67	Sub-task	ILMT2-65	1399916	1396311	Closed	Fixed	Test Language Selection	ALloyd1	16/02/2017 09:12	27/02/2017 04:45
68	Sub-task	ILMT2-63	1399908	1396675	Closed	Fixed	Test Blank Rate Card	ALloyd1	16/02/2017 09:10	27/02/2017 05:13
69	Sub-task	ILMT2-62	1399907	1396288	Closed	Fixed	Test Captured Vendor Details	ALloyd1	16/02/2017 09:09	27/02/2017 05:08
70	Sub-task	ILMT2-61	1399905	1396282	Closed	Fixed	Test Create Vendor Wizard	ALloyd1	16/02/2017 09:07	27/02/2017 05:12
71	Sub-task	ILMT2-60	1397825	1396463	Closed	Fixed	create trados grid model in database	ALloyd1	15/02/2017 13:39	27/02/2017 04:36
72	Sub-task	ILMT2-54	1397808	1396288	Closed	Fixed	Allow vendors to be made inactive or deleted	ALloyd1	15/02/2017 13:34	27/02/2017 05:10
73	Sub-task	ILMT2-53	1397800	1396288	Closed	Fixed	Create vendor edit view and successfully apply edit to stored object	ALloyd1	15/02/2017 13:32	27/02/2017 05:09
74	Sub-task	ILMT2-52	1397798	1396288	Closed	Fixed	Create vendor details view	ALloyd1	15/02/2017 13:32	27/02/2017 05:09
75	Sub-task	ILMT2-51	1397795	1396282	Closed	Fixed	Create vendor creation wizard	ALloyd1	15/02/2017 13:31	27/02/2017 05:12
76	Story	ILMT2-50	1396675		Closed	Fixed	As a LMT Administrator I want a Create Rate Card Wizard to be available This will output a Blank Rate Card, the details w	ALloyd1	15/02/2017 04:16	27/02/2017 08:57
77	Story	ILMT2-48	1396489		Closed	Fixed	As a LMT Administrator I want the ability to upload Completed Vendor Rate Cards easily and have all data from the rate	ALloyd1	15/02/2017 02:13	06/03/2017 02:43
78	Story	ILMT2-47	1396486		Closed	Fixed	As a Vendor Manager I want the ability to view a summary of the Blank Rate card prior to creation,to review all data fiel	ALloyd1	15/02/2017 02:08	27/02/2017 09:02

McAfee JIRA (2)

FIGURE 2.3: JIRA Stories 3



79	Story	ILMT2-46	1396482		Closed	Fixed	As a LMT Administrator I want the ability to add, edit, make inactive/delete a Project Management Fee	ALloyd1	15/02/2017 02:04	27/02/2017 08:57
80	Story	ILMT2-43	1396463		Closed	Fixed	As a LMT Administrator I want the correct TRADOS grid to be associated when the job type is charged Per Language	ALloyd1	15/02/2017 01:55	27/02/2017 04:38
81	Story	ILMT2-42	1396461		Closed	Fixed	As a LMT Administrator I want the ability to add, edit, make inactive/delete a Per Hour or Per Word Rate	ALloyd1	15/02/2017 01:53	27/02/2017 08:55
82	Sub-task	ILMT2-41	1396459	1396452	Closed	Fixed	As a Vendor Manager I want the ability to easily manage and get charged on other costs (Job Types)	ALloyd1	15/02/2017 01:53	27/02/2017 04:43
83	Story	ILMT2-40	1396452		Closed	Fixed	As a LMT Administrator I want the ability to add, edit, make inactive/delete job type which can be re-used throughout t	ALloyd1	15/02/2017 01:49	27/02/2017 04:44
84	Story	ILMT2-39	1396311		Closed	Fixed	As a LMT Administrator I want the ability to add, edit, make inactive/delete languages which can be re-used throughout	ALloyd1	15/02/2017 01:40	27/02/2017 06:12
85	Story	ILMT2-38	1396307		Closed	Fixed	As a vendor Manager I want all captured details in a completed Rate Card to be commited to Database	ALloyd1	15/02/2017 01:38	06/03/2017 02:44
86	Story	ILMT2-35	1396288		Closed	Fixed	As a vendor manager I want the ability to easily edit, delete or make inactive captured vendor details	ALloyd1	15/02/2017 01:30	27/02/2017 05:11
87	Story	ILMT2-34	1396282		Closed	Fixed	As a vendor manager I want to capture details on our vendor companies. This will be captured using a wizard	ALloyd1	15/02/2017 01:28	27/02/2017 07:34
88	Story	ILMT2-33	1376404		Closed	Fixed	As a vendor manager I want the ability to easily manage and get charged on various Job Types	ALloyd1	06/02/2017 04:48	27/02/2017 05:15
89	Story	ILMT2-20	1367424		Closed	Fixed	As a globalization engineer, I want to use the existing Business Intelligence portal code to create a basic interface so tha	ALloyd1	31/01/2017 02:58	15/02/2017 04:07
90	Story	ILMT2-19	1367421		Closed	Fixed	As a globalization engineer, I want to create a basic API using the new database so that I can begin working on end to en	ALloyd1	31/01/2017 02:53	31/01/2017 05:03
91	Story	ILMT2-36	1396296		Closed	Fixed	As a localization engineer I want to create the user authentication system so that users can log in and use the system wi	ALloyd1	15/02/2017 01:31	12/04/2017 16:09
92	Story	ILMT2-110	1414106		New		As a Program Manager I want to see number of words (planned and actual) that are with a translation vendor	ALloyd1	21/02/2017 07:08	12/04/2017 16:04
93	Story	ILMT2-109	1414092		New		As a Program Manager I want to see all vendor holidays that are active during a Project	ALloyd1	21/02/2017 07:05	21/02/2017 07:17
94	Story	ILMT2-107	1414085		New		As a Program Manager I want the ability to edit multiple Status for jobs at the one time	ALloyd1	21/02/2017 07:04	11/03/2017 03:25
95	Story	ILMT2-106	1414084		New		As a Program Manager I want the ability to edit multiple PO numbers at the one time	ALloyd1	21/02/2017 07:04	11/03/2017 03:24
96	Story	ILMT2-103	1414035		New		As a Program Manager I want the ability to add, edit, delete PO number	ALloyd1	21/02/2017 06:45	29/03/2017 11:16
97	Story	ILMT2-95	1413995		New		As a Program Manager I want to input and view Vendor Translation Capacity on the Project Dashboard relative to Holida	ALloyd1	21/02/2017 06:28	21/02/2017 07:16
98	Story	ILMT2-94	1413993		New		As a Program Manager I want to see a dashboard based on the choices and dates made for a project and display alongsid	ALloyd1	21/02/2017 06:24	21/02/2017 07:14
99	Story	ILMT2-84	1413672		New		As a Program Manager, I want to show vendor holidays during Job Scheduling	ALloyd1	21/02/2017 03:57	29/03/2017 10:48
100	Story	ILMT2-45	1396478		New		As a LMT Administrator I want the ability to take account of Volume discount rates automatically once the threshold has	ALloyd1	15/02/2017 02:03	12/04/2017 16:08
101	Story	ILMT2-44	1396464		New		As a LMT Administrator I want the ability to add, edit, make inactive/delete a Volume Discount Fee	ALloyd1	15/02/2017 01:55	12/04/2017 16:08

2.3.0.1 Sprint Planning

On the first day of sprint, the stories for the sprint were created or pulled from the backlog. Any stories left uncompleted from the last sprint were either moved back into the backlog or moved forward depending on priority. Time estimates were then given against each to ensure there was sufficient time to complete all stories within the 3 week period.

2.3.0.2 Prototyping

As each sprint started development of a new component, the next step was prototyping the interface for the required functionality. This was completed first using paper and then moving on to prototyping software and eventually implemented wire-frames.

Any domain knowledge or business need uncertainty was cleared up by meeting with the stakeholders. This often involved multiple meetings and some disagreement between stakeholders which needed to me mediated and compromises reached.

Technical issues were discussed on resolved by meeting with my technical supervisor and ensuring that these decisions were in line with the business needs outlines beforehand.

2.3.0.3 Acceptance

As stories were completed, the code was pushed to a private Git repository and a pull request was issued. This notified the technical supervisor to review the code before being approved and pushed to the operations repository. Once pushed, the Jenkins build system would pick up the changes and deploy the new build to the DEV and QA environments.

Once live on the QA server, testing stories were created and completed by members of the team. From this, changes were requested and bugs identified and logged.

Due to time constraints, any non-critical bugs were scheduled to be fixed at the end of the project if sufficient time remained.

2.3.0.4 End of Sprint

At the end of each sprint the completed component was demonstrated to the stakeholders; the sprint was completed and the cycle began again.

2.3.1 College

Along with weekly reports and demos at work weekly meetings were conducted with my college supervisor Mary Davin.

During these meetings, we would discuss the work completed that week and any issues that arose. We would discuss the plans for next week and make plans for the final report content.

Following each weekly meeting with Mary, it was required that I provide a weekly report document containing the minutes of our meeting, work completed and work to be completed next week. This was to track my progress and ensure that work was progress was consistent along with adding content to the final report.

***The full reports can be found in the Appendix - Supervisor Reports section.**

2.4 Final Implementation vs Requirements

As is the nature with the agile development process, the requirements changed constantly throughout the project. As stakeholders reviewed and tested the functionality, changes were suggested and some implemented. While minor changes were implemented, the base requirements initially outlined remained the same with the only differences being functionality that was not implemented due to time/feasibility. These are outlined in the conclusion of this report.

The technology changed in a more significant manner however. While much of the technology choices were mandated by the company, some flexibility was given with regard to the choice of JavaScript framework and the inner workings of the back-end C# solution.

2.4.1 Angular/React

The majority of the companys software is developed using angular.js, a client side JavaScript framework. However, current trends are moving towards react.js and it is being used in some of their newer projects.

The decision was made to stick with angular.js for this project as I was familiar with it already and, while it would have been interesting, the overheads of learning a new framework would have been too great. Furthermore, react.js is only a view model based

framework, so further extension with other frameworks/libraries would have been required to create a robust client application.

2.4.2 JavaScript vs Typescript vs CoffeeScript

While CoffeeScript was mandated by the company initially, consideration was given to the JavaScript framework would be used, so were consideration given to which version of JavaScript or JavaScript pre-processor to use.

While I would have been more comfortable using vanilla JavaScript without any pre-processor at all, the scale of the project would have benefited from using the more object oriented and extensible Typescript. The decision not to go with Typescript was again down to overheads. The overhead of learning the syntax form scratch were deemed too great considering the time restrictions.

CoffeeScript was settled on as it was the teams preferred syntax. It also allowed me to leverage some of their existing angular directives to speed up development and to get more efficient help from the team due to their familiarity with it.

2.4.3 LINQ to SQL vs Entity

The requirements specification did not mandate any technologies for the management of the back-end code other than C# and SQL server. As such I was free to choose how the code was structured and what frameworks to use.

LINQ to SQL is an ORM (object relational mapping) framework which uses the LINQ query language to make it easier to manage collections of data while abstracting the SQL statements to make the code easier to read/write. While this would have been a solid choice, I decided to go a step further with this process of abstraction and use the Entity framework.

Entity is an ORM framework, much like Link to SQL, that offers the capability of generating the database automatically from the code. This is referred to as "code-first" and means that there is no need to ever interact directly with the SQL database tables but rather with objects within the code. Entity handles all of the relationships (once set) and creates link tables to persist these relationships. This choice allowed me to conceptualise and visualise the domain data much easier and allowed me to make large changes without needing to recreate database tables. It also allowed me to more easily leverage the repository pattern which will be discussed later in this report.

2.5 Non-Technical Challenges

Of the challenges presented by this project, the non-technical ones were those that proved most demanding.

Taking on an extremely large project in a complex domain was always going to be a struggle. However, the realities of being the sole developer working on such a large project, despite the support of the team at Intel, were extremely challenging.

2.5.1 Domain knowledge

As this document will show in chapter 3, the domain for this project is extremely large, involving over 30 entity types and complex relationships between them. I initially attempted to model the data and the relationships within the system based purely on the requirement specification. This was somewhat successful but, as I began work on each component, it quickly became clear that much of what I had designed was unsuitable.

During our sprint planning meetings, we would discuss the features being developed and the models that this functionality relied upon. In these meetings, it would often become clear that the initial model I had designed would not be suitable as I had not completely understood the relationships between the models or their intended function. Trying to fully grasp, and improve, a system that encompassed multiple work-flows and job roles was massively challenging. Often it was only at the end of a sprint, when I had completed the functionality, that I fully understood the work I had done and how it fit into the larger picture.

2.5.2 Remote Work

As this project was designed, not for myself, but for a company, I had to rely on weekly meetings and sprint demos to ensure that I was on the correct path. Often, I was not aware of a mistake I was making in the design until a week later when it was noticed in our demo. This resulted in many setbacks and a great deal of frustration throughout.

I believe that this was inevitable and would not have been the case were I working with Intel on a daily basis. It is included here merely to highlight the challenges I faced.

2.5.3 Lack of Prototype/Planning Time

Unfortunately, the research phase module was unsuitable and of minimal benefit to my project, and potentially other projects with existing specifications. The first semester was spent doing research and theorising about options which had no bearing since the project had been defined already.

The first semester could have been more wisely spent meeting with my team at Intel, designing the data model and prototyping the interface. This would have alleviated many of the above issues and resulted in a far more robust application at the end. As it stands, I had to do all of the above as I was actually writing the application which resulted in mistakes, changes and ultimately, wasted time.

While it is understandable that one module must fit all projects and this one was an outlier, the time spent doing this should have been used more wisely and may have alleviated some of the challenges I faced during this project.

2.6 Technical Challenges

Given the overall challenge of developing a project of this magnitude in such a short time, it was inevitable that I would face substantial technical challenges.

2.6.1 .NET Web API

In my previous experience, I had touched on the .Net framework but had never created a RESTful API with it. Learning the .NET web API pattern and applying it to a project of this scale was very challenging.

2.6.2 Code Quantity

The scope of this project and the complex domain involved meant that I had to plan out the structure of the code carefully. Making use of inheritance and generic and repository design patterns, I was able to overcome this and make a lean code base.

2.6.3 Data Integrity

Of the challenges that arose from learning and using the Entity framework, data integrity was the greatest. With all of the power the framework provides, when non-standard functionality is required, it proved difficult.

2.6.3.1 Cascading Deletes

As touched on earlier, Entity abstracts the database layer by using defined code models and relationships to create the tables for the developer. This has many benefits but it began to create problems when the data models became more complex.

The below example outlines one of the main issues encountered and how it was resolved.

When modelling the Vendor entity, it needed to have a collection of Languages associated with it. Using Entity, it was possible to model this by assigning the AvailableLangauges property to be a collection of Language entities as shown in the snippet below.

```
public virtual ICollection<Language> AvailableLanguages { get; set; }
```

In order to keep track of this relationship in the database, Entity would create a foreign key column in the Language table referring to the Vendor. This works very well for one to many relationships but in this case a many to many relationship was needed. As any Language entity may be associated with any Vendor (or other Entity), the foreign key column was irrelevant and caused many errors. This was especially notable when deleting the parent Vendor which caused a cascade delete on all associated Language entities.

To resolve this, and other similar issues, the model creation rules were modified to explicitly tell Entity not to create these foreign key relationships but rather to use the WithMany() method to tell it to create a link table mapping one entity to the other. Using this method, when the parent entity was deleted, only the entry in the link table was deleted leaving the child entity intact as seen in 2.4.

2.6.3.2 Foreign Keys and Collections

Conversely to the above issue, there were points when Entity did not, or incorrectly created foreign key relationships for child and parent entities. This resulted in child collections not being recorded in the database and ultimately lost once they left run-time memory.

```
modelBuilder.Entity<Vendor>().HasMany(v => v.AvailableLanguages).WithMany();
modelBuilder.Entity<RateCard>().HasMany(r => r.Languages).WithMany();
modelBuilder.Entity<Project>().HasMany(p => p.Languages).WithMany();
modelBuilder.Entity<Project>().HasMany(p => p.Activities).WithMany();

modelBuilder.Entity<RateCard>().HasMany(r => r.HourlyCostsFromEnglish).WithMany();
modelBuilder.Entity<RateCard>().HasMany(r => r.HourlyCostsToEnglish).WithMany();


// Navigation Properties

modelBuilder.Entity<User>()
    .HasOptional(a => a.Vendor)
    .WithMany(c => c.Users)
    .HasForeignKey(a => a.VendorId).WillCascadeOnDelete(true);

modelBuilder.Entity<Category>()
    .HasRequired(a => a.Group)
    .WithMany(c => c.Categories)
    .HasForeignKey(a => a.GroupId).WillCascadeOnDelete(true);

modelBuilder.Entity<BusinessUnit>()
    .HasRequired(a => a.Category)
    .WithMany(c => c.BusinessUnits)
    .HasForeignKey(a => a.CategoryId).WillCascadeOnDelete(true);
```

FIGURE 2.4: Data Integrity

To overcome this issue, Entity was explicitly told to create the foreign keys, what to call the foreign key column and what property to look for when identifying a collection. The WillCascadeOnDelete flag was also set where the child collection needed to be removed when the parent was 2.5.

The Vendor entity was a collection of Users. The User entity has an associated Vendor and a VendorId. The manual mapping below ensures that when a User is added to the Vendors child collection, the correct VendorId foreign key is created 2.6.

2.6.4 Angular limitations

As this project used Angular 1.x, it was not possible to leverage the advanced object oriented functionality provided by Typescript and Angular 2 which would have given me the ability to use class based inheritance to streamline my code base. As the project grew though, it became clear that there would be a lot of duplicate code to create similar functionality across multiple components.

While not conventional, I was able to implement some basic inheritance into my controllers and services by creating generic versions which is outlined in the Generic Implementation section below.

```
[ForeignKey("Vendor")]
1 reference
public int? VendorId { get; set; }
1 reference
public virtual Vendor Vendor { get; set; }
1 reference
public DateTime LastModifiedDate { get; set; }
```

FIGURE 2.5: Foreign Key

```
4 references
public virtual ICollection<User> Users { get; set; }

[ForeignKey("Currency")]
0 references
public int CurrencyId { get; set; }
2 references
public virtual Currency Currency { get; set; }
2 references
public string PhoneNumber { get; set; }
1 reference
public string FTPLink { get; set; }
2 references
public bool HasSLAFile { get; set; }
2 references
public bool HasSOWFile { get; set; }
2 references
public bool HasMSAFile { get; set; }
3 references
public virtual ICollection<RateCard> RateCards { get; set; }
2 references
public DateTime CreatedDate { get; set; }
1 reference
public DateTime LastModifiedDate { get;  set; }
9 references
public virtual ICollection<Language> AvailableLanguages { get; set; }
```

FIGURE 2.6: Foreign Key 2

2.6.5 Stateless Authentication

As the server side API will need to communicate, not only with the client but with other machines, a simple and secure authentication system was required. To remove the need for sessions and conventional login (form based) methods, the decision was made to implement a stateless token based authentication system.

This differs from conventional methods such that no session is created on the server side to persist the users login state. Instead, the user logs in once and a token is created and sent back to them. This token is stored in the users browser and attached to the header of every request that follows. The server side checks this token to ensure it is valid and provides access to the API routes based on this. This removes the need to send authentication credentials repeatedly and, this the case of this system allows tokens to be assigned to machines to allow communication with the API.

While this was something that I have created in the past, implementing it in .NET was very different.

To secure each API route, an Auth attribute was created which was applied above each route and functions as middle-ware and executed before the method body 2.7.

This attribute gets the authorisation headers from the request and extracts the username and the token. It validates the credentials and grants/denies access 2.8.

```
0 references
public class ActivityController : GenericController<Activity>
{
    /// <summary>
    /// Returns the basic display fields requried for the listing view of this collection
    /// </summary>
    // GET/EntityName/display
    [Auth]
    [HttpGet]
    [Route("api/activity/getdisplay")]
    0 references
    public virtual IHttpActionResult GetDisplay()
    {
        return Ok(base.Repository.GetAll().Select(o => Format(o)));
    }
```

FIGURE 2.7: Authentication Attribute

```
53 references
public class AuthAttribute : AuthorizationFilterAttribute
{
    0 references
    public override void OnAuthorization(System.Web.Http.Controllers.HttpActionContext acti
    {
        AuthController auth = new AuthController();

        var authHeader = actionContext.Request.Headers.Authorization;

        if (authHeader != null)
        {
            if (!String.IsNullOrWhiteSpace(authHeader.Parameter))
            {
                var credArray = GetCredentials(authHeader);
                var userName = credArray[0];
                var token = credArray[1];

                if (auth.Login(userName, token))
                {
                    var currentPrincipal = new GenericPrincipal(new GenericIdentity(userNam
                    Thread.CurrentPrincipal = currentPrincipal;
                    return;
                }
            }
        }
```

FIGURE 2.8: Authentication Attribute 2

2.6.6 Parsing XLSX

A large part of the project revolves around creating and using a vendors Rate Card to calculate pricing. These rate cards contain language and hourly costs and are filled out by the vendor company and returned to Intel using an Excel spreadsheet. The application allows users to create these spreadsheets using an intuitive wizard and allows them to upload completed rate cards to input the data into the system.

```
1 reference
public void PopulateValues(ExcelWorksheet ws, RateCard rateCard)
{
    rowIndex = 2;
    colIndex = 2;

    // Populate per word costs per job type per language for From English
    foreach (JobTypeCost jobTypeCost in rateCard.JobTypeCosts)
    {
        if (jobTypeCost.Activity.ChargeType == ChargeType.PERWORD)
        {
            // Iterate through all of the languagePrices within this jobTypecost and
            foreach (LanguagePrice langPrice in jobTypeCost.ToLanguagePrices)
            {
                ws.Cells[rowIndex, colIndex].Value = langPrice.Cost;
                rowIndex++;
            }
            colIndex++;
            rowIndex = 2;
        }
    }

    // Populate the hourly costs per language
    foreach (LanguagePrice langPrice in rateCard.HourlyCostsFromEnglish)
    {
        ws.Cells[rowIndex, colIndex].Value = langPrice.Cost;
```

FIGURE 2.9: Excel Parsing

Creating the spreadsheet required code that would populate all of the cells in the spreadsheet correctly, colour/style them and set any relevant data types before sending the file back to the web browser for download.

To achieve this, helper method was created which tracks its location on the spreadsheet by a row and column index. It uses these indices, and the stored rate card data to input the correct values in each cell 2.9.

To read in an uploaded rate card a similar method was created that did the opposite. It created a virtual spreadsheet in memory form the uploaded file and, using the same indices, populated the rate card data. This was then stored in the database for later use 2.10.

```
cellValue = float.Parse(workSheet.Cells[rowIndex, colIndex].Value.ToString().Trim());
rateCard.PmPercent = cellValue;
rowIndex++;

cellValue = float.Parse(workSheet.Cells[rowIndex, colIndex].Value.ToString().Trim());
rateCard.DiscountPercent = cellValue;
rowIndex++;

cellValue = float.Parse(workSheet.Cells[rowIndex, colIndex].Value.ToString().Trim());
rateCard.MinimumCharge = cellValue;
rowIndex++;

// Skip 2 rows
rowIndex = rowIndex + 2;

// Iterate over hourly job types and populate
foreach (JobTypeCost jobTypeCost in rateCard.JobTypeCosts)
{
    if (jobTypeCost.Activity.ChargeType == ChargeType.PERHOUR)
    {
        cellValue = float.Parse(workSheet.Cells[rowIndex, colIndex].Value.ToString().T
        jobTypeCost.HourlyCost = cellValue;
        rowIndex++;
    }
}
```

FIGURE 2.10: Excel Parsing 2

2.7 Design Patterns and Benefits Gained

2.7.1 Code structure

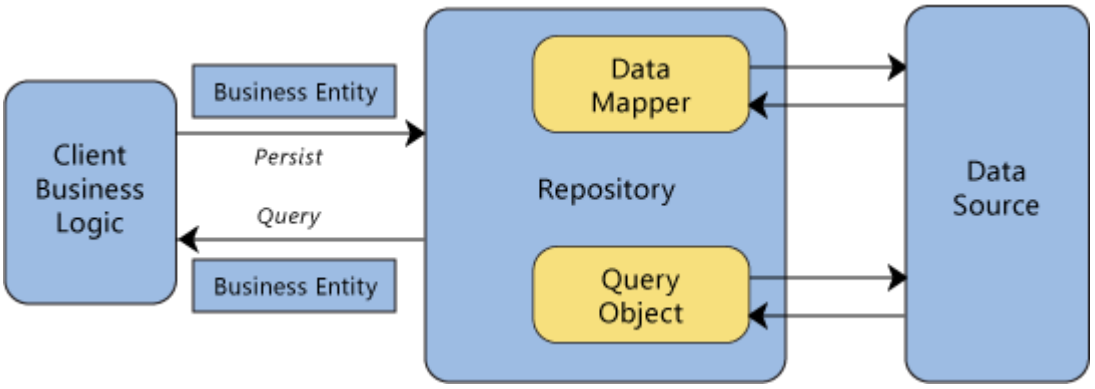
Due to the scale of this project, multiple design patterns were used to reduce duplicated code and improve the stability of the application while allowing for future extension. The points below will outline the ones that yielded the most measurable results.

2.7.1.1 Repository Pattern

Due to the number of entities being modelled and stored in the database, the project would have required a class to control the communication between the ORM and the database for each of the entities. This would have been extremely time consuming and would have created a great deal of duplicate code along with a higher potential for errors.

By using the repository pattern, it was possible to separate the business logic from the database interactions by adding a layer in between them called a repository. This layer outlines methods for basic CRUD functions. A generic interface was also created for this which abstracts the data access layer such that it can be implemented for both production data and for test data without the need for multiple data access methods.

FIGURE 2.11: Repository Pattern [**?**]



As can be seen in the snippets below, the generic IRepository interface is associated with a TEntity class which is mapped to a concrete entity when the repository is implemented. This interface outlines the method bodies of all core data access methods.

The implementation shown in the snippets shows the interface being implemented and associated with a ServerContext object which contains all of the collections the database will manage. In an alternative implementation of the interface a TestingContext may be passed in this using the same data access methods with a different data set.

The only overhead to using this pattern is perhaps the readability of the code to a junior developer. Otherwise it is the recommended means of implementing a robust data access layer.

FIGURE 2.12: Repository Pattern 2

```
public class Repository<TEntity> : IRepository<TEntity> where TEntity
{
    private ServerContext _context;
    private DbSet<TEntity> _dbSet;

    2 references
    public Repository(ServerContext context)
    {
        _context = context;
        _dbSet = context.Set<TEntity>();
    }

    17 references
    public void Commit()
    {
        _context.SaveChanges();
    }

    2 references
    public void Delete(object id)
    {
        TEntity entity = _dbSet.Find(id);
        if(entity == null)
        {
            throw new ArgumentNullException("Cannot find entity with i
        }
        _dbSet.Remove(entity);
    }

    2 references
    public void Delete(TEntity entity)
    {
        _dbSet.Remove(entity);
    }

    1 reference
    public void Dispose()
    {
        _context?.Dispose();
    }

    11 references
    public IEnumerable<TEntity> GetAll()
    {
```

FIGURE 2.13: Repository Pattern 3

```
public class Repository<TEntity> : IRepository<TEntity> where TEntity : class
{
    private ServerContext _context;
    private DbSet<TEntity> _dbSet;

    2 references
    public Repository(ServerContext context)
    {
        _context = context;
        _dbSet = context.Set<TEntity>();
    }

    17 references
    public void Commit()
    {
        _context.SaveChanges();
    }

    2 references
    public void Delete(object id)
    {
        TEntity entity = _dbSet.Find(id);
        if(entity == null)
        {
            throw new ArgumentNullException("Cannot find entity with id: " + id);
        }
        _dbSet.Remove(entity);
    }

    2 references
    public void Delete(TEntity entity)
    {
        _dbSet.Remove(entity);
    }
```

FIGURE 2.14: Repository Pattern 4

```
public class GenericController<TEntity> : LoggingApiContr
{
    private ServerContext _context;
    private Repository<TEntity> _repository;

    0 references
    public GenericController()
    {
        _context = new ServerContext();
        _repository = new Repository<TEntity>(_context);
    }
```

2.7.1.2 Generic Implementation

Generic implementation involves creating one class that will work with any number of models. This is achieved by creating code that does not know what model it will be using but rather uses a template or does not need to know what model it is at all. The result of this being that one controller, service or other programming construct can be used for multiple models reducing the amount of duplicate code and increasing test-ability and robustness.

2.7.1.2.1 Server Side The .NET and Entity frameworks allows the use of the TEntity class which is a template entity. It is effectively a blank model which will match any other entity model when implemented. See Fig 2.15.

The GenericController class accepts a TEntity and the class uses this TEntity, in conjunction with the repository model outlined above, to provide basic API routes such as GET which returns all of the entities in a data set. As can be seen, this implementation allows for consistent error handling and removes the need to create API routes for all of the 20+ entities in this project thus massively reducing the code needed.

As can be seen in the next snippet, any controllers requiring non-standard routes simply extend the base controller and add/override the base routes. The implementation of the base controller and the association with the concrete entity is done in the controller definition by passing it Job in the snippet. This is passed to the base class and used in place of TEntity from then on. See Fig 2.16.

The overheads of using this pattern are only in the error logging, such that the generic implementation only offers generic error messages.

FIGURE 2.15: Generic Pattern

```
ublic class GenericController<TEntity> : LoggingApiController where TEntity : class
{
    private ServerContext _context;
    private Repository<TEntity> _repository;

    0 references
    public GenericController()
    {
        _context = new ServerContext();
        _repository = new Repository<TEntity>(_context);
    }

    // GET/EntityName
    [Auth]
    [HttpGet]
    0 references
    public virtual IHttpActionResult Get()
    {
        try
        {
            return Ok(_repository.GetAll());
        }
        catch (Exception ex)
        {
            string e = "Could not get the requested items, the collection may be empty";
            Error(ex, e);
            return BadRequest(e);
        }
    }
```

FIGURE 2.16: Generic Pattern 2

```
public class JobController : GenericController<Job>
{

    // GET/EntityName/display
    [Auth]
    [HttpGet]
    [Route("api/job/getdisplay")]
    0 references
    public virtual IHttpActionResult GetDisplay()
    {
        return Ok(base.Repository.GetAll().Select(o => Format(o)));
    }

    1 reference
    private object Format(Job job)
    {
        return new
        {
            id = job.id,
            Name = job.Name,
            Requester = job.Requester.FullName,
            RequestDate = job.CreatedDate,
            ReturnDate = job.ReturnDate
        };
    }

    [Auth]
    [HttpPost]
    [Route("api/job/exists")]
    0 references
```

2.7.1.2.2 Client Side On the client side, the same issues presented themselves in that there were large amounts of models to be managed. JavaScript and Angular 1.x however, are not fully object oriented and do not support conventional inheritance and abstraction so an unconventional method was used.

A generic controller was created which provided functionality to get and display collections of objects and provide common CRUD functions in the interface. This generic controller was then injected into the named controllers and their scope was passed back to the generic controller. This provided all of the functionality from the generic controller using the scope of the named controller. See Fig 2.17.

FIGURE 2.17: Generic Pattern Client

```
angular.module('app')

  .controller 'LanguageCtrl', (
  $controller
  $scope
) ->

    # Get the type of entity to be queries
    $scope.type = 'language'

    $controller('GenericCtrl',{$scope:$scope});
```

For simple objects this allowed creation of very thin controllers as seen in the snippet below and for more complex ones, to be able to add and encapsulate the more advanced functionality.

A similar effect was used with the generic service that was created. The controller passes a type variable when making calls to the generic service and that is interpolated into the API URL string to specify the correct controller route on the server side. This allows me to simulate a generic implementation of all basic CRUD routes and removes the need for a service for each object. See Fig 2.18.

FIGURE 2.18: Generic Pattern Client 2

```
.service 'GenericService', ($http, $q, $config) ->

  baseUrl = "#{$config.serverAddress}"

  # Handles all of the get routes by appending the provided t
  @get = (type, url) ->
    if(!url)
      url=""
    url = baseUrl + type + '/' + url
    def = $q.defer()
    $http.get(url).then(
      (response) ->
        def.resolve response.data
      (response) ->
        def.reject(response)
    )
    def.promise

  # Update the entity
  @update = (type, object) ->
    def = $q.defer()
    $http.put(baseUrl + type + "/" + object.id, object).then(
      (response) ->
        def.resolve response.data
      (response) ->
        def.reject(response)
    )
    def.promise
```

The overheads for this are greater than that of the .NET implementation as it is non-standard. It does decrease the readability of the code and, without the structure being explained, it may confuse many developers.

2.7.2 Interface Design

Similar to the design of the code base, the interface also implements design patterns. These patterns involve using commonly uses elements and layouts that the user will find easy to learn. I will also displaying data in convenient ways.

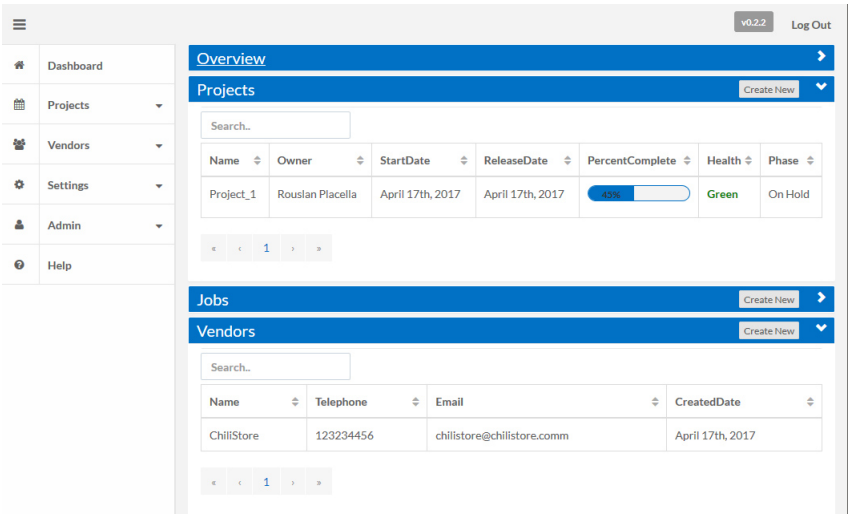
2.7.2.1 Dashboard

As this system will be used by multiple types of user, each with different permissions, a way to easily display all data that was relevant for each user was needed. To achieve this, the dashboard pattern was used. See Fig 2.19.

This pattern allows the user to see relevant data immediately and conveniently as they log in. This removes the need for them to navigate to other sections of the application to complete common tasks and also offers the potential for customisable views to personalise their data feed even more.

The overheads involved in implementing this pattern are in the extra code and views needed to customise the data feed for each user.

FIGURE 2.19: Dashboard



2.7.2.2 Consistent design

While not a conventional design pattern, much of the interface design was mandated to be consistent with existing Intel software. This was to reduce the learning curve for users and allow the system to fit in seamlessly with the existing suite of software already in use.

This allowed me to leverage existing style and code libraries to speed up development and integrate further with existing systems.

While this did speed up development it hindered creativity and often workarounds had to be created for parts of the system that were not suitable for the existing design.

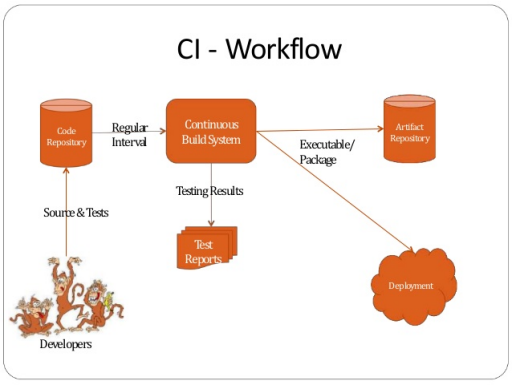
2.8 Verification and Validation

2.8.1 Continuous Integration

As with all engineering projects at Intel, this project was integrated into their continuous integration system. Continuous integration is the process of multiple developers (in this case only 1) working on the same code base via source control and their work being verified by an automate build system. In this case, we used Git as our source control repository and Github to manage it with Jenkins as the build system.

Jenkins watches the code repository for changes and, when found, runs a stored script which will run tests and build the application for deployment before copying it to the target environment. See Fig 2.20. In this case Jenkins deployed the application to 2 separate virtual machines, one for QA and one for DEV.

FIGURE 2.20: Jenkins [**?**]



2.8.2 Form Validation

To reduce the potential for error when submitting entity creation forms, validation was implemented on each step of the creation wizards. This ranged from simply ensuring that the field was completed to fully checking the validity of the input with the server before allowing the user to move onto the next step.

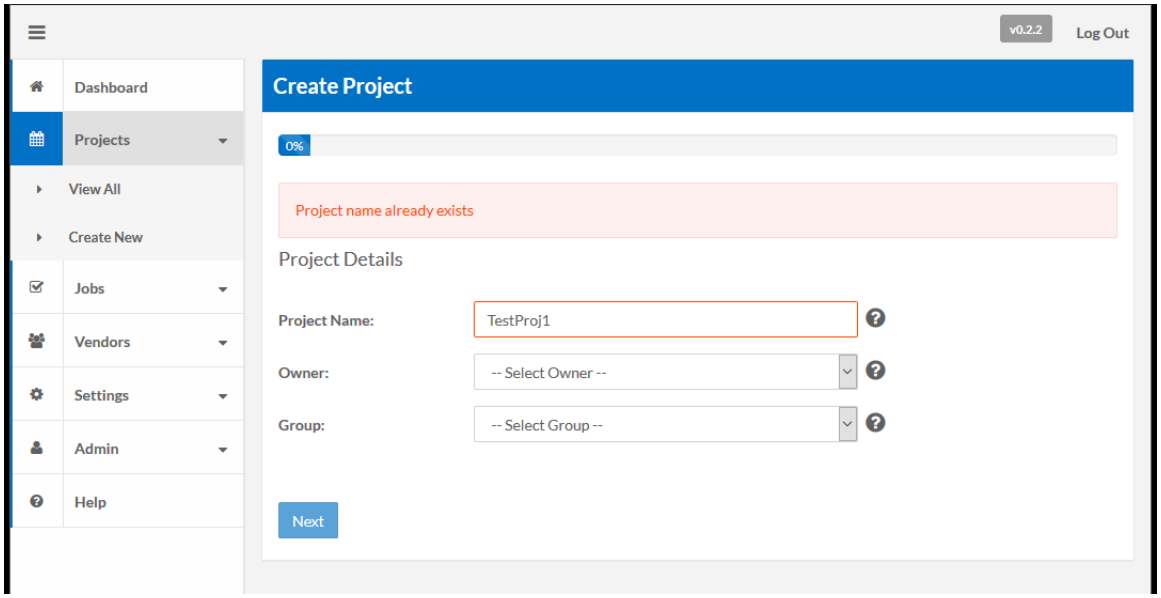
To create a consistent, reusable means of validating that the entity being created has a unique name on the server. an angular.js directive was created. See Fig 2.21. This is and can be applied as attribute to any HTML element and will validate via the API that the value entered is unique. It is generic and minimal and causes a relevant error message to appear on the wizard if a unique name is found. See Fig 2.22.

FIGURE 2.21: Directive

```
.directive('uniqueName',  (GenericService) ->

  return {
    restrict: 'AE'
    require: 'ngModel'
    link: (scope, elem, attrs, model) ->
      model.$asyncValidators.nameExists =  ->
        GenericService.exists(attrs.object, elem.val()).then(
          (response) ->
            scope.nameExists = response
            model.$setValidity('nameExists', response)
        )
  }
)
```

FIGURE 2.22: Directive



2.8.3 Unit testing

While initially mandated by the requirements, there was insufficient time to create the functionality required and also write tests for it. As such only minimal tests were written to test basic angular controller functions.

To be properly integrated into the CI system a full suite would be required to ensure future functionality can be created safely without causing instability.

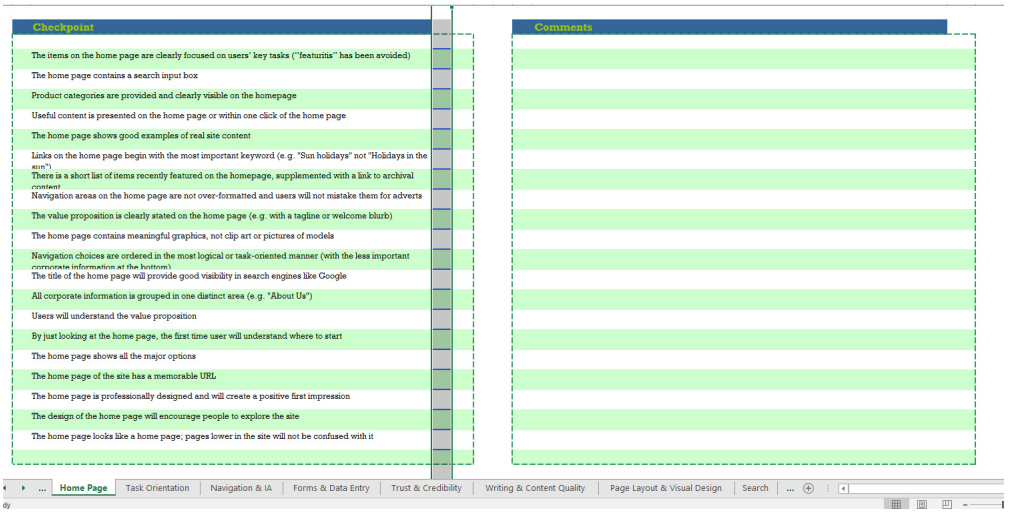
Jasmine was used for the test code as it is written in JavaScript as is the client side code it will be testing. Tests written in Jasmine are easily readable and the output is descriptive as it provides more verbose errors telling the developer where the problem is in English rather than a complex stack trace.

Selenium is the software which will be used to automate the web browser. The test code will connect to a locally running Selenium instance and issue commands to the browser via selenium which will simulate the actions of a user.

2.8.4 Usability testing

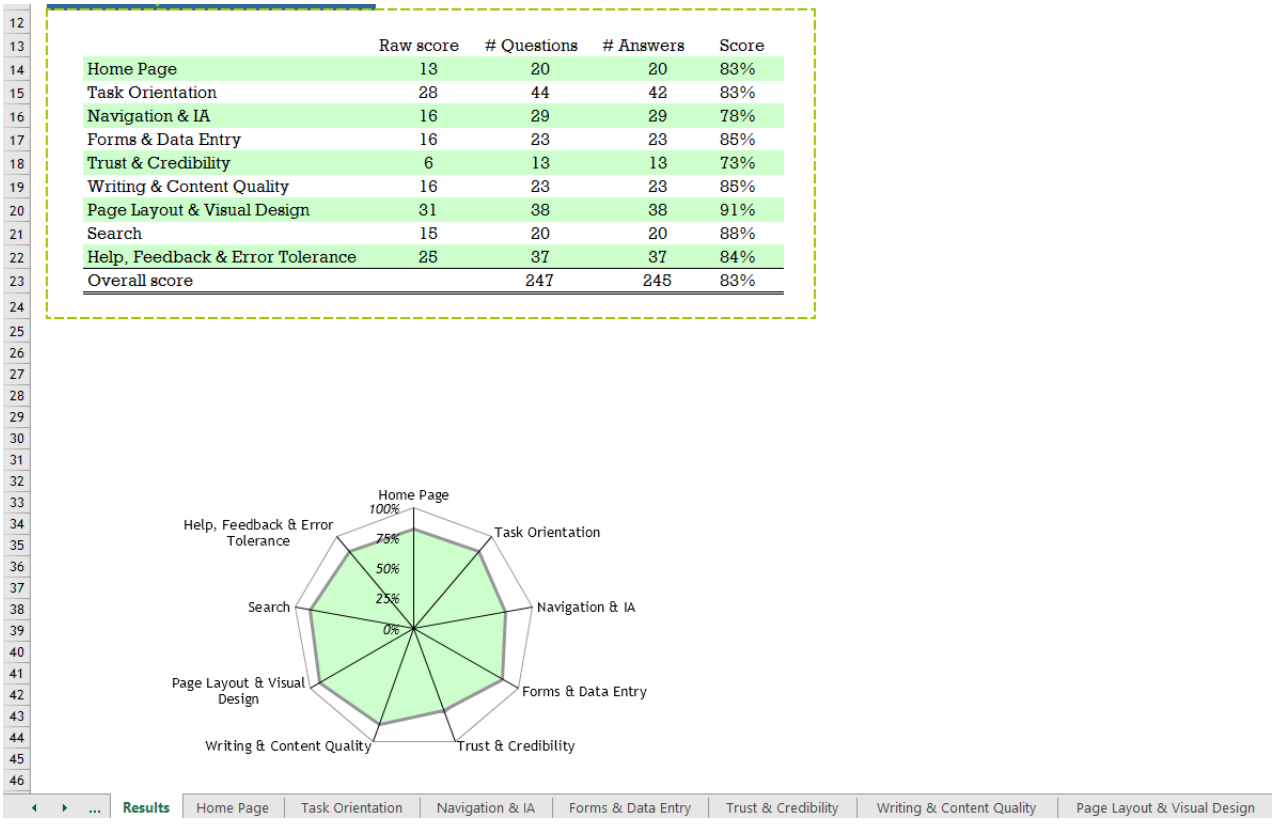
Usability testing was carried out using the Expert Review checkpoint spreadsheet provided by my supervisor. See Fig 2.23. The interface of the application was separated into multiple sections each consisting of a series of questions for the testing user to answer. These answers are either -1, indicating that the application does not comply with the guideline 1, indicating that the application does comply with the guideline or 0, indicating a middle ground.

FIGURE 2.23: Usability Spreadsheet



The many sections of this spreadsheet were completed by one of the stakeholders and can be seen in the Appendix - Usability Spreadsheet section. The output from all of this data was compiled and is displayed in Fig 2.24 showing the breakdown and the overall usability score.

FIGURE 2.24: Usability Spreadsheet Output



	Raw score	# Questions	# Answers	Score
Home Page	13	20	20	83%
Task Orientation	28	44	42	83%
Navigation & IA	16	29	29	78%
Forms & Data Entry	16	23	23	85%
Trust & Credibility	6	13	13	73%
Writing & Content Quality	16	23	23	85%
Page Layout & Visual Design	31	38	38	91%
Search	15	20	20	88%
Help, Feedback & Error Tolerance	25	37	37	84%
Overall score		247	245	83%

2.8.5 Security

As this system will eventually be customer facing, security was a major concern.

On the client side, angular.js comes with a built-in module to sanitize form inputs to prevent command injection attempts. However, with the inherent security issues with client side code, most of the security measures were implemented on the server side.

Along with the authorisation requirement to access API routes outlined earlier, permission based restrictions were also placed on sensitive routes to prevent unauthorised access. This was achieved simply by checking the authenticated users Type property against the one required to access that route.

To ensure that Cross Origin Requests (Requests from other servers) were not allowed this option was turned off in the API configuration class. However, cross origin was required for testing locally so it needed to be conditionally enabled in DEBUG runtimes. To do this .NET conditional code blocks were used as can be seen in the snippet to surround the cross-origin code as can be seen in Fig 2.25.

Command/SQL injection protection is offered by the Entity framework so was not a concern for this project.

FIGURE 2.25: Conditional Runtime code

```
using System.Net;
using System.Web.Http;

#if DEBUG
using System.Web.Http.Cors;
#endif

namespace ServerAPI
{
    1 reference
    public static class WebApiConfig
    {
        1 reference
        public static void Register(HttpConfiguration config)
        {
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls;

#if DEBUG
            config.EnableCors(new EnableCorsAttribute("http://localhost:9000", "*", "*") { SupportsCredentials = true });
#endif
            // Web API routes
            config.MapHttpAttributeRoutes();
            config.Formatters.JsonFormatter.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Chapter 3

Solution Demonstration

The core problem was to refine and automate the work-flow of the software localisation department of Intel Security by providing a modern web application to fulfill their requirements.

The resultant application fulfills their requirements as they were laid out at the inception of this project.

3.1 Stakeholder Definition

The stakeholders of the system were outlined in the requirements document provided to me at the beginning of this project and represent the main user roles that will be interacting with this system.

3.1.1 Admin

A localisation system administrator who is responsible for managing the user and project data within the localisation systems. They are familiar with engineering and administrative tasks.

3.1.1.1 Persona

David is a system admin, for the current financial and project management systems of Intel Securitys Software localisation department.

He has worked in the engineering department for 12 years and is familiar with software localization processes both technical and managerial.

He is responsible for managing account and project data within the system.

- David McCourt
- Age 39
- Software Localisaiton Systems Administrator.

David will use the application to set up new user accounts and modify existing accounts. He needs to be able to create accounts, look up accounts and easily modify account permissions. He also sometimes needs to add and edit projects to the system.

3.1.1.1.1 Use Cases

- As an Administrator, I want to modify the list of languages codes the system uses.
- As an Administrator, I want to modify the user list.

3.1.1.1.2 Red Routes

- As an Administrator, I want to manage user accounts so that I can add new users to the system.
- As an Administrator, I want to be able to create a new project on the system so that translations jobs can be associated with it

3.1.2 Program Manager

A software localisation project manager who is responsible for managing the project at a high level. They deal with the translation vendors; the software engineers and upper management so are a hub between them all.

3.1.2.1 Persona

Derry is the director of program management at Intel Security. He is responsible for overseeing the various localisation projects that Intel Security handles and ultimately paying the translation vendors for their services once the jobs are completed.

- Derry Leary

- Age 48
- Program Manager

Derry uses the system to view the status of projects and translation jobs, to assign financial data to these jobs and to calculate the payment due to the translation vendors once completed.

3.1.2.1.1 Use Cases

- As a Program Manager, I want to create and modify projects.
- As a Program Manager, I want to see a summary of a project before submission.
- As a Program Manager, I want to see a dashboard on Projects.
- As a Program Manager, I want to filter data in the dashboard.
- As a Program Manager, I want to search for specific projects.
- As a Program Manager when I click on a project I want to see project specific data.
- As a Program Manager, I want to see a job on the dashboard.
- As a Program Manager, I want to see the full details of the job (Outlined in Engineering User Stories)

3.1.2.1.2 Red Routes

- As a Program Manager, I want to set up and modify projects.
- As a Program Manager, I want to search for specific projects.
- As a Program Manager when I click on a project in the dashboard, I want to see detailed project data.
- As a Program Manager, I want to see the full details of the job (Outlined in Engineering User Stories)

3.1.3 Engineer

A localisation software engineer is responsible for maintaining the various applications and frameworks that the localisation department uses.

3.1.3.1 Persona

Ross is a senior software engineer at Intel Security. He has worked for the company for 3 years and is currently lead developer for the continuous integration team.

He is extremely familiar with all of Intels engineering processes.

- Ross Place
- Age 31
- Software Localisaiton Engineer

Ross uses the system primarily via the API but frequently needs to edit translation requests directly in the system.

3.1.3.1.1 Use Cases

- As an Engineer, I want to modify jobs.
- As an Engineer, I want to view the details of a Submitted job.
- As an Engineer, I want to view the language details of a job and see individual cost information.

3.1.3.1.2 Red Routes

- As an Engineer, I want to modify jobs.
- As an Engineer, I want to view the details of a Submitted job.

3.1.4 Vendor

A translation vendor is a company who provides translation services. There will be many of these vendors used for each software localisation project and they are located all over the world.

3.1.4.1 Persona

Martina is the lead Quality Manager at ChilliStore Technologies, a localisation company based in Dublin.

She is responsible for ensuring the quality of the translations that are sent back from her team are as accurate as possible and in the correct context for the application being developed.

- Martina Bower
- Age 35
- Localisation Quality Manager

Martina uses the system to view translation jobs assigned to her team and to download the translation documents provided by Intel Security which are then fed into her systems and translated by her translators. Upon completion, she also uses the system to upload completed translation documents back into the system.

3.1.4.1.1 Use Cases

- As a vendor, I want a dashboard that will allow me to view all jobs and languages assigned to me
- As a vendor, I want to upload logs and have costs calculated automatically by the system.

3.1.4.1.2 Red Routes

- As a vendor, I want a dashboard that will allow me to view all jobs and languages assigned to me
- As a vendor, I want to upload logs and have costs calculated automatically by the system.

3.1.5 Vendor Manager

A vendor manager is responsible for managing accounts of translation vendor companies in the system. They maintain the prices that these companies charge and calculate payment due from translation jobs.

3.1.5.1 Persona

Morris is a vendor manager for the software localisation department of Intel Security, he has worked at Intel for 10 years and is familiar with managerial processes.

He is responsible for managing the localisation of software projects and liaising with the translation vendors and engineers to ensure the project runs smoothly.

- Morris Peavey
- Age 45
- Vendor Manager

Morris will use the system to create vendor accounts, projects and manage the vendor price plans.

He needs an intuitive system that allows him to set up accounts quickly and repeat the task with the same options afterwards to add multiple accounts.

He also needs the ability to set up and modify the vendor price plans which detail how much each vendor chargers for their services.

3.1.5.1.1 Use Cases

- As a vendor manager, I want to create and modify vendor companies.
- As a vendor manager, I want to upload, edit, make inactive or delete vendor Price plans (also known as rate cards)
- As a vendor manager, I want to modify Project categories.
- As a vendor manager, I want to manage costs for non-linguistic services (development, external QA, Screenshot Creation, etc.)
- As a vendor manager, I want to create and modify vendor companies.

3.1.5.1.2 Red Routes

- As a vendor manager, I want to create and modify vendor companies.
- As a vendor manager, I want to create and modify vendor Price plans (also known as rate cards)

3.2 The Vendor Component

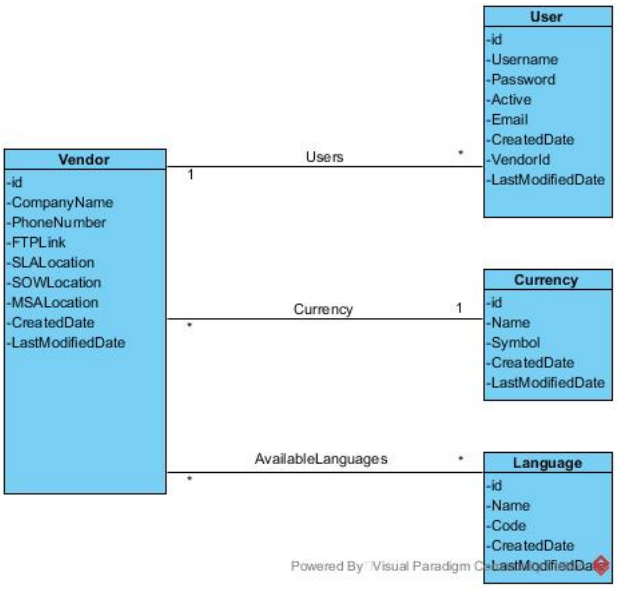
The first component developed was the vendor component. The vendor entity is at the core of all financial operations within the system and the creation and management of these entities was decided to be a good first step as the pricing information contained within their rate card would be used for many other parts of the system.

This component involved storing vendor details, associating them with rate cards and enabling the vendor manager to download and upload these rate cards in the form of Excel spreadsheets.

3.2.1 Entities involved

The first part of this component was simply to create Vendor entities. As they will be using the system, they require a User entity to be associated with them to provide authentication. This user is created at the same time as the vendor and given a temporary password. The Vendor also must be associated with a Currency entity to track the currency they use to charge Intel. Finally they have a child collection of AvailableLanguages which stores the languages that this vendor supports for translation services.

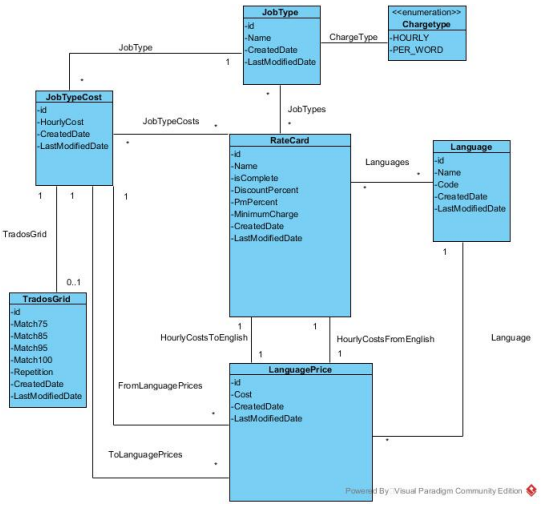
FIGURE 3.1: Vendor Domain Model



Next, the RateCard entity was implemented and this was far more complex. The RateCard entity will represent all of the hourly and per word costs for each language that the Vendor supports. For the per word costs it will also store the TradosGrid match rates which determine what percentage of the full cost the word is charged at. This can be

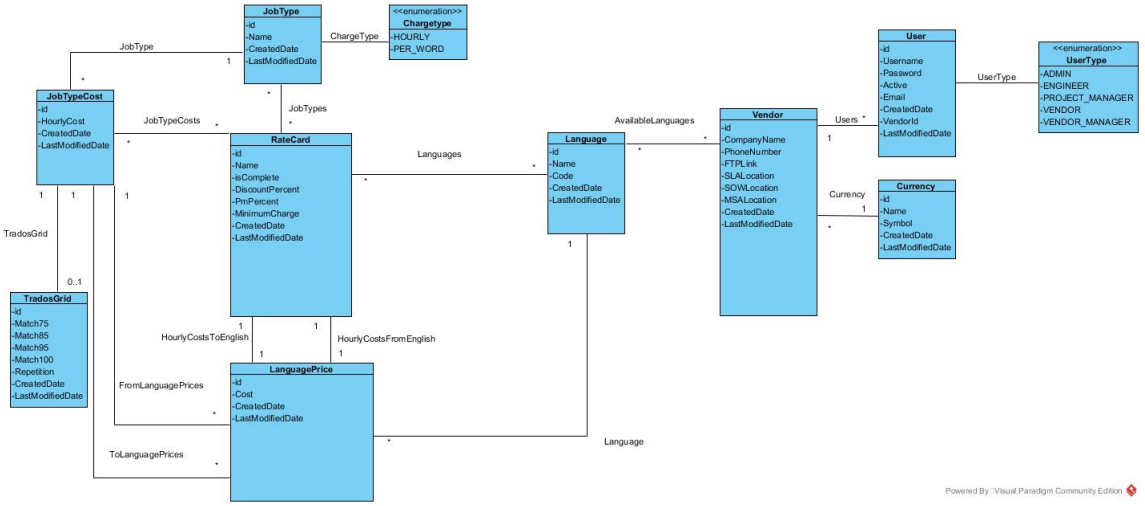
anywhere from 0%, new words that are not in their translation memory to 100%, words that completely match a word in the translation memory.

FIGURE 3.2: Vendor Domain Model



The model for the entire Vendor component, as seen in the diagram, is quite large but lays the ground work for many of the features which were developed later. The RateCard specifically is used to perform many of the calculations within the Job management component.

FIGURE 3.3: Vendor Domain Model



3.2.1.1 Primary Use Cases

- As a vendor manager, I want to create and modify vendor companies.
- As a vendor manager, I want to upload, edit, make inactive or delete vendor price plans (also known as rate cards).

- As a vendor manager, I want to manage costs for non-linguistic services.

3.2.1.2 Example Use Case

Morris wants to add a new vendor to the system and add their ratecard to the system so that they can be assigned to projects.

*** Due to formatting constraints of LaTeX, the accompanying screen-shots for this use case are in the Appendix - Vendor Management Use Case Screen-shots**

1. Morris logs into the system using his username and password.
2. From the Vendor Management page, Morris clicks on the Create New button.
3. This opens the Vendor creation wizard.
4. From here he inputs the vendors company name, email, phone number and preferred currency.
5. He also fills in the login username field to provide login credentials for the vendor which will be supplied to them.
6. On the next step of the wizard, he selects the languages that this vendor company provides services in.
7. Finally, a confirmation screen is presented to him. Where he reviews the information before creating the new Vendor in the system.
8. Now that the new vendor is created, it appears in the list of vendors and can be viewed.
9. He clicks on the newly created vendor and then on the Rate Cards tab.
10. As this is a newly created vendor, it does not have any rate cards associated with it so one must be created. To do this he clicks on the Create Rate Card button.
11. This opens the Rate Card creation Wizard.
12. A randomly generated name is automatically created but can be replaced if desired.
13. Next the set of languages that this rate card will store prices for is selected.
14. Next the set of activities that this rate card will store prices on is selected.

15. Finally a summary is displayed before he clicks the Confirm and Create button to add the new rate card to the system.

16. Now when he goes back into the details view of that vendor, the new rate card and related options can be seen.

3.3 Project Management Component

The project management component is used to track projects which contain multiple individual jobs.

A project can contain contain multiple activities from which jobs can be created for each activity.

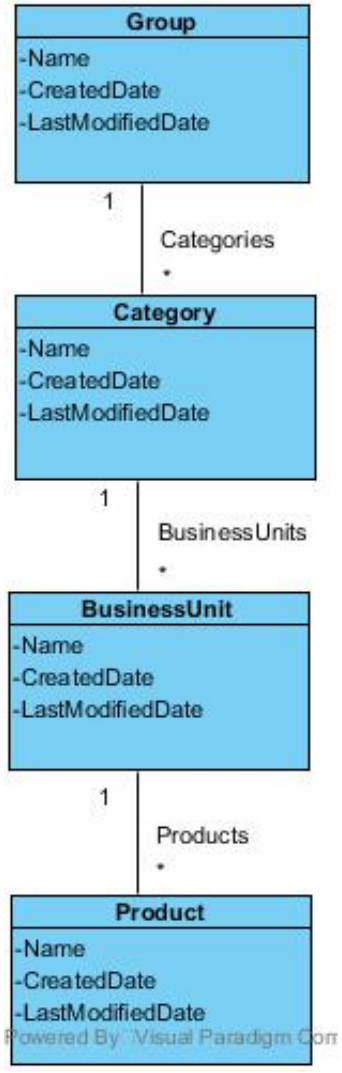
This component allows program managers to have a high level overview of projects and track their progress over time.

3.3.1 Entities Involved

Before a project is created, it must be assigned to a specific product. These products are part of Business Units, these Business Units are part of Categories that are part of Groups 3.4.

This separation allows for more concise browsing of projects and potentially more granular reporting if implemented later.

FIGURE 3.4: Groups Domain Model



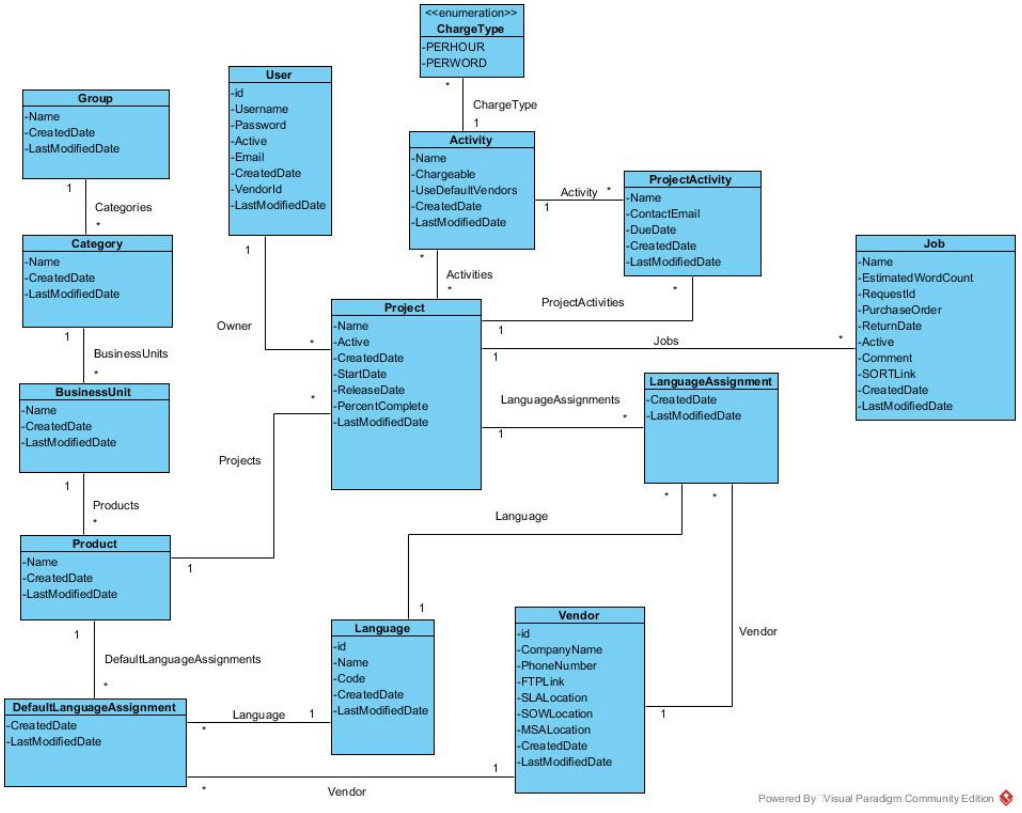
Once associated with a product, the project is created and becomes quite complex 3.5.

It must be associated with multiple activities which are used to create named project activities which represent the different types of work that will be completed within this project.

A vendor must be assigned to work on each language that the project will contain jobs for and this is stored as a language assignment entity. This will pull from a predefined list of default language assignments which can be set in the settings menu for each product. It can, however, be overridden when creating a project and later a job.

Finally the job is associated with a user entity defined as the project owner. This property will be used at a later date to provide filtering options within the dashboard and project views.

FIGURE 3.5: Project Management Domain Model



3.3.2 Primary Use Cases

- As a Program Manager, I want to create and modify projects
- As a Program Manager, I want to see a summary of a project before submission

- As a Program Manager, I want to search for specific projects
- As a Program Manager, when I click on a project I want to see project specific data
- As a Vendor Manager, I want to modify project categories.

3.3.2.1 Example Use Case

Derry is a program manager and wants to create a new project so that jobs can be associated with it and progress can be tracked.

*** Due to formatting constraints of LaTeX, the accompanying screen-shots for this use case are in the Appendix - Project Management Use Case Screen-shots**

1. Derry logs into the system using his username and password.
2. From the Project Management page he clicks on the Create New button.
3. This opens the Project Creation Wizard.
4. Derry enters the name of the project, the owner (usually himself) and selects a group for this project to be a part of.
5. This causes additional options to be displayed and he chooses a Category, BusinessUnit and a product for the project before choosing a start date and a release date.
6. Next, he chooses the languages which this project will use.
7. Based on this selection he is presented with options to assign specific vendors to each language or to assign them all to one vendor.
8. The activities that this project will manage are selected next and then he is prompted to name them and add a return date for each one. It is also possible for him to add multiple instances of the same activity here should it be repeating. In this case he repeats UA.
9. Finally, he is presented with a summary screen and asked to confirm before creating the new project.
10. The project now appears in the project list and its details can be viewed.

3.4 Job Management Component

The job management component builds on the project component to create child job entities which are contained within projects.

These jobs represent a single activity that a project manages and are associated with one or more vendors. These vendors complete work and upload their logs into the system whereby the financial data is calculated and invoices can be created upon completion of work.

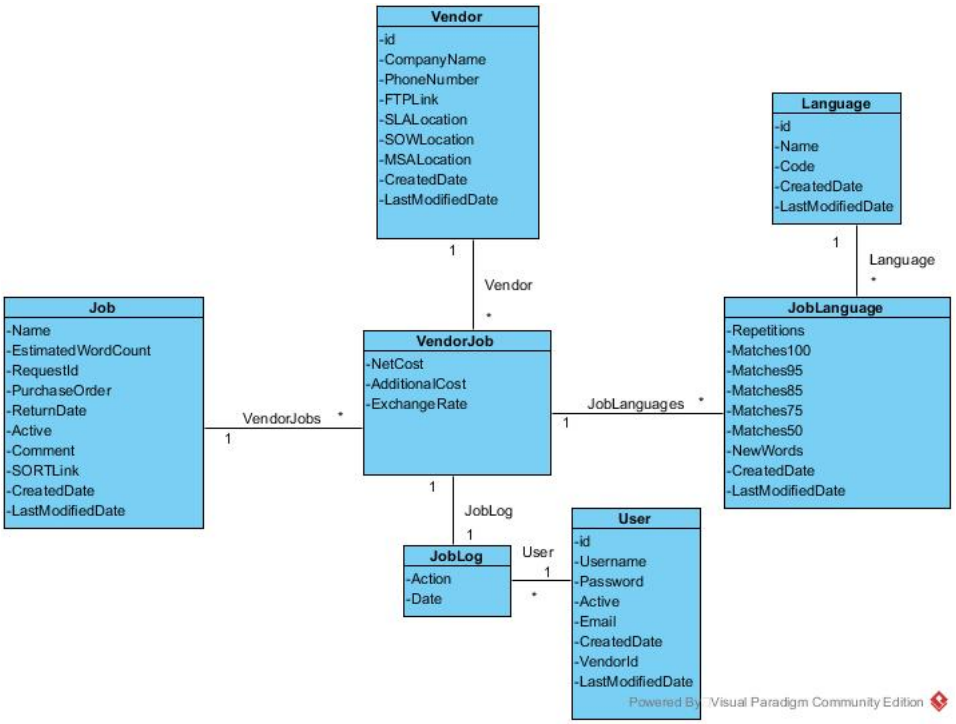
3.4.1 Entities Involved

The job entity is primarily a container for the more complex vendor job entities. It contains high level job data that is applicable to all contained vendor jobs 3.6.

These vendor jobs are associated with a vendor and contain a separate entity for each language within the vendor job. This allows separate logs and information to be stored for each language.

Finally a job log entity is associated with the job which tracks changes made over time.

FIGURE 3.6: Job Management Domain Model



3.4.2 Primary Use Cases

- As an Engineer, I want to modify translation requests.
- As an Engineer, I want to view the details of a Submitted job.
- As a Program Manager, I want to see full details we store of the job.
- As a vendor, I want a dashboard that will allow me to view all jobs and languages assigned to me.
- As a vendor, I want to upload logs and have costs calculated automatically by the system.

3.4.2.1 Example Use Case

Ross is an engineer and he wants to create a new translation job on the system so that translation work can be completed, tracked and invoiced.

*** Due to formatting constraints of LaTeX, the accompanying screen-shots for this use case are in the Appendix - Job Management Use Case Screenshots**

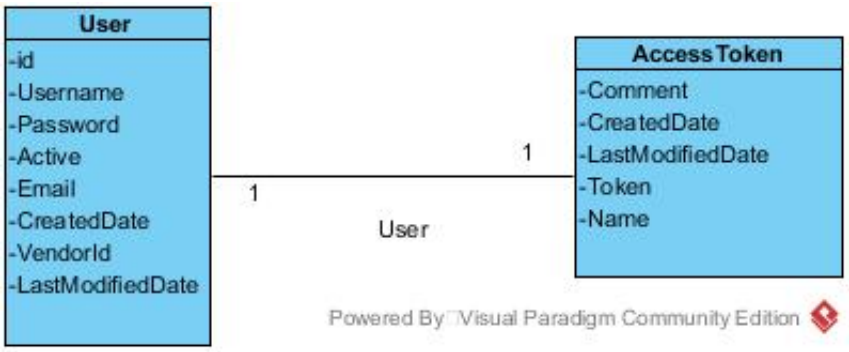
1. He logs into the system with his username and password.
2. From the Job Management screen, he clicks on the Create New button.
3. Here he enters the job name, the request id, purchase order number and the requesting user (usually himself)
4. Next, he associates the job with a project by selecting a group, category, business unit, product and then project. He also chooses an activity for this job to complete.
5. Next, he selects a return date for the job.
6. Next, he adds a link to the screen-shots on SORT if available and uploads the attachment files which contain the content to be translated.
7. He is then asked to confirm and create the new job.
8. The newly created job now appears in the list of jobs and its details can be viewed.

3.5 Authentication/Dashboard Component

The authentication system, at a conceptual level merely associates a user entity with an access token to allow secure stateless authentication via the API.

3.5.1 Entities Involved

FIGURE 3.7: Authentication Domain Model



3.5.2 Primary Use Cases

- As a Program Manager, I want to see a dashboard on Projects
- As a Program Manager, I want to filter data in the dashboard
- As a Program Manager, I want to see a translation request on the dashboard

3.5.2.1 Example Use Case

Derry wants to log into the system and view a dashboard on projects.

*** Due to formatting constraints of LaTeX, the accompanying screen-shots for this use case are in the Appendix - Authentication Use Case Screenshots**

1. He logs into the system with his username and password.
2. He is then presented with a dashboard page containing relevant information.

3.6 The Full System

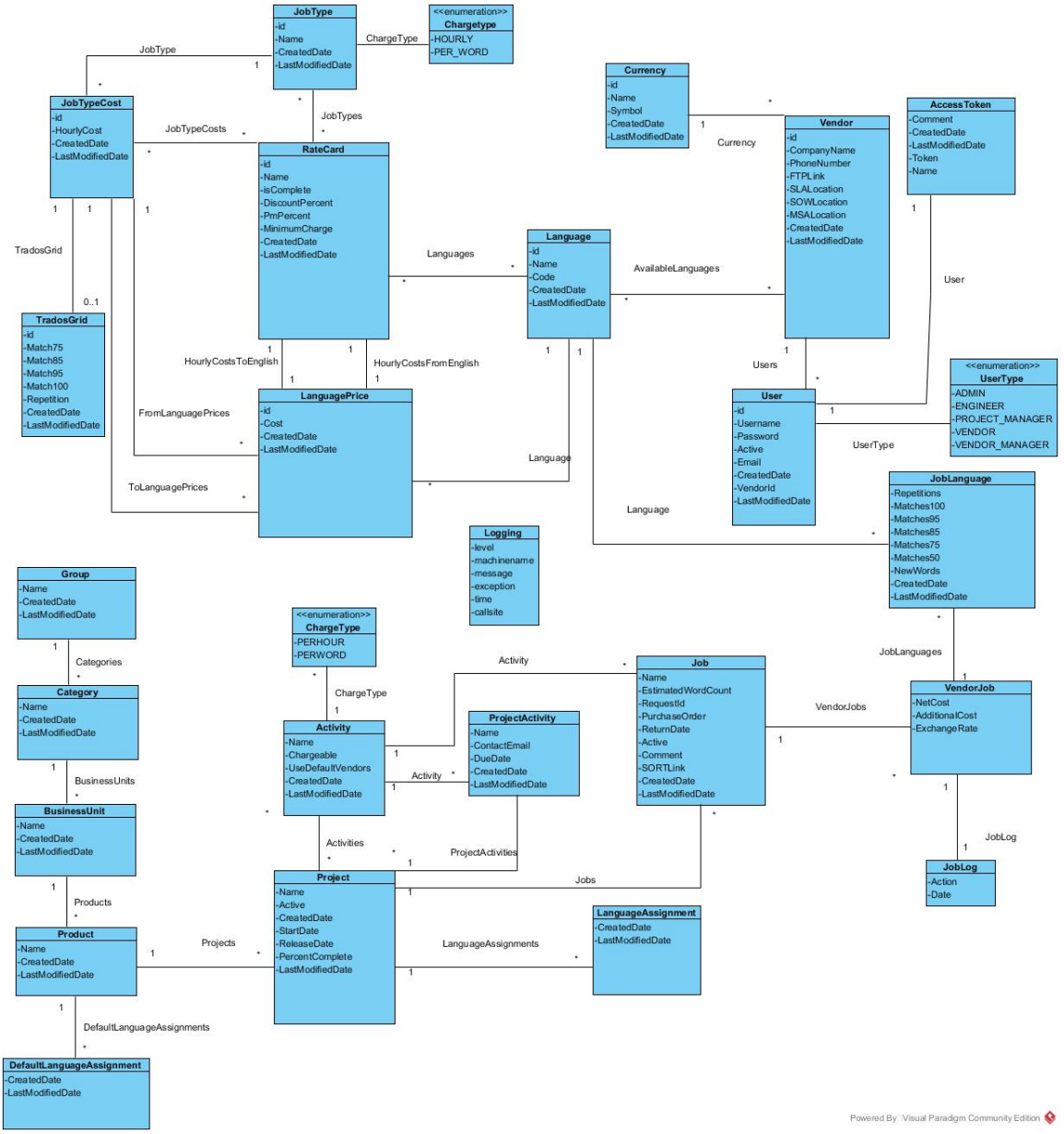
All of the above components fit together to form the entire system which, while still rather simplistic in relation to its potential, is complex and functional.

At the centre of the system is the Vendor component which connects the projects and jobs to pricing information stored in the rate cards. This is the reason it was developed first.

Common language, currency, activity and other simple entities are used throughout the system and all reference the same objects. Child collections are used to create a hierarchy and preserve entity separation in the system to keep things as manageable as possible.

While there is no use case which would encompass the entire system, the separate use cases for each of the components can be seen in the above sections. Their interactions each make up a use case for a specific job role at Intel.

FIGURE 3.8: Full System Domain Model



Chapter 4

Conclusions and Future Work

4.1 Solution Review

The requirements for this project were defined at the start and the look and feel had to reflect existing Intel applications. This left little room for innovation from an interface standpoint but also freed up my time that would have been spent designing to focus on other, more technical aspects.

4.2 Key Skills

Throughout this project, my existing skills were tested to a high degree. The new skills I gained are extremely valuable and the experience working on an enterprise product prepared me well for a career in professional development.

4.2.1 Skills Developed

Working with CoffeeScript was a large challenge as it changes the JavaScript syntax completely. By learning this skill I feel more confident about exploring other JavaScript pre-processors in the future such as TypeScript and Babel which will serve me well in the future.

The decision to use the Entity framework within .NET was risky but provided me with great experience working with a highly abstracted ORM framework on a very large scale. The problems and solutions I encountered while working with this have been very educational.

4.2.2 Skills Leveraged

When I agreed to take on this project I had a small amount of experience using angular.js and .NET but I had not used it on a such a large scale. It did, however, allow me to skip the initial steep learning curve involved with the 2 frameworks and focus on tacking the more challenging technical aspects of the project.

My education and experience with web design allowed me to quickly adapt the Intel HTML template and styles to suite the unique needs of this project while saving time.

My experience working with REST APIs allowed me to conceptualise the API routes prior to creation to ensure a consistent implementation for all functions.

4.2.3 Enterprise Software Development

Working within the constraints and systems of Intel provided great experience but also many challenges. Managing the needs of the stakeholders, the various development environments and doing all this while the company was changing from Intel to McAfee provided unique challenges.

Overall I am glad I completed the project for a company as it looks great on my CV and I have gained skills I will use throughout my career.

4.3 Future Work

It was clear at the beginning of this project that I would not have sufficient time to complete it to its full potential. However, it was clear that the team had lofty plans for the system in the future so it was crucial to consider these while developing the core functionality.

The basic project management component will be scaled up to become a full featured system to replace the multiple systems currently in use. This would allow their project management work-flows to be captured and integrated completely with the system.

The vendor component will be scaled up to provide more automation and information about vendor capacity so that projects can be planned more effectively.

The job management component will be scaled up to allow for exporting of various formats and, along with the project management component, provide robust reporting functions.

The authentication system will be fleshed out as it is still quite bear bones. API routes will be properly secured to ensure the system is not vulnerable when it is deployed to production.

A full suite of unit and end to end tests will be completed to ensure stability while developing new features.

4.4 Incomplete Work

The vast majority of the original requirements have been fulfilled in this project, however, there were some that did not. I was also given the choice on certain technologies to be used and some choices I made differ from the initial specification.

4.4.1 As a vendor manager, I want the ability to take account of Volume discount rates automatically once the threshold has been reached.

This story did not get completed due to time constraints. While the data to handle, this feature is captured, it is not reflected in the UI.

This story remains in the backlog.

4.4.2 As a vendor manager, I want to record vendor language capacity and indicate this when new projects are created to ensure that vendors do not get assigned more work that they can handle.

This feature, while good on paper, did not work in the real world. None of the stakeholders were able to decide how this would be implemented in such a way that there was not a massive error rate or without micro management of the data.

It is still in the backlog and may be figured out later on in the projects development.

4.4.3 As a vendor manager, I want to record vendor holidays for specific language staff so that projects can be assigned and completed without delay during holiday seasons.

This story was ultimately shelved. Nobody could figure out a way to gather and update this data without devoting too much time to maintaining it or relying on the vendors themselves to keep it current.

This story was removed from the backlog as it was deemed impractical.

4.4.4 As an application owner, I want a Restful API set that will accept log data from Translation Manager and assign it to the correct job

This story, along with any other interoperability stories did not get completed due to time constraints. While all efforts were made to ensure that the new software will be backwards compatible, some time will be required up on completion to make the minor adjustments required to implement the API functionality outlined in the requirements document.

The interoperability stories remain in the backlog.

4.5 Conclusion

Overall this project was a success. At the inception it was unclear if this was even possible given the amount of time I had but I felt I was up to the challenge.

It was extremely difficult to get a grasp on the vast domain while simultaneously developing the features that involved the domain. As mentioned earlier in the document, I would have benefited greatly from a stronger focus on planning and development in the first project phase.

The resultant application, while somewhat bare bones, will provide a solid platform for the company to build on. It will give them a massive head start towards creating the application they envisioned and, that I was able to complete it, is a massive personal victory.

Bibliography

Appendix A

Vendor Management Use Case Screenshots

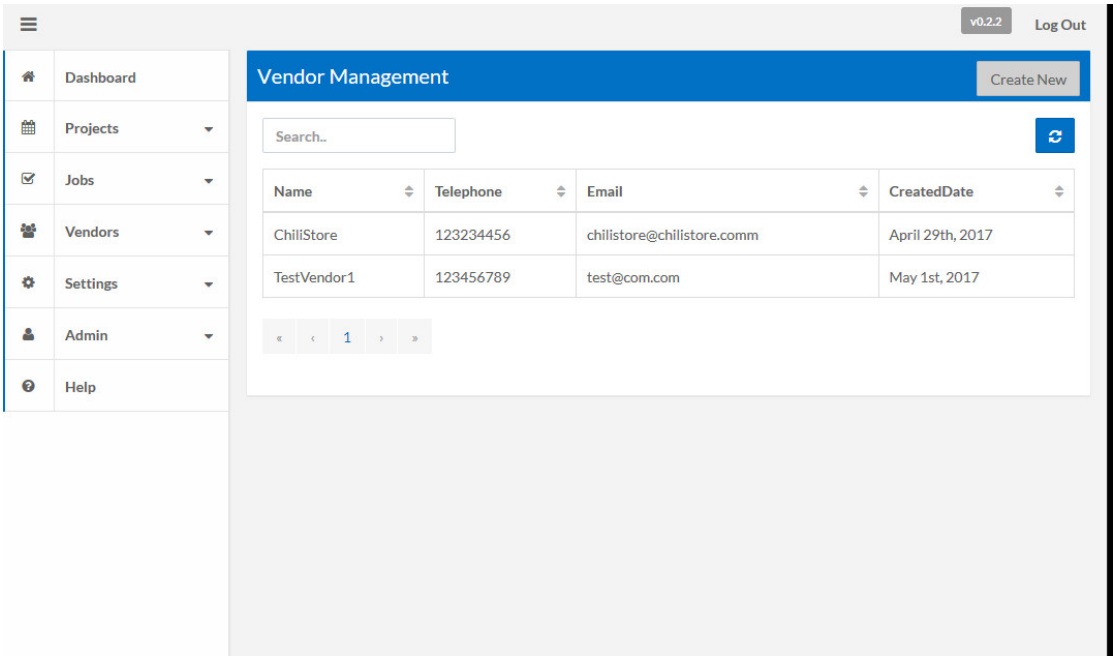


FIGURE A.1: Vendor Screen 1

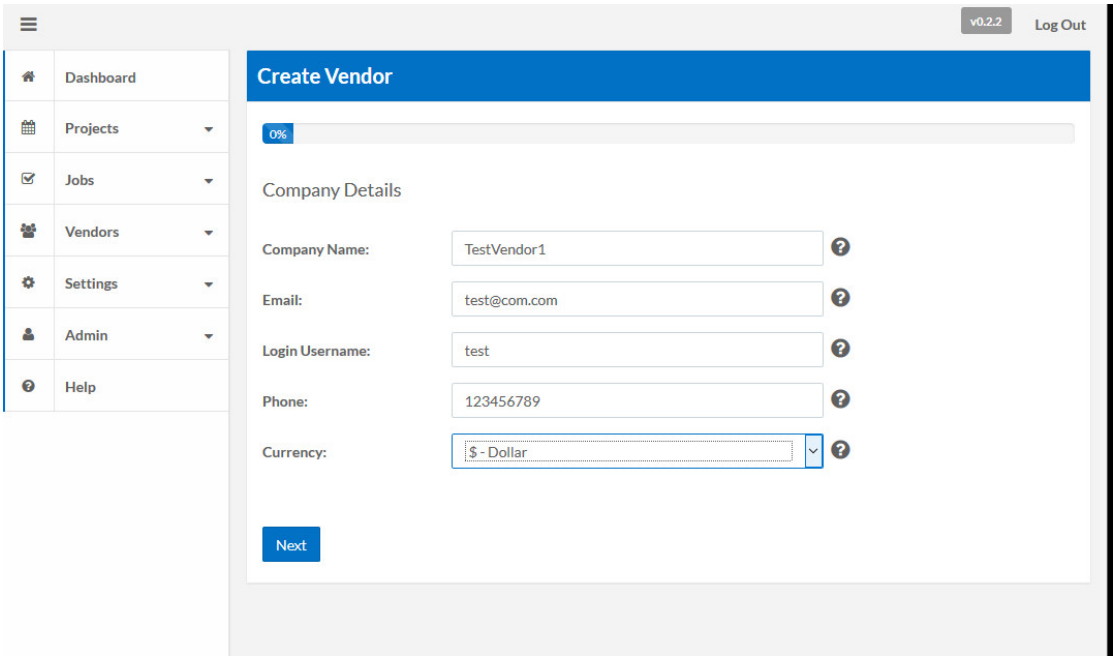


FIGURE A.2: Vendor Screen 2

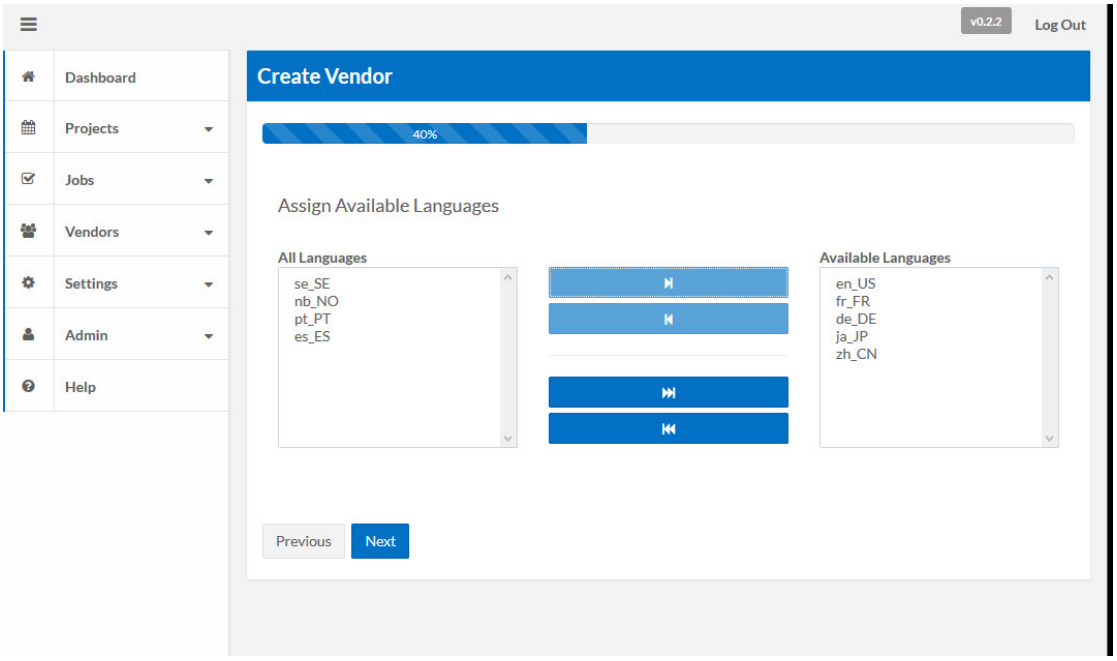


FIGURE A.3: Vendor Screen 3

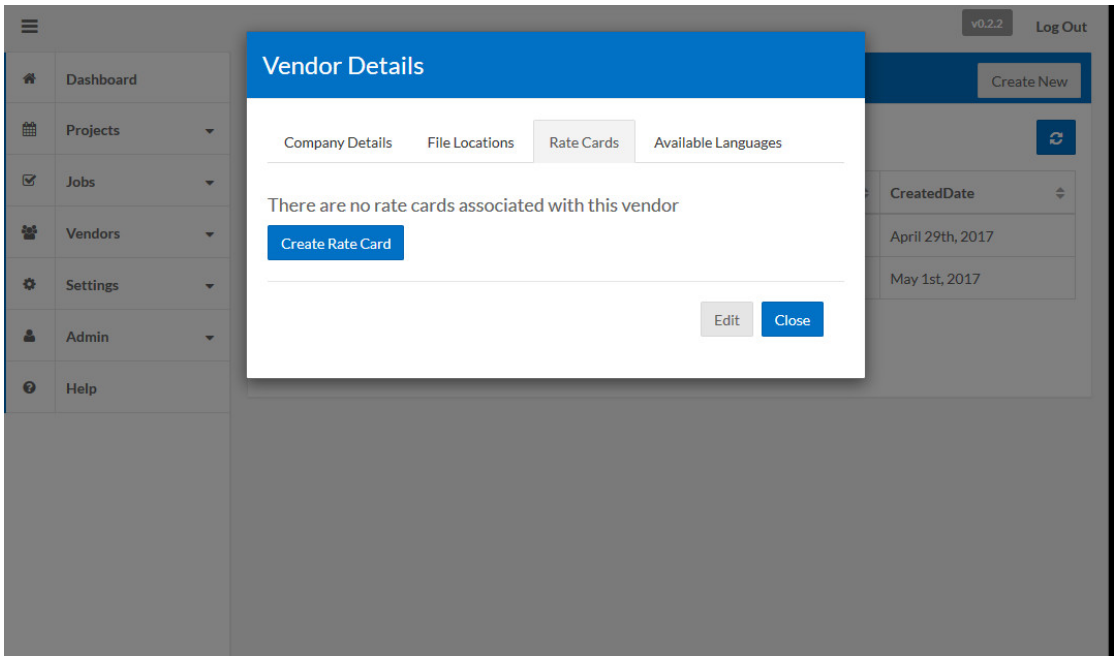


FIGURE A.4: Vendor Screen 4

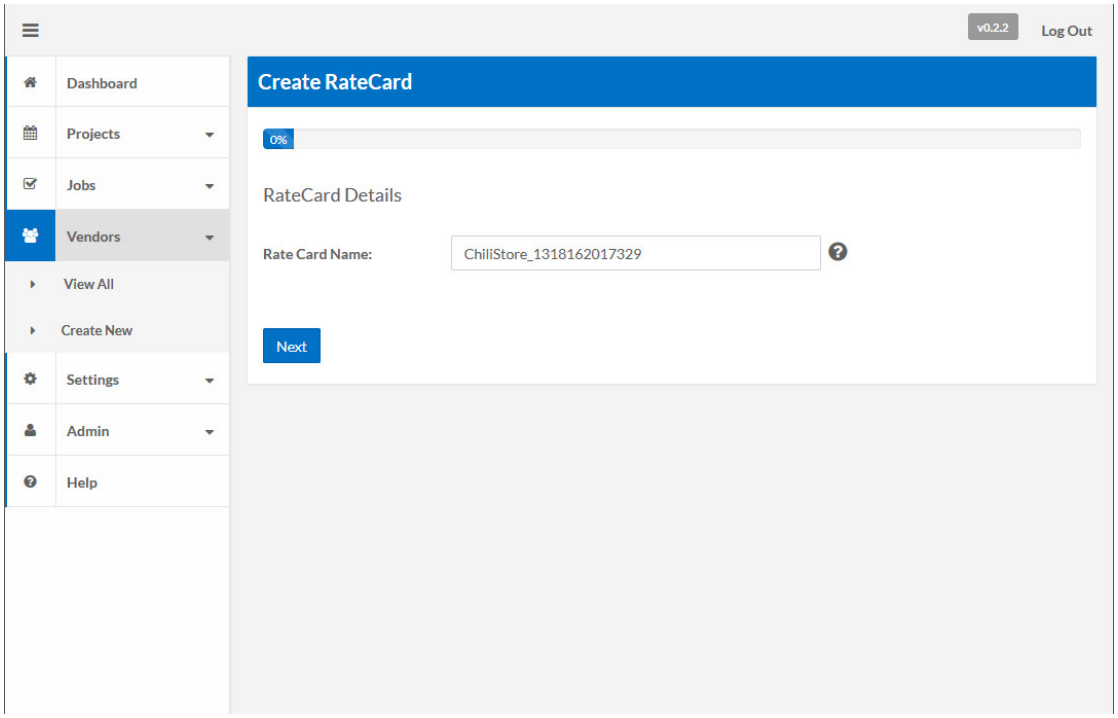


FIGURE A.5: Vendor Screen 5

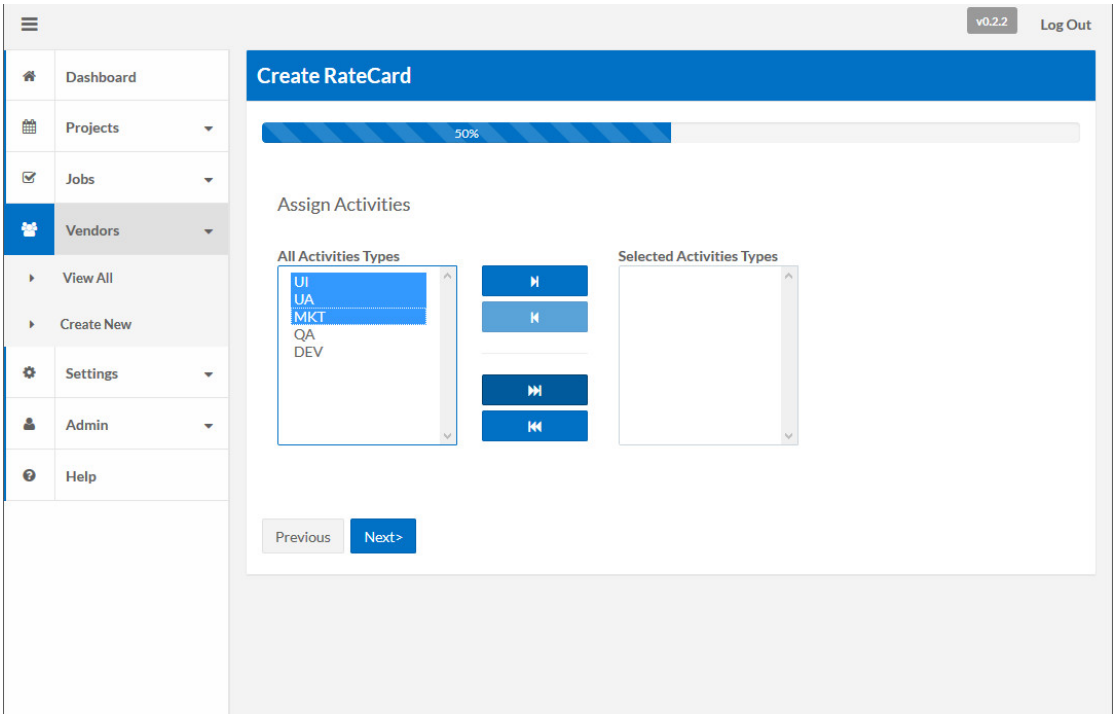


FIGURE A.6: Vendor Screen 6

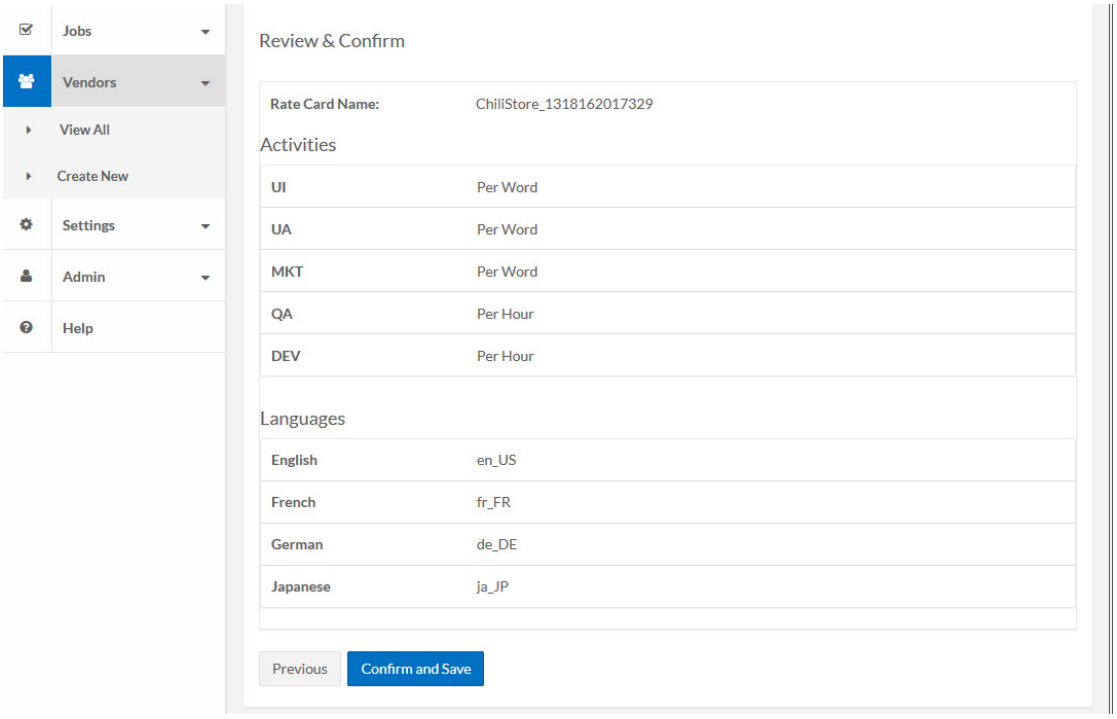


FIGURE A.7: Vendor Screen 7

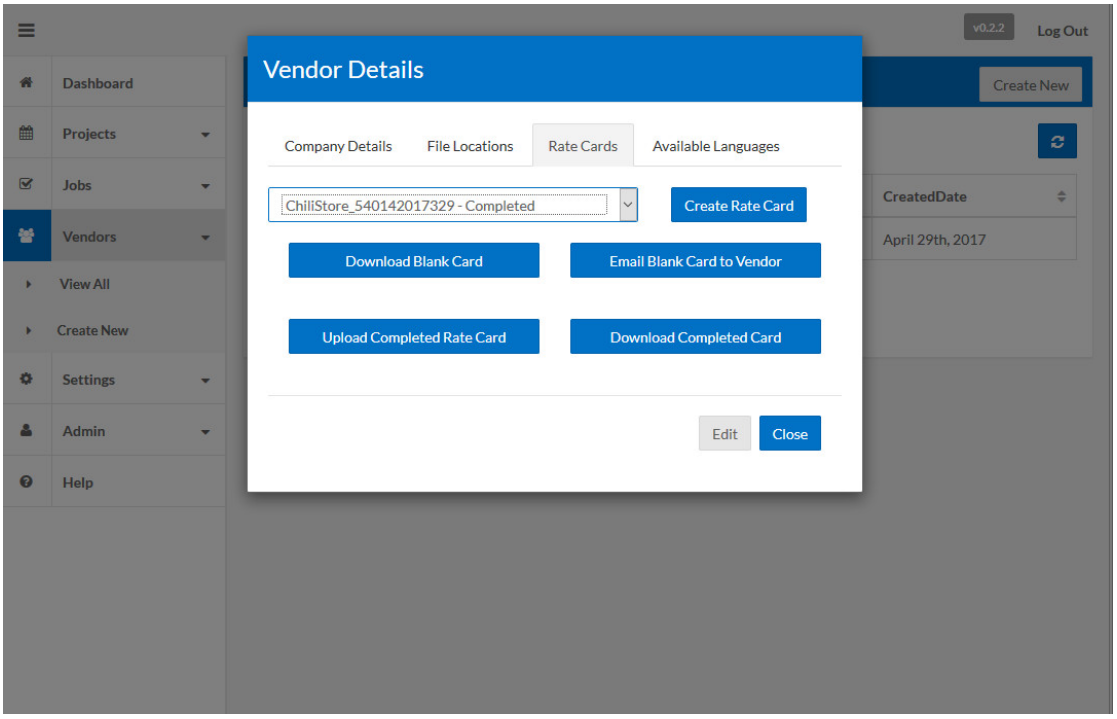


FIGURE A.8: Vendor Screen 8

Appendix B

Project Management Use Case Screenshots

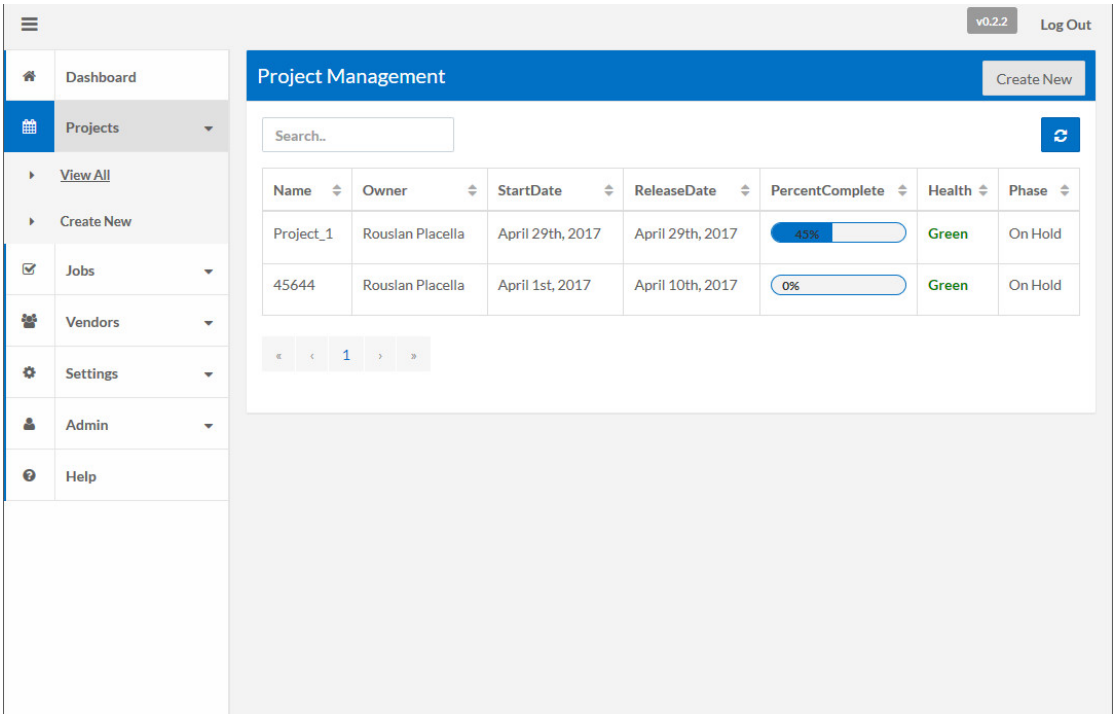


FIGURE B.1: Project Screen 1

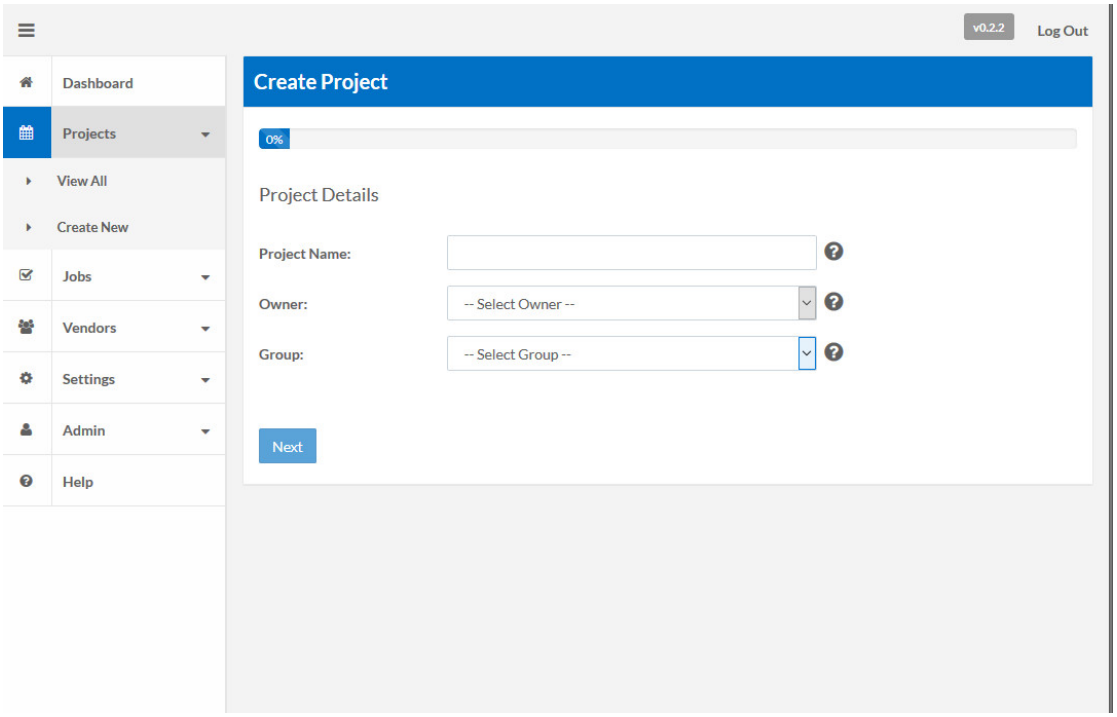


FIGURE B.2: Project Screen 2

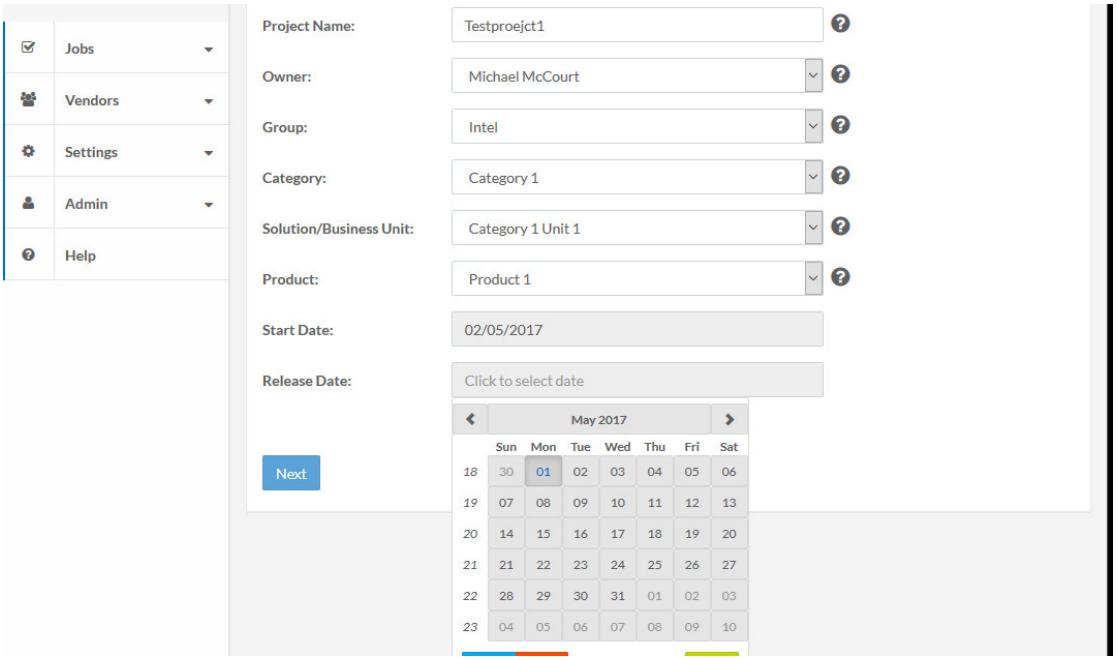


FIGURE B.3: Project Screen 3

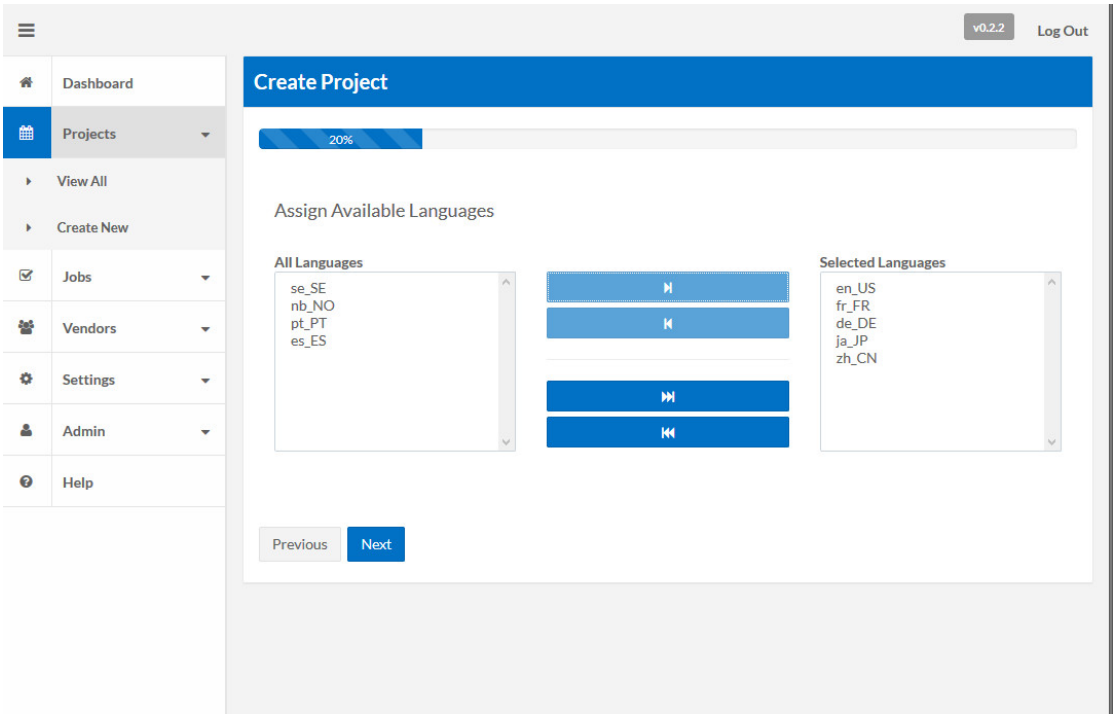


FIGURE B.4: Project Screen 4

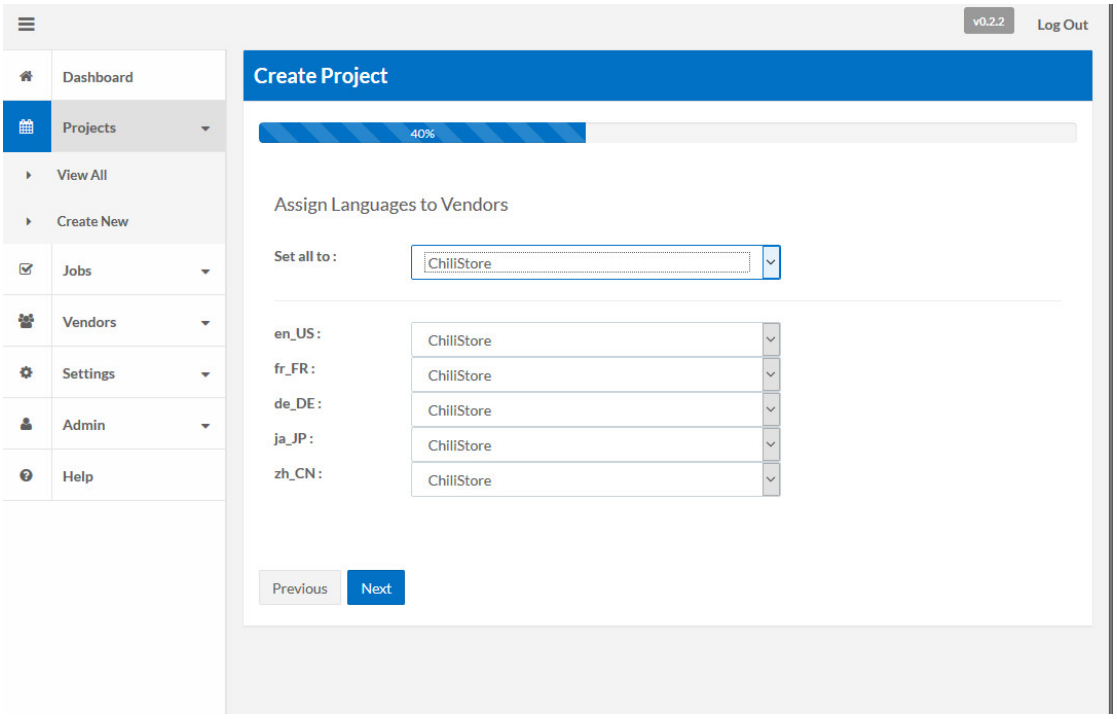


FIGURE B.5: Project Screen 5

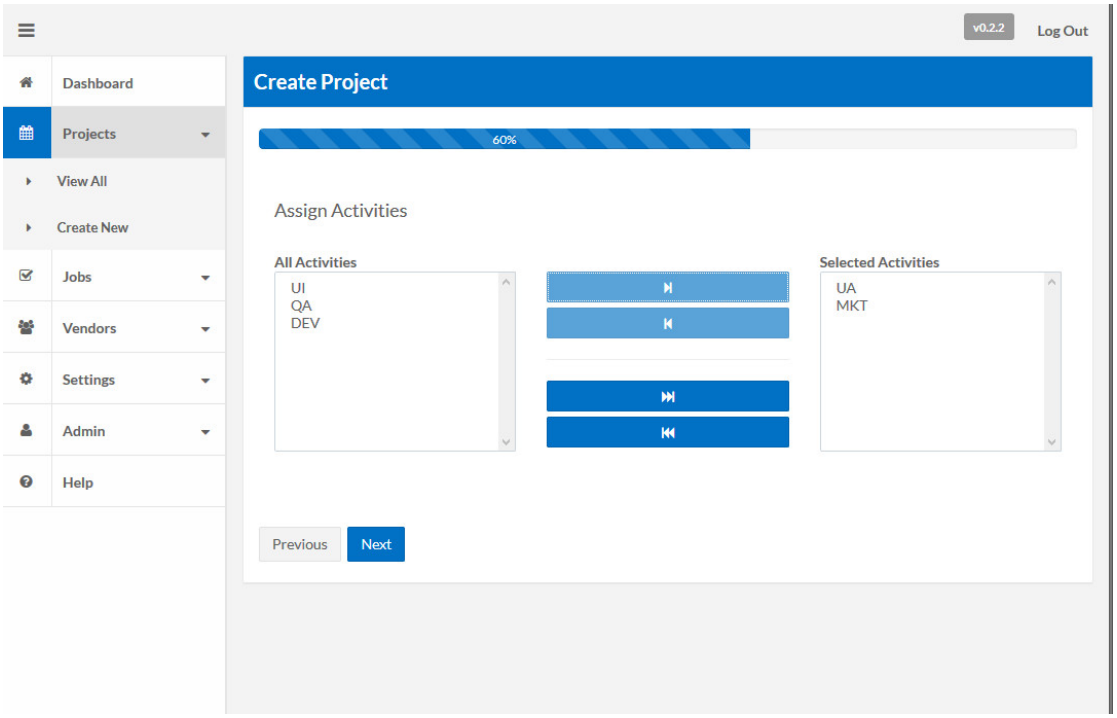


FIGURE B.6: Project Screen 6

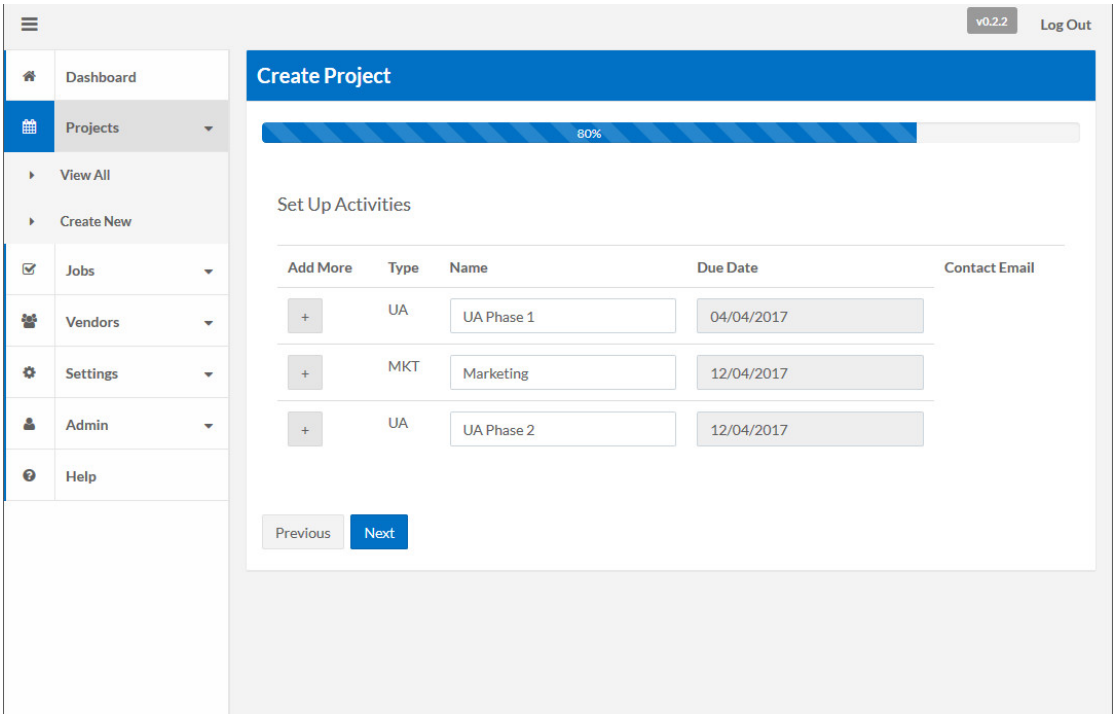


FIGURE B.7: Project Screen 7

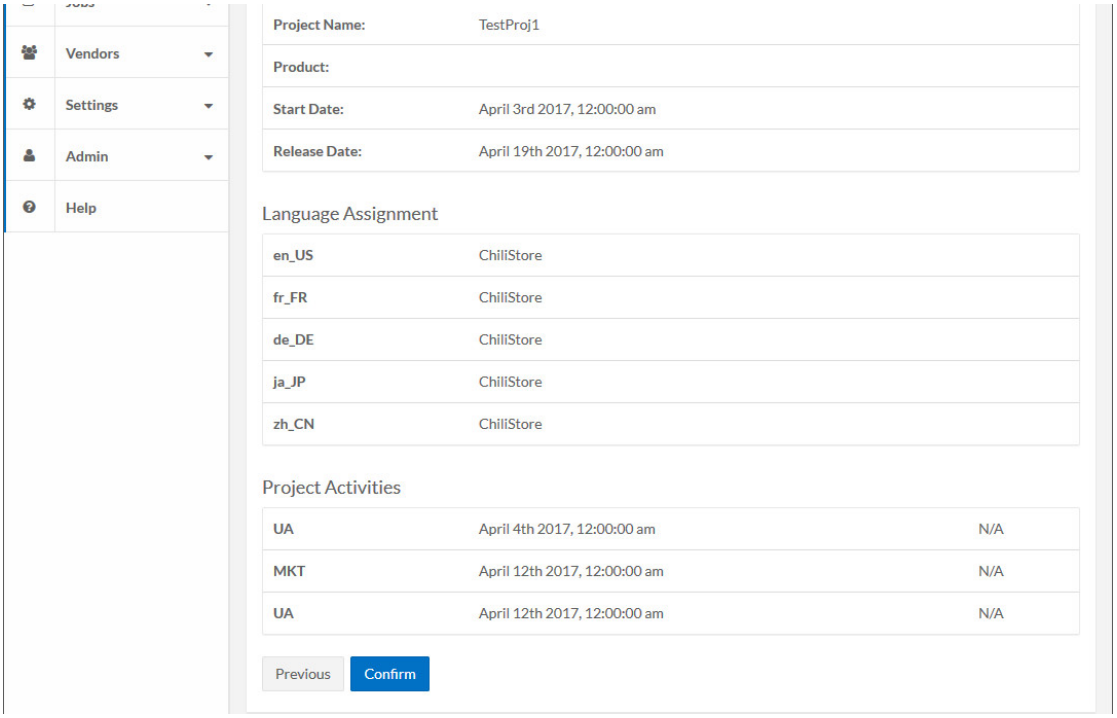


FIGURE B.8: Project Screen 8

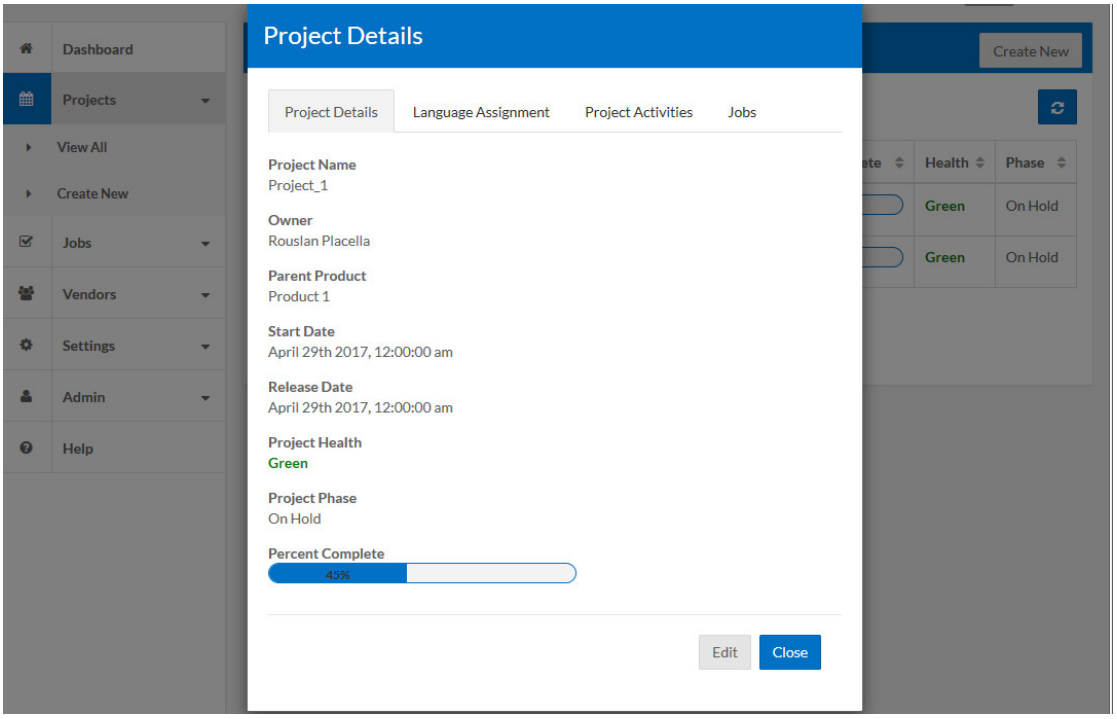


FIGURE B.9: Project Screen 9

Appendix C

Job Management Use Case Screenshots

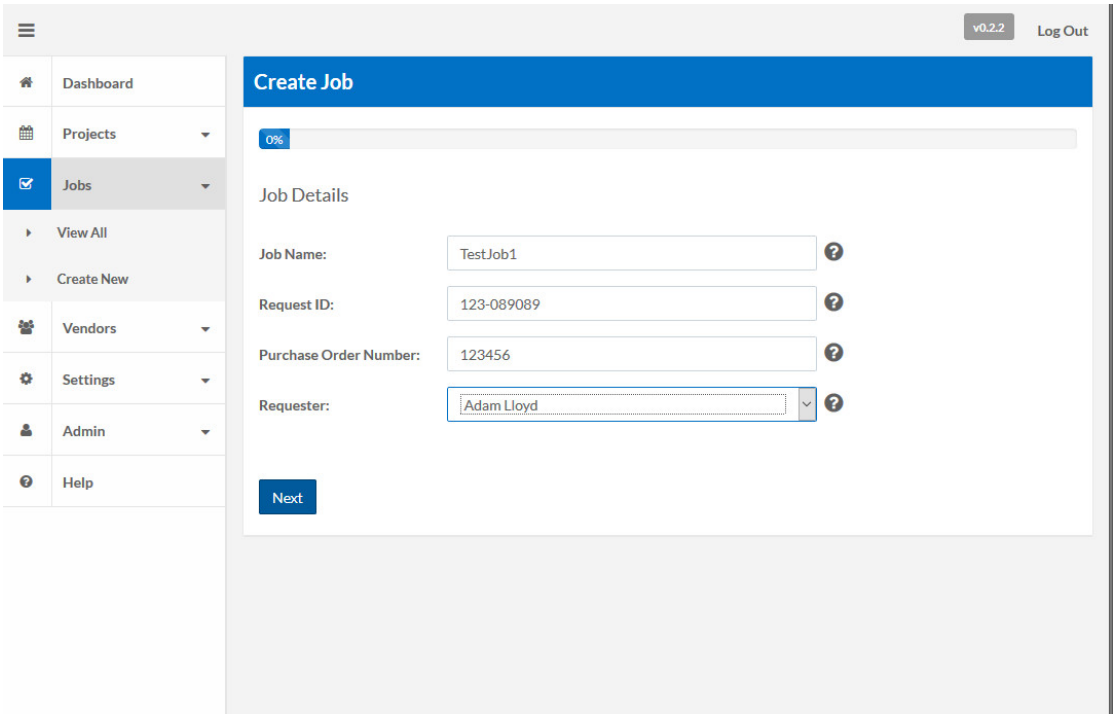


FIGURE C.1: Job Screen 1

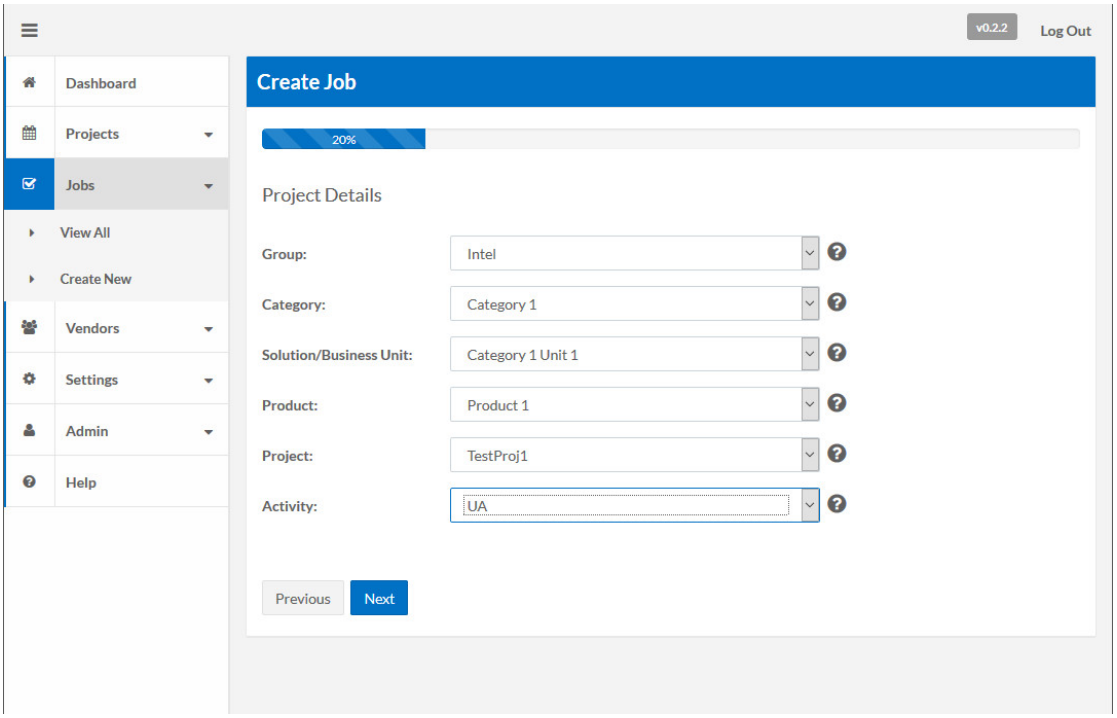


FIGURE C.2: Job Screen 2

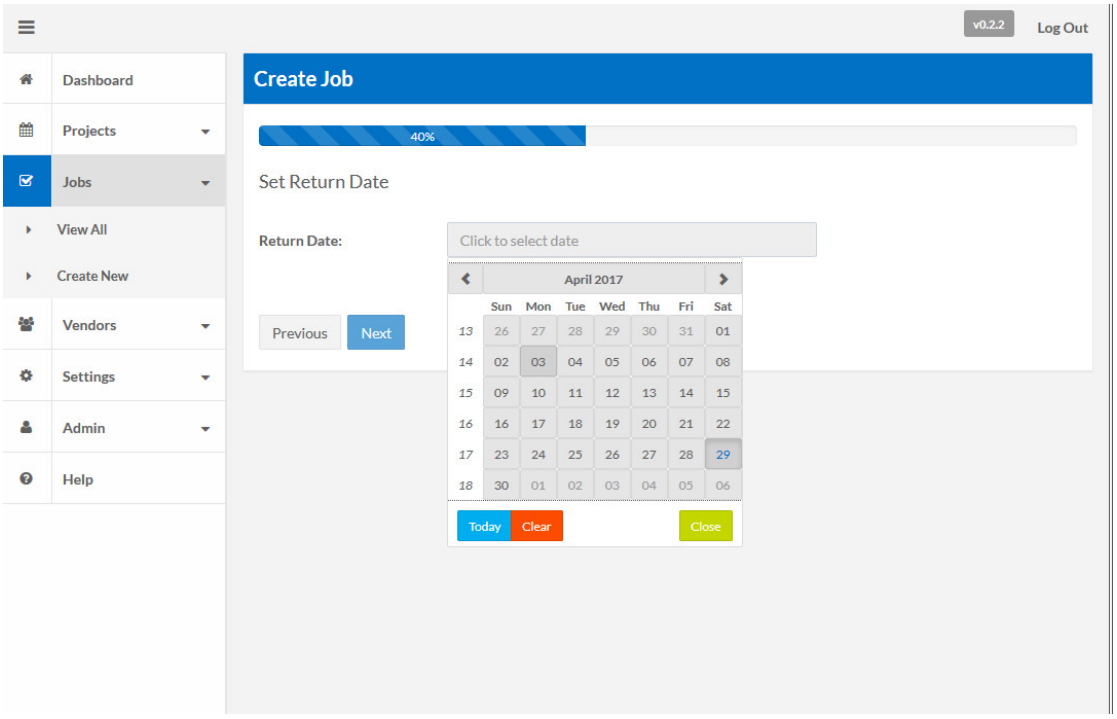


FIGURE C.3: Job Screen 3

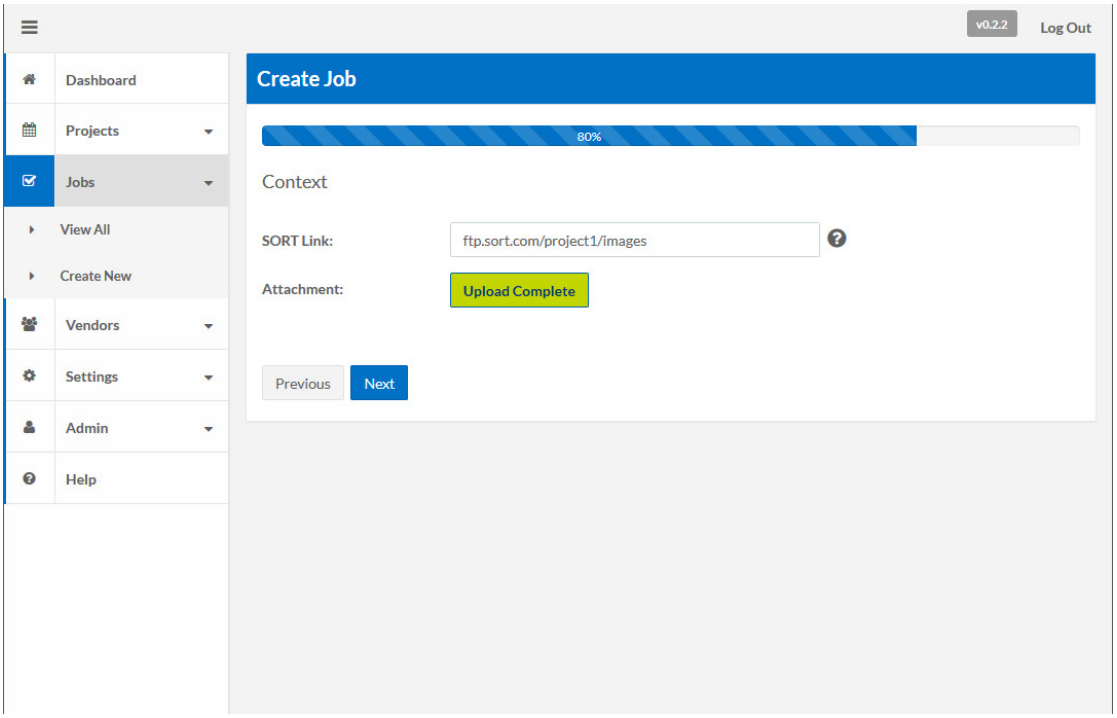


FIGURE C.4: Job Screen 4

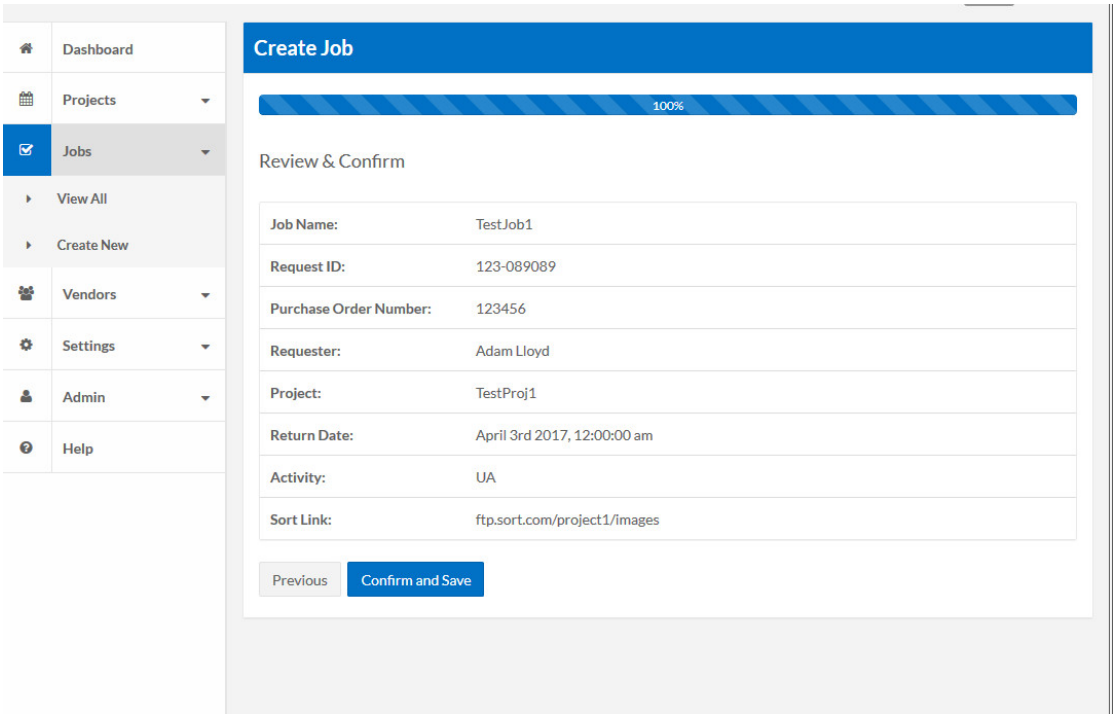


FIGURE C.5: Job Screen 5

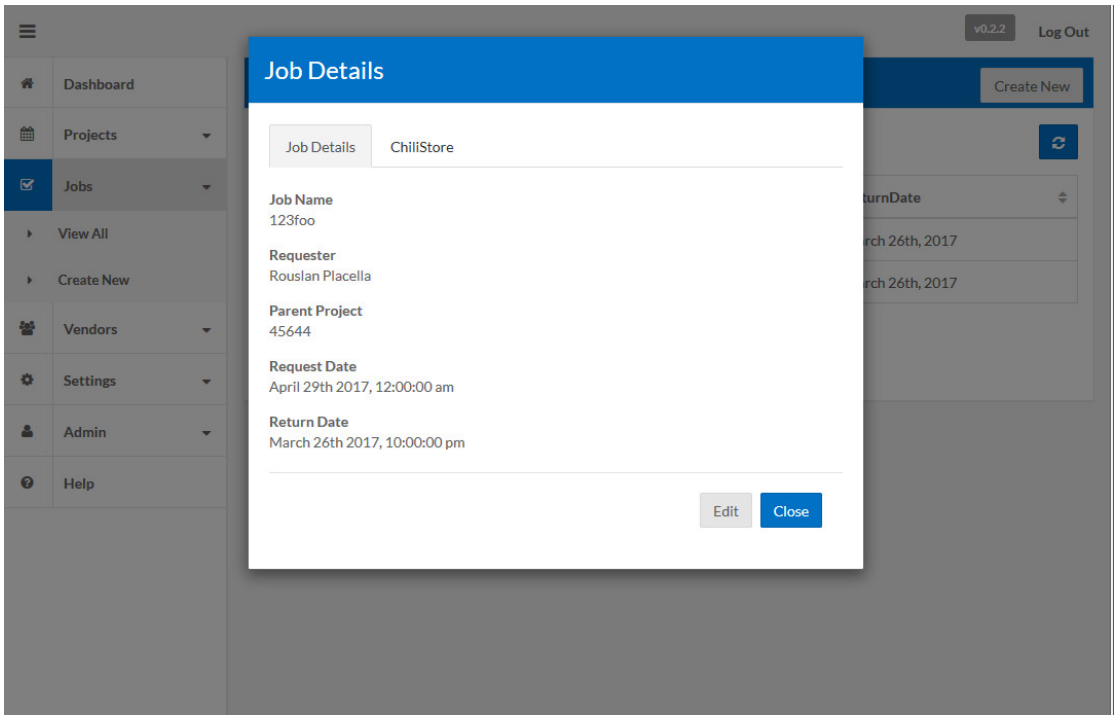


FIGURE C.6: Job Screen 6

Appendix D

Authentication Use Case Screenshots

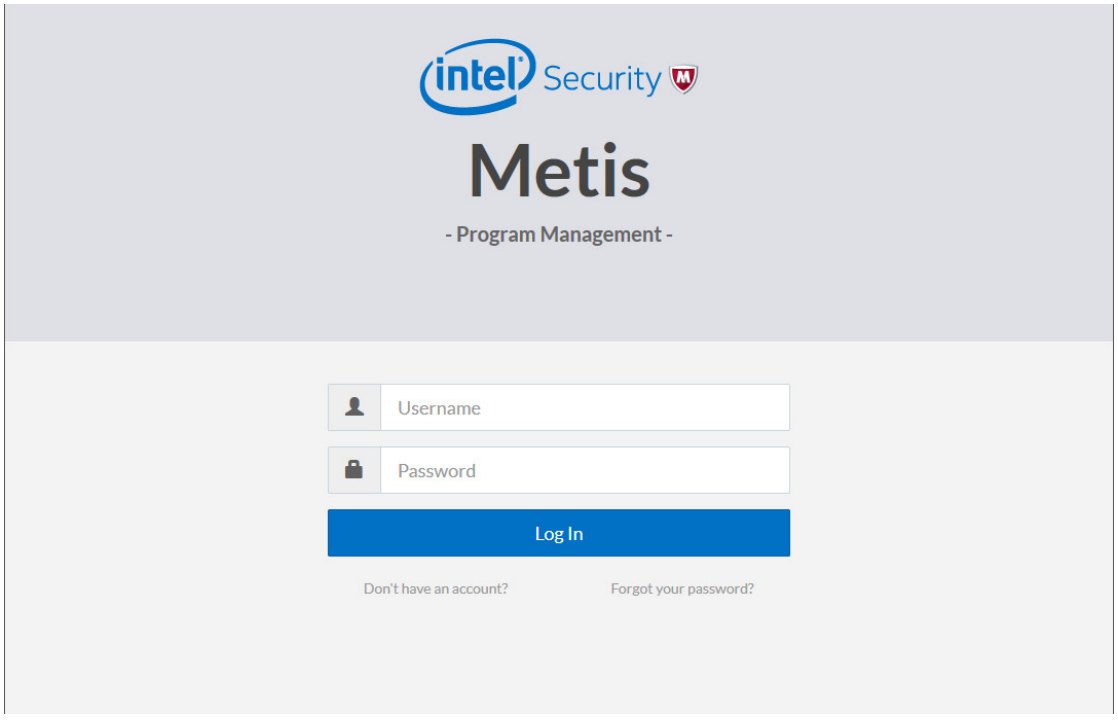


FIGURE D.1: Auth Screen 1

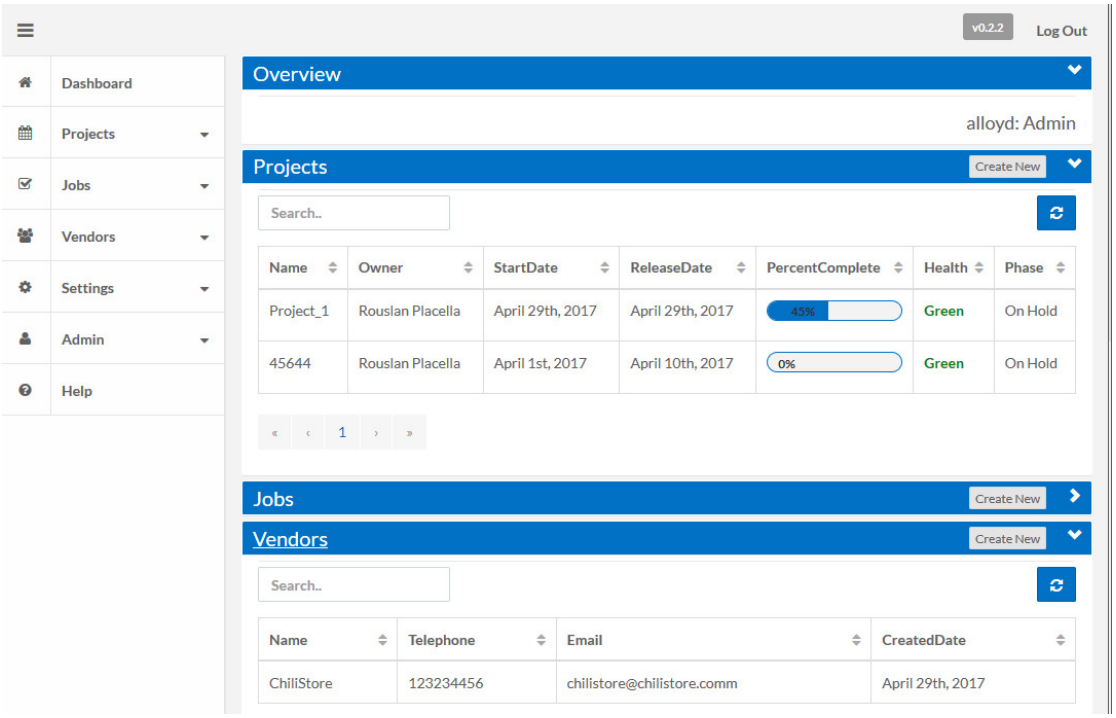


FIGURE D.2: Auth Screen 2

Appendix E

Supervisor Reports

E.1 Week 1

E.1.1 Minutes

- Discussed project research phase in general
- Discussed implementation plan (3 week sprints and weekly meetings)
- Displayed work completed so far
- Discussed technologies being used and design patterns

E.1.2 Items Completed

- .Net Code first Server side API using Entity Framework
- Basic API Routes for some entities for testing
- AngularJs client with basic functionality for testing

E.1.3 Items Planned

- Vendor Creation Wizard
- Vendor client side CRUD functions

E.1.4 Issues

- None

E.2 Week 2

E.2.1 Minutes

- We did not have a meeting this week due to scheduling difficulties

E.2.2 Items Completed

- Vendor Creation Wizard
- Vendor client side CRUD functions
- Streamlined client code by moving all common controller operations and services to a generic implementation for all simple entities.
- Worked with Intel team to prepare project for deployment onto live server for testing

E.2.3 Items Planned

- Rate Card generation wizard
- Exploration into Excel export of generated rate card
- Error handling on Vendor creation and rate card wizard

E.2.4 Issues

- Some confusion over rate card details caused minor delays.

E.3 Week 3

E.3.1 Minutes

- We discussed the status report and meeting minute formats
- We discussed points of interest to focus on for the project report
- Demonstrated new functionality

E.3.2 Items Completed

- Vendor Creation Wizard Updated per feedback from Intel
- Rate card generation wizard

E.3.3 Items Planned

- Export Excel spreadsheet after rate card generation
- Error handling for rate card and vendor wizard. This was pushed back due to previous issues/delays

E.3.4 Issues

- More confusion over rate cards causing minor delays.

E.4 Week 4

E.4.1 Minutes

- We agreed that, to simplify the conceptual models involved in each component, I would create domain models for each of the major parts of the system that I have completed so far.
- Demonstrated new functionality

E.4.2 Items Completed

- Rate card generation wizard updated per feedback form Intel
- Rate card export function to generate XLSX file
- Validation in wizard forms

E.4.3 Items Planned

- Import completed Excel spreadsheet and populate database with values
- Begin project management component. Basic views/controllers/services/routes

E.4.4 Issues

- As I begin to delve deeper into complex entity updates and creations I am finding problems with entity duplication inside child collections. I am having to change the way updates and creates are done.

E.5 Week 5

E.5.1 Minutes

- We discussed the domain model I had provided per my supervisors request at our last meeting.
- We found them to need refinement to properly explain the relationships implemented in my code first database.
- I was tasked with finding out how exactly entity stores "many to many" relationships without foreign key mapping.
- Unfortunately, I was unable to demo my progress this week as a file was missing from GIT due to some changes that were made to make the project compatible with Intels continuous integration system.

E.5.2 Items Completed

- Completed Rate card XLSX file uploaded in web browser.
- Uploaded date being stored in the rate card model and committed to the database.
- Began Project management component by creating the first steps of the creation wizard.

E.5.3 Items Planned

- Correct domain models.
- Complete Project creation wizard

E.5.4 Issues

- The aforementioned continuous integration changes causes problems demoing the software but all is on track for this sprint.

E.6 Week 7

E.6.1 Minutes

- As this was coming up to the end of sprint for me, there was not much to discuss as the majority of my work was completing the issues I had been working on for the last 2 weeks.
- I showed the initial stages of the project creation wizard and discussed the issues I was having with it and scope creep in the project in general.

E.6.2 Items Completed

- Completed initial stages of the project creation wizard (Awaiting input from Intel to complete final stages).
- Completed the back-end functionality for storing projects.
- Implemented peer review feedback from Intel supervisor.

E.6.3 Items Planned

- Complete Project creation wizard final stages and store in the database.
- Plan the next sprint for creating the job management component.

E.6.4 Issues

- Minor scope creep and the company changing their source control to a new type (which I do not yet have access to) caused some delays but nothing major.

E.7 Week 8

E.7.1 Minutes

- We looked at the marking scheme for the project module.
- We discussed the best options for presenting this project in May and for my report.
- We discussed the poster, its format and what should be highlighted to give the best impression possible.
- Discussed data integrity issues that may arise in the project.

E.7.2 Items Completed

- Completed initial stages of the project creation wizard (Awaiting input from Intel to complete final stages).
- Completed the back-end functionality for storing projects.
- Implemented peer review feedback from Intel supervisor.

E.7.3 Items Planned

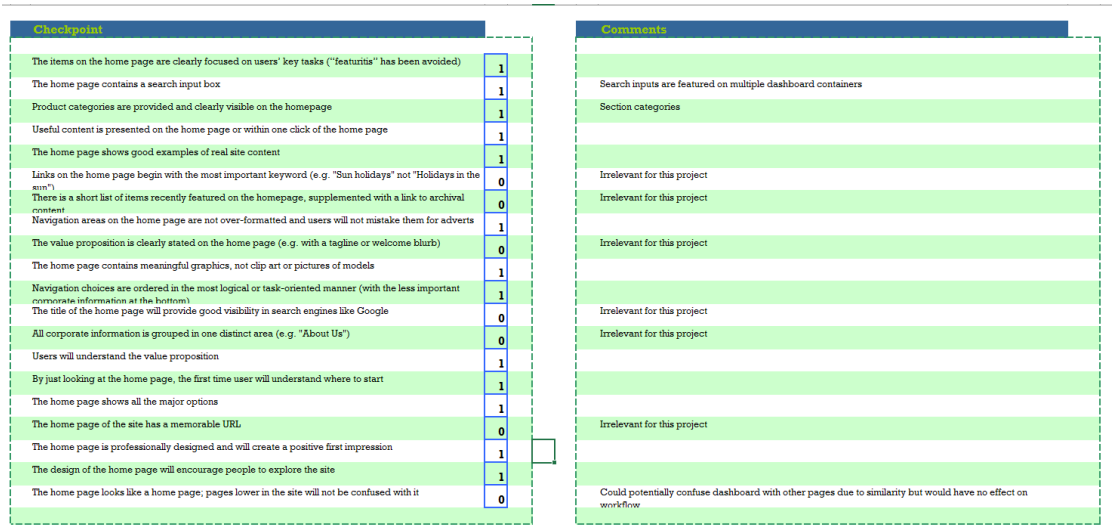
- Completed Project creation wizard
- Planned the final sprint as there will only be time for 1 more until project is due.

E.7.4 Issues

- Some code had to be refactored to accommodate the project component. Job type was merged with activity.

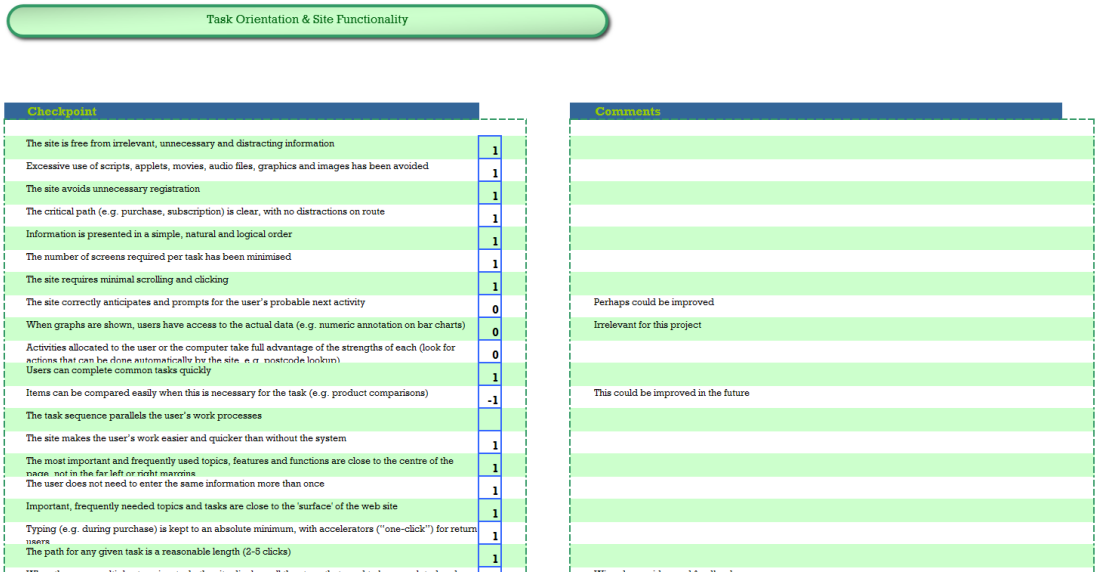
Appendix F

Usability Spreadsheet



Checkpoint		Comments
The items on the home page are clearly focused on users' key tasks ("featuritis" has been avoided)	1	
The home page contains a search input box	1	Search inputs are featured on multiple dashboard containers
Product categories are provided and clearly visible on the homepage	1	Section categories
Useful content is presented on the home page or within one click of the home page	1	
The home page shows good examples of real site content	1	
Links on the home page begin with the most important keyword (e.g. "Sun holidays" not "Holidays in the sun")	0	Irrelevant for this project
There is a short list of items recently featured on the homepage, supplemented with a link to archival content	0	Irrelevant for this project
Navigation areas on the home page are not over-formatted and users will not mistake them for adverts	1	
The value proposition is clearly stated on the home page (e.g. with a tagline or welcome blurb)	0	Irrelevant for this project
The home page contains meaningful graphics, not clip art or pictures of models	1	
Navigation choices are ordered in the most logical or task-oriented manner (with the less important corporate information at the bottom)	1	
The title of the home page will provide good visibility in search engines like Google	0	Irrelevant for this project
All corporate information is grouped in one distinct area (e.g. "About Us")	0	Irrelevant for this project
Users will understand the value proposition	1	
By just looking at the home page, the first time user will understand where to start	1	
The home page shows all the major options	1	
The home page of the site has a memorable URL	0	Irrelevant for this project
The home page is professionally designed and will create a positive first impression	1	
The design of the home page will encourage people to explore the site	1	
The home page looks like a home page; pages lower in the site will not be confused with it	0	Could potentially confuse dashboard with other pages due to similarity but would have no effect on workflow

FIGURE F.1: Sheet 1



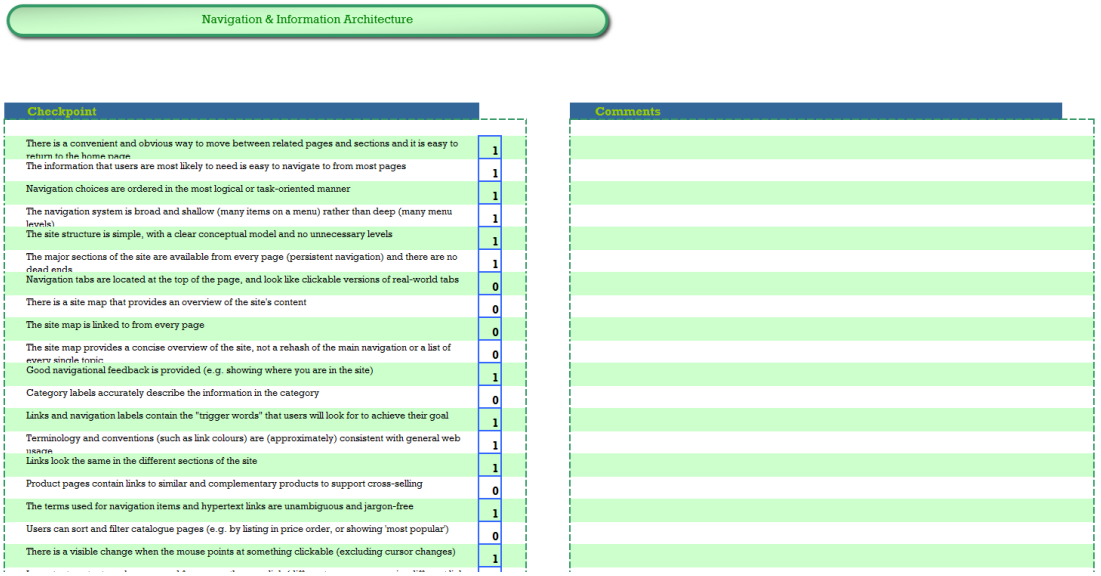
Task Orientation & Site Functionality

Checkpoint		Comments
The site is free from irrelevant, unnecessary and distracting information	1	
Excessive use of scripts, applets, movies, audio files, graphics and images has been avoided	1	
The site avoids unnecessary registration	1	
The critical path (e.g. purchase, subscription) is clear, with no distractions on route	1	
Information is presented in a simple, natural and logical order	1	
The number of screens required per task has been minimised	1	
The site requires minimal scrolling and clicking	1	
The site correctly anticipates and prompts for the user's probable next activity	0	Perhaps could be improved
When graphs are shown, users have access to the actual data (e.g. numeric annotation on bar charts)	0	Irrelevant for this project
Activities allocated to the user or the computer take full advantage of the strengths of each (look for actions that can be done automatically by the site, e.g. postcode lookup)	0	
Users can complete common tasks quickly	1	
Items can be compared easily when this is necessary for the task (e.g. product comparisons)	-1	This could be improved in the future
The task sequence parallels the user's work processes		
The site makes the user's work easier and quicker than without the system	1	
The most important and frequently used topics, features and functions are close to the centre of the page, not in the far left or right margins	1	
The user does not need to enter the same information more than once	1	
Important, frequently needed topics and tasks are close to the 'surface' of the web site	1	
Typing (e.g. during purchase) is kept to an absolute minimum, with accelerators ("one-click") for return users	1	
The path for any given task is a reasonable length (2-5 clicks)	1	

FIGURE F.2: Sheet 2

The path for any given task is a reasonable length (2-5 clicks)	1	
When there are multiple steps in a task, the site displays all the steps that need to be completed and provides feedback on the user's current position in the workflow	1	Wizards provide good feedback
Price is always clearly displayed next to any product	0	Irrelevant for this project
The site's privacy policy is easy to find, especially on pages that ask for personal information, and the policy is simple and clear	0	Irrelevant for this project
Users of the site do not need to remember information from place to place	1	
The use of metaphors is easily understandable by the typical user	1	
Data formats follow appropriate cultural conventions (e.g. miles for UK)	1	
Details of the software's internal workings are not exposed to the user	1	
The site caters for users with little prior experience of the web	1	
The site makes it easy for users to explore the site and try out different options before committing themselves	0	Irrelevant for this project
A typical first-time visitor can do the most common tasks without assistance	1	
When they return to the site, users will remember how to carry out the key tasks	1	
The functionality of novel device controls is obvious	0	
On the basket page, there is a highly visible 'Proceed to checkout' button at the top and bottom of the page		
Important calls to action, like 'Add to basket', are highly visible	1	Create New buttons featured proinently where applicable
Action buttons (such as "Submit") are always invoked by the user, not automatically invoked by the system when the last field is completed	1	
Command and action items are presented as buttons (not, for example, as hypertext links)	1	
If the user is half-way through a transaction and quits, the user can later return to the site and continue from where he left off	-1	This functionality will be completed in the future
When a page presents a lot of information, the user can sort and filter the information	1	
If there is an image on a button or icon, it is relevant to the task	1	Glpyhicons are used on buttons where applicable
The site prompts the user before automatically logging off the user, and the time out is appropriate	0	Not yet implemented
Unwanted features (e.g. Flash animations) can be stopped or skipped	0	Irrelevant for this project
The site is robust and all the key features work (i.e. there are no javascript exceptions, CGI errors or broken links)	1	
The site supports novice and expert users by providing different levels of explanation (e.g. in help and error messages)	1	
The site allows users to rename objects and actions in the interface (e.g. naming delivery addresses or accounts)	0	
The site allows the user to customise operational time parameters (e.g. time until automatic logout)	1	

FIGURE F.3: Sheet 3



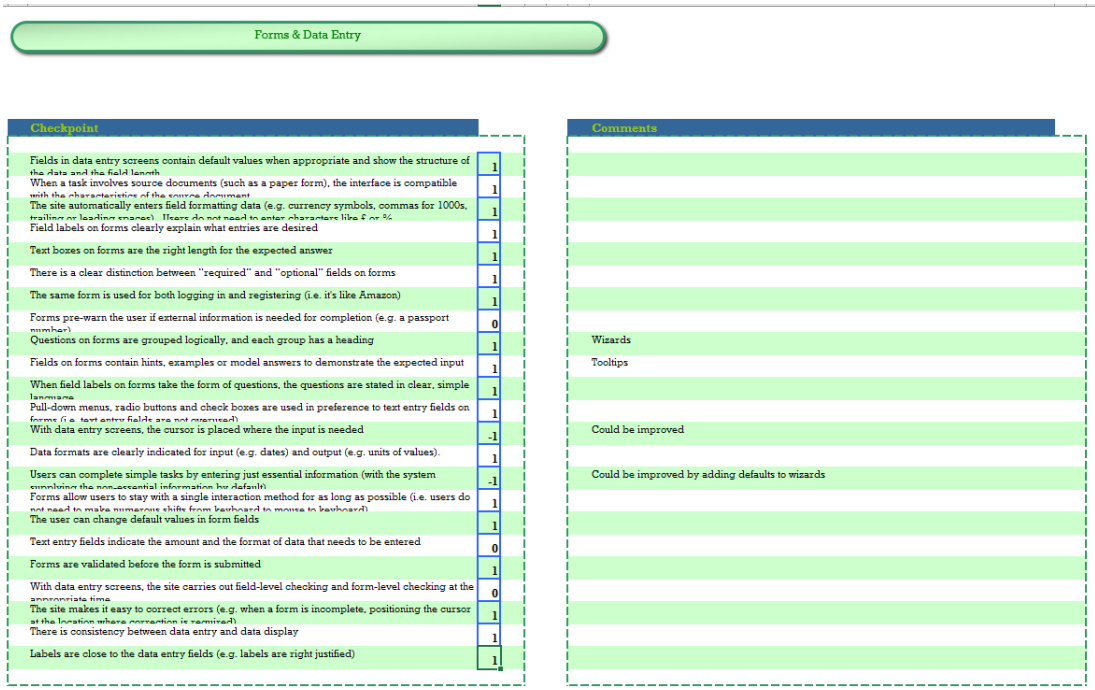
Navigation & Information Architecture

Checkpoint		Comments
There is a convenient and obvious way to move between related pages and sections and it is easy to return to the home page	1	
The information that users are most likely to need is easy to navigate to from most pages	1	
Navigation choices are ordered in the most logical or task-oriented manner	1	
The navigation system is broad and shallow (many items on a menu) rather than deep (many menu levels)	1	
The site structure is simple, with a clear conceptual model and no unnecessary levels	1	
The major sections of the site are available from every page (persistent navigation) and there are no dead ends	1	
Navigation tabs are located at the top of the page, and look like clickable versions of real-world tabs	0	
There is a site map that provides an overview of the site's content	0	
The site map is linked to from every page	0	
The site map provides a concise overview of the site, not a rehash of the main navigation or a list of every single topic	0	
Good navigational feedback is provided (e.g. showing where you are in the site)	1	
Category labels accurately describe the information in the category	0	
Links and navigation labels contain the "trigger words" that users will look for to achieve their goal	1	
Terminology and conventions (such as link colours) are (approximately) consistent with general web usage	1	
Links look the same in the different sections of the site	1	
Product pages contain links to similar and complementary products to support cross-selling	0	
The terms used for navigation items and hypertext links are unambiguous and jargon-free	1	
Users can sort and filter catalogue pages (e.g. by listing in price order, or showing 'most popular')	0	
There is a visible change when the mouse points at something clickable (excluding cursor changes)	1	

FIGURE F.4: Sheet 4

levels)	1	
The site structure is simple, with a clear conceptual model and no unnecessary levels	1	
The major sections of the site are available from every page (persistent navigation) and there are no dead ends	1	
Navigation tabs are located at the top of the page, and look like clickable versions of real-world tabs	0	
There is a site map that provides an overview of the site's content	0	
The site map is linked to from every page	0	
The site map provides a concise overview of the site, not a rehash of the main navigation or a list of every single topic	0	
Good navigational feedback is provided (e.g. showing where you are in the site)	1	
Category labels accurately describe the information in the category	0	
Links and navigation labels contain the "trigger words" that users will look for to achieve their goal	1	
Terminology and conventions (such as link colours) are (approximately) consistent with general web usage	1	
Links look the same in the different sections of the site	1	
Product pages contain links to similar and complementary products to support cross-selling	0	
The terms used for navigation items and hypertext links are unambiguous and jargon-free	1	
Users can sort and filter catalogue pages (e.g. by listing in price order, or showing 'most popular')	0	
There is a visible change when the mouse points at something clickable (excluding cursor changes)	1	
Important content can be accessed from more than one link (different users may require different link labels)	0	
Navigation-only pages (such as the home page) can be viewed without scrolling	1	
Hypertext links that invoke actions (e.g downloads, new windows) are clearly distinguished from hypertext links that load another page	1	
The site allows the user to control the pace and sequence of the interaction	0	
There are clearly marked exits on every page allowing the user to bale out of the current task without having to go through an extended dialog	-1	could be better or implement drafts
The site does not disable the browser's "Back" button and the "Back" button appears on the browser toolbar on every page	1	
Clicking the back button always takes the user back to the page the user came from	0	
A link to both the basket and checkout is clearly visible on every page	0	
If the site spawns new windows, these will not confuse the user (e.g. they are dialog-box sized and can be easily closed)	1	
Menu instructions, prompts and messages appear on the same place on each screen	1	

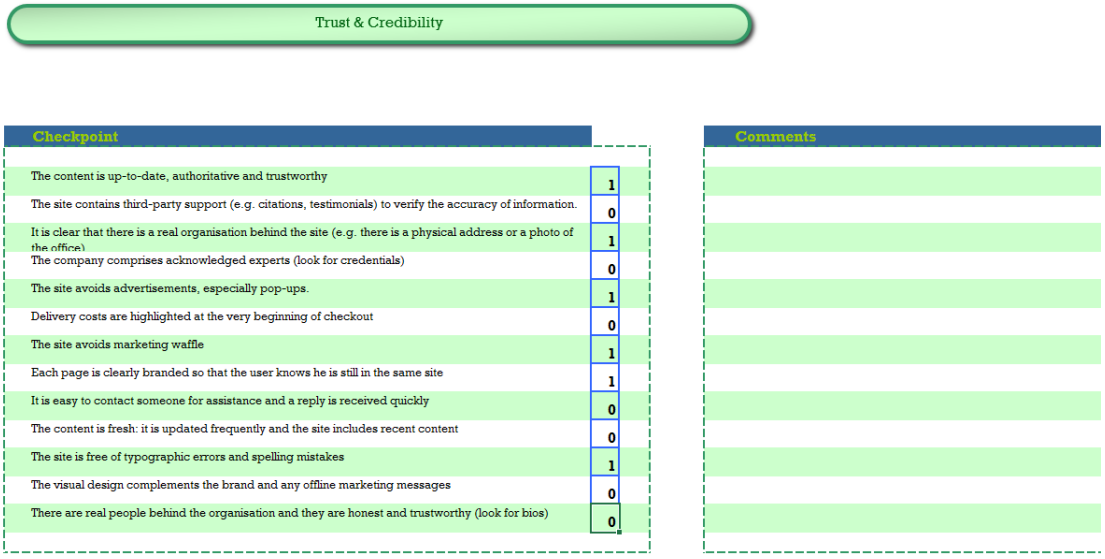
FIGURE F.5: Sheet 5



Forms & Data Entry

Checkpoint		Comments
Fields in data entry screens contain default values when appropriate and show the structure of the data and the field length	1	
When a task involves source documents (such as a paper form), the interface is compatible with the characteristics of the source document	1	
The site automatically enters field formatting data (e.g. currency symbols, commas for 1000s, trailing or leading spaces). Users do not need to enter characters like $ or %	1	
Field labels on forms clearly explain what entries are desired	1	
Text boxes on forms are the right length for the expected answer	1	
There is a clear distinction between "required" and "optional" fields on forms	1	
The same form is used for both logging in and registering (i.e. it's like Amazon)	1	
Forms pre-warn the user if external information is needed for completion (e.g. a passport number)	0	
Questions on forms are grouped logically, and each group has a heading	1	Wizards
Fields on forms contain hints, examples or model answers to demonstrate the expected input	1	Tooltips
When field labels on forms take the form of questions, the questions are stated in clear, simple language	1	
Pull-down menus, radio buttons and check boxes are used in preference to text entry fields on forms (i.e. text entry fields are not overused)	1	
With data entry screens, the cursor is placed where the input is needed	-1	Could be improved
Data formats are clearly indicated for input (e.g. dates) and output (e.g. units of values).	1	
Users can complete simple tasks by entering just essential information (with the system supplying the non-essential information by default)	-1	Could be improved by adding defaults to wizards
Forms allow users to stay with a single interaction method for as long as possible (i.e. users do not need to make numerous shifts from keyboard to mouse to keyboard)	1	
The user can change default values in form fields	1	
Text entry fields indicate the amount and the format of data that needs to be entered	0	
Forms are validated before the form is submitted	1	
With data entry screens, the site carries out field-level checking and form-level checking at the appropriate time	0	
The site makes it easy to correct errors (e.g. when a form is incomplete, positioning the cursor at the location where correction is required)	1	
There is consistency between data entry and data display	1	
Labels are close to the data entry fields (e.g. labels are right justified)	1	

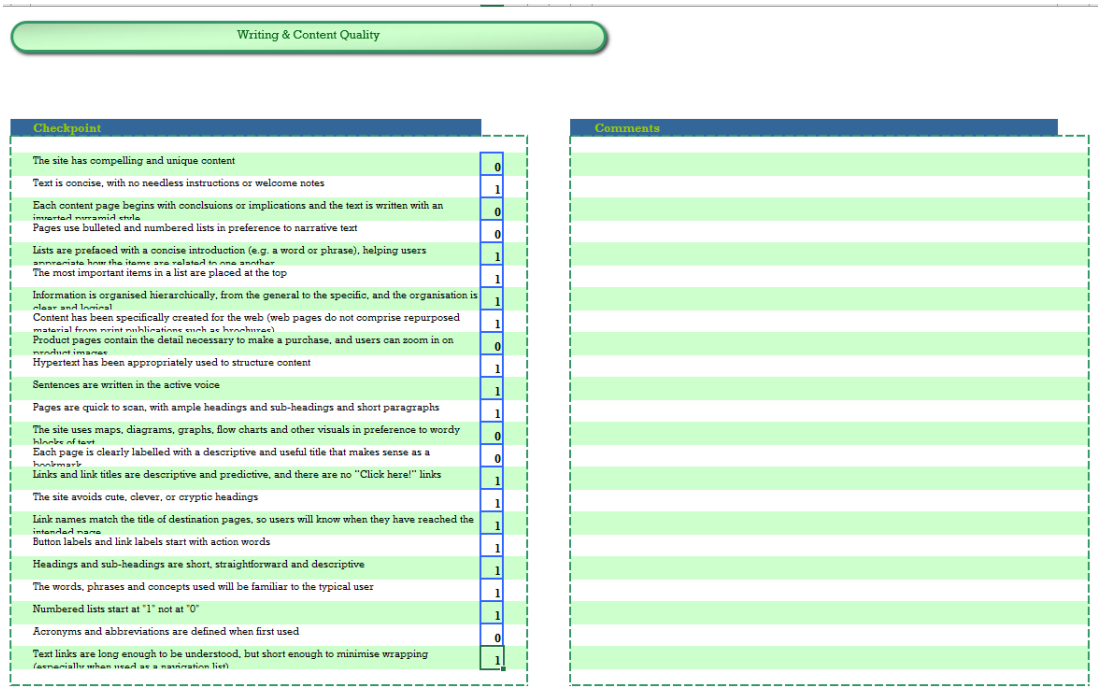
FIGURE F.6: Sheet 6



Trust & Credibility

Checkpoint		Comments
The content is up-to-date, authoritative and trustworthy	1	
The site contains third-party support (e.g. citations, testimonials) to verify the accuracy of information.	0	
It is clear that there is a real organisation behind the site (e.g. there is a physical address or a photo of the office)	1	
The company comprises acknowledged experts (look for credentials)	0	
The site avoids advertisements, especially pop-ups.	1	
Delivery costs are highlighted at the very beginning of checkout	0	
The site avoids marketing waffle	1	
Each page is clearly branded so that the user knows he is still in the same site	1	
It is easy to contact someone for assistance and a reply is received quickly	0	
The content is fresh: it is updated frequently and the site includes recent content	0	
The site is free of typographic errors and spelling mistakes	1	
The visual design complements the brand and any offline marketing messages	0	
There are real people behind the organisation and they are honest and trustworthy (look for bios)	0	

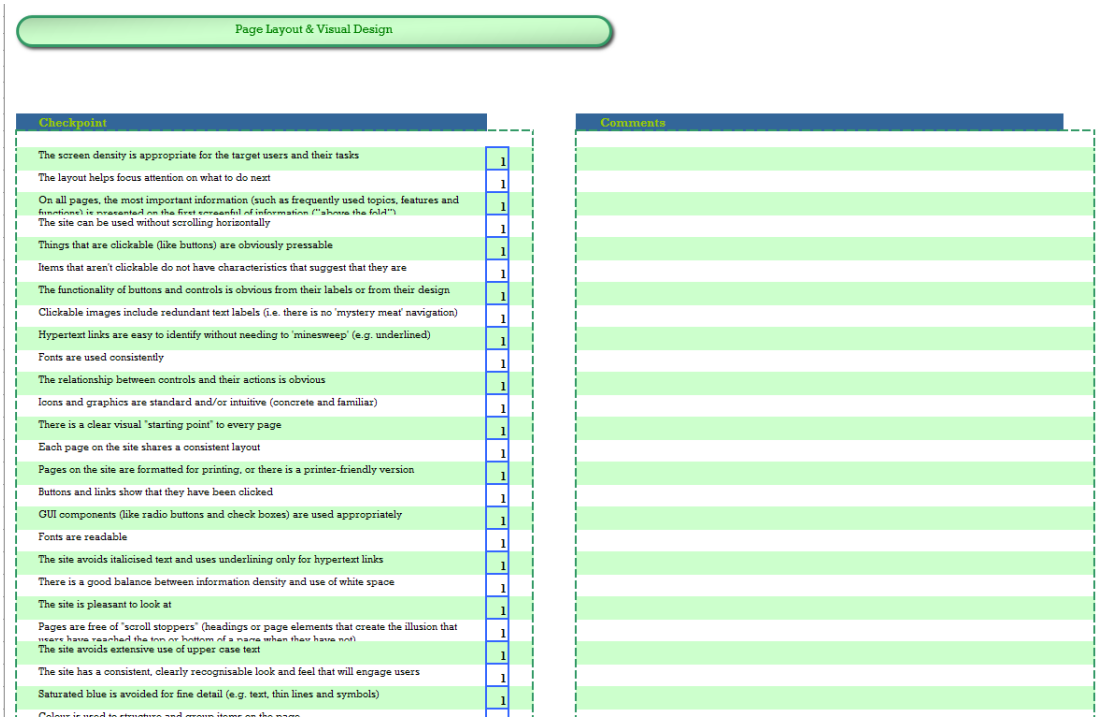
FIGURE F.7: Sheet 7



Writing & Content Quality

Checkpoint		Comments
The site has compelling and unique content	0	
Text is concise, with no needless instructions or welcome notes	1	
Each content page begins with conclsuions or implications and the text is written with an inverted pyramid style	0	
Pages use bulleted and numbered lists in preference to narrative text	0	
Lists are prefaced with a concise introduction (e.g. a word or phrase), helping users appreciate how the items are related to one another	1	
The most important items in a list are placed at the top	1	
Information is organised hierarchically, from the general to the specific, and the organisation is clear and logical	1	
Content has been specifically created for the web (web pages do not comprise repurposed material from print publications such as brochures)	1	
Product pages contain the detail necessary to make a purchase, and users can zoom in on product images	0	
Hypertext has been appropriately used to structure content	1	
Sentences are written in the active voice	1	
Pages are quick to scan, with ample headings and sub-headings and short paragraphs	1	
The site uses maps, diagrams, graphs, flow charts and other visuals in preference to wordy blocks of text	0	
Each page is clearly labelled with a descriptive and useful title that makes sense as a bookmark	0	
Links and link titles are descriptive and predictive, and there are no "Click here!" links	1	
The site avoids cute, clever, or cryptic headings	1	
Link names match the title of destination pages, so users will know when they have reached the intended page	1	
Button labels and link labels start with action words	1	
Headings and sub-headings are short, straightforward and descriptive	1	
The words, phrases and concepts used will be familiar to the typical user	1	
Numbered lists start at "1" not at "0"	1	
Acronyms and abbreviations are defined when first used	0	
Text links are long enough to be understood, but short enough to minimise wrapping (especially when used as a navigation list)	1	

FIGURE F.8: Sheet 8



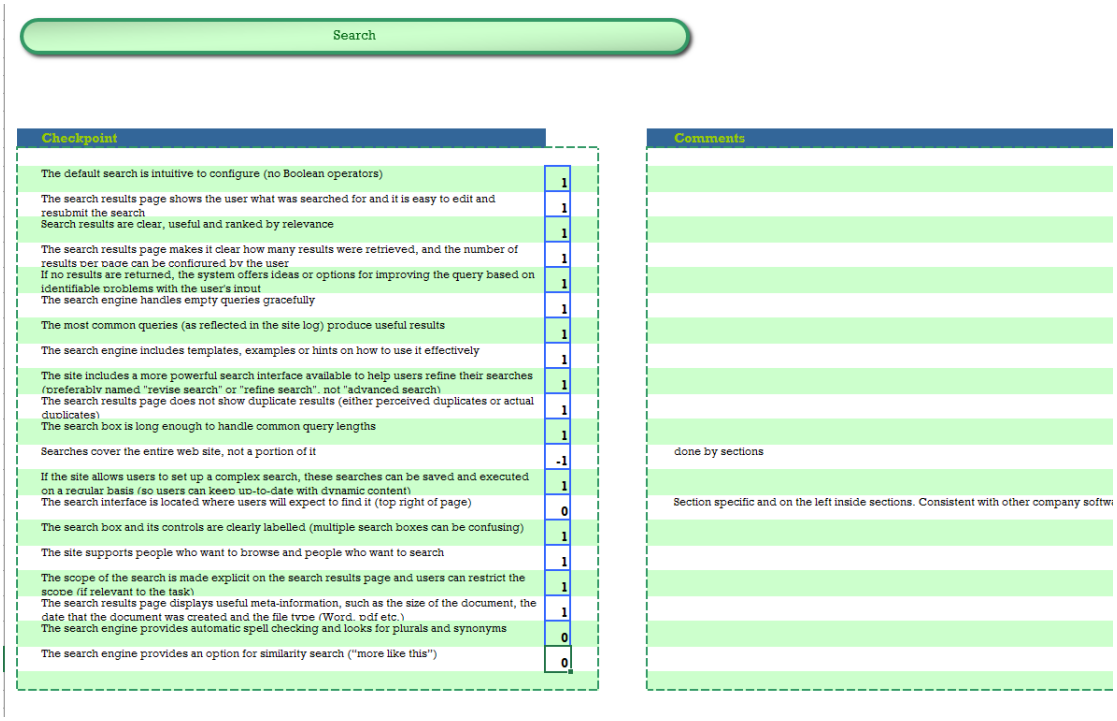
Page Layout & Visual Design

Checkpoint		Comments
The screen density is appropriate for the target users and their tasks	1	
The layout helps focus attention on what to do next	1	
On all pages, the most important information (such as frequently used topics, features and functions) is presented on the first screenful of information ("above the fold")	1	
The site can be used without scrolling horizontally	1	
Things that are clickable (like buttons) are obviously pressable	1	
Items that aren't clickable do not have characteristics that suggest that they are	1	
The functionality of buttons and controls is obvious from their labels or from their design	1	
Clickable images include redundant text labels (i.e. there is no 'mystery meat' navigation)	1	
Hypertext links are easy to identify without needing to 'minesweep' (e.g. underlined)	1	
Fonts are used consistently	1	
The relationship between controls and their actions is obvious	1	
Icons and graphics are standard and/or intuitive (concrete and familiar)	1	
There is a clear visual "starting point" to every page	1	
Each page on the site shares a consistent layout	1	
Pages on the site are formatted for printing, or there is a printer-friendly version	1	
Buttons and links show that they have been clicked	1	
GUI components (like radio buttons and check boxes) are used appropriately	1	
Fonts are readable	1	
The site avoids italicised text and uses underlining only for hypertext links	1	
There is a good balance between information density and use of white space	1	
The site is pleasant to look at	1	
Pages are free of "scroll stoppers" (headings or page elements that create the illusion that users have reached the top or bottom of a page when they have not)	1	
The site avoids extensive use of upper case text	1	
The site has a consistent, clearly recognisable look and feel that will engage users	1	
Saturated blue is avoided for fine detail (e.g. text, thin lines and symbols)	1	
Colour is used to structure and group items on the page		

FIGURE F.9: Sheet 9

Fonts are used consistently	1
The relationship between controls and their actions is obvious	1
Icons and graphics are standard and/or intuitive (concrete and familiar)	1
There is a clear visual "starting point" to every page	1
Each page on the site shares a consistent layout	1
Pages on the site are formatted for printing, or there is a printer-friendly version	1
Buttons and links show that they have been clicked	1
GUI components (like radio buttons and check boxes) are used appropriately	1
Fonts are readable	1
The site avoids italicised text and uses underlining only for hypertext links	1
There is a good balance between information density and use of white space	1
The site is pleasant to look at	1
Pages are free of "scroll stoppers" (headings or page elements that create the illusion that users have reached the top or bottom of a page when they have not)	1
The site avoids extensive use of upper case text	1
The site has a consistent, clearly recognisable look and feel that will engage users	1
Saturated blue is avoided for fine detail (e.g. text, thin lines and symbols)	1
Colour is used to structure and group items on the page	1
Graphics will not be confused with banner ads	1
Emboldening is used to emphasise important topic categories	0
On content pages, line lengths are neither too short (<50 characters per line) nor too long (>100 characters per line) when viewed in a standard browser width window	0
Pages have been designed to an underlying grid, with items and widgets aligned both horizontally and vertically	1
Meaningful labels, effective background colours and appropriate use of borders and white space help users identify a set of items as a discrete functional block	1
The colours work well together and complicated backgrounds are avoided	1
Individual pages are free of clutter and irrelevant information	0
Standard elements (such as page titles, site navigation, page navigation, privacy policy etc.) are easy to locate	0
The organisation's logo is placed in the same location on every page, and clicking the logo returns the user to the most logical page (e.g. the home page)	0
Attention-attracting features (such as animation, bold colours and size differentials) are used sparingly and only where relevant	0
Icons are visually and conceptually distinct yet still harmonious (clearly part of the same family)	0
Related information and functions are clustered together, and each group can be scanned in a single fixation (5-deg, about 4.4cm diam circle on screen)	1

FIGURE F.10: Sheet 10



Search

Checkpoint		Comments
The default search is intuitive to configure (no Boolean operators)	1	
The search results page shows the user what was searched for and it is easy to edit and resubmit the search	1	
Search results are clear, useful and ranked by relevance	1	
The search results page makes it clear how many results were retrieved, and the number of results per page can be configured by the user	1	
If no results are returned, the system offers ideas or options for improving the query based on identifiable problems with the user's input	1	
The search engine handles empty queries gracefully	1	
The most common queries (as reflected in the site log) produce useful results	1	
The search engine includes templates, examples or hints on how to use it effectively	1	
The site includes a more powerful search interface available to help users refine their searches (preferably named "revise search" or "refine search", not "advanced search)	1	
The search results page does not show duplicate results (either perceived duplicates or actual duplicates)	1	
The search box is long enough to handle common query lengths	1	
Searches cover the entire web site, not a portion of it	-1	done by sections
If the site allows users to set up a complex search, these searches can be saved and executed on a regular basis (so users can keep up-to-date with dynamic content)	1	
The search interface is located where users will expect to find it (top right of page)	0	Section specific and on the left inside sections. Consistent with other company softw
The search box and its controls are clearly labelled (multiple search boxes can be confusing)	1	
The site supports people who want to browse and people who want to search	1	
The scope of the search is made explicit on the search results page and users can restrict the scope (if relevant to the task)	1	
The search results page displays useful meta-information, such as the size of the document, the date that the document was created and the file type (Word, pdf etc.)	1	
The search engine provides automatic spell checking and looks for plurals and synonyms	0	
The search engine provides an option for similarity search ("more like this")	0	

FIGURE F.11: Sheet 11

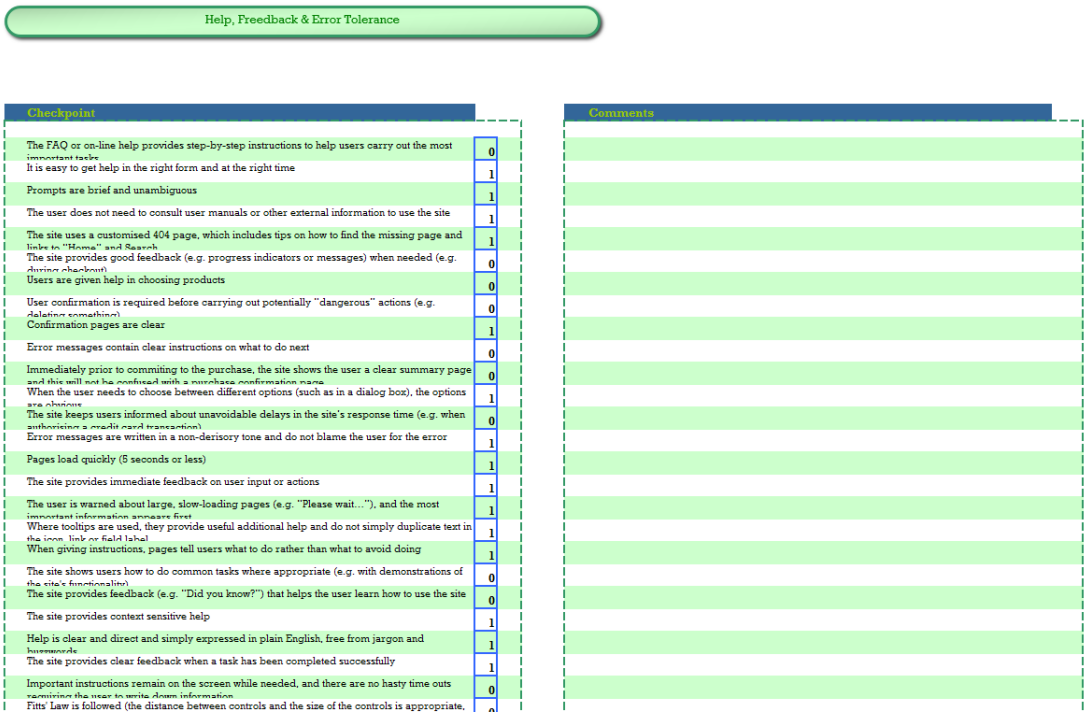


FIGURE F.12: Sheet 12

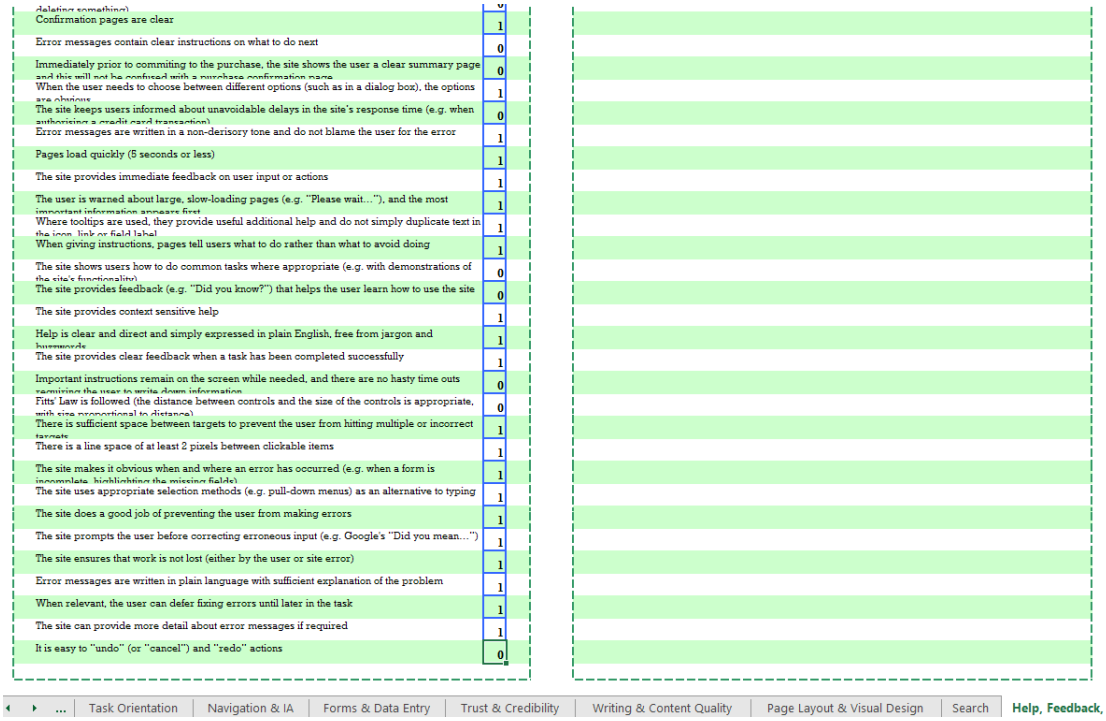
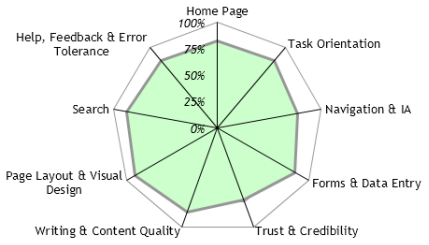


FIGURE F.13: Sheet 13

	Raw score	# Questions	# Answers	Score
Home Page	13	20	20	83%
Task Orientation	28	44	42	83%
Navigation & IA	16	29	29	78%
Forms & Data Entry	16	23	23	85%
Trust & Credibility	6	13	13	73%
Writing & Content Quality	16	23	23	85%
Page Layout & Visual Design	31	38	38	91%
Search	15	20	20	88%
Help, Feedback & Error Tolerance	25	37	37	84%
Overall score		247	245	83%



Results | Home Page | Task Orientation | Navigation & IA | Forms & Data Entry | Trust & Credibility | Writing & Content Quality | Page Layout & Visual Design

FIGURE F.14: Sheet 14