**Project Specification Document**

Data61

Adam Hart

Mitchell Gerard

Nicholas Feikema

Tyler Losinski

**Table of Contents**

Section Page

[1. Project Vision and Objectives 1](#_Toc433722625)

[1.1 Project Scope and Vision 1](#_Toc433722626)

[1.2 Project Goals and Objectives 1](#_Toc433722627)

[2. Project Planning 2](#_Toc433722628)

[2.1 Project Lifecycle 2](#_Toc433722629)

[2.2 Project Setup 2](#_Toc433722630)

[2.3 Stakeholders 2](#_Toc433722631)

[2.4 Project Resources 3](#_Toc433722632)

[2.5 Assumptions 3](#_Toc433722633)

[3. Project Tracking 4](#_Toc433722634)

[3.1 Tracking 4](#_Toc433722635)

[3.2 Communication Plan 4](#_Toc433722636)

[3.3 Deliverables 5](#_Toc433722637)

[3.4 Project Metrics 6](#_Toc433722638)

[4. Requirements (User Stories) 7](#_Toc433722639)

[4.1 Overall Description 7](#_Toc433722640)

[4.2 Users and Roles 7](#_Toc433722641)

[4.3 Use Case Diagrams 8](#_Toc433722642)

[4.4 User Stories (Requirements) 8](#_Toc433722643)

[4.5 Constraints and Limitations 13](#_Toc433722644)

[5. Design 14](#_Toc433722645)

[5.1 Introduction 14](#_Toc433722646)

[5.2 Scope 14](#_Toc433722647)

[5.3 High-Level Component Design 14](#_Toc433722648)

[5.4 Class Diagram 15](#_Toc433722649)

[5.5 Activity Diagrams 16](#_Toc433722650)

[5.6 Sequence Diagram 17](#_Toc433722651)

[5.6 Alternative Designs and Design Rationale 17](#_Toc433722652)

[5.7 Data Architecture 18](#_Toc433722653)

[6. User Interface 19](#_Toc433722654)

[6.1 UI Description 19](#_Toc433722655)

[6.2 UI Mockup 19](#_Toc433722656)

[7. Project Closure 20](#_Toc433722657)

[7.1 Goals / Vision 20](#_Toc433722658)

[7.2 Delivered Solution 20](#_Toc433722659)

[7.3 Remaining Work 20](#_Toc433722660)

[8. Definitions and Acronyms 21](#_Toc433722661)

# 1. Project Vision and Objectives

## 1.1 Project Scope and Vision

Data61 receives many datasets collected from each province of Australia, and each province collects its data using its own independent schema. Current attempts to reconcile these data with commercial data integration tools has proved to be slow and frustrating. To address this problem, Data61 is developing its own data integration tool tailored to provide the tools and functionality they require in integrating their data. This project is concerned with laying down the framework of the back-end API which will consume the unintegrated data from Australia's provinces and reformat it into a pre-defined schema that can be consumed by a front-end system currently in development. This framework will be able to consume small datasets and act as a prototype for future expansion into a full suite of functions. The final delivery of the project will be in early May of 2017.

## 1.2 Project Goals and Objectives

|  |  |
| --- | --- |
| **#** | **Goal or Objective** |
| 1 | Make the code generalized – future developers should be able to look at our code and use it as a base case in writing code to handle other data sets. |
| 2 | Document our work well - our code should be easy to maintain while we work and continue to be maintainable by future developers after we are gone |
| 3 | Make the system support multiple data types - our code should have some form of support for common data set formats including but not limited to CSV, JSON, and Excel |
| 4 | Have a working prototype ready by late February/early March – this prototype should be able to correctly integrate a single test data set as a proof of concept. |
| 5 | Have a second working prototype ready by late March/early April – this prototype should demonstrate the API's ability to be extended to support multiple data sets |
| 6 | Learn about the agile software development process – behind the deliverable, the highest priority goal of the project is to enrich ourselves with the experience of developing software in an environment that emulates the real world. Our work in this project should set a precedent for how we approach future job opportunities. |
| 7 | Have fun working on the project |

# 2. Project Planning

## 2.1 Project Lifecycle

Our team will use a type of agile approach with a two-week sprint system. Every sprint will be divided into two week-long phases, the first for implementing the user story tasks assigned for that sprint and the second for validation, refinement, testing the implementation and buffer in case the user stories reach unexpected troubles. User story tasks will be organized via Trello boards so we can keep a constant monitor on who is assigned what task and what stage of completion each task is in. Meetings will occur with the sponsor once per week (schedule allowing) to assess progress and rebase our sprint goals, with small in-person meetings for members of the team occurring after class or by appointment as necessary. Slack will be used for continuous short-form communication and event organization among the team members.

## 2.2 Project Setup

|  |  |
| --- | --- |
| **#** | **Decision Description** |
| 1 | Programming languages to use. C#, Java, or JavaScript |
| 2 | Type of agile methodology |
| 3 | Type of Database to use. Oracle, MySQL, MongoDB, or Microsoft SQL |

## 2.3 Stakeholders

|  |  |
| --- | --- |
| **Stakeholder** | **Role** |
| Iman Avazpour | Sponsor |
| Alex Radermacher | Instructor |
| Dean Knudson | Instructor |
| Adam Hart | Team member, Scrum Master |
| Mitchell Gerard | Team member |
| Nicholas Feikema | Team member |
| Tyler Losinski | Team member, Project Manager |

## 2.4 Project Resources

|  |  |  |
| --- | --- | --- |
| **Resource** | **Resource Description** | **Quantity** |
| Slack | Communication tool provided for free online | 1 |
| Capstone Team | Our team of students who will be the primary developers of the project. | 4 |
| GitHub | Open source repository that we will use for version control | 1 |
| Trello | Organization tool for keeping track of our groups progress | 1 |

## 2.5 Assumptions

|  |  |
| --- | --- |
| **#** | **Assumption** |
| A1 | The capstone team and mentors will be able to meet face to face once a week. |
| A2 | All team members have access to the tools needed to program effectively |
| A3 | Team members know how to use version control and all the tools we will use to communicate |
| A4 | We will have sufficient time to complete the project our company has proposed thus far |

# 3. Project Tracking

## 3.1 Tracking

|  |  |  |
| --- | --- | --- |
| **Information** | **Description** | **Link** |
| Code Storage | Project code will be stored in a public GitHub repository | Link |
| Task Assignment | Sprint and task assignments will be handled with Trello | Link |
| Project Documents and Assignments | Weekly reports, specification and design documents, etc. will be stored on Slack | Link |
| Continuous Integration | Continuous integration will be dependent on what language ends up being used | Link |
| Regression Testing | Regression testing will be dependent on what language we end up using | Link |

## 3.2 Communication Plan

Regularly Scheduled Meetings

|  |  |  |
| --- | --- | --- |
| Meeting Type | Frequency/Schedule | Who Attends |
| Conference Call/Skype | Weekly (Wednesday at 4:30PM) | Project team and mentor |
| Team Meeting | Weekly | Project team |
| Short Meeting | Weekly in class | Project team |
| Sprint Planning Meeting | Start of each sprint | Project team and mentor |
| Sprint Retrospective Meeting | End of each sprint | Project team |
| Sprint Review Meeting | End of each sprint | Project team, mentor, and sponsor |

Information To Be Shared Within Our Group

|  |  |  |  |
| --- | --- | --- | --- |
| Who? | What Information? | When? | How? |
| Project team | Task assignments & General scrum information | Weekly | Team meetings, listing in Project Specification. |

Information To Be Provided To Other Groups

|  |  |  |  |
| --- | --- | --- | --- |
| Who? | What Information? | When? | How? |
| Sponsor and mentor | Final deliverables | At completion of project | Project specification doc., code, Power Point presentation |
| Sponsor and mentor | Weekly report | Weekly | Email and Trello/Slack access |
| Sponsor and mentor | Progress on project | At the end of each sprint | Access to GitHub repository |

Information Needed From Other Groups

|  |  |  |  |
| --- | --- | --- | --- |
| Who? | What Information? | When? | How? |
| Sponsor and mentor | Requirement changes | Start of each sprint | Conference call or meeting with sponsor and mentor. |

## 3.3 Deliverables

|  |  |
| --- | --- |
| **#** | **Deliverable** |
| 1 | Study results ***(if any)*** |
| 2 | Code |
| 3 | Test and test results |
| 4 | Build process documents***(if any)*** |
| 5 | Install process documents***(if any)*** |
| 6 | Administrator or user manual |
| 7 | Postmortem document |
| 8 | Final report (final PowerPoint presentation, 3 minute video, and final sprint) |

## 3.4 Project Metrics

|  |  |  |
| --- | --- | --- |
| Metric | Frequency | Location |
| Estimated User Story Points | Per Sprint at the start of each sprint | At the beginning of Individual Sprint  (Section 4.5) |
| Actual User Story Points Completed (Velocity) | Per Sprint at the end of each sprint | At the beginning of Individual Sprint  (Section 4.5) |

# 4. Requirements (User Stories)

## 4.1 Overall Description

This project aims to be a proof of concept for a future data integration system that can accept two or more schemas along with a JSON instruction file and use them to output arbitrary code that, when run on the end-user system, can merge the datasets described by the input schemas to a defined target. This system will eventually be wrapped in a ReSTful API and tied to a front-end program currently in development. The end goal is to have a backbone that communicates with JSON, but a varied number of filetypes should be supported in the future.

The first iterations of this prototype will simply be a standalone application that can accept small dataset samples and small well-defined JSON instructions and perform integrations internally, outputting the result data sets. This is so we can verify that the prototype functions as desired and has potential to be expanded into a full project later. After the prototype is completed, this integration functionality should be abstracted into generation of arbitrary code that can be run on any system.

## 4.2 Users and Roles

The nature of this project being a backend prototype means that there is generally only one actor, the User. More actors may be introduced as the project evolves.

|  |  |
| --- | --- |
| **User** | **Description** |
| User | A hypothetical user of the program who wants to perform integration on a given set of datasets. |

## 4.3 Use Case Diagrams

N/A

## 4.4 User Stories (Requirements)

**SPRINT 1**

**Estimated User Story Points: ???**

**Actual Completed User Story Points:** TBA

**Main User Story:** As a User, I want the system to be able to parse two data sets and an instruction file so that I can transform one data set and append it to the other

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| 100 | 02-13-2017 | As a user,  I want the system to be able to parse CSV files  So that it can extract the schema and load datasets | **T** | **5** |  | | **0%** |
| **Acceptance Criteria** | | | **Verification** | | | | | |
| **110** | Must be able to correctly import arbitrary data from the test datasets | | **Write a test case that reads in 5 lines of each test dataset and assert that the read data matches hardcoded values** | | | | | |
| **111** | Must be able to extract the schema of the test datasets | | **Write a test case that reads in each test dataset and assert that the read data matches hardcoded values** | | | | | |
| **112** | Must be scalable to a CSV of a relatively small arbitrary width | | **Write a test case that makes the program read both test files and ensure that the program does not fault** | | | | | |
| **ID** | **Tasks** | | | | | **Resource** | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| 200 | 02-13-2017 | As a user,  I want the system to be able to parse JSON files  So that it can interpret how the data should be transformed | **T** | **3** |  | | **0%** |
| **Acceptance Criteria** | | | **Verification** | | | | | |
| **210** | Must be able to correctly parse JSON | | **Write a test case that reads a test JSON file and verifies the data against hardcoded values** | | | | | |
| **ID** | **Tasks** | | | | | **Resource** | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| 300 | 02-13-2017 | As a user,  I want the system to accept a source dataset and merge it with a target data set using a JSON file as instructions on what to merge So that I may combine my datasets | **T** | **8** |  | | **0%** |
| **Acceptance Criteria** | | | **Verification** | | | | | |
| **310** | Program must be able to differentiate a source dataset from a target dataset | | **Write a test case that can identify flags that are passed with input datasets** | | | | | |
| **311** | The correct data must be loaded into the target dataset as dictated by the JSON file | | **Write a test case that takes a subset of the input datasets and verifies the output** | | | | | |
| **ID** | **Tasks** | | | | | **Resource** | | |

## 4.5 Constraints and Limitations

|  |  |
| --- | --- |
| **Constraint** | **ID** |
| Team will be working with an overseas company we may not be able to get a hold of at short notice. | 1 |

# 5. Design

## 5.1 Introduction

<< Briefly describe the major aspects of the design and, if applicable, how a developer will use it. **Example:** “Create and post a General Ledger transaction using the glTrx routines. Perform account inquiries with gjJournal routines.” >>

## 5.2 Scope

<< Provide a brief overview of the scope of this design. Also touch on anything that will not be included in this document.

**Example:** This document is a high level overview of the design decisions and paradigms. The primary component that needs design is the R visualization library which has a variety of functions for creating different visualizations. An example of Azure ML machine learning will be shown using the cloud based tools, this is primarily designed through a UI and supplemented by custom R scripts that will be written as needed. The MapReduce function is likely beyond the scope of this project, and thus will not be included in the initial revision of this design document. >>

## 5.3 High-Level Component Design

<< Create a diagram of the high-level components or modules in the program, linking them with arrows to show any dependencies. Also complete the tables to provide a description of each module as well as the table which traces components to their related requirements. >>

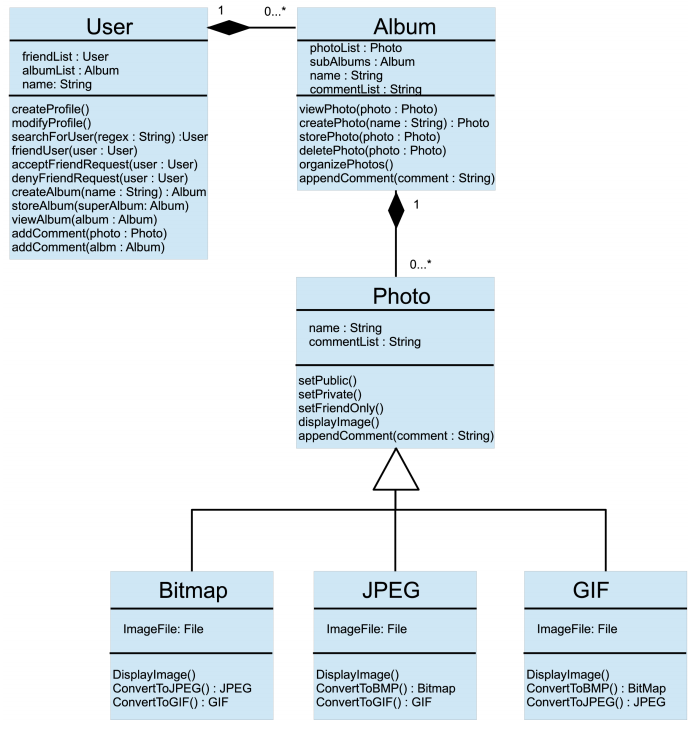
**Example:**

|  |  |  |
| --- | --- | --- |
| **Component** | **Related Requirements** | **Description** |
| R Data Import Function | 100, 300 | Takes the reduced telemetry data, imports to R, cleans and orders the correct fields |
| R Statistics Function | 300 | Allows the user to choose from a variety of statistics about an individual session or form, or an overview of the entire data set. |
| R Visualization Function | 200, 900 | Allows the user to choose the type of visualization desired and generate any of number of them based upon the data that was imported |
| Machine Learning Model | 400, 500, 1000 | Takes data from a test data set and ultimately predicts the users next action, this will ultimately be able to dynamically process telemetry as it happens |

## 5.4 Class Diagram

<< Include a class diagram for all classes to be designed. Optionally include major data elements of those classes and important methods and functions that will be used by other classes. >>

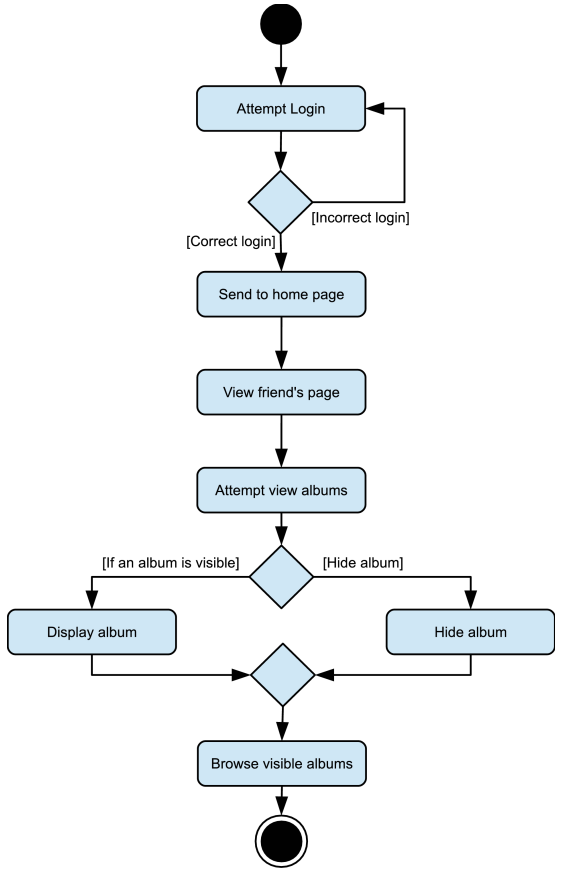
**Example:** Online Photo Collection



## 5.5 Activity Diagrams

<< Include activity diagrams for important workflows in the program. At least one diagram should be included for the main workflow in the program. Optionally include labels that indicate which component is responsible for that part of the activity. Activity diagrams for components which perform complex tasks should also be included. >>

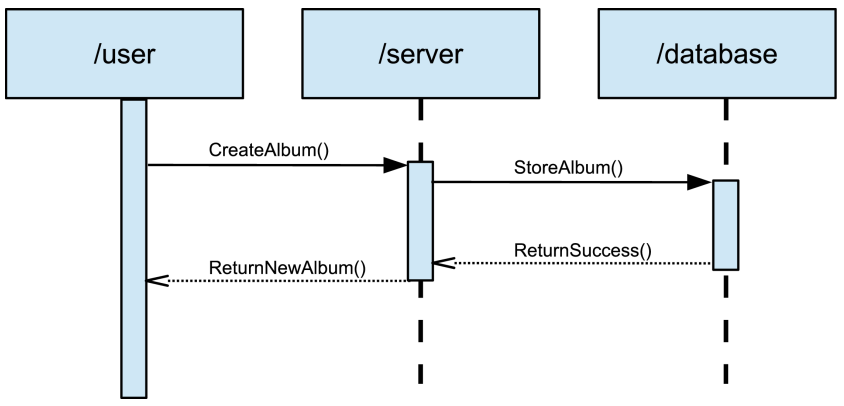
**Example:** View friend’s photo album



## 5.6 Sequence Diagram

<< Include sequence diagrams for important functionality of the program to indicate control flow. These diagrams should include classes found in the class diagram and use the methods for those classes to show the interaction between them. >>

**Example:** Create new album



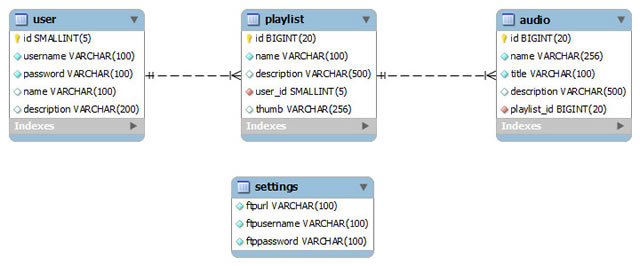
## 5.6 Alternative Designs and Design Rationale

<< Provide information regarding other designs which were considered, but not chosen. Provide rationale for why the chosen design was selected over the alternative candidate designs. >>

## 5.7 Data Architecture

<< Include any information or diagrams that provide details about databases, xml configuration files, or other data structures that are a part of the system. If a very specific format is required, it may be worthwhile to provide a more robust description or a detailed design such as a database schema. >>

Example: Online Audio Gallery



# 6. User Interface

## 6.1 UI Description

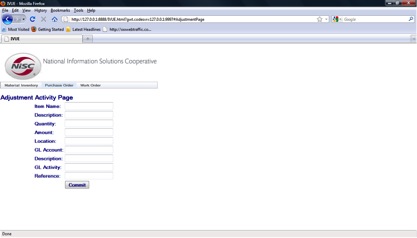
<< Provide a brief description of the UI that will be used in this program and how users will interact with the program.

**Example:** We are creating a research project using R and Azure ML. R is a statistical programming language that uses the R console, which will be the primary means of interacting with our code. This uses standard console UI, and it is not in the scope of our project to create a UI on top of that. The Azure ML machine learning model has its own built-in drag and drop UI for the different code modules we create, so there is no need to generate any sort of AI for that portion of the project either. >>

## 6.2 UI Mockup

<< Create a mockup of the user interface. This can be a simple drawing that demonstrates key parts of the user interface or a screenshot of a prototype created within an IDE. >>

**Example:**

****

# 7. Project Closure

## 7.1 Goals / Vision

**<<** Provide an update to the vision statement that was originally stated in the Project Initiation document.

**Example:** Our original goals for this project were to take telemetry data from Dynamics AX, analyze and visualize that data, then create machine learning models in Azure ML to predict user navigation based upon previous actions. Through the course of the project, these goals were altered so that the primary goal became creating a well-documented, extensible R language package that facilitated cleaning and importing telemetry data, and contained a variety of useful analysis and visualization functions to make the raw data more understandable. >>

## 7.2 Delivered Solution

<< Provide a high-level description of what was planned and what is being delivered.

**Example:** Our solution delivered primarily consisted of a fully documented, fully featured R language package that contained the functions for importing telemetry data, cleaning, separating, and isolating that data, then performing a variety of analysis, statistics, and visualizations on that data. This R package has standard R documentation, a full suite of unit tests, and an integrated manual and help documentation to allow anyone with a basic familiarity with the R environment to utilize our functions. >>

## 7.3 Remaining Work

<< Provide a short summary of what should be done next, ways of further improving the project, or any additional recommendations.

**Example:** We created a prototype of a web-based UI for our R package using Shiny in an attempt to make the functionality more accessible for those unfamiliar with R. There are a few minor functions implemented, and a basic UI setup, but a good deal of work would be required to flesh out this web app with all of the functions currently available in the R package, should Microsoft choose to pursue this UI overlay further. >>

# 8. Definitions and Acronyms

<< This section provides a definition for terms or acronyms used in this document which may not be familiar for all users. >>

**Example:**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Enterprise Resource Planning (ERP) Software | Software that tracks payroll, inventory, labor, capital, etc. for businesses to increase efficiency and manageability. |
| Dynamics AX | Microsoft’s popular ERP software solution with the capabilities to manage small and large businesses. |
| Microsoft Azure | Microsoft’s cloud computing platform that houses modules such as Azure ML (Machine Learning) and Azure HDInsights (Hadoop). |
| R | Open source statistical analysis software used by many developers in data science and analysis. |
| Form | A window or application page in Dynamics AX that contains a set of controls or functions. |